Decentralised Multi-Robot Exploration using Monte Carlo Tree Search

Sean Bone[1], Luca Bartolomei[1], Florian Kennel-Maushart[2] and Margarita Chli[1]
[1]Vision For Robotics Lab, ETH Zürich Switzerland and University of Cyprus
[2]Computational Robotics Lab, ETH Zürich Switzerland

***Abstract*— Autonomous robotic systems are useful in automating tasks such as inspection and surveying of unknown areas, where speed is often an important factor. In order to effectively reduce the time required to complete missions, an efficient exploration and coordination strategy is needed. In this spirit, this work proposes an approach based on the Monte Carlo Tree Search (MCTS) algorithm to guide robots during exploration missions. Our method first expands a search tree of possible actions from the robot's position towards unknown regions, and then selects the sequence of movements that best drive the exploration process forward with respect to a given reward function. The proposed approach, which is able to balance short- and long-term decision-making, is then extended to accommodate the presence of multiple robots, in a bid to push the efficiency of exploration further. Our method allows for the coordination of the robots' movements in a decentralized manner, relying on point-to-point communication. This results in an efficient strategy, which we refer to as Decentralized Monte Carlo Exploration (DMCE). The experimental results demonstrate that our pipeline outperforms a greedy exploration approach, as well as state-of-the-art planners, with up to 30% reduction in exploration times in a series of real-world maps.**

I. Introduction

Adopting mobile robots to automate tasks such as inspection, 3D reconstruction and search-and-rescue holds the potential for faster completion times and safer operations. While these applications have their own unique challenges, they share a common need for a reliable navigation system to guide the process, especially when a robot has to traverse unknown environments. The process of navigating unknown scenes, aiming at the complete coverage of an area of interest, is commonly referred to as *exploration*, and is one of the most widely studied topics in the robotics research community [1]. In order to boost the efficiency of missions, this paradigm has also been extended to employ multiple robots [2], which makes it possible to cover large areas faster, while benefiting from increased robustness to failures of any individual robotic agent. However, achieving efficient coordination of a robotic team is a complex task which poses significant challenges from a path-planning perspective [3], [4]. While solutions to co-localize multiple robots in a shared map exist [5], [6], coordinating the movements of the agents efficiently, avoiding duplicated work, is still an open research question. To this end, simple methods based on space partitioning are popular [7]; nonetheless, they are unable to cope with more complex environments. Other, more flexible approaches [2], [8] drive the exploration process using the map of the scene generated online, and outsource the coordination problem to an external computational unit, where all the available information, such as robots' poses and global maps, is stored. Therefore, since this class of planners solves the planning problem *centrally*, assuming continuous communication with the agents, it is characterized by poor scalability with respect to the number of robots in the team and the complexity of the environment. By contrast, *decentralized* methods [9]–[11] do not require a central planning station, instead allowing each robot to plan autonomously while communicating with peers in their vicinity. The increase in robustness and the lower requirement of computational power come at the cost of more complex networking solutions, as well as the risk of less efficient coordination.

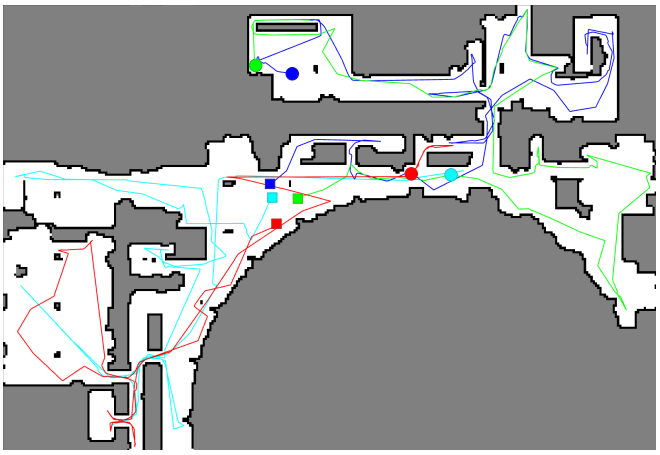


Fig. 1: Top-view of the paths followed by four robots exploring the map *Urban* [12], shown in different colors. Obstacles are depicted in black, and free space in white. The proposed exploration algorithm, dubbed *DMCE*, minimizes the time required to cover the area of interest by coordinating peers in a decentralized fashion. Start and end locations are shown as squares and circles, respectively.

Motivated by these challenges, in this work we propose an exploration strategy based on the Monte Carlo Tree Search (MCTS) algorithm [13], aiming at the efficient coverage of unexplored areas of interest while maintaining flexibility with respect to a variety of mission goals and requirements. Furthermore, we demonstrate how our approach can be extended to multi-robot settings in a decentralized fashion (Fig. 1). In a nutshell, the proposed planner attempts to minimize the time required to cover the areas of interest

This work was supported by NCCR Robotics, the Amazon Research Awards and the HILTI group.

Video – `https://youtu.be/3RfDq8wlnPI`
Code – `https://github.com/VIS4ROB-lab/dmce`

by first creating a tree of possible actions that the robots can undertake, and then selecting the one that maximises the expected exploration gain. In order to allow for the changes to the map as the robots traverse the environment, our strategy continuously adjusts the underlying tree structure, and balances short-sighted greedy behaviour with longer-term actions. We demonstrate that the proposed solution yields better results than a greedy approach and state-of-the-art planners, and that it can be scaled up to multiple robots following the paradigm of decentralized coordination.

