Energy-constrained multi-robot exploration for autonomous map building

Karumanchi, Sambhu; Rokaha, Bhagawan; Schperberg, Alexander; Vinod, Abraham P.

TR2025-131 September 05, 2025

Abstract

We consider the problem of building the map of an unknown environment using multiple mobile robots that have physical limitations arising from dynamics and a limited onboard battery. We consider the setting where the unknown environment has a set of charging stations that the robots must discover and visit often to recharge their battery during the map building process. We propose an iterative approach to solve the resulting energy-constrained multi-robot exploration problem. Our approach uses a combination of frontier-based exploration, graph-based path planning, and multi-robot task assignment. We show that our algorithm admits a computation- ally inexpensive implementation that enables rapid replanning, and propose sufficient conditions for recursive feasibility and finite-time termination. We validate our approach in several Gazebo-based realistic simulations.

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2025

Energy-constrained multi-robot exploration for autonomous map building

Sambhu H. Karumanchi, Bhagawan Rokaha, Alexander Schperberg, and Abraham P. Vinod*

Abstract— **We consider the problem of building the map of an unknown environment using multiple mobile robots that have physical limitations arising from dynamics and a limited onboard battery. We consider the setting where the unknown environment has a set of charging stations that the robots must discover and visit often to recharge their battery during the map building process. We propose an iterative approach to solve the resulting energy-constrained multi-robot exploration problem. Our approach uses a combination of frontier-based exploration, graph-based path planning, and multi-robot task assignment. We show that our algorithm admits a computationally inexpensive implementation that enables rapid replanning, and propose sufficient conditions for recursive feasibility and finite-time termination. We validate our approach in several Gazebo-based realistic simulations.**

I. Introduction

Simultaneous localization and mapping (SLAM) is a critical component in several applications, including autonomous navigation [1], infrastructure monitoring [2], and disaster management [3]. Existing SLAM literature has considered the problem of map building using multiple autonomous robots [1], [4]. In this work, we tackle a specific challenge in autonomous map building — building the map using multiple mapping robots quickly, while respecting the physical constraints on the mapping robots, e.g., dynamics and limited battery. The physical limitations on the mapping robots may become particularly severe when using drones for autonomous map building [1]. We propose an iterative algorithm to solve the energy-constrained, multi-robot exploration problem. We show that *our approach has a computationally-inexpensive implementation enabling rapid replanning for efficient exploration, and enjoys guarantees of recursive feasibility and finite termination*.

Autonomous map building using SLAM has been an area of active research, where frontier-based exploration provides an effective strategy [4]–[9]. These exploration algorithms are iterative in nature, and repeatedly assign robots to visit frontiers and update the occupancy grid until no frontier remains [5] (see Figure 1 for terminology). For the purposes of coordinating multiple mapping robots using frontier-based exploration algorithms, approaches based on multi-robot task assignment [4], [6] and map segmentation [7] have been proposed. Recently, frontier-based exploration algorithms have also been extended to consider communication constraints in [8]. The authors in [9] impose the energy constraints on the mapping robots as soft constraints by minimizing the collective energy used by the team. However, these approaches may not be able to ensure that an energy-limited mapping robot will not get stranded during mapping, as these approaches do not explicitly account for the energy limitations of the robots as hard constraints.

A closely related problem to the considered energy-constrained multi-robot exploration problem is that of routing and persistent monitoring using multiple robots under constraints [10]–[14]. In these works, multiple robots with physical limitations coordinate their motion to persistently achieve certain goals. For example, [10] considers the problem of routing a single drone through multiple static charging stations under energy constraints using insights from the traveling salesman problem literature. Alternatively, [14] tackles the problem considered in this work using insights from vehicle routing literature. However, [14] requires solving a mixed-integer linear program (MILP) to design feasible exploration paths for the mapping robots. Since solving an MILP typically requires significant computational effort, such approaches may not be amenable to replanning and consequently may not adapt quickly to dynamic maps typically encountered in autonomous map building problems.

We use a combination of frontier-based exploration and multi-robot task assignment to solve the energy-constrained, multi-robot exploration problem of interest. Instead of directly assigning frontier centers to the mapping robot as done in [4], [6], [7], our approach considers two stages of path assignment for the robot in order to satisfy physical constraints on the mapping robot. The first path assignment identifies *energy-feasible paths* for each robot to a charging station, where energy-feasible paths are paths that satisfy the dynamics and energy constraints on the robots. The second path assignment identifies deviations from these paths such that the resulting path remains energy-feasible and allows the mapping team to visit the most informative parts of the unexplored environment. We cast both path assignment problems as linear assignment problems for which well-known polynomial-time algorithms exist [15]. When used in conjunction with graph-based planning algorithms like A* or Dijkstra [16], our approach admits a computationally-inexpensive implementation that allows for rapid replanning and adaptation to dynamic maps using only polynomial-time algorithms. Additionally, we provide sufficient conditions under which our approach has guarantees of recursive feasibility and finite-time termination.

*Corresponding author.
S. H. Karumanchi is with The University of Illinois Urbana-Champaign, Urbana, IL 61801, USA. Email: shk9@illinois.edu.
B. Rokaha is with Mitsubishi Electric Corp., Japan. Email: Rokaha.Bhagawan@df.MitsubishiElectric.co.jp.
A. Schperberg and A. P. Vinod are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA. Email: {schperberg@merl.com, abraham.p.vinod@ieee.org}.
This work was completed during Mr. Karumanchi's internship at MERL.

The main contributions of our work are 1) an iterative solution to the energy-constrained, multi-robot exploration problem, 2) a computationally-inexpensive implementation that allows for rapid replanning, and 3) sufficient conditions to guarantee finite-time termination and recursive feasibility.

II. Preliminaries & Problem statement

Notation: For a finite set $\mathcal{S}$, we denote its cardinality by $|\mathcal{S}|$. We use $\mathbb{N}_{[a,b]}$ to denote the set of natural numbers between (and including) $a, b \in \mathbb{N}$ with $a \leq b$.

