Multi-Robot Exploration for the CADRE Mission

Sharan Nayak[1*], Grace Lim[2*], Federico Rossi[2], Michael Otte[1], Jean-Pierre de la Croix[2]

[1]Department of Aerospace Engineering, University of Maryland, College Park, College Park, 20903, Maryland, USA.
[2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, 91011, California, USA.
[*]These authors contributed equally.

*Corresponding author(s). E-mail(s): snayak18@terpmail.umd.edu; grace.lim@jpl.nasa.gov;
Contributing authors: federico.rossi@jpl.nasa.gov; otte@umd.edu; jean-pierre.de.la.croix@jpl.nasa.gov;

Abstract

We present the design, implementation and testing of a multi-robot exploration algorithm for NASA's upcoming Cooperative Autonomous Distributed Robotic Exploration (CADRE) lunar technology demonstration mission. The CADRE mission, among its various objectives, entails utilizing a trio of autonomous mobile robots to collaboratively explore and construct a map of a designated area of the lunar surface. Given the mission's inherent constraints, including limited mission duration, constrained power resources, and restricted communication capabilities, we formulate an exploration algorithm to improve exploration efficiency, facilitate equitable workload distribution among individual agents, and minimize inter-robot communication. To achieve these requirements, we employ a semi-centralized exploration algorithm that partitions the unexplored area, regardless of its shape and size, into a series of non-overlapping partitions, assigning each partition to a specific robot for exploration. Each robot autonomously explores its designated region without intervention from other robots. We explore the design space of the proposed algorithm and evaluate its performance under diverse conditions in simulations. Finally, we validate the algorithm's functionality through two sets of hardware experiments: the first utilizes prototype rovers using a ROS-based navigation software stack for feasibility testing, while the second employs high-fidelity development model rovers running CADRE's custom flight-software stack for flight-like performance validation. Both sets of experiments are conducted in the Jet Propulsion Laboratory's (JPL) lunar-simulated rover testing facilities, demonstrating the algorithm's robustness and readiness for lunar deployment.

Keywords: Multi-robot systems, Exploration, Planetary surface robotics, Collaborative robotics

1 Introduction

In robotics, exploration refers to the problem of using autonomous mobile robots (agents) to explore an unknown environment and build a data map. This problem has been extensively studied in the context of both single [1] and multi-robot teams [2]. Multi-robot teams hold a

distinct advantage over their single robot counterparts, as they can collaboratively complete exploration tasks more efficiently. Moreover, they exhibit greater fault tolerance, a crucial attribute in challenging environments.

Multi-robot exploration has found valuable applications across diverse domains, including planetary exploration [3] and search and rescue operations during disaster missions [4]. Notably, NASA has recently funded the development of CADRE (Cooperative Autonomous Distributed Robotic Exploration) [5, 6], a pioneering multi-robot technology demonstration lunar mission. CADRE's primary objective is to deploy a team of three autonomous mobile robots that will work collaboratively to explore and gather data from the Reiner Gamma Lunar region [7]. The preliminary mission objective is to explore and construct a surface occupancy map, which will serve as a foundational resource for subsequent mission operations. The success of CADRE holds great promise for inspiring future multi-robot space expeditions that will look for potential signs of life and analyze the geology and mineral composition of Solar System bodies.

In the CADRE mission, agents must collaborate to explore an assigned area on the lunar surface. The mission's constraints, which include limited mission duration, scarce power resources, and constrained communication capabilities, dictate the exploration algorithm we employ. These constraints are driven by several factors, including the need to actively manage the rovers' thermal state in the face of extreme lunar temperatures, the need for slow, periodic rover recharging using solar panels, the necessity to share limited communication bandwidth with other rover applications, and concerns about electromagnetic interference between communication, driving, and payload activities. To effectively address these constraints, we propose a multi-robot exploration algorithm that employs a "divide and conquer" strategy. This strategy involves selecting a "leader" agent that partitions the unexplored area into sub-regions, with each sub-region then assigned to a robot for autonomous exploration (Fig. 1). Previous studies [8–10] have demonstrated the effectiveness of this map-based partitioning approach in expediting exploration missions.

This approach offers a number of attractive properties:

1. It facilitates equitable distribution of computational resources among agents, resulting in faster exploration.
2. It reduces inter-agent coordination demands, especially during driving, as communication solely occurs between the leader agent and its counterparts and no communication is required while the robots drive.
3. It reduces power requirements and the necessity to manage inter-robot collisions, given the absence of overlap between regions.

The main contribution of this work lies in the design, implementation, and testing of a map-partition-based multi-robot exploration algorithm tailored for the CADRE spaceflight mission. While the concept of a map-partition-based algorithm [9, 10] is not novel, our proposed algorithm distinguishes itself from previous works by introducing a *semi-centralized* approach. In this approach, a central agent (dubbed a "leader") performs map division and agent assignment, and all agents subsequently autonomously explore their assigned regions in a decentralized manner without intervention from the leader. In contrast, earlier approaches that employ central entities for map partitioning also retain centralized control over agent movements, as discussed in Section 2.2. This key distinction sets our algorithm apart and makes it well-suited for achieving the objectives of the CADRE mission.

We note that the decision to use three robots in this study is informed by the operational requirements of the CADRE space mission. For transportation to the lunar surface, robots are planned to be attached to a lander [11] with stringent physical and weight limitations, which restricts the number of deployable robots. Additionally, the mission operates under significant budget constraints, further limiting the feasibility of including more than three robots. However, we emphasize that our proposed algorithm is designed to scale efficiently, supporting larger numbers of robots if mission resources allow.

We also emphasize that a key contribution of this work is the testing, verification, and validation of the proposed algorithms to the standards required by a flight project. While a number of exploration algorithms have been proposed in the academic literature, infusion of such algorithms

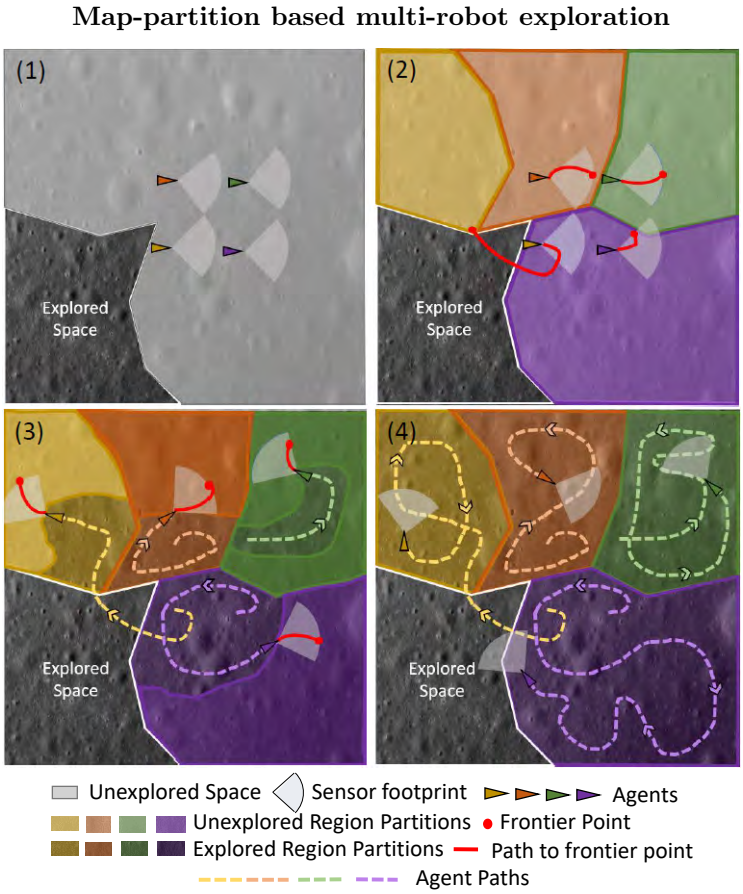

Fig. 1: (1) shows a map with initial location of agents, explored and unexplored space. (2) Unexplored space is divided into partitions and agents start moving to their assigned partitions. (3) Agents autonomously explore their assigned partition. (4) Agents complete exploring their partitions.

in a spaceflight mission introduces strict requirements in terms of computational complexity; inter-agent communication; inclusion of telemetry for observability, under tight constraints on overall downlink bandwidth; and handling of interactions with the rest of the autonomy stack.