In brief, the contributions of this work are:

- the design of an exploration strategy based on a novel version of the MCTS algorithm, modified to work in the context of robotic exploration,
- the integration of frontier-based actions, boosting the efficiency of the exploration process,
- the extension of the single-robot case to a multi-robot decentralized approach,
- extensive evaluations in simulation, demonstrating better performance than the state of the art, and
- the source code of the planning strategy and simulation.

II. Related Works

Exploration planning has been a topic of extensive research in robotics due to its broad applicability. With the outlook of practical application, the common goal is often fast scene exploration by eliminating the unknown space as quickly as possible. To this end, frontier-based methods have been particularly successful, since they target regions on the boundary between known and unknown space [14]. There are different criteria used to decide which frontier to explore next, such as their proximity to the robot [15], following a greedy selection strategy [16] or having global planning dictate their selection [17]. Other works focus on the problem of active Simultaneous Localization and Mapping (SLAM) for indoor robotic navigation, with the aim of reducing the localization error at the cost of longer travelled distances [18], [19]. While these works concentrate on the coverage problem, another flourishing research direction is 3D reconstruction of unknown structures, whereby the task is to propose viewpoints for accurate surface estimation [20].

Aiming at improving their efficiency and reducing mission times, these different paradigms of exploration planners have been extended to accommodate for the presence of multiple robots. Even though this implies harder challenges from a co-localization and mapping perspective as mentioned in [3], in this work we focus exclusively on the coordination aspect. While single-robot strategies can be extended to multi-agent setups by partitioning the area of interest according to the number of robots [7], this does not ensure efficient collaboration between them or resilience to single-robot failures. With the objective of addressing this limitation, cooperative frontier-based approaches have been proposed in centralized [2], [8], [21] and decentralized [10], [22] fashions. Centralised methods feature a hub or ground station where a global multi-robot plan is first computed, and then communicated to the agents. To this end, the early work by Burgard *et al.* [2] aims at reducing the overlap in explored areas, by down-scaling the information gain of a candidate frontier if another robot is assigned to a different frontier in its vicinity. The work in [21] proposes a similar solution, but adopts a sampling-based approach, where frontiers are found and assigned by means of a Rapidly-exploring Random Tree (RRT). Alternatively, Tian *et al.* [23] solve a Multiple Travelling Salesman Problem (mTSP) to allocate each agent to a candidate frontier. While this method holds the potential of finding the globally optimal solution, its computational complexity scales exponentially with the number of robots. Furthermore, in exploration tasks, only partial knowledge of the environment is available, severely limiting the *a priori* optimality of the plans. Alternatively, decentralized approaches allow each robot to plan autonomously, and coordination is ensured by allowing robots to share local information with the other agents in their vicinity. While these methods are characterized by higher flexibility and better scalability with respect to the team size, they sacrifice the global optimality of the trajectories as they rely on local information only. Kabir and Lee [11] aim at mitigating this limitation basing their approach on optimal transport theory, while other solutions utilize Artificial Potential Fields (APFs) [24]; however, this last class of methods is well-known to be bound to find only local optima. This effect is exacerbated in multi-robot settings, where the presence of multiple robots highlights the suboptimality of their frontier-assignment process. Aiming at overcoming this limit, Yu *et al.* [25] propose a strategy, dubbed Multi-robot Multi-target Potential Field (MMPF), where the objective is to eliminate local minima to boost the efficiency of target assignment. Another line of research focuses instead on the Monte Carlo Tree Search (MCTS) algorithm. To this end, Best *et al.* [9] propose a decentralized MCTS-based planner for robotic active perception tasks, whereby the coordination problem is addressed by performing a distributed optimization over the joint action-space of the robots. However, this method cannot be easily transferred to frontier-based exploration scenarios, since their focus is on long-term optimal plans, which cannot be efficiently computed given the short-horizon nature of exploration planning. Lauri and Ritala [26] model exploration as a Partially Observable Markov Decision Process (POMDP), using frontier-based planning as a fallback strategy. However, they do not address the multi-robot case.

Inspired by these shortcomings, this work proposes an exploration strategy based on MCTS, that is then extended to multi-robot settings in a decentralized fashion. The proposed planner, titled Decentralized Monte-Carlo Exploration (DMCE), generates actions guiding the robots towards the most informative frontier. Coordination is ensured by allowing robots to share information about the map and their current plan with nearby agents, effectively reducing the overlap between assigned regions and redundancy in the exploration process. Consequently, our approach yields better exploration performance in terms of time required to explore a region of interest compared to other state-of-the-art approaches [21], [25] and a greedy strategy, effectively boosting the efficiency

of robotic exploration missions.

III. Problem Description

The overall problem considered in this work is the exploration of 2D areas of interest using autonomous robots, with the objective of minimizing the time required to complete the mission, i.e. to fully explore the reachable area. We base the proposed exploration strategy on MCTS, and we extend it to multiple agents following a decentralized approach, aiming at efficient fleet coordination. To this end, we assume that the robots are holonomic and co-localized in a common reference frame, and that their poses, together with measurements from range sensors, such as LiDARs, are available at a constant rate. Using this data, we generate an occupancy map of the environment online, and we consider the exploration process complete when all unknown regions are cleared.