A. Linear assignment problem

Given finite sets of workers $\mathcal{W}$ and tasks $\mathcal{T}$, and a cost function $C : \mathcal{W} \times \mathcal{T} \to \mathbb{R}$, the *linear assignment problem* solves the following optimization problem,

$$\underset{f \in \mathscr{A}(\mathcal{W},\mathcal{T})}{\text{minimize}} \sum_{w \in \mathcal{W}} C(w, f(w)). \qquad (1)$$

Here, $\mathscr{A}(\mathcal{W},\mathcal{T})$ is the set of *assignment functions* $f : \mathcal{W} \to \mathcal{T}$ that assigns a task in $\mathcal{T}$ to each worker in $\mathcal{W}$, where each task gets assigned at most one worker and each worker gets assigned at most one task [17]. The optimal solution of (1) minimizes the cumulative assignment cost. Note that (1) is a special class of integer linear program that admits polynomial time solutions and is mathematically identical to the *weighted bipartite matching problem* in graph theory [15]. Existing literature has used linear assignment problems like (1) in multi-agent motion planning and task assignment problems [4], [6], [12]. A popular approach to solve (1) is the Hungarian algorithm [15].

B. Problem statement

We consider the problem of deploying multiple energy-constrained mobile robots to autonomously build a map of an unknown environment. The designed motion plans must enable rapid, collaborative mapping of the environment, while respecting the physical constraints on the robots such as the dynamics and energy limitations. We assume that the environment has a finite set of *static* charging stations that are dispersed spatially. Throughout the mapping task, each robot must rendezvous with a station before running out of their battery. We assume that the locations of the stations are unknown *a priori*, and the robots "discover" these stations as they map the environment. Such settings are commonplace in post-disaster situations where robot teams must map an environment with spatially dispersed charging stations, and it is unclear *a priori* if feasible paths exist between a robot and a charging station. Additionally, the stations can recharge only a single robot at a time, and the team must coordinate to account for the charging capacity constraint at the stations.

C. Modeling

Robot team: We assume that the mapping team consists of N_T homogeneous mobile robots, each equipped with necessary sensors (e.g., LiDAR or camera) for mapping.

Motion constraints from occupancy grid: We consider an *occupancy grid*, which approximates the environment being mapped as a grid environment. Occupancy grids are a

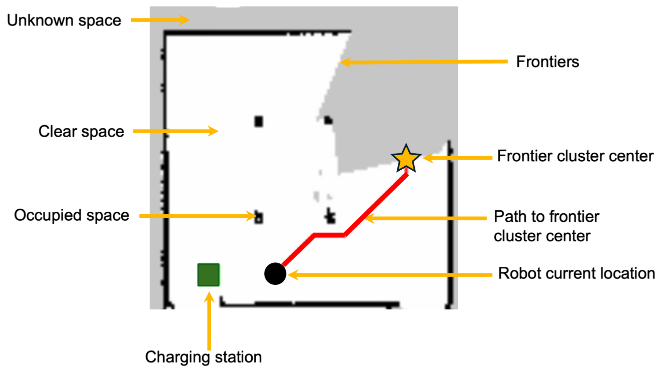

Fig. 1. Screenshot of a typical occupancy grid obtained from `gmapping` package annotated with the various terms used in the paper.

popular output of SLAM algorithms, where the algorithms update the probability of a grid cell being occupied based on the collected sensed data [5], [18]. Figure 1 annotates an instance of the occupancy grid with some key terminology used in the paper.

We assume that each robot in the mapping team can travel on a graph $\mathscr{G}_t = (\mathcal{V}, \mathcal{E}_t)$ defined over the occupancy grid at any time $t \in \mathbb{N}$ during the mapping process, with vertex set $\mathcal{V}$ and edge set $\mathcal{E}_t$. Each cell in the occupancy grid is a vertex in $\mathcal{V}$. At any time $t \in \mathbb{N}$ during the mapping process, $\mathcal{V} = \mathcal{C}_t \cup \mathcal{O}_t \cup \mathcal{U}_t$ may be partitioned into three sets — 1) $\mathcal{C}_t$ is the set of cells that is known to be *clear* of any obstacles, 2) $\mathcal{O}_t$ is the set of *occupied* cells, and 3) $\mathcal{U}_t$ is the set of cells whose occupancy status is *unknown*. The objective of the map building problem is to identify the partition $\mathcal{C}_\infty$, $\mathcal{O}_\infty$, and $\mathcal{U}_\infty$, where $\mathcal{C}_\infty$, $\mathcal{O}_\infty$, and $\mathcal{U}_\infty$ are the true sets of clear, occupied, and unknown cells. Note that $\mathcal{U}_\infty$ is not necessarily empty, but must have a boundary comprised solely of occupied cells.

The edge set $\mathcal{E}_t$ enforces the physical constraints on the robot arising from its dynamics. Specifically, every vertex of the graph $\mathscr{G}_t$ is connected to cells in $\mathcal{C}_t$ that are also its neighbors in any one of the cardinal or diagonal directions (like the king piece in chess).

Finally, we define $\mathcal{F}_t \subseteq \mathcal{C}_t$ as the set of *frontier* cells, vertices with at least one unknown cell as its neighbor [5].

Sensing set: We define $\mathbb{S}(v) \subset \mathcal{V}$ as the set of cells that a robot at vertex $v \in \mathcal{V}$ can sense and use to perform SLAM. For LiDAR sensors with a fixed sensing radius $r > 0$, $\mathbb{S}(v)$ is a ball of radius r centered at v.

Robot path: We define a path $\mathcal{P}$ as a walk over the graph $\mathscr{G}_t$ limited to $\mathcal{C}_t$, i.e., a sequence of vertices in $\mathcal{C}_t$ where each adjacent vertices in the sequence $\mathcal{P}$ are connected by an edge in $\mathcal{E}_t$. Given $\mathscr{G}_t$ and vertices $v_S, v_G \in \mathcal{C}_t$, we can find a path $\mathcal{P}$ that starts in v_S and ends in v_G using well-known graph-based path planning algorithms like A^* or Dijkstra [16].