The rest of the article is organized as follows. Section 2 presents related work on multi-robot exploration and places this paper's contribution in the context of the state of the art. Section 3 provides a formal problem definition. Section 4 discusses the design of the proposed multi-robot exploration approach. Section 5 presents the setup and results of the simulation experiments. Section 6 provides the setup, results, and lessons learned from the hardware experiments. Finally, Section 7 concludes by summarizing the contributions and main results, and lays out directions for future work.

2 Related work

In this section, we review the literature on multi-robot exploration, focusing on key algorithms and methodologies. We begin with an overview of various multi-robot exploration strategies, categorizing them into frontier-based, information theory-based, and potential field-based approaches. We then discuss frontier-based exploration architectures, differentiating between centralized and decentralized schemes for assigning frontier points to agents. Next, we highlight map-partition-based methods and their advantages in

Testing the proposed multi-robot exploration algorithm at the Moon yard at JPL

Fig. 2: Testing the proposed algorithm at the Moon yard at JPL. The yard is cluttered with rocks, craters and uneven surfaces to resemble lunar surface. The violet and green rectangles represent the map partitions output by our algorithm corresponding to the two robots.

optimizing exploration efficiency compared to previous techniques. We also address the role of communication in multi-robot exploration, examining how different strategies affect coordination and overall exploration effectiveness. We conclude with a discussion on multi-robot coordination in lunar environments.

2.1 Multi-robot exploration algorithms

In the existing literature, various multi-robot exploration algorithms have been proposed, with notable approaches falling into different categories, including frontier-based [2, 9], information theory-based [12, 13], and potential field-based [4, 14]. Frontier-based algorithms guide robots towards "frontiers", defined as points on the boundary between explored and unexplored regions. Information theory-based methods direct robots to areas that either reduce the overall entropy of the map or maximize mutual information [15]. Meanwhile, potential field methods employ a potential field function [16] to guide robots towards unexplored areas while keeping them away from already explored regions. Among these approaches, frontier-based exploration stands out as the most commonly adopted due to its effectiveness and simplicity. Multiple frontier-based methods have been proposed [17] to minimize time [8], avoid redundant coverage [18, 19], and reduce communication [20]. While frontier-based and information-theoretic approaches have been presented as distinct methodologies, they are not mutually exclusive; specifically, frontier points may be assessed using information-theoretic metrics [21–23], a concept that will be elaborated upon in the subsequent discussion.

In frontier-based approaches, a critical operation in each iteration involves the determination of frontier points. The initial frontier-detection method proposed by Yamauchi [1], suffered from inefficiencies in that the entire map had to be processed. In pursuit of enhanced computational efficiency, various faster frontier detection methods have been proposed including Wavefront Frontier Detector (WFD) [24], Fast Frontier Detector (FFD) [24] and Expanding Wavefront Frontier Detector (EWFD) [25]. These approaches internally use Breadth First Search (BFS) [26] to iterate through only the explored cells to find the boundary between explored and unexplored cells. An alternate category of frontier-detection approaches [27, 28] rely on employing Rapidly Exploring Random Tee (RRT) algorithm [29]. The advantage of RRT is that its growth is heavily biased towards the unknown regions of the map

which inherently aids in exploration. In our implementation, we opt for an exploration approach based on WFD due to its simplicity and efficiency.

When frontier-detection methods are run at every iteration, they yield a set of candidate frontier points. From the pool of these candidate frontier-points, a best frontier point is determined. A prevailing methodology for selecting the best frontier point is the utilization of utility or cost functions, as highlighted by Daniel da Silva Lubanco et al. [30]. These functions employ various criteria such path cost [21], information gain [21] , sensor data acquisition cost [21], geometry of the environment [31] and likelihood of communication success. In our implementation, designed to meet mission constraints, we employ a cost function that combines path cost, information gain, and rotation cost enabling efficient exploration. We acknowledge that this cost function is not novel and a modified version (only the first two terms) has been used in previous work [32]. Further insights into this cost function are presented in Section 4.

2.2 Frontier-based exploration architecture approaches

Frontier-based approaches can be categorized into centralized [19, 33] and decentralized schemes [34], depending on the entity responsible for generating frontier points. In centralized schemes, a central entity, such as a base station or a leader agent, is tasked with assigning frontier points to individual agents. For instance, Banfi et al. [33] employ Integer Linear Programming (ILP) on a central base station to allocate frontier points to each agent. This allocation ensures efficient exploration while adhering to recurrent connectivity constraints. In contrast, decentralized schemes involve individual agents autonomously selecting their next frontier points. For example, Bautin et al. [34] has individual agents determine their next frontier point by considering factors such as the proximity of other agents to frontiers and the number of agents closer to a particular frontier. In market-driven multi-robot exploration approaches [35, 36], each agent generates a list of candidate frontier points in a decentralized fashion and requests other agents to bid for the frontier; the highest bidder gets awarded the frontier. Another noteworthy category of approaches, particularly advantageous in communication-restricted environments, is role-based frontier-based exploration [37]. In role-based frontier exploration, each agent assumes one of two roles: explorer or relay. The explorers are responsible for exploring the environment using frontier-based exploration and sharing knowledge such as environment information (e.g. obstacles, landmarks) and status update (e.g. battery, location) with a relay when they come within communication range. The relays, in turn, transmit this data to a central command center. In our proposed scheme, we introduce a map-partition-based approach that integrates elements from both centralized and decentralized approaches.

2.3 Map-partition based frontier-based exploration

A key aspect of multi-robot frontier-based exploration is assigning frontiers to different robots to reduce the overall mission time for exploration [9]. Previous approaches [8–10] have suggested achieving this objective by partitioning the environment into distinct regions and assigning frontier points within these regions to individual robots for exploration. For instance, Wurm et al. [9], Lopez Perez et al. [10], and Wu et al. [38] have utilized Voronoi partitioning to segment environments like rooms and corridors, while Solanas and Garcia [8] employed K-means-based partitioning for environment segmentation. Subsequently, these methods utilized the robot's location to determine the robot-region assignments.

In alignment with these techniques, we propose the utilization of a map-partition approach in our work. Our approach diverges from these methods in several key aspects. Firstly, Wurm et al. [9] and Solanas and Garcia [8] divided the map into regions and assigned frontiers to robots within a region in a centralized manner. Wurm et al. [9] also recalculated new partitions at each iteration. In contrast, our algorithm empowers agents to individually determine frontier points within their assigned static regions in a decentralized fashion. Secondly, Lopez Perez et al. [10] shared maps between different robots, with agents individually calculating their Voronoi partitions. In our approach, we employ a centralized entity to partition the map. Our approach effectively combines key advantages of both decentralized and centralized methods, enabling agents

to autonomously explore individual subregions (which reduces inter-agent communication while agents drive) while reducing duplication of computation of Voronoi partitions, and removing the possibility of disagreement between the agents on the subregions themselves. While introducing a central entity does raise concerns about a single point of failure, the CADRE project mitigates this issue by implementing a leader election scheme (described in [5, 39]) to select an alternate leader agent if the current one becomes disabled or otherwise unable to perform its duties.