IV. MCTS-based Exploration

In its simplest form, the MCTS algorithm generates a tree of candidate action sequences by randomly sampling the space of feasible actions, with the aim of maximising a given reward function. In the context of robotic exploration in 2D space, each node x_s^r, or *state*, in the search tree for robot r represents the robot's position $\mathbf{x}^r \in \mathbb{R}^2$, together with the occupancy map of the environment, estimated by simulating the action sequence from the tree root to x^r. From each initial state x_0^r, the robot can carry out a plan $p^r \in \mathcal{P}^r$, defined as a sequence of feasible actions $a_i^r \in \mathcal{A}(x^r)$ with $i = 0, ..., T$ over the planning horizon T. Thus, a path can be understood as a sequence of states $\{x_0^r, x_1^r, ..., x_T^r\}$ originating from the root x_0^r to the leaf x_T^r of the tree, with each edge between consecutive nodes representing an action. Given a reward function $V(x_0^r, p^r)$, which assigns a score to every state-plan pair, the robot adopts the plan that maximises the expected reward for the current state x_0^r:

$$\hat{p}^r(x_0^r) := \arg\max_{p \in \mathcal{P}^r} \mathbb{E}\left[V(x_0^r, p)\right]. \tag{1}$$

Note that in standard MCTS implementations the reward function considers the final state (i.e., $V(x_T^r, a_{T+1}^r)$), which implies the necessity to store a partially explored map for each tree node. This is memory-intensive and leads to sub-optimal action selection, since the maps quickly become outdated; consequently we define our reward function over the path p, and regenerate x_T^r from the current map. As our objective is rapid exploration, a natural choice for V is the total explored area on the map. Using 2D occupancy grids for mapping, this amounts to:

$$V(x, p) := \frac{N_{known}^f}{N_{cells}} = 1 - \frac{N_{unknown}^f}{N_{cells}}, \tag{2}$$

where N_{known}^f and $N_{unknown}^f$ are the number of known and unknown cells, respectively, after the simulated execution of the plan. Notice that V is normalised with respect to the total number of cells N_{cells} in the grid, in order to account for different map resolutions and environment scales.

In the following, we detail the steps of the iterative growth of the search tree, as depicted in Fig. 2. Notice that MCTS is an *any-time* algorithm, i.e. a solution can always be queried after any complete iteration.

A. Selection

The first step of the algorithm aims at finding the most promising branch of the tree to expand. To this end, the tree is traversed, starting from the root and iteratively selecting the child node that maximises a given utility function $\mathcal{U}$. A common choice for $\mathcal{U}$ is the Upper Confidence bound for Trees (UCT) [27], which balances exploitation of known action sequences versus the exploration of new pathways:

$$\mathcal{U}(x_s) := \mathbb{E}[V_s] + \lambda\sqrt{\frac{\log N_s}{n_s}}, \tag{3}$$

where $\mathbb{E}[V_s] := \tilde{V}_s / n_s$ is the expected value at s, N_s indicates the number of times the parent node of s has been visited. The values $\{\tilde{V}_s, n_s\}$ are stored on each tree node; $\tilde{V}_s$ is the value of traversing node s estimated over n_s passes during the growth of the tree. λ is a parameter that controls the exploration-exploitation trade-off: exploitation is favored when λ is set to zero, as the highest-value path at each iteration is selected, whereas larger values of λ tend to favor exploration and push the iteration closer to a breadth-first approach. Generally, the optimal value of λ is application-specific, and is determined empirically to maximise performance. This process continues until a leaf state x_L^r is found, i.e. until a node with one or more unexpanded children is hit.

1) Accounting for Action Duration: In traditional MCTS, the expected reward for carrying out an action a_s for robot r is estimated as the average value obtained over multiple simulated action sequences. However, due to the varying travel distances to target locations, the different actions require varying lengths of time for execution: the longer an action takes, the more uncertain its outcome, since longer routes imply a higher probability of sudden changes in the map as the environment is traversed. To alleviate this shortcoming, we aim at biasing the planner towards shorter-term actions by modifying the reward expectation as follows:

$$\mathbb{E}[V_s] = \tau^{t_s} \frac{\tilde{V}_s}{n_s}, \tag{4}$$

where $\tau \in (0, 1]$ is a discounting factor and t_s is the duration of action a_s, which is calculated using the straight-line distance to the target and the robots' maximum velocity.

B. Expansion

In this step, the tree is expanded from the leaf x_L^r by selecting an action a at random from the set of feasible actions $\mathcal{A}$. The corresponding state $\tilde{x}_{L+1}^r$ is obtained by simulating a from x_L^r, and then added to the tree. We determine the feasible actions $\mathcal{A}$ by sampling straight-line movements of fixed length in random directions (Fig. 3). In order to ensure safety, actions that would cause the robot to collide with an obstacle are rejected; furthermore, to keep the tree size bounded, avoid redundancy, and improve the computational efficiency of the algorithm, directions that deviate by less than $\alpha = 15°$ from previously sampled

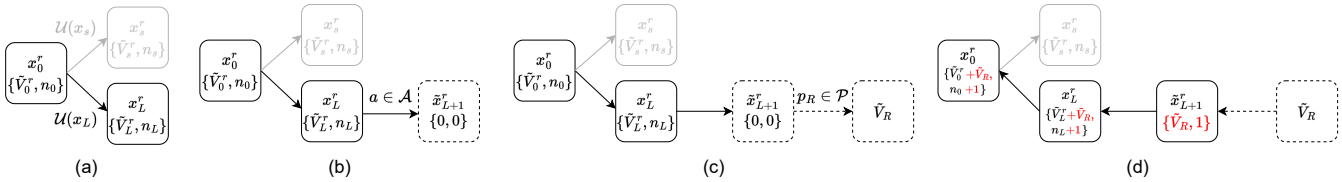


Fig. 2: Schematic overview of the iterative expansion of the search tree in the MCTS algorithm for a single robot r. Each node s stores the value $\tilde{V}_s^r$ of traversing s estimated over n_s passes through the node during the growth of the tree. (a) In the *Selection* step, the tree is traversed from the root x_0^r and the leaf node x_L^r with highest utility $\mathcal{U}(x_L^r)$ is selected. (b) The leaf node is then expanded (*Expansion*) by simulating a random action a and the resulting child node x_{L+1}^r is added to the tree. (c) In the *Rollout*, a path p_R of random actions starting from x_{L+1}^r is simulated and the resulting rollout reward $\tilde{V}_R^r$ is calculated. (d) Finally, during *Backpropagation*, the value of the newly added node (in red) is propagated by traversing the tree in reverse until the root is reached.