Energy management: Let $\mathcal{S}$ denote the set of *static* stations, whose locations and count are *a priori* unknown. At any time t, the team is aware of the stations at $\mathcal{R}_t = \mathcal{S} \cap \mathcal{C}_t$, the set of station locations that have been declared clear at

Algorithm 1 Map building using energy-constrained robots

```
Input: Graph $\mathscr{G}_0 = (\mathcal{V}, \mathcal{E}_0)$ associated with the initial occupancy grid, initial station locations $\mathcal{R}_0$ with number of robots $N_T \leq |\mathcal{R}_0|$, weight parameter $\alpha \geq 0$, path length upper bound $B \in \mathbb{N}$, collision avoidance budget $\eta \in \mathbb{N}, \eta < B$, sensing set $\mathbb{S}$
Output: Partition of $\mathcal{V} = \mathcal{C}_\infty \cup \mathcal{O}_\infty \cup \mathcal{U}_\infty$
1:  Initialize $t \leftarrow 0$.
2:  Find the set of frontier cluster centers $\mathscr{F}_t$.
3:  while $\mathscr{F}_t$ is not empty do
4:      Optional: Perform any necessary redistribution of robots that are already at the station.    ▷ See Fig. 2.
5:      Obtain $M_i$ for each $i \in \mathbb{N}_{[1,N_T]}$, the number of moves (excluding collision avoidance) made by robot $i$ since last rendezvous with a station.
6:      Solve (1) with cost (2) to assign a unique station $s_i^* \in \mathcal{R}_t$ to each robot $i \in \mathbb{N}_{[1,N_T]}$.
7:      Solve (1) with cost (4) to assign a unique frontier cluster $c_t^* \in \mathscr{F}_t$ that robot $i \in \mathbb{N}_{[1,N_T]}$ must visit on its way to $s_i^*$, when possible.
8:      while robots are yet to reach their assigned goal do
9:          Execute paths and update map.
10:     end while
11:     Increment $t \leftarrow t + 1$.
12: end while
```

time t. Additionally, we assume that all robots start at stations in $\mathcal{R}_0 \subset \mathcal{S}$, where $|\mathcal{R}_0| \geq N_T$.

We model the energy limitations on the robot by restricting the length of the paths executed by the robots between recharging. Throughout the mapping process, we require the algorithm to only select *energy-feasible paths* for the robots.

Definition 1. (ENERGY-FEASIBLE PATHS) *For some user-specified path length bound $B \in \mathbb{N}$, a path $\mathcal{P}$ is an* energy-feasible path *if $\mathcal{P}$ starts and terminates at (possibly different) stations in $\mathcal{R}_t$ and $|\mathcal{P}| \leq B$.*

Note that the robot paths may also be a concatenation of energy-feasible paths. In Definition 1, the path length bound B models the physical limitation on the mapping robots due to their limited onboard battery. By definition, lower B implies a more severe energy limitation on the robots.

III. METHOD

We summarize our approach for addressing the energy-constrained, multi-robot exploration problem in Algorithm 1. It combines standard frontier-based exploration with energy-constrained task assignment and replanning. Similarly to existing frontier-based exploration algorithms [5], Algorithm 1 is iterative, and terminates only when the occupancy grid has no frontier cells left to visit. At each iteration, it identifies a *goal* for each robot, where the goal may be either a cell in $\mathcal{C}_t$ or a station. The robots continue mapping the environment while executing paths towards their respective goals, until a *replanning event* occurs. In this work, a replanning event occurs whenever a robot reaches its goal.

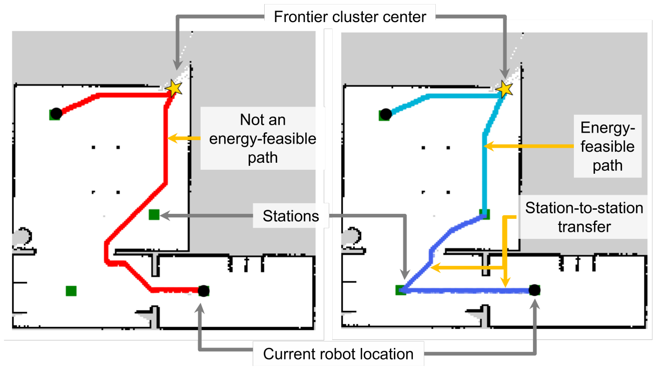

Fig. 2. Station-to-station transfers to improve robot reach despite energy constraints. (Left) A frontier cluster $c \in \mathscr{F}_t$ may be inaccessible for a robot at a far-away station. (Right) To mitigate this issue, robots first execute a series of station-to-station transfers to reach a closer station (Step 4 of Algorithm 1), and then visit c using an energy-feasible path.

Algorithm 1 identifies a goal for each robot using two stages of linear assignment — the first stage identifies a terminating station cell for each robot, and the second stage identifies a deviation for each robot so that the robot can explore the environment as it heads to a terminating station, when possible. The two-staged approach is motivated by the need for recursive feasibility, as shown later in Section IV.

In the first stage, we assign paths for the team from their current location at time t to the set of known stations $\mathcal{R}_t$. We set up and solve a linear assignment problem (1), where the assignment takes into account the robot dynamics, energy constraints, as well as the limitation of the station to serve only a single robot at a time. Here, the workers in the assignment problem are the robot team. The task set $\mathcal{T}_{\text{stage-1}}$ consists of (shortest) paths over the graph $\mathscr{G}_t$ computed for each pair of robot $i \in \mathbb{N}_{[1,N_T]}$ and station $s \in \mathcal{R}_t$, where each path starts at the current location of robot i and terminates at s and $|\mathcal{T}_{\text{stage-1}}| = N_T|\mathcal{R}_t|$. We define the cost function in (1) of the first stage,

$$C_{\text{stage-1}}(i, \mathcal{P}_{\text{stage-1}}) = \begin{cases} |\mathcal{P}_{\text{stage-1}}|, & |\mathcal{P}_{\text{stage-1}}| \leq B - M_i - \eta, \\ \infty, & \text{otherwise}, \end{cases} \tag{2}$$

where $\eta \in \mathbb{N}$, $\eta < B$ is a user-specified budget of the battery set aside for local inter-agent collision-avoidance and tracking errors. In (2), M_i for each $i \in \mathbb{N}_{[1,N_T]}$ denotes the number of cells robot i has visited since its last rendezvous with a station, excluding any additional cell visits for the purposes of (local) collision avoidance. By construction, the cost (2) eliminates paths that violate energy limitations. We solve (1) with $C_{\text{stage-1}}$ to obtain $\mathcal{P}^*_{\text{stage-1}}$, where each robot $i \in \mathbb{N}_{[1,N_T]}$ is assigned with a unique station $s_i^* \in \mathcal{R}_t$ and a dynamically-feasible $\mathcal{P}^*_{\text{stage-1}}(i)$ that starts at the current location of robot i and terminates at s_i^* with $|\mathcal{P}^*_{\text{stage-1}}(i)| \leq B - M_i - \eta$.