An additional distinction from previous works lies in the testing environment. Previous studies predominantly focused on testing either in simulation [8, 10] or within indoor office-like structured environments [9]. In contrast, we have conducted extensive verification and validation for our exploration algorithm in a Lunar analog outdoor setting featuring obstacles and craters that closely resemble the lunar surface (Fig. 2). This choice is critical to align our testing conditions with the unique challenges of CADRE's mission objectives, and ensure that the proposed approach is sufficiently mature for infusion in a flight project.

2.4 Communication in multi-robot exploration

Communication is essential for multi-robot exploration as it affects the coordination between robots [40]. Inefficient coordination may lead to poor exploration efficiency, with the same areas being visited multiple times by different robots. Prior research has delved into diverse communication strategies [33] to facilitate multi-robot exploration coordination. While providing an exhaustive treatment of these strategies lies beyond the scope of this work, we highlight three common scenarios frequently employed: 1) Full communication, where communication is always available, 2) Periodic communication [41], where communication is available periodically and 3) Recurrent communication [33], where communication is available only at the deployment locations. In the CADRE mission, agents frequently turn off their on-board CPUs to manage their thermal and power state [5], and collectively wake up their CPUs at agreed-upon times at 30-minute intervals. The approach in this paper only requires inter-agent communication when the agents wake up to share exploration progress and recompute and share subregions to explore; in this sense, the approach is reminiscent of periodic communication. This communication requirement could be further relaxed to have the agents only communicate at the beginning of the mission (to assign subregions) and at the end of the mission (to report the locally explored maps), resulting in further reduced communication at the cost of potentially slower exploration.

2.5 Multi-robot coordination in lunar environments

There are several works that address discuss multi-robot lunar exploration, co-operation and task allocation. For example, Cordes et al. [42] introduce LUNARES: Lunar Crater Exploration with Heterogeneous Multi-Robot Systems, where they describe a system for exploring lunar craters using a heterogeneous team of robots, including ground and aerial vehicles that collaboratively explore complex lunar terrains. They leverage each robot's unique capabilities—such as mobility in rough terrain or an aerial perspective—to overcome specific environmental challenges, achieve efficient coverage, and map previously inaccessible areas. Rocamora Jr. et al. [43] present a multi-robot framework designed to efficiently search for, excavate, and transport lunar mineral resources. Their study emphasizes collaborative strategies among simulated robots to navigate a realistic lunar environment while tackling challenges such as mobility hazard estimation, immobility recovery, and cooperative task planning. Additionally, the work by Cordes et al. [22] discusses a modular approach to developing multi-robot systems for exploration tasks, emphasizing the flexibility of heterogeneous robots that can be reconfigured to adapt to various mission requirements and environmental challenges. Finally, Thomas et al. [44] focus on task allocation and management for robots involved in lunar habitat construction, emphasizing adaptability to changing conditions and energy optimization during tasks such as transporting, mapping, and clearing.

Our work differentiates itself by focusing on a comprehensive exploration pipeline that was matured to a high technology readiness level

(TRL) [45] to meet the demands of lunar surface exploration. Specifically, centralized partitioning and subregion allocation at the leader ensures conflict free decisions across a team of this size (primarily imposed by mass-volume payload constraints). Secondly, subregion exploration decisions (i.e., frontier selections) are delegated to individual rovers to eliminate the overhead of coordination with the leader or other rovers unless a team-level event requires replanning (e.g., attrition). Consequently, the network is kept free for downlink unless otherwise necessary.

A shortened and modified version of this document appears as a chapter in Sharan Nayak's dissertation [46].

3 Problem definition

In this section, we first define the general multi-robot exploration problem and then describe its specific instantiation as addressed by the CADRE mission.

Let $\mathcal{M} \subset \mathbb{R}^2$ denote the workspace in which n agents $\mathbf{A} = \{a_1, \ldots, a_n\}$ operate, with corresponding initial positions $\mathbf{Q} = \{q_1, \ldots, q_n\}$. Let $\mathcal{U} \subseteq \mathcal{M}$ represent the unexplored space and $\mathcal{E} \subseteq \mathcal{M}$ represent the (explored) space, such that $\mathcal{M} = \mathcal{U} \cup \mathcal{E}$. Each agent a_i constructs its own local environment representation R_i as it explores its assigned region. The global environment representation R_{global} is created by merging all local representations R_i, yielding a unified understanding of the workspace. Given ($\mathbf{A}$, $\mathbf{Q}$, $\mathcal{U}$), the objective of multi-robot exploration is to move agents in $\mathcal{U}$ such that they collectively explore $\mathcal{U}$ (i.e., the agents cover every point in $\mathcal{U}$ with their sensor footprint, with the exception of points that cannot be reached because they are surrounded by impassable obstacles).

Specifically, the approach proposed in this paper decomposes the multi-agent exploration problem in four subproblems.

- **Partitioning:** Partition the desired region to explore $\mathcal{M}$ with unexplored space $\mathcal{U} \subseteq \mathcal{M}$ into n regions $\mathbf{R} = \{r_1, \ldots, r_n\}$ such that $\mathcal{M} = \bigcup_{i=1}^{n} r_i$ and $r_i \cap r_j = \emptyset$ for $i \neq j$.
- **Assignment:** Compute an assignment $\mathbf{K} : \mathbf{Q} \mapsto \mathbf{R}$ such that agent a_i explores region r_j.
- **Single-Robot Exploration:** For each agent a_i, compute a sequence of locations within r_i to be visited, ensuring complete coverage of r_i by the agent's sensor footprint.
- **Merging:** During exploration, periodically merge the local environment representations R_i generated by each agent into a unified global representation R_{global}, ensuring an accurate and cohesive depiction of the explored space.

Note that although the workspace is specified as two-dimensional to simplify the problem definition, 3D elevation maps incorporating terrain slope and height are utilized during hardware validation. This integration, which allows the robots to navigate challenging terrain such as rocks and craters typical of lunar environments, is detailed in Section 6.2.1 and 6.2.2.

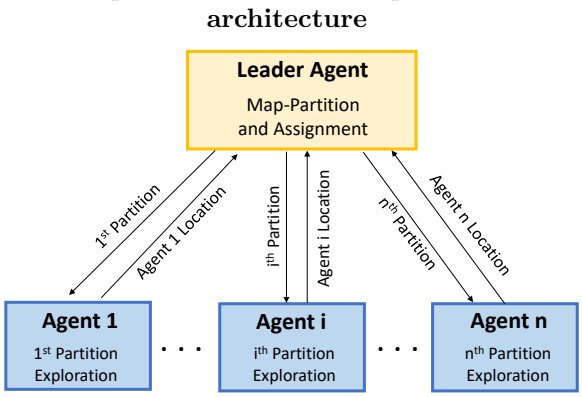

Fig. 3: Leader agent divides the unexplored space into disjoint partitions and then assigns a partition to an agent. Each agent individually explores its partition without communication of its frontier points with other robots.

4 Algorithm design

The work in this paper is motivated by key mission objectives including the elimination of redundant coverage, reduction in mission time, and judicious use of communication resources among the participating agents. These objectives are achieved through the implementation of a map-partition-based exploration algorithm. In this algorithm, an elected leader agent (as depicted in Fig. 3) undertakes the responsibility of dividing the unexplored space into a collection of non-intersecting

partitions and subsequently assigning these partitions to individual agents (Algorithm 1). Once assigned, each agent proceeds to autonomously explore its designated partition without any external intervention from other agents (Algorithm 3).