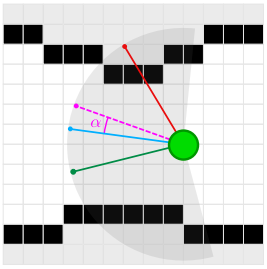


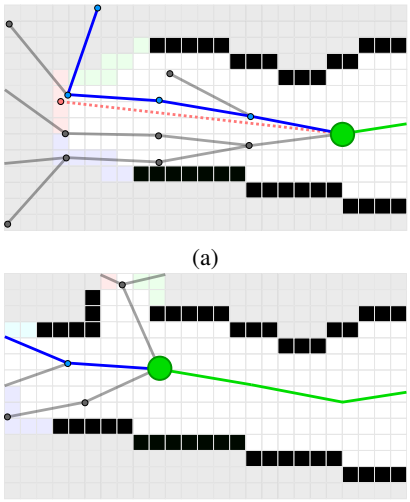
Fig. 3: Schematic overview of the action sampling process in a 2D occupancy map. Given the robot's current position (green blob) and previously selected actions in $\mathcal{A}$ (cyan), candidates are sampled in random directions within a forward-facing arc (light gray area). Valid actions (dark green) are added to the set of feasible actions $\mathcal{A}$, while those leading to collisions (red) are discarded. Similarly, displacements with angular distance α from previously selected actions lower than a threshold (dotted purple line) are rejected.

displacements are discarded. Similarly, in order to prevent unnecessary back-and-forth movements, the sampling of new directions is restricted by a maximum turn angle with respect to the parent action. However, this restriction is relaxed if no suitable actions can be found, e.g. when sampling in the vicinity of an obstacle. The performance of MCTS is very sensitive to the branching factor β: while low values keep the cost of the tree search manageable, they risk producing poor-quality plans. On the other hand, high values of β lead to large computational costs, which must be contained for real-time applications.

Robotic exploration presents further challenges, which we address with the following modifications to the standard MCTS algorithm:

1) Changes in the Environment: During exploration, the map of the environment is subjected to continuous changes, possibly leading to some actions previously considered feasible suddenly becoming invalid, as shown in Fig. 4. In order to overcome this limitation, we propose to re-check the feasibility of actions when traversing the tree during each MCTS iteration, and to prune branches corresponding to infeasible displacements. If a node would be left without children, its set of feasible actions is generated anew.

2) Improving Planning Range: During the sampling process, feasible actions are generated as fixed-length displacements, which directly tie the effective range of the planner to the depth of the search tree. Thus, the computational cost of detecting unexplored areas increases exponentially with their distance from the root of the tree; in practice,



(a)

(b)

Fig. 4: Schematic overview of the re-growing process of the search tree when a new obstacle is discovered. (a) The tree is initially grown from the robot's current position (green blob) by expanding randomly-sampled branches (gray). Additionally, actions connecting the root to the centroids of the clustered frontiers (shown as coloured cells) are added to the tree (red dotted line). The path maximising a given reward function is then selected for execution (dark blue). (b) As the robot moves, the map is explored. During navigation, the feasibility of actions in the tree are checked at a constant rate and invalid actions are pruned. The tree is then re-grown, and a new path is found.

this often causes the search to deteriorate to a random walk, especially in complex environments. In order to allow the robot to detect unexplored regions far from the current position, frontier cells in the map are clustered. The set of actions is then augmented with displacements from the tree root to the n closest frontier cluster centroids. Note that this differs from the approach presented in [26], where frontier-based exploration is only used as a fallback solution.

C. Rollout

After the tree is expanded, a plan p_R composed of a series of randomly selected actions is simulated starting from the state $\tilde{x}_{L+1}^r$ in the *rollout* phase. The length of p_R is limited to a fixed number of actions, known as *rollout depth*. The rollout reward $\tilde{V}_R^r = V(\tilde{x}_{L+1}^r, p_R)$ is then evaluated, and the result stored as the initial value estimate $\tilde{V}_{L+1}^r$ for the new node.

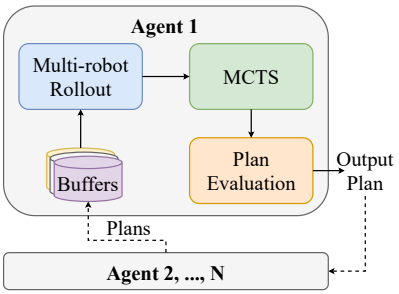


Fig. 5: Schematic overview of the DMCE pipeline. The plan for every robot in the fleet is computed using a multi-agent variant of the proposed MCTS algorithm and then communicated to its peers, while the received plans are stored locally in buffers. The presence of the other team members is considered in DMCE during the multi-robot rollout stage, where we use the the history of the last n_P plans of the other agents to drive the robots towards unassigned areas.