The second stage computes deviations for each path in $\mathcal{P}^*_{\text{stage-1}}$ to visit frontiers. These deviations are designed to facilitate rapid exploration of the unknown environment and autonomous map building. Motivated by existing frontier-based algorithms [5], we also use the so-called *information gain* metric to enforce a preference over frontiers, where the information gain $I_t : \mathcal{V} \to \mathbb{N}$ is,

$$I_t(v) = |\mathcal{U}_t \cap \mathbb{S}(v)|. \tag{3}$$

Here, $\mathbb{S}$ is the sensing set as defined in Section II-C. Informally, $I_t(v)$ for a vertex $v \in \mathcal{V}$ is the number of cells within $\mathbb{S}(v)$ whose status in the occupancy grid is unknown. To simplify second-stage planning, we use standard clustering techniques to group frontiers in $\mathcal{F}_t$ into N_C clusters based on their proximity, and use $\mathscr{F}_t \subset \mathcal{C}_t$ to denote the set of the resulting cluster centers.

In the second task, we set up another linear assignment problem (1) with the task set $\mathcal{T}_{\text{stage-2}}$ that consists of (shortest) paths over the graph $\mathscr{G}_t$ computed for each pair of robot $i \in \mathbb{N}_{[1,N_T]}$ and frontier cluster center $c \in \mathscr{F}_t$, where each path starts at the current location of robot i, visits c, and terminates at s_i^*. Consequently, $|\mathcal{T}_{\text{stage-2}}| = N_T N_C$. We use the following cost function in (1) of the second stage,

$$C_{\text{stage-2}}(i, \mathcal{P}_{\text{stage-2}}) = \begin{cases} -I_t(c) + \alpha|\mathcal{P}_{\text{stage-2}}|, & |\mathcal{P}_{\text{stage-2}}| \leq B - M_i - \eta, \\ \infty, & \text{otherwise,} \end{cases} \tag{4}$$

where $\alpha \geq 0$ is a parameter that trades off the information gain associated with a path with its path length. Lower α prefers more informative paths over shorter paths. Similarly to (2), (4) eliminates paths that violate energy limitations of the mapping robots. By solving (1) with $C_{\text{stage-2}}$, each robot i is assigned with a *unique* frontier cluster center $c_i^* \in \mathscr{F}_t$ and a dynamically feasible $\mathcal{P}^*_{\text{stage-2}}(i)$ that starts at the current location of robot i, visits c_i^*, and terminates at s_i^* with $|\mathcal{P}^*_{\text{stage-2}}(i)| \leq B - M_i - \eta$. When a robot i is not assigned any path in $\mathcal{P}_{\text{stage-2}}$ due to B, we assign the robot to execute the path in $\mathcal{P}^*_{\text{stage-1}}(i)$ computed at the first stage.

Algorithm 1 inherently disperses the robots away from each other in order to maximize collective information gain, which reduces the possibility of inter-agent collisions. However, one may enforce local multi-agent collision avoidance among the mapping robots by suitably modifying the low-level controller on the robots in Step 9. For example, we could use *optimal reciprocal collision avoidance* (ORCA) [19] to identify minimum norm corrections to the velocity commands. Alternatively, we can generate collision-free trajectories using other techniques like optimization-based safety filtering [20] or task swapping [12].

Step 4 of Algorithm 1 aims to mitigate the inherent greedy nature of the approach. Specifically, by relying on (1), Algorithm 1 uses a greedy strategy to select the frontier and stations to assign to the team of mapping robots. Consequently, the following undesirable scenario may occur during mapping — the set of frontier cluster centers is *not* empty and are far-away from the current stations charging the robots (see Figure 2). To avoid such a scenario, Step 4 redistributes the robots at far-away stations to stations closer to the frontier cluster centers. Such transfers are possible among any pair of stations connected by an energy-feasible path.

IV. Performance guarantees

We now show that Algorithm 1 enjoys finite-time termination and recursive feasibility guarantees. To simplify the analysis, we will assume that the localization error of the robots during the mapping process is low. Prior works [21], [22] show that revisiting previously explored areas reduces localization and mapping errors. Since robots visit charging stations often by design during exploration, the stations can serve as landmarks to correct the occupancy grid from SLAM, thereby improving localization and overall navigation accuracy.

Proposition 1. *(*Finite-time termination*) Let the following assumptions hold:*
1) (Reachability) For every cell $v \in \mathcal{C}_\infty$, there exists two stations $s_1, s_2 \in \mathcal{S}$ (not necessarily distinct) and a cell $c \in \mathcal{C}_\infty$ such that $v \in \mathbb{S}(c)$ and c is visited by an energy-feasible path connecting s_1, s_2.
2) (Station-to-station transfer) For any pair of stations in $\mathcal{S}$, there exists an energy-feasible path (or their concatenation) that can transfer a robot at one station to another.
3) (Representative clusters) The frontier set $\mathcal{F}_t$ at all iterations of the algorithm is adequately represented by the frontier cluster centers $\mathscr{F}_t$, i.e., $\mathcal{F}_t \subseteq \cup_{c\in\mathscr{F}_t}\mathbb{S}(c)$.
Then, Algorithm 1 terminates in at most $|\mathcal{U}_0 \setminus \mathcal{U}_\infty|$ iterations.