It is important to emphasize that, while the leader agent plays a pivotal role in the partition assignment process, the proposed algorithm is best described as semi-centralized rather than centralized. This distinction is drawn because, despite the leader agent's involvement in the initial allocation of partitions, the agents subsequently operate independently within their assigned regions, exploring without reliance on the leader agent. The leader agent gets re-involved in the exploration process only if one of the agents gets stuck or malfunctions during the exploration process. Thus, although the robots explore their assigned regions independently, the mission still requires a level of coordination between the leader agent and other agents for partitioning and maintaining the overall system's functionality.

A broader discussion of leader selection methods is outside the scope of this paper; we refer the interested reader to [39] for a description of the leader election algorithm used by CADRE.

We present the algorithm part that runs on the leader agent in subsection 4.1, and the algorithm part that runs on the agent in subsection 4.2.

4.1 Region generation on leader

The leader agent executes three key computations (Fig. 4) - map partition, robot-partition assignment, and boundary computation.

4.1.1 Map partition

The map-partitioning algorithm (line 1 in Algorithm 1) is responsible for dividing the unexplored map into n non-intersecting partitions, one for each robot. To satisfy the mission constraints of uniform work allocation among the robots, we require a map-partitioning algorithm that produces approximately uniform-sized partitions. We considered two algorithms - Voronoi-based partition [47] and K-means-based partition [48]; additional detail on these approaches is provided in the Appendix. Following extensive experimentation and evaluation, described in Section 5, we chose to implement the K-means algorithm (Algorithm 2). The final step in the map-partitioning algorithm (Algorithm 2) uses the computed centroids from K-means to generate the regions such that $\mathcal{M} = \bigcup_{i=1}^{n} r_i$. That is, the cluster centroids are computed using exclusively the unexplored space $\mathcal{U}$, ensuring that the unexplored space is partitioned approximately uniformly among the agents; however, *every* point in the workspace, including explored points, is assigned to an agent. This allows the frontier selection algorithm to consider all prior knowledge of the workspace from other agents, including obstacles, so that a point that has been marked as an obstacle by an agent is not selected as a frontier point by another agent. Furthermore, the frontier search described in Section 4.2 requires the regions to be spatially connected in order to find the optimal solution. We achieve this by labeling the entire workspace, mitigating the likelihood of generating spatially-disconnected regions.

We use a grid map representation of the environment. The input to the map-partitioning algorithm are the unexplored grid cells, and the initial location of robots. Initially, no prior information about the internal structure of the area or specific obstacles are provided. The initial unexplored map layout is transmitted by the ground operators on Earth. Periodically, as agents wake up in a coordinated fashion, they re-partition the region to explore. To do this, as mentioned in Section 3, the robots send their local maps to the leader, allowing the leader to utilize an updated merged map for partitioning. The location of obstacles is inflated by the enclosing radius of the rover, which allows the exploration planner to treat individual robots as points.

Map-partition and assignment in leader agent

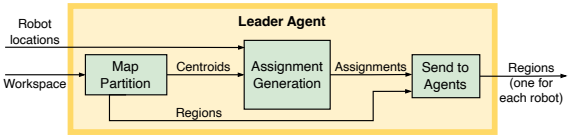

Fig. 4: Leader agent performs map-partition to divide map into regions, computes the assignments, and sends the region to each corresponding agent.

Algorithm 1: `LeaderAgent` $(\mathcal{M}, \mathcal{U}, n, \mathbf{Q}, \mathbf{A})$

```
1 R ← MapPartition(M, U, n)
2 K ← GetAssignment(Q, R)
3 for i ← 1 to n do
4     SendToAgent(a_i, r_i)
```

Algorithm 2: `MapPartition` $(\mathcal{M}, \mathcal{U}, n)$

```
  // KmeansAlgorithm. C is the centroids set
1 C ← InitializeCentroids(U, k)
2 for i ← 1 to maxIterations do
3     R ← AssignClusters(U, C)
      // C_new is the updated centroid set
4     C_new ← updateCentroids(U, C, k)
5     if CentroidsConverge(C, C_new) then
6         break
7     C = C_new
8 // Assign clusters one last time with M
9 R ← AssignClusters(M, C)
10 return R
```

4.1.2 Robot-partition assignment

This module assigns a partition to an agent (line 2 in Algorithm 1). We use the Hungarian algorithm [49] to determine the optimal mapping between partition and agent. A variety of cost functions can be used within the Hungarian algorithm to determine this mapping; in our experiments, we have employed the distance between the robot's current location and the centroid of the partitioned regions.

Algorithm 3: `Agent`$_i$ $(b_i, c(f))$

```
1 r_i ← Region(b_i)
2 while U ∩ UnexploredRegion(b_i) ≠ ∅ do
3     F_i ← GetFrontierPoints(r_i)
      f_{b_i} ← GetBestFrontierPoint(F_i, c(f))
4     MoveRobotToFrontierPoint(f_{b_i})
```

4.2 Frontier-based exploration on every agent

The agents use a frontier-based algorithm [1] for exploration due to its speed and simplicity. There are three components (Fig. 5) to the proposed frontier-based exploration approach: 1) Frontier points generation within the assigned partition 2) Calculation of the best frontier point using a user-defined cost-function, and 3) Navigation to the frontier point.

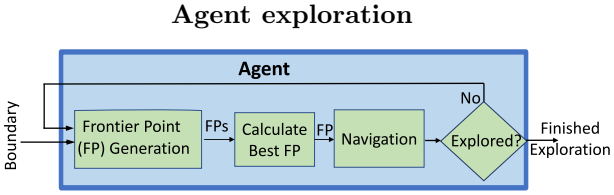

Fig. 5: Each agent calculates frontier points within the assigned partition. The agent then uses the cost function in Equation (1) to calculate the best frontier point. This frontier point is input to the rovers' single-agent surface mobility stack, which calculates the best path and move towards to it. This process is repeated until the region is fully explored - i.e., there are no more reachable, unknown parts of the region.

4.2.1 Frontier point generation

A frontier is defined as the boundary between explored and unexplored space. We employ BFS [26] to determine the frontier cells and use Divisive Hierarchical K-means clustering [50] to cluster the frontier regions. The frontier region's size is determined by the sensor footprint view radius, which is a function of the sensor's depth and the Field of View (FOV).

We use the region centroids as the corresponding frontier points (line 3 in Algorithm 3). The frontier points represent the potential locations the robot can move towards to uncover new space. We note that although each agent maintains a copy of the global unexplored map, Our frontier generation algorithm ensures that the search and selection of frontier points are restricted to the agent's assigned region or partition. This is done by simply not considering any explored cells outside the partition while iterating through explored cells while performing BFS. We note that in our initial simulation and Mercury-7 experiments, we represented the region as a convex hull, with points outside the region ignored using the even-odd algorithm [51, 52]. However, for the more recent Development Model (DM) experiments, we replaced the convex hull representation with

a labeled map. In this labeled map, grid pixel values of "1" indicate assigned areas, while values of "0" denote non-assigned areas. Although this approach increases the size of the transferred regions, it ensures that there is no overlap between regions in degenerate cases and facilitates efficient rejection of unassigned points during the BFS process.

Due to the non-zero size of the robot's sensor footprint, there is a possibility of some overlapping coverage when the robots explore the area near the partition boundaries, even though the frontier points lie inside a robot's assigned partition. This is reduced by periodically merging all robots' local maps into a global map, and updating the robot's local map with the global map.