D. Backpropagation

After determining the value of the newly added node in the rollout, this information has to be propagated through the tree. During backpropagation, the search tree is traversed in reverse starting from the new state, and the stored values of each parent node s are updated until the root is reached:

$$\begin{aligned} \tilde{V}_s^r &\leftarrow \tilde{V}_s^r + \tilde{V}_R^r, \\ n_s &\leftarrow n_s + 1. \end{aligned} \tag{5}$$

V. Decentralized Multi-Robot MCTS

We now extend the proposed exploration strategy to a decentralised multi-robot setting, with point-to-point communication. The extended, multi-agent pipeline, named Decentralized Monte-Carlo Exploration (DMCE), is shown in Fig. 5. Assuming the robots are co-localized in a common reference frame, they exchange information about the plan to execute. After collecting information from peers, each robot plans its own actions. Notice that here the map of the environment is built collectively, and we consider the exploration process complete once no new frontiers can be extracted from the map.

A. Coordination by Plan Sharing

After running the MCTS algorithm as descibed in Sec. IV, each robot shares its plan with the other fleet members. The agents store the received information in separate per-robot buffers, containing the most recent n_P plans. Before starting another iteration of MCTS, the robot randomly samples one of these stored plans for each peer, and the map of the environment is modified by simulating each of the selected plans in a multi-robot rollout. This indirectly modifies the reward function in Eq. (2) to discourage the robot from exploring areas that other agents are planning to visit, since those locations are marked as explored during this rollout process. While the overall computational complexity of this approach scales with $\mathcal{O}(N^2)$, it is important to note that, being a decentralised method, the limiting factor is actually the $\mathcal{O}(N)$ scaling of a single MCTS rollout. Furthermore, a benefit of this strategy is its resilience to imperfect communication since, in the case no plan is received from another peer, it is simply not simulated, and the exploration process continues as normal. This implies that in many complex, communications-restricted environments, the typical rollout cost may even scale with $\mathcal{O}(1)$, as most peers do not share plans at all times.

B. Local Rewards

A limitation of the proposed coordination approach is the fact that the reward function of Eq. (2) does not differentiate between the contributions of peers and the impact of the robot's self-plan. To increase the sensitivity of the reward function to the selected plan p^r, we adopt the concept of *local reward* [9]. This is defined as the difference between the reward when robot r performs plan p^r and the case of simply idling with an empty plan $p^\emptyset$:

$$\begin{aligned} V_{loc}(x^r, p^r) &= V(x^r, p^r) - V(x^r, p^\emptyset) \\ &= \frac{N^i_{unknown} - N^f_{unknown}}{N_{cells}}, \end{aligned} \tag{6}$$

where $N^i_{unknown}$ and $N^f_{unknown}$ are the number of undiscovered cells in the map before and after the rollout of plan p^r, respectively. This reward is used for planning, replacing the former definition of Eq. (2).

C. Rollout Discounting

The nature of exploration processes implies the possibility of sudden, large changes in the map of the environment. These can cause discontinuities in the value function $V(x, p)$, leading to the estimates $\tilde{V}_s^r$ being outdated. This has a large impact not only on the exploration performance of the robots, as described in Sec. IV-B.1, but also on the coordination of the fleet. To mitigate this effect, we introduce a discounting factor $\gamma \in (0, 1]$ to the backpropagation step of Eq. (5) as follows:

$$\begin{aligned} \tilde{V}_s^r &\leftarrow \gamma \cdot \tilde{V}_s^r + \tilde{V}_R^r, \\ n_s &\leftarrow \gamma \cdot n_s + 1. \end{aligned} \tag{7}$$

This effectively discounts the contributions of older rollouts, weighting the value estimation in favour of the most recent ones, which are more representative of the current state of the exploration process.

VI. Experiments

To assess the performance of the proposed method, the pipeline is tested in maps of varying difficulty, namely *Urban* and *Tunnels* [12] from the DARPA Subterranean Challenge, and the natural scene *Forest* [28]. These are obtained from scans of real-world environments, and are characterized by different topology of the space, as well as density of obstacles. The experiments are conducted in a custom kinematic simulator, where robots are modelled as circular omnidirectional platforms equipped with LIDAR sensors (1° ray spacing) and only agent-to-environment collisions are considered. We initialise the robots in random positions within a starting area, and we report the total time and travelled distance required to explore the environment, i.e. until no reachable space is left unexplored. The simulated robots' sensor range is $10\,\mathrm{m}$, and the reconstructed map

resolution is $0.25\,\mathrm{m}$ (*Urban* scenario) or $0.5\,\mathrm{m}$ (*Tunnels* and *Forest* scenarios). Our approach is compared against a greedy frontier-selection strategy, and the methods by Umari and Mukhopadhyay [21] and Yu *et al.* [25]. While the greedy planner clusters frontiers and assigns each robot to the closest cluster centroid, Umari and Mukhopadhyay [21] grow an RRT towards unknown areas to passively search for frontiers. These are then clustered, evaluated in terms of expected exploration gain, and finally assigned to the agents. To address fleet coordination, Yu *et al.* [25] create instead a continuous potential field over the navigation area. The potential function for every point in the map is calculated summing the informativeness of frontier clusters in terms of exploration gain, scaled by the length of the shortest collision-free path connecting each candidate point to the centroid of the cluster. To achieve fleet coordination, the potential field is modified by adding a penalty if a point lies within an agent's sensing range; finally, each robot is assigned the most informative location in the field. For fairness of comparison, we use the coordination strategy from [25] discarding the multi-agent state estimation components of the pipeline.