Proof. Observe that the reachability and station-to-station transfer assumptions together imply that the robot team can ensure that every $v \in \mathcal{C}_\infty$ may be covered by $\mathbb{S}(c)$ for some cell c. At each iteration t of Algorithm 1, the robot to visit a frontier cluster center c_i^*. Consequently, a visit to c_i^* removes at least one cell from the set of unknown cells $\mathcal{U}_t$, by the representative cluster assumption. Consequently, the set sequence $\mathcal{U}_t$ is a monotonically decreasing sequence of sets with $\mathcal{U}_{t+1} \subset \mathcal{U}_t$ for all $t \in \mathbb{N}$. Recall that the boundary of the unknown set $\mathcal{U}_t$ can either be frontier cells or occupied cells. Since Algorithm 1 continues until no frontier cells are left and $\mathcal{U}_0$ is a finite set, Algorithm 1 terminates in at most $|\mathcal{U}_0 \setminus \mathcal{U}_\infty|$ iterations. □

The reachability assumption in Proposition 1 is necessary for the team to complete the map, i.e., identify the partition of $\mathcal{V}$. The station-to-station transfer assumption may be restrictive in scenarios where the spatial distribution of the stations is skewed to certain parts of the environment over others. Existing literature in facility location problems [23] can help in identifying station locations that satisfy the station-to-station transfer assumption in Proposition 1. The representative clustering requirement may require a large N_C at certain iterations of Algorithm 1 that can increase the computation time. Empirically, we found that Algorithm 1 remained recursively feasible even when the clustering assumption was relaxed. Specifically, Section V uses a constant $N_C = 30$ for all t.

Proposition 2. (RECURSIVE FEASIBILITY) *Assume that the local collision-avoidance budget η is never violated, i.e., the local multi-agent collision-avoidance forces the robot to visit at most η additional cells (apart from its assigned path) between any consecutive visits to stations. Then, Algorithm 1 is recursively feasible, i.e., there always exists a feasible assignment of paths for the robot team at each iteration t.*

Proof. We provide the proof by induction. We first prove that the base case holds, i.e., a feasible assignment exists at $t = 0$. The proof follows from the observation that, since all robots can stay at their respective stations, the feasible space of the assignment problem (1) is non-empty.

Assume, for induction, that the robots have been assigned energy-feasible paths at iteration $t \in \mathbb{N}, t > 0$. We need to show that there are energy-feasible paths to be assigned to every robot at iteration $t + 1$.

At time t, the robots at the station $\mathcal{R}_t$ have at least one energy-feasible path, i.e., the robots stay at the station. Consequently, to prove recursive feasibility at $t+1$, it suffices to focus on the robots that are not at the station ($M > 0$).

Consider robot $i \in \mathbb{N}_{[1,N_T]}$. By design, the energy-feasible paths $\mathcal{P}^*_{\text{stage-2}}$ assigned to the robots at iteration t are such that each path $\mathcal{P}^*_{\text{stage-2}}(i)$ starts at the current location $v_i(t)$ of the robot $i \in \mathbb{N}_{[1,N_T]}$, visits $v_i(t+1)$ next, terminates at a station $s_i^* \in \mathcal{R}_t$, and $|\mathcal{P}^*_{\text{stage-2}}(i)| \leq B - M_i - \eta$, and M_i excludes visits to cells for the purposes of collision avoidance. We have also assumed that the collision avoidance leaves the robot at $v_i(t+1)$ after the necessary avoidance maneuver and at most η cells are visited between any two consecutive visits to stations. Then, at iteration $t+1$, the path $\mathcal{P}^*_{\text{stage-2}}(i) \backslash \{v_i(t)\}$ is a feasible trajectory that starts at $v_i(t+1)$ and terminates at s_i^* with path length $|\mathcal{P}^*_{\text{stage-2}}(i)| \leq B - M_i - \eta$, where M_i is incremented to $M_i + 1$ in Step 5. Thus, a feasible path assignment exists for every robot i that is not at the station at $t + 1$, which completes the proof. $\square$

Excluding any computation necessary for collision avoidance, Algorithm 1 requires only polynomial-time computation at each iteration. Specifically, each iteration of Algorithm 1 computes $N_T(|\mathscr{F}_t| + |\mathcal{R}_t|)$ paths, and solves two linear assignment problems. Both of these operations require polynomial time, and have efficient off-the-shelf implementations [24], [25].

V. NUMERICAL VALIDATION

We validate our approach in several Gazebo-based realistic simulations. We show that our approach scales to moderately-sized mapping robot teams and can map environments of moderate to large sizes in a reasonable time. We also study the effect of the path length bound on the mapping quality. As expected, increasing the battery capacity typically improved the performance of the approach. Finally, we compared our approach to a vanilla frontier-based exploration algorithm, based on [5], to show that our approach generates energy-feasible paths for robots and achieves coverage despite physical limitations, with only a moderate increase in computational cost compared to [5].

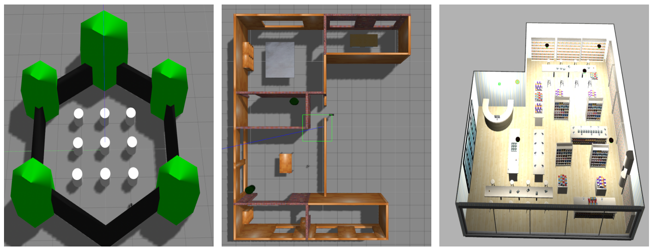
Fig. 3. ROS2 Gazebo simulation environments used in Section V. (Left) Turtlebot world, (Middle) House, (Right) Bookstore.

We performed all simulations on a standard computer (Intel $i9$ processor, 125 GB RAM) running Ubuntu 22.04.

A. Set up for the numerical validation

For our numerical experiments, we used three indoor template worlds built on the ROS2 Gazebo simulator — the TurtleBot world ($\approx$ 50 m^2), House ($\approx$ 150 m^2), and bookstore world ($\approx$ 250 m^2). See Figure 3.