Algorithm 4: `UpdateTabooList` (P_{frontier}, P_{anchor}, h_{anchor} r_{taboo})

```
   // d is the distance between new frontier and
      current candidate taboo point
1  d ← GetDistance(P_frontier, P_anchor)
   // If the new frontier point is far from the
      candidate taboo point, reset the candidate
2  if (d > r_taboo) or (h_anchor = 0) then
3  |  P_anchor ← P_frontier
4  |  h_anchor ← 1
   // If the new frontier point is close to the
      previous one, increase the number of hits
5  else
6  |  h_anchor ← h_anchor + 1
   // If we sent enough points close to the
      candidate, add it to the taboo list
7  if h ≥ h̄_anchor then
8  |  taboo_list.append(P_anchor)
9  |  h ← 0
```

4.2.2 Determination of the best frontier point

The best frontier point, denoted as f_{b_i} for agent a_i, is selected from a set of candidate frontier points $\mathbf{F}_i$ based on its utility or cost function (line 3 in Algorithm 3). Various utility and cost functions have been utilized in prior frontier-based exploration research, and a comprehensive review of these functions can be found in [30]. In this work, the cost function $c(f)$ incorporates three terms, as expressed in equation (1), tailored to meet the mission constraints. The first term, $d(f)$, represents the distance between the current robot's location and the potential new frontier point. We introduce this term to penalize longer travel distances, ultimately reducing power consumption. The second term, $u(f)$, quantifies the information gain, reflecting the amount of unexplored space that would be uncovered upon reaching the frontier point. This term serves to enhance exploration efficiency. The third term $r(f)$, captures the smallest rotation amount needed to rotate the robot from its current heading to the frontier point "heading". The frontier point heading is the heading of the vector connecting the robot and the frontier point. We include this term to study penalizing rotational motion as it causes more wear and tear on the wheels than linear motion.

The mathematical notation for $c(f)$ is as follows:

$$c(f) = w_1 d(f) + w_2 u(f) + w_3 r(f) \tag{1}$$

where w_1, w_2, and w_3 are the weights associated with $d(f)$, $u(f)$ and $r(f)$ respectively. The frontier point is chosen as the one with the lowest cost.

In practice, the weights can be selected through various approaches like empirical tuning based on mission priorities, grid or random search [53], optimization based approaches [54] or machine learning [55] approaches. A higher w_1 will prioritize frontier points closer to its current location whereas a lower w_1 will choose frontier points away from its current location. A higher w_2 improves the quality of exploration, especially in areas with high uncertainty or unexplored regions whereas a lower w_2 causes robot to visit frontier points that are closer or easier to reach, possibly leaving high-information areas unexplored. Finally a higher w_3 prioritizes frontier points that require less rotation to align with the robot's current heading, making the robot's movements smoother. However a lower w_3 lead to missed exploration opportunities, as the robot might avoid significant heading changes even when they could yield high information gain. For CADRE, the weights were tuned empirically after significant experimentation, described in Section 6.

4.2.3 Taboo list search

An application-inspired addition to the frontier selection algorithm was the addition of a feature to keep track of frontier areas that a rover repeatedly

fails to reach, and exclude them from subsequent frontier point calculations. We observed in hardware experiments that, occasionally, the rovers' navigation stack would be unable to navigate to frontier points, despite the existence of a free path between the rover's location and the frontier point in the map. This phenomenon can be caused by discrepancies between local maps used for collision avoidance and the map used for exploration, or by wheel slip causing insufficient progress toward a goal. In order to allow the exploration algorithm to autonomously recover from such situations, we implemented a "taboo list search", shown in Algorithm 4.

When candidate frontier points are scored according to Equation 1, frontier points that lie within the prescribed radius of a point in the taboo list are excluded; this ensures that, after sufficient failures to progress to a given region, the rover will move on to a different region.

The taboo list is periodically cleared when the robots reboot, ensuring that temporary surface mobility misbehavior (caused, e.g., by transient lighting conditions) does not cause the robot to ignore parts of the environment permanently.

The taboo list was only used for the DM hardware experiments described in Section 6.2, which demonstrated the effectiveness of the approach in challenging environments, as reported in Section 6.2.2 and Table 3.

4.2.4 Surface Mobility

The final operation in the agent-based exploration is navigating to the frontier point (line 4 in Algorithm 3). CADRE's surface mobility planner combines a global and local planner, similar to [56]. The global planner uses the currently available global map to build a high-level collision-free path to the frontier point. The local planner is myopic and uses incoming sensor data to plan a path in the robot's immediate vicinity. In our simulation and Mercury-7 hardware experiments, the agents use Dijkstra's algorithm (specifically, the Navfn ROS implementation) [56] for the global planner and TEB [57] for the local planner while for the DM experiments, both global and local planners use a Time Elastic Band (TEB) implementation over different horizons to plan a rover's path to a frontier point. The local planner is also capable of turn-in-place maneuvers that are limited to 90° to avoid excessive sinkage in the lunar regolith. This capability is also required due to the rover's non-zero minimum turn radius when driving forwards (or backwards). The process of calculating and moving to the frontier points is repeated until a desired percentage of the region of interest (including potentially the entire region) is explored (line 1 in Algorithm 3).

4.3 Event handling

Autonomous exploration relies on the capability to handle, and in some cases recover from, events on-board a rover (i.e., local) and across the team (i.e., system-level). Specifically, individual rovers must be able to handle the case where they are unable to reach a selected frontier point, while the team must be able to recover from the event that a rover is unable to continue to participate in exploration. We will describe how these two events are handled by our implementation.

4.3.1 Local event handling: surface mobility failures

Exploration manages the feedback from the planners of the surface mobility stack. The planners may report a failure due to inability to reach the frontier point before a provided timeout, or exceeding a number of failed attempts to plan a trajectory to the waypoint. These failures can occur, for instance, if mapping reports a region as traversable but the surface mobility planners are unable to find a feasible trajectory within that region. Another reason is when the frontier point is placed too close to an obstacle. Recall that, before the frontier point is computed, the obstacles in the grid map input are inflated by the rover's enclosing radius to avoid placing frontier points at unreachable locations. However, as the rover drives towards a frontier point, mapping can reveal an obstacle in a region that was previously marked unknown and near the frontier point. This can result in the surface mobility planners being unable to find a feasible trajectory to the waypoint. Both cases are handled by recomputing a new frontier point with an updated rover pose and an updated merged map. If the same frontier point is computed multiple times sequentially, but not reached, then exploration inserts the point into the taboo list described in Section