A. Performance Metrics

Performance is tracked by measuring the explored fraction of the *discoverable* portion of the environment. Notice that this differs from the *traversable* region of the environment, as not all traversable map cells can necessarily be reached by the robots (e.g. tight passages). As a result, the maximum discoverable area is determined empirically, and exploration is considered complete when 95% of the discoverable area has been mapped. In an attempt to isolate the exploration rate from the quality of the generated plans, we report the *time* required by each strategy to complete exploration, as well as the total *travelled distance* covered by all the robots. While completion times are a strong indicator of effective coordination, as lower values indicate better spread of the robots in the environment, the lengths of the travelled paths are directly linked to the efficiency of the proposed plans – namely, longer paths imply less efficient planning, for instance back-and-forth movements. Furthermore, we cap the maximum allowed mission time in each scenario, and we mark the run as failure if exploration is not completed within the time limit. This is set to $900\,\mathrm{s}$ and $450\,\mathrm{s}$ in the map *Urban* for the single- and multi-robot cases, while the experiments in *Forest* and *Tunnels* are limited to $350\,\mathrm{s}$ and $900\,\mathrm{s}$, respectively.

B. Results

1) Ablation Study: We study the effect of the proposed coordination strategy by comparing the fully coordinated DMCE pipeline against an MCTS-based strategy which performs no plan-sharing or peer rollouts. The results in Table I indicate that, while DMCE is able to explore the same environment marginally faster than its uncoordinated variant for 2 and 3 robots, its performance degrades with 4 agents. The reason is the increased planning time due to the multi-robot rollout; this is evidenced by the fact that DMCE is able to consistently plan the shortest paths, even with 4 robots, thanks to effect of coordination. Note that, due to practical limitations, the computational resources used were kept constant for all simulations.

Strategy	Number of Robots 1	2	3	4
Ours				
Time [s]	$\mathbf{477 \pm 70.4}$	$\mathbf{262 \pm 55.1}$	$\mathbf{213 \pm 40.1}$	193 ± 48.7
Travelled Distance [m]	$\mathbf{443 \pm 65.9}$	$\mathbf{443 \pm 95.7}$	$\mathbf{498 \pm 92.8}$	$\mathbf{520 \pm 144}$
Ours (uncoord.)				
Time [s]	489 ± 102	$\mathbf{262 \pm 50.1}$	221 ± 44.5	$\mathbf{170 \pm 25.9}$
Travelled Distance [m]	445 ± 83.6	454 ± 68.8	566 ± 97.7	582 ± 99.7

TABLE I: Mean and standard deviation of time and cumulative travel distance required to complete exploration of the *Urban* scenario with DMCE (*Ours*) and its uncoordinated variant (*Ours (uncoord.)*), over 10 runs. The best performance is highlighted in bold.

2) Comparisons to Baselines: We compare our proposed strategy to the baseline methods with two robots in multiple environments, and report the results in Figure 6. In all cases, DMCE outperforms its competitors by a significant margin. While both Umari and Mukhopadhyay [21] and Yu *et al.* [25] perform better than the greedy planner, our method outperforms all three, both in the time to completion as well as in efficiency over the total travelled distance. These differences are due to the different underlying strategies: while the baseline methods drive the robots towards the best frontier in terms of information gain, we explicitly formulate our goal as the uncovering of previously unexplored areas, which allows for more flexible and efficient navigation.

3) Multi-robot Scaling: The three scenarios were chosen to represent different types of environments, ranging from mostly open terrain (*Forest*), where the planners perform most similarly, to corridors with limited connectivity (*Tunnels*) where coordination is most important. The third environment, *Urban*, is middle-ground between these extremes, and for this reason it was chosen for a more in-depth comparison of the multi-robot scaling of each planner. Fig. 7 reports the performance of each planner in the *Urban* environment with varying numbers of robots. While increasing the agent count benefits every strategy, our method is consistently able to outperform other methods.

4) Impact of Communication Range: To further assess the robustness of the different approaches, we test them in the case of limited communication range for point-to-point communication. We report the results in terms of time required to complete the mission for a team composed of two robots. As Fig. 8 shows, even under restricted communication range, our planner outperforms its competitors, leading to lower completion times.

VII. Conclusions and Future Work

Inspired by the need for effective and flexible exploration approaches, this work presents an online exploration strategy based on the MCTS algorithm, allowing for time-efficient coverage of areas of interest. By creating a tree of possible actions, the robot selects the path that maximises the