We considered robot teams with $N_T \in \{2, 4, 6\}$ Turtlebots [26], with each mapping robot equipped with a LiDAR scanner with a range of 6 m. We restricted the robot's maximum linear and angular speeds to 0.15 m/s and $\frac{\pi}{4}$ rad/s, respectively. We used a ROS2 *gmapping* package [27] for SLAM and occupancy grid generation, `m-explore-ros2` [28] for merging maps from the mapping robots, and `pyastar2d` [29] and `NAV2` [30] for planning and control of the mapping robots. In order to facilitate map merging, we assumed that the global positions of the initial stations $\mathcal{R}_0$ were known. We set $\alpha = 0$ in (4) to focus on most informative path plans. We considered various path length bounds $B \in \{8, 16, 32\}$ (in m). We computed path length $|\cdot|$ by measuring the actual distance traveled by the robot, and consequently, we measured B in meters. While we did not implement any collision avoidance controller, we chose a collision avoidance budget $\eta = 2$ (in m) to account for tracking errors exhibited by the robots. We compute the frontier cluster centers $\mathscr{F}_t$ similarly to [31]. Alternative clustering methods like the pixel-connectivity method in [5] or RRT-based exploring trees [32] may be considered as well.

All tables and plots report the median and the maximum deviation from the median evaluated over three runs.

B. Different environments and team sizes

We evaluate the performance of Algorithm 1 by varying N_T and the environment. We set B and $\mathcal{S}$ appropriately to satisfy Proposition 1. For the House environment with $N_T = 2$, all three runs triggered the redistribution step (Step 4 of Algorithm 1) to complete the mapping. No other problem instance triggered Step 4 of Algorithm 1.

Table I reports various performance metrics for different environments and team sizes. As expected, Algorithm 1 did not violate the energy constraint in any of the 15 runs, and the travel distance between charging is below the corresponding path length bound B in all cases. Additionally, Algorithm 1 successfully explored the different environments with high

TABLE I

PERFORMANCE METRICS OF ALGORITHM 1 FOR DIFFERENT ENVIRONMENTS AND TEAM SIZES.

Environment type	**Turtlebot**	**House**			**Bookstore**
Number of robots N_T	2	2	4	6	6
Path length bound B [m]	8	8	8	8	16
Maximum distance between charging [m]	4.2 ± 2.7	6.1 ± 0.5	5.6 ± 2.1	5.3 ± 0.6	13.4 ± 2.5
Total distance traveled by the team [m]	8.2 ± 1.4	73.1 ± 7.5	31.9 ± 8.6	27.9 ± 10.5	81.6 ± 28.9
Number of charging instances	1.0 ± 1.0	5.0 ± 3.0	1.5 ± 1.5	1.0 ± 1.0	2.0 ± 1.0
Exploration percentage [%]	99.8 ± 0.1	99.2 ± 0.7	100.0 ± 0.0	100 ± 0.0	99.9 ± 0.1
Overall mapping time [s]	52.1 ± 13.8	530.1 ± 62.1	258.4 ± 105.3	156.4 ± 53.7	361.2 ± 203.1
Computation time/Iter [s]	0.1 ± 0.0	0.2 ± 0.0	0.2 ± 0.2	0.9 ± 0.7	2.0 ± 0.5

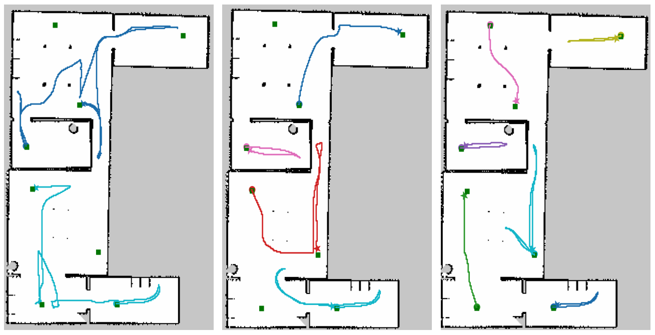

Fig. 4. Paths traveled by robot teams with $N_T \in \{2, 4, 6\}$ while mapping the House environment using Algorithm 1. Here, green squares show the charging station locations, white region shows the clear spaces, black cells show the occupied cells, and grey region shows the unknown cells.

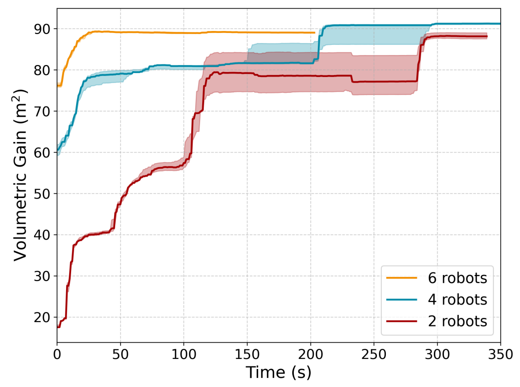

Fig. 5. Variation in the volumetric gain (area of the environment declared as clear) for Algorithm 1 over the map building process in the House environment for varying team sizes.

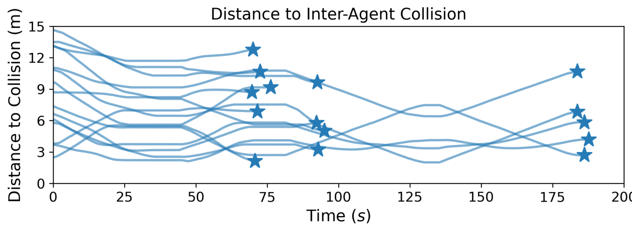

Fig. 6. Variation of DistToCollision (5) over the map building process for House environment with $N_T = 6$. For each of the resulting 15 robot pairs, the distance to collision is positive, which implies no collision.

exploration percentage in reasonable mapping times. Finally, Algorithm 1 exhibited a moderate increase in computation time for the House environment as the team size increased from $N_T = 2$ to $N_T = 6$. Notably, the computation time at each iteration of Algorithm 1 across all simulations was below 3 seconds, despite using a (non-optimized) Python implementation of Algorithm 1. Figure 4 shows the paths traversed by the mapping robots for the House environment.

Figure 5 shows the trend in volumetric gain with varying team size for the House environment. Recall that volumetric gain refers to the area of clear space, which is proportional to $|\mathcal{C}_\infty|$, in the environment. For the House environment, the volumetric gain is about 90 m^2. Thus, Figure 5 shows the effectiveness of Algorithm 1 in mapping the environment using multiple robots despite the energy constraints.