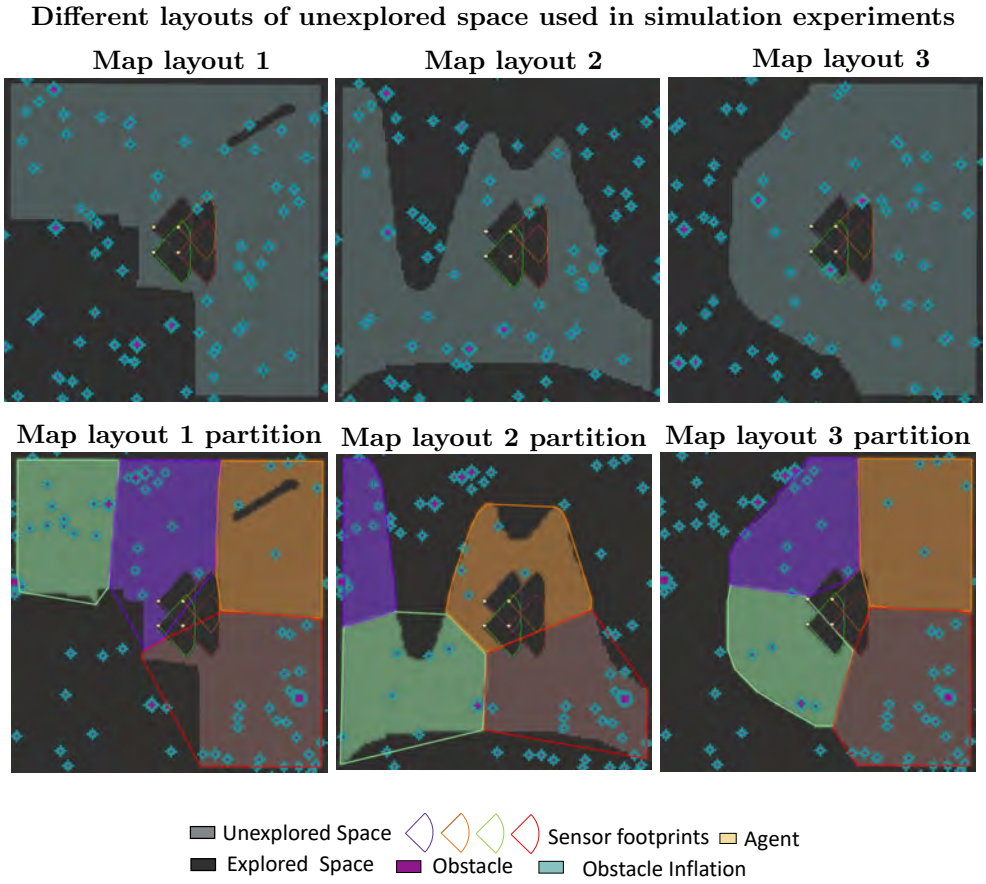

Fig. 6: For each layout, we run 50 trials with randomized obstacle environments. Note that the obstacle locations are unknown to all robots at the mission start and until the locations are detected using the robots' sensors.

4.2.3. Consequently, exploration is robust to scenarios where the surface mobility system reports failure, without compromising additional regions to explore.

4.3.2 System-level event handling: changes in team participation

During exploration, agents may malfunction or become stuck, requiring a failure detection mechanism to maintain mission continuity. Our system achieves this through the notion of a semi-static participation list (i.e., who should be participating?) and a network link status (i.e., is this agent awake and running FSW?).

Each rover maintains a list of agents "participating" in exploration. The list can be updated by ground operators; an early implementation also allowed agents to marked other agents as non-participating based on inter-agent heartbeat pings; however, the functionality was descoped from the final flight software.

Changes in the participation list do not result in immediate replanning: if a rover is added to the exploration team, it starts idle, and, if a rover is removed from the exploration team, its region remains temporarily unexplored. Upon the next reboot, however, the team planner re-partitions the region based on the set of participating agents, allowing for addition or removal of agents during exploration.

An early implementation of exploration reassigned regions to agents immediately whenever a rover joined or left the team; the change to only re-assign regions upon reboot was motivated

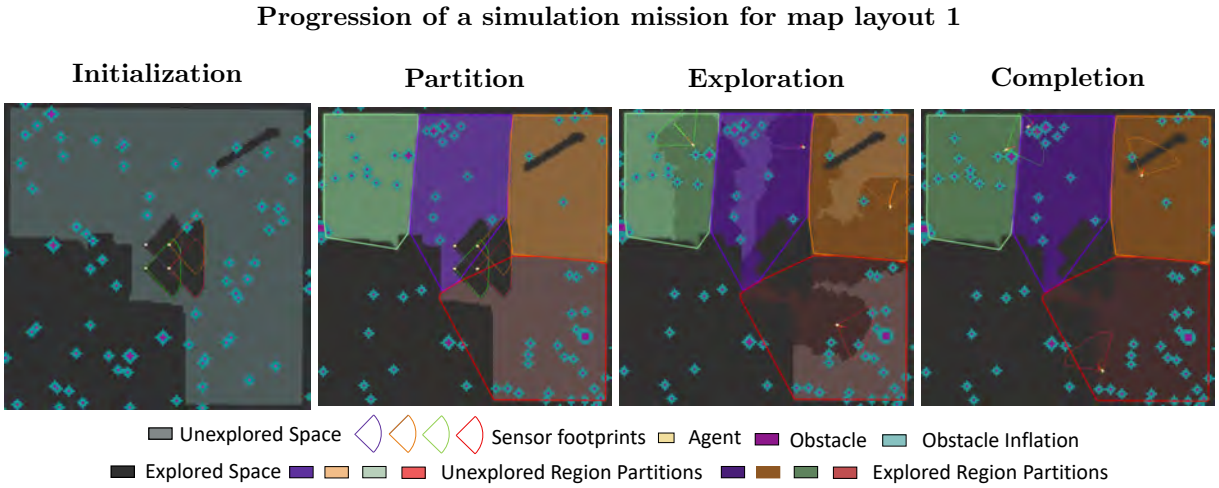

Fig. 7: Workspace initialization. The second figure (from left) shows the workspace partition into 4 regions, one corresponding to each robot. The third figure shows region exploration by their assigned robots. The last figure shows exploration completion.

by concerns about "chattering" where one malfunctioning agent may leave and rejoin the team repeatedly, resulting in frequent replanning and lack of progress.

In the next two sections, we present the simulation (Section 5) and hardware experiments (Section 6) conducted to validate our proposed approach.

5 Simulation experiments

We conduct a series of simulation experiments to thoroughly evaluate the effectiveness of our design choices. We provide the simulation setup in subsection 5.1 and the simulation results in subsection 5.2. In both simulation and Mercury-7 hardware experiments, we consider the exploration mission to be completed when at least 95% of the total area has been explored.

5.1 Simulation setup

The simulation environment is ROS-based of size 30m × 30m (Fig. 7) with obstacles interspersed between free space. The obstacle locations are unknown to the rovers at the start of the experiments. We perform planning on a grid map overlaid on top of the environment with a size of 150 x 150 cells (0.2 m/cell resolution). We run experiments for 3 scenarios (Fig. 6), each having a different layout of unexplored space. The layouts feature distinct characteristics, including peaks, troughs, rounded and straight edges, which allow us to evaluate the performance of the proposed exploration algorithm under diverse conditions. The three different layouts were selected to demonstrate two primary aspects: first, that the partitioning algorithm can adapt to various shapes of unexplored space, and second, that the frontier-based exploration algorithm, combined with the proposed cost function, can effectively navigate and explore areas of diverse geometries.

For each scenario, we run 50 trials using environments with randomized obstacle locations, each with 3% obstacle Cumulative Fractional Area, or CFA [58] — a very conservative estimate of the CFA that CADRE expects to encounter at Reiner Gamma. The 4 rovers in the experiments have differential drive dynamics, and dimension of 0.25m (length) and 0.20m (width). The agents have a sensor footprint of depth 2m and FOV 90^o, and a maximum velocity of 2m/s. For each experiment, we evaluate as performance metrics the total exploration time, average distance traversed by agents, the maximum distance travelled by any agent, and number of messages sent by the base station to agents. We implement the algorithms in C++ and run the experiments on a system with Intel i7-7700 4-core CPU with 32GB RAM.

In the comparative analysis, we evaluate our design choices by comparing four versions of

the proposed algorithm. The first two versions, Proposed-K and Proposed-V, utilize K-means and Voronoi map partitions, respectively. These algorithms initialize K-means centroid locations and Voronoi generator points using agent locations. Frontier-point selection in these algorithms is guided by the cost function defined in equation (1), with parameters set as $w_1 = 1.0$, $w_2 = 0.6$, and $w_3 = 0.25$. The last two versions, Random-K and Random-V, differ in that they select frontier points randomly, without considering the specified cost function. A summary of the distinctions between these algorithm variations is provided in Table 1. Additionally, we include a centralized algorithm [33], referred to as "centralized" in our experiments, which relies on a central entity to generate frontier points for each agent at each iteration. Specifically, we adopt the second of the two centralized approaches proposed by [33], known as the two-stage approach in their work. This algorithm operates under recurrent connectivity constraints to ensure connectivity at frontier locations. The centralized strategy was included in the comparison for two primary reasons. First, the precursor to CADRE, known as A-Puffer [59], was evaluated using the centralized approach in [33], which was found to be comparatively inefficient in terms of communication overhead and distance traveled by the robots. Including this strategy provides historical context and underscores the improvements achieved by our semi-centralized approach. Second, the centralized strategy serves as a baseline for evaluating metrics such as distance traveled and communication overhead, highlighting the advantages of the proposed semi-centralized approach, which demonstrates clear improvements over the commonly employed centralized strategy.