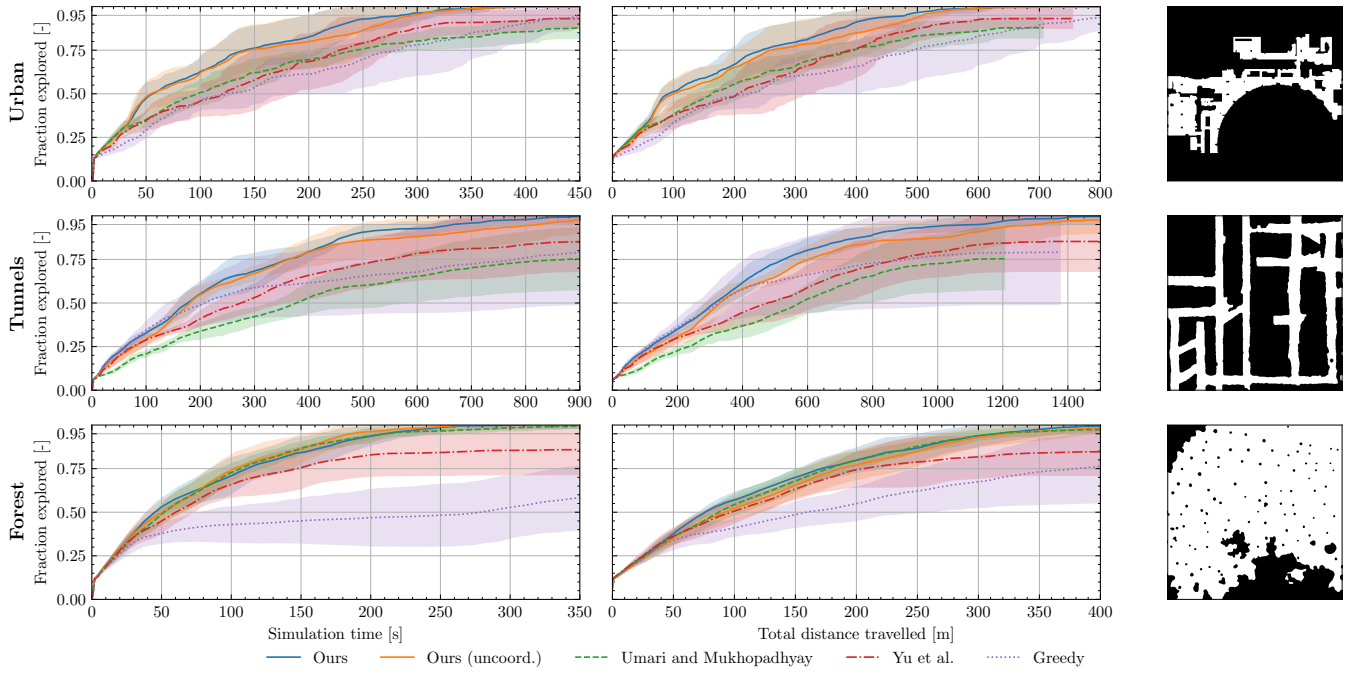


Fig. 6: Planner performance for two-robot exploration of the test environments. Each planner was evaluated for 10 runs; reported are mean (lines) and standard deviation (shaded regions) of the successful runs. Our approach and its variant are in blue and orange, respectively, while the greedy approach is in purple. The method by Yu *et al.* [25] is in red, whereas Umari and Mukhopadhyay [21] is in green. On the far right, we show a top-view of the maps, where the white area is the discoverable space. The map *Urban* has a dimension of $75\,\mathrm{m} \times 50\,\mathrm{m}$, while *Tunnels* of $100\,\mathrm{m} \times 100\,\mathrm{m}$ and *Forest* of $60\,\mathrm{m} \times 60\,\mathrm{m}$.

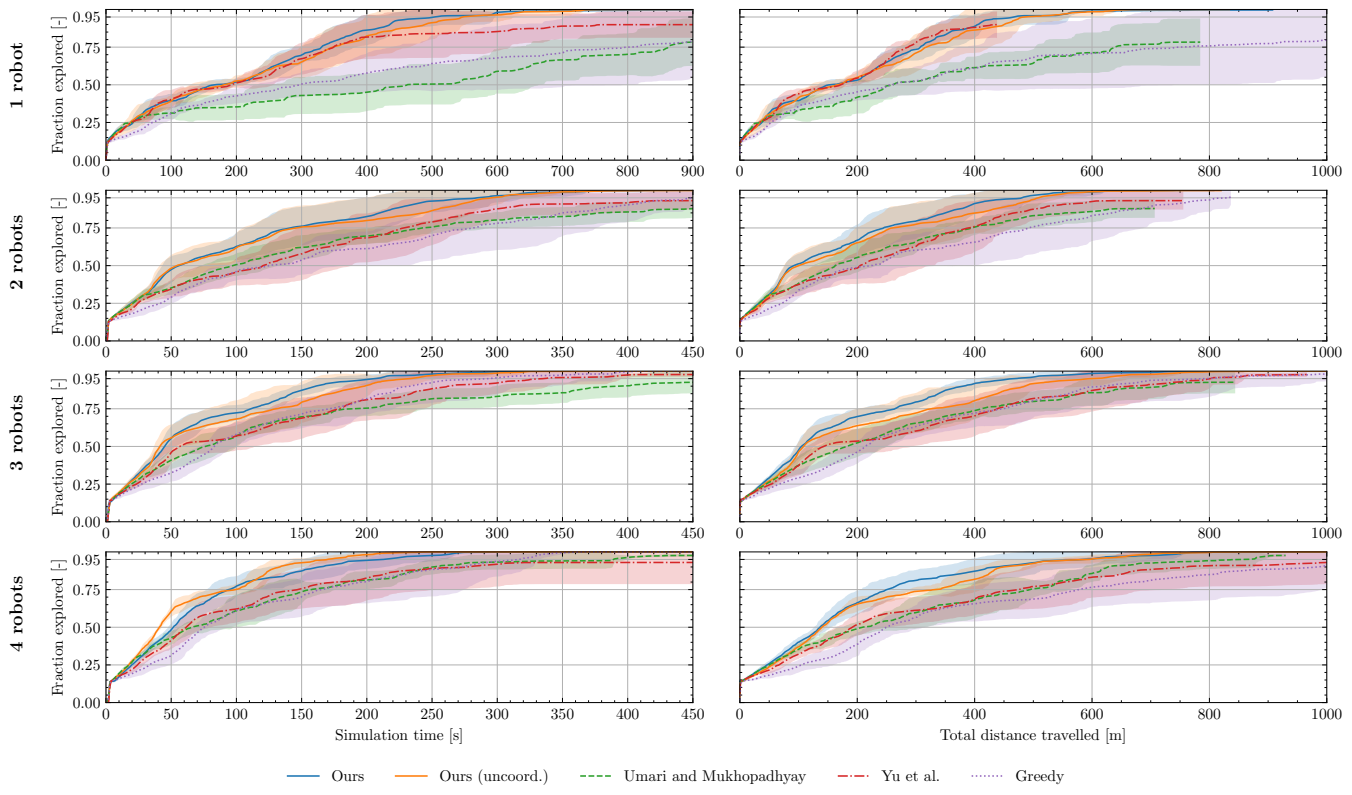


Fig. 7: Planner performance for varying numbers of robots in the *Urban* scenario. Each planner was evaluated for 10 runs; reported are mean (lines) and standard deviation (shaded regions) of the successful runs.