Figure 6 shows the distance to collision between every pair of the mapping robots for $N_T = 6$ while mapping the House environment. Here, the distance to collision between any pair $i, j \in \mathbb{N}_{[1,N_T]}$, $i \neq j$ positioned at $p_i(t)$ and $p_j(t)$ at time t is defined as (with $r = 0.3$ m as the Turtlebot radius),

$$\mathrm{DistToCollision}(p_i(t), p_j(t)) = \|p_i(t) - p_j(t)\| - 2r. \quad (5)$$

From (5), two mapping robots collide when their corresponding distance to collision is negative. Figure 6 shows that all pairwise distances are positive, implying no collisions.

C. *Effect of path length bound B and station locations $\mathcal{S}$*

We analyze the effect of $B, \mathcal{S}$ on overall performance.

Figure 7 illustrates the variations in total time, total distance traveled, and the number of charging instances for path length bounds $B \in \{8, 16, 32\}$ and $N_T \in \{2, 4, 6\}$ for the House environment. As anticipated, increasing path length bound B (more onboard battery capacity) reduces the number of recharges, leading to a decrease in the total distance traveled and the total time required for exploration.

We also considered three distinct station configurations $\mathcal{S}$ that satisfy the reachability condition of Proposition 1 for the House environment, and repeated the mapping experiment with Algorithm 1 for $N_T \in \{2, 4, 6\}$. Table II shows that the station locations do not appear to have an impact on the efficiency of the exploration or the energy constraints. Additionally, Table II reinforces the observations made in Table I on scalability and performance of Algorithm 1.

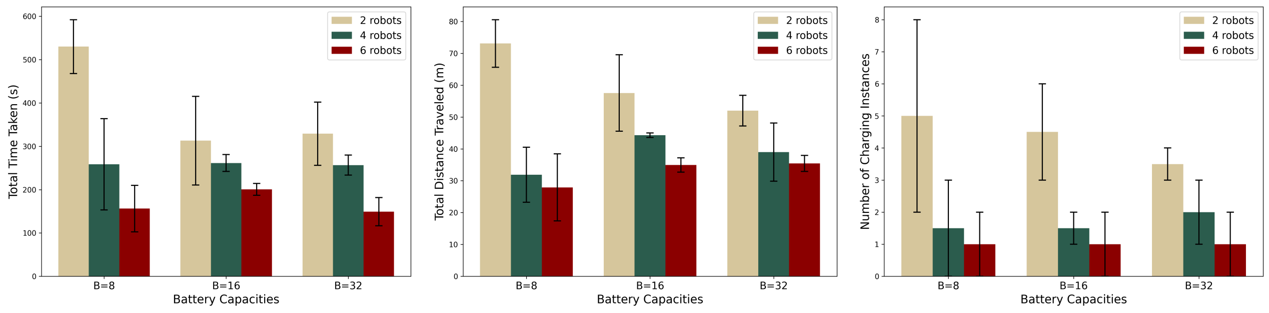

Fig. 7. Variation in overall mapping time, total distance traveled by the team, and the number of charging instances required for different path length bounds B (battery capacities) for the House environment.

TABLE II
PERFORMANCE METRICS FOR ALGORITHM 1 FOR DIFFERENT TEAM SIZES N_T AND STATION CONFIGURATIONS IN THE HOUSE ENVIRONMENT.

(N_T, B)	Station Configurations	Maximum distance between charging [m]	Total distance traveled by the team [m]	Number of charging instances	Exploration percentage [%]	Overall mapping time [s]	Computation time/Iter [s]
(2,16)	1	9.6 ± 1.8	57.6 ± 12.0	4.5 ± 1.5	98.1 ± 1.5	313.0 ± 102.4	0.2 ± 0.0
	2	11.5 ± 1.4	46.6 ± 13.9	3.5 ± 2.5	100.0 ± 0.0	308.1 ± 132.8	0.2 ± 0.0
	3	9.3 ± 0.5	34.7 ± 2.4	2.5 ± 0.5	99.9 ± 0.1	152.9 ± 34.0	0.2 ± 0.0
(4,16)	1	10.5 ± 0.8	44.3 ± 0.7	1.5 ± 0.5	99.7 ± 0.2	261.4 ± 19.6	0.6 ± 0.0
	2	9.9 ± 0.0	42.7 ± 7.1	2.0 ± 1.0	99.8 ± 0.2	303.8 ± 72.3	0.5 ± 0.1
	3	8.5 ± 0.1	37.5 ± 1.4	1.0 ± 1.0	100.0 ± 0.0	161.7 ± 10.2	0.5 ± 0.1
(6,16)	1	9.1 ± 3.9	35.0 ± 2.2	1.0 ± 1.0	99.8 ± 0.2	200.5 ± 13.6	1.2 ± 0.1
	2	6.5 ± 6.5	25.5 ± 4.8	1.0 ± 1.0	99.6 ± 0.3	225.7 ± 82.3	1.4 ± 0.5
	3	6.1 ± 2.4	28.2 ± 4.6	1.0 ± 1.0	99.7 ± 0.3	169.2 ± 62.3	1.1 ± 0.6

D. Comparison with greedy algorithm based on [5]

We evaluate the performance of our algorithm against the greedy algorithm, based on [5]. The greedy algorithm replaces Steps 6 and 7 with a round-robin assignment of frontier cluster center from the set $\mathscr{F}_t$ that is closest to the robot, while ensuring that no robot is assigned the same frontier cluster center. Since the greedy approach does not impose any energy constraints, it fails to complete the exploration of the environment in all 9 runs for $B = 16$, as expected.

Figure 8 shows that Algorithm 1 generates energy-feasible paths that address the energy-constrained, multi-robot exploration problem while only incurring a modest increase in computation time as compared to the greedy algorithm. Algorithm 1 computes $N_T(|\mathscr{F}_t| + |\mathcal{R}_t|)$ paths and solves two linear assignment problems, while the greedy algorithm computes only $N_T|\mathscr{F}_t|$ paths and performs a round-robin assignment via N_T linear searches for the shortest path.