	Partition		Frontier-selection	
Algorithm	K-means	Voronoi	Cost func. (1)	Random
Proposed-K	✓		✓	
Proposed-V		✓	✓	
Random-K	✓			✓
Random-V		✓		✓

Table 1: Different variations of our proposed algorithms used in simulation experiments to test our design choices

5.2 Simulation results

We show a simulation mission progression in Fig. 7 and the experiment results in Fig. 8. The proposed exploration algorithm (Proposed-K) performs the best in the metrics of mission completion time, average distance traveled, and maximum distance traveled. Proposed-K performs better than Proposed-V as the former produces better uniform-sized partitions, which leads to a more even distribution of work among the agents. In addition, our algorithms perform better than their corresponding versions (Random-K,V) that use the random frontier selection policy, which shows the advantage of using our cost function to improve exploration efficiency.

6 Hardware experiments

Once the partitioning strategy was selected, we evaluated the performance of the proposed approach through extensive hardware experiments on two sets of hardware: a mid-fidelity hardware platform, dubbed Mercury-7, equipped with a ROS-based mobility stack that allowed for rapid iteration and experimentation; and high-fidelity Development Models (DM) that faithfully reproduce the hardware and flight software that will be used by CADRE on the lunar surface. The Mercury-7 rovers had a Dubins-vehicle dynamics with a minimum turning radius whereas the DM rovers had Differential-drive dynamics that allowed rotation in place. We describe the hardware configuration, testing environment, and results of the Mercury-7 and Development model rovers in subsection 6.1 and subsection 6.2, respectively, and the lessons learned in subsection 6.3.

6.1 Mercury-7 rovers

We utilize two prototype rovers that provide a mobility and sensing platform representative of the CADRE flight rovers (Fig. 9). Each rover is equipped with a ModalAI Voxl-2 board featuring a Snapdragon 821 processor for sensor data processing and execution of flight code. The rovers are equipped with stereo cameras offering a resolution of 1600 by 1300 pixels and a maximum frame rate of 30 Hz; in operations, the frame rate is reduced to 5 Hz to manage data processing load. The CPU and cameras are analogous to the flight articles'. To drive the rover, each wheel employs

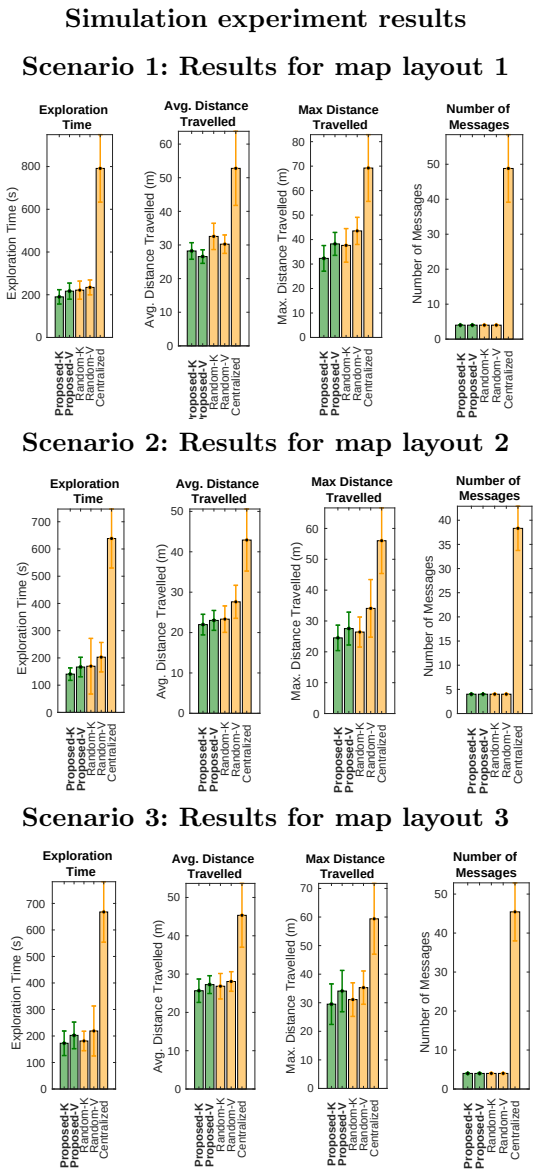

Fig. 8: Performance metrics for scenarios 1 (top), 2 (middle), and 3 (bottom). For each scenario, we run 50 trials of each algorithm. The algorithms proposed in this paper are highlighted in green.

a DYNAMIXEL servo dc motor; the motors are connected in a daisy chain configuration, linked to the processor via a USB communication converter for serial-to-USB conversion. A lithium-ion battery provides a 12V power source for the rover, and power distribution is managed through a dedicated power distribution board. The rovers are equipped with a protruding WiFi antenna to enable long-range communication with the base station. The rover's chassis and wheels are 3D printed, with the wheels featuring grousers for enhanced traction and the ability to navigate over small obstacles like rocks. The maximum linear velocity of the rovers is set to 0.05m/s.

A stationary laptop with an Intel i7-8750H 6-core CPU with 16GB RAM acts as the leader agent, which performs the region partition, and then relays the regions to the robots using 802.11n WiFi. Throughout the exploration process, each robot continuously updates its individual obstacle map and periodically transmits these updates to the leader. The leader agent employs the multi-robot merge package [60] to consolidate the local maps into a unified merged map.

6.1.1 Mercury-7 test environment

We perform the hardware experiments on the Mercury-7 rovers at JPL's Moon Yard. The yard is cluttered with rocks, small boulders, and craters to mirror the lunar surface. We design three distinct scenarios to evaluate the effectiveness of our proposed algorithm. The first two scenarios involve the exploration of rectangular regions measuring 6m by 6m and 8m by 6m, respectively, with variations in obstacle and crater layouts. In the third scenario, we conduct a resilience test in 6m by 6m region to ensure that exploration can continue even if one of the robots fails to explore. In all the described scenarios, we use K-means algorithm to perform map partitioning. The weights in equation (1) are set to $w_1 = 1.0$, $w_2 = 0.6$, and $w_3 = 0.0$, with the rotational cost excluded to reduce complexity.

6.1.2 Visual-guided navigation pipeline

On the Mercury-7 rovers, we employ a stereo imaging system to perceive the environment. The choice of using a stereo camera system is driven by its advantages in terms of lightweight design, low power-consumption and cost-effectiveness. The term "lightweight" refers to the compact size and reduced physical weight of the stereo camera hardware, particularly when compared to sensors like LIDAR or depth cameras. The stereo camera generates a synchronized pair of images which is given as input to the stereo module and localization module (Fig. 10). The stereo module comprises a disparity sub-module responsible for generating

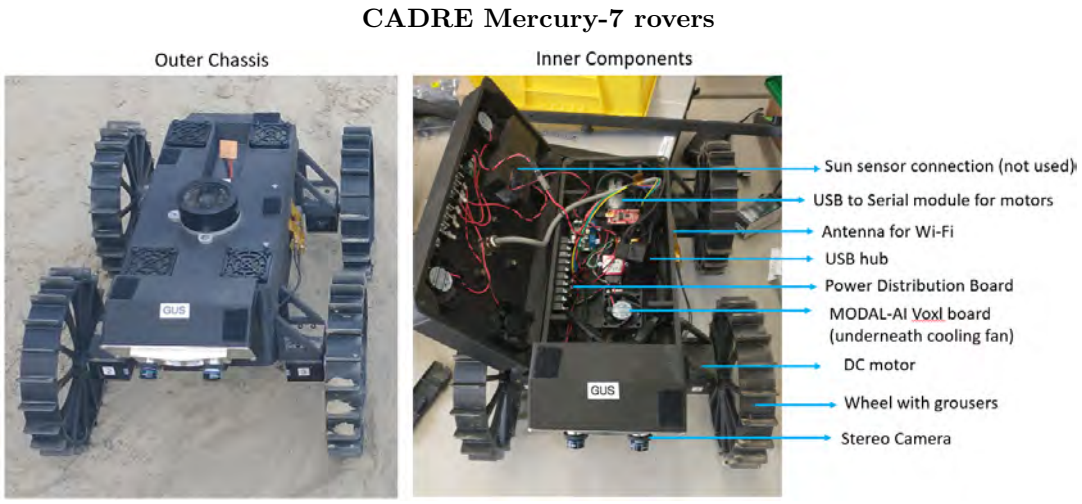

Fig. 9: Close-up view of prototype Mercury-7 Lunar rovers that employ a ROS-based navigation stack.