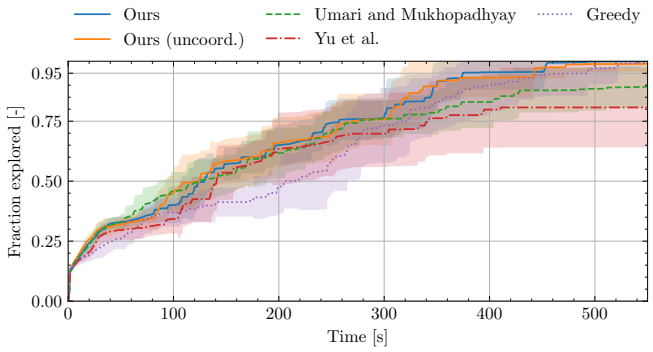


Fig. 8: Two-robot exploration of the *Urban* scenario with line-of-sight restricted communications. Reported are mean and standard deviation of the fraction of explored terrain, as measured at the robots' common depot, averaged over 10 runs.

expected information gain, striking an effective trade-off between short and long-term planning. Furthermore, adding frontier-based actions helps the planner to avoid deadlocks. To account for the changing environment as it gets explored, we continuously adjust the underlying tree structure online, ensuring safety during path execution. Following a decentralized coordination paradigm, the proposed strategy is also extended to accommodate the presence of multiple robots in the environment. Comparisons on a testbed of different real-world maps reveal consistently faster exploration, as well as more effective coordination than the state of the art. Future work will focus on the integration of the proposed pipeline in real platforms, and investigate more advanced strategies to deal with line-of-sight restricted communications. Another interesting avenue for future studies would be the exploration of different kinds of actions, for example through the abstraction of the global environment into a dynamic and distributed topological graph. This would allow the method to be applied to large-scale environments, as well as a more accurate estimation of the displacement times. Finally, while the effectiveness of the proposed coordination strategy is demonstrated by its ability to plan shorter paths compared to the uncoordinated planner, the high computational cost of the multi-robot rollouts reduces the benefits in terms of time required for exploration. To address this limitation, more cost-effective coordination strategies will be investigated.

References

[1] F. Amigoni, J. Banfi, and N. Basilico, "Multirobot exploration of communication-restricted environments: A survey," *IEEE Intelligent Systems*, 2017.

[2] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[3] L. Bartolomei, M. Karrer, and M. Chli, "Multi-robot Coordination with Agent-Server Architecture for Autonomous Navigation in Partially Unknown Environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[4] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics (T-RO)*, 2005.

[5] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "An online multi-robot slam system for 3d lidars," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[6] Z. Zhu, W. Jiang, L. Yang, and Z. Luo, "Indoor multi-robot cooperative mapping based on geometric features," *IEEE Access*, 2021.

[7] 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*, 2020.

[8] S. Dong, K. Xu, Q. Zhou, A. Tagliasacchi, S. Xin, M. Nießner, and B. Chen, "Multi-robot collaborative dense scene reconstruction," *ACM Transactions on Graphics*, 2019.

[9] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, 2019.

[10] K. M. Brian Lee, F. Kong, R. Cannizzaro, J. L. Palmer, D. Johnson, C. Yoo, and R. Fitch, "An Upper Confidence Bound for Simultaneous Exploration and Exploitation in Heterogeneous Multi-Robot Systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[11] R. H. Kabir and K. Lee, "Efficient, Decentralized, and Collaborative Multi-Robot Exploration using Optimal Transport Theory," in *American Control Conference (ACC)*, 2021.

[12] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, and K. Alexis, "CERBERUS in the DARPA Subterranean Challenge," *Science Robotics*, 2022.

[13] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo Tree Search methods," *IEEE Transactions on Computational Intelligence and AI in games*, 2012.

[14] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97*, 1997.

[15] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[16] D. Duberg and P. Jensfelt, "Ufoexplorer: Fast and scalable sampling-based exploration with a graph-based planning structure," *IEEE Robotics and Automation Letters*, 2022.

[17] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "FUEL: Fast UAV Exploration Using Incremental Frontier Structure and Hierarchical Planning," *IEEE Robotics and Automation Letters*, 2021.

[18] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active SLAM," in *IEEE International Conference on Robotics and Automation*, 2012.

[19] G. Georgakis, B. Bucher, A. Arapin, K. Schmeckpeper, N. Matni, and K. Daniilidis, "Uncertainty-driven planner for exploration and navigation," in *International Conference on Robotics and Automation (ICRA)*, 2022.

[20] Y. Kompis, L. Bartolomei, R. Mascaro, L. Teixeira, and M. Chli, "Informed Sampling Exploration Path Planner for 3D Reconstruction of Large Scenes," *IEEE Robotics and Automation Letters*, 2021.

[21] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[22] R. G. Colares and L. Chaimowicz, "The next frontier: Combining information gain and distance cost for decentralized multi-robot exploration," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016.

[23] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple UAVs," *The International Journal of Robotics Research*, 2020.

[24] C. W. Warren, "Global path planning using artificial potential fields," in *IEEE International Conference on Robotics and Automation*, 1989.

[25] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[26] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robotics and Autonomous Systems*, 2016.

[27] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Machine Learning: ECML 2006*. Springer, 2006.

[28] A. Ahmad, V. Walter, P. Petráček, M. Petrlík, T. Báča, D. Žaitlík, and M. Saska, "Autonomous Aerial Swarming in GNSS-denied Environments with High Obstacle Density," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.