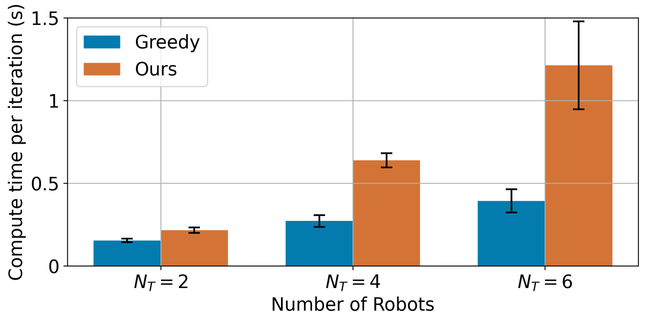

Fig. 8. Comparison of computation time per-iteration for Algorithm 1 with the greedy approach inspired from [5] for $N_T \in \{2, 4, 6\}$ in the House environment.

VI. CONCLUSION AND FUTURE WORK

In this paper, we addressed the energy-constrained multi-robot exploration problem with a focus on building maps of unknown environments. Our approach uses a combination of frontier-based exploration and multi-robot task assignment to generate energy-feasible paths for the mapping robots. We also proposed a computationally-inexpensive implementation of the algorithm that enables the mapping robot to account for changes in the evolving occupancy grid during the map building process via replanning. Finally, we characterized sufficient conditions under which Algorithm 1 is guaranteed to have recursive feasibility and terminate in finite iterations.

Our approach has some shortcomings, which we hope to address in future work. First, our approach uses robot redistribution to overcome the undesirable scenario of robots terminating at stations far away from the remaining frontier clusters (see Step 4 in Algorithm 1). Consequently, the total distance traveled by the mapping robots may be much higher than the optimal distance to be traveled. Second, our approach can only accommodate *static* charging stations. Third, we focussed on generating shortest paths at each stage of Algorithm 1, which may result in suboptimal exploration when B is moderately large (a setting ignored in this paper). All these limitations arise from our focus on reducing the computational burden of Algorithm 1. Our future work will also focus on validating the proposed approach using experiments with physical robots.

References

[1] Arias-Perez, et. al., "Exploring unstructured environments using minimal sensing on cooperative nano-drones," *Robotics Automation Letters*, 2024.

[2] L. Iocchi, L. Marchetti, and D. Nardi, "Multi-robot patrolling with coordinated behaviours in realistic environments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2796–2801, IEEE, 2011.

[3] Y. Liu and G. Nejat, "Multirobot cooperative learning for semiautonomous control in urban search and rescue applications," *Journal of Field Robotics*, vol. 33, no. 4, pp. 512–536, 2016.

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

[5] B. Yamauchi, "Decentralized coordination for multirobot exploration," *Robotics and Autonomous Systems*, vol. 29, no. 2-3, pp. 111–118, 1999.

[6] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, "A practical, decision-theoretic approach to multi-robot mapping and exploration," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3232–3238, 2003.

[7] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1160–1165, 2008.

[8] M. Tellaroli, M. Luperto, M. Antonazzi, and N. Basilico, "Frontier-based exploration for multi-robot rendezvous in communication-restricted unknown environments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024.

[9] A. Benkrid, A. Benallegue, and N. Achour, "Multi-robot coordination for energy-efficient exploration," *Journal of Control, Automation and Electrical Systems*, vol. 30, no. 6, pp. 911–920, 2019.

[10] K. Yu, A. K. Budhiraja, and P. Tokekar, "Algorithms for routing of unmanned aerial vehicles with mobile recharging stations," in *Proceedings of IEEE International Conference on Robotics & Automation*, pp. 5720–5725, 2018.

[11] J.-R. Fan, M. Thummalapeta, and Y.-C. Liu, "Energy-constrained persistent coverage control on multi-quadrotor systems," in *Proceedings of International Conference on Control, Automation and Systems*, pp. 1035–1040, 2023.

[12] X. Lin, S. Nayak, S. Di Cairano, and A. P. Vinod, "Data-driven spatial classification using multi-arm bandits for monitoring with energy-constrained mobile robots," *arXiv preprint arXiv:2501.08222*, 2025.

[13] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," *IEEE Transactions on Robotics*, 2015.

[14] D. Mitchell and N. Michael, "Persistent multi-robot mapping in an uncertain environment," in *Proceedings of IEEE International Conference on Robotics & Automation*, pp. 4552–4558, 2019.

[15] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment problems: revised reprint*. SIAM, 2012.

[16] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[17] D. W. Pentico, "Assignment problems: A golden anniversary survey," *European Journal of Operational Research,*, vol. 176, no. 2, pp. 774–793, 2007.

[18] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021.

[19] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems: The 10th international symposium*, pp. 203–216, Springer, 2013.

[20] A. P. Vinod, S. Safaoui, T. H. Summers, N. Yoshikawa, and S. Di Cairano, "Decentralized, safe, multiagent motion planning for drones under uncertainty via filtered reinforcement learning," *IEEE Transactions on Control System Technology*, 2024.

[21] C. Stachniss, D. Hahnel, and W. Burgard, "Exploration with active loop-closing for FastSLAM," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1505–1510, 2004.

[22] Y. Zhao, J. S. Smith, S. H. Karumanchi, and P. A. Vela, "Closed-loop benchmarking of stereo visual-inertial slam systems: Understanding the impact of drift and latency on tracking accuracy," in *Proceedings of IEEE International Conference on Robotics & Automation*, pp. 1105–1112, 2020.

[23] L. Pronzato, "Minimax and maximin space-filling designs: some properties and methods for construction," *Journal de la Société Française de Statistique*, vol. 158, no. 1, pp. 7–36, 2017.

[24] P. Virtanen and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[25] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using networkx," tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

[26] "Turtlebot3." `https://github.com/ROBOTIS-GIT/turtlebot3`.

[27] "gmapping." `https://wiki.ros.org/gmapping`.

[28] "m-explore-ros2." `https://github.com/robo-friends/m-explore-ros2`.

[29] "pyastar." `https://pypi.org/project/pyastar2d/`.

[30] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2718–2725, IEEE, 2020.

[31] Uslu, Erkan, e.t. al., "Implementation of frontier-based exploration algorithm for an autonomous robot," in *Proceedings of International Symposium on Innovations in Intelligent SysTems and Applications*, pp. 1–7, 2015.

[32] S. Mukhopadhyay, H. Umari, and K. Koirala, "Multi-robot map exploration based on multiple rapidly-exploring randomized trees," *SN Computer Science*, vol. 5, no. 1, p. 31, 2023.