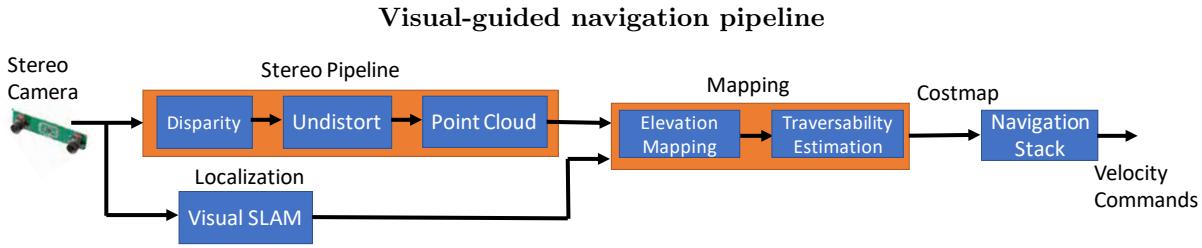

Fig. 10: Visual-guided navigation pipeline used in the CADRE rovers.

the disparity image [61] which is then given as input to the "undistorted" sub-module to produce the undistorted image. The undistortion relies on knowledge of a camera distortion model, which is derived via camera calibration [62] before the start of the mission. We use the Kalibr toolbox [63] to perform camera calibration. The undistorted image is given as input to a point cloud library (PCL)[64] that produces a detailed point cloud representation of the image (Fig. 11). Simultaneously, the stereo images are fed to the localization module. This module tracks key feature points across images, leading to the generation of visual odometry [65]. The point cloud data and odometry are both fed to an elevation mapping module [66] which outputs an elevation map, capturing height variations. The elevation map, in turn, is fed to the traversability estimation module [67] to produce a traversability costmap. This costmap is a 2D grid with each grid cell representing the terrain traversability based on its height and slope. We use a threshold to determine if a cell is traversable or not based on the robot's capability. Primilarily, the robot's traversability is determined by its wheel design including wheel diameter and grouser setup. We intentionally avoid complex features like surface friction in the costmap to reduce computational complexity. The costmap is used by the navigation stack to traverse the terrain. We use the ROS `move_base` [68] package to handle accepting goal inputs and move the robot to the desired positions using the ROS navigation stack.

6.1.3 Mercury-7 experiment results

We present the results of the Mercury-7 rover hardware experiments for the first two scenarios in Table 2, where each data point represents the mean of three trials. The robots demonstrated their capability to efficiently compute intermediate frontier points and autonomously explore their designated regions, as visualized in Fig. 12. As

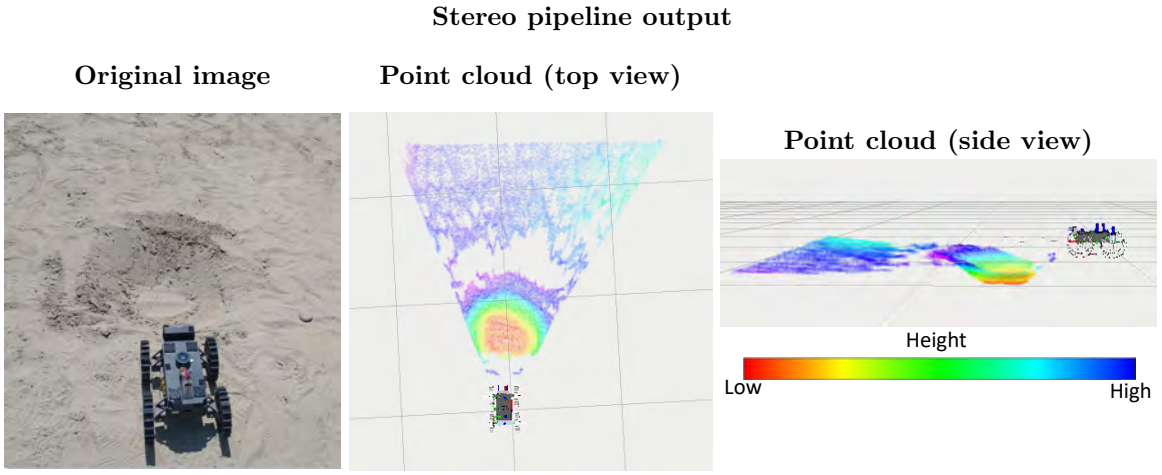

Fig. 11: Original image showing a crater (negative obstacle) in the front of the rover and the corresponding point cloud in top view and side views. The different point colors represent different heights.

expected, the exploration of the larger area in Scenario 2 naturally required more time compared to the exploration of the smaller area in Scenario 1.

The resiliency experiment's progression is visualized in Fig. 13. Initially, two robots embark on the exploration mission, but a simulated failure occurs where one robot fails proceed with exploration. This failure mimics situations where a robot has to stop due to thermal and power constraints, or hardware failures. In response, the base station dynamically re-partitioned the regions, taking into account only the active robot. Subsequently, the active robot continued its exploration and successfully completed the mission. Note that in our experiment, we manually simulated robot failure and updated robot count from 2 to 1 to recompute the partitions. However, the CADRE mission plans to implement a set of fault-protection mechanisms to autonomously detect robot availability, and trigger a replan in response to changes in the robots' statuses.

6.2 Development-Model (DM) rovers

A second set of hardware experiments were performed on the base station and two development model (DM) rovers based on the flight models described in [5], to assess the performance of the proposed approach when integrated in CADRE's flight software stack. The avionics and sensors on the development model rovers (Fig. 15) are largely similar to the flight model (Fig. 14), with some missing hardware, such as the solar cells on the panels and the radiator. The DM base station is identical to the rovers in terms of its compute and communication capabilities, but does not have the capability to drive. The base station is responsible for the communication of the data from the agents to the lander and vice versa, functioning as the gateway for uplink and downlink between ground operators and the system. Surface mobility and localization are provided by a bespoke navigation stack developed for the mission. Individual rovers' maps are stored, shared, and merged by a bespoke distributed mapping database, MoonDB. We refer the interested reader to [69] for a detailed discussion of the database. Communication between the processes used NASA's F' framework [70]. The hardware experiments did not include the lander,

Metric	Mean (with std) - 6mx6m	Mean (with std) - 6mx8m
Exploration Time (min)	18.28 ± 3.83	22.88 ± 4.13
Avg. Distance Travelled (m)	22.11 ± 4.97	31.70 ± 7.49
Max Distance Travelled (m)	25.15 ± 2.77	35.05 ± 3.37

Table 2: Mercury-7 experiment results for CADRE multi-robot exploration tests. Each point is a mean of three trials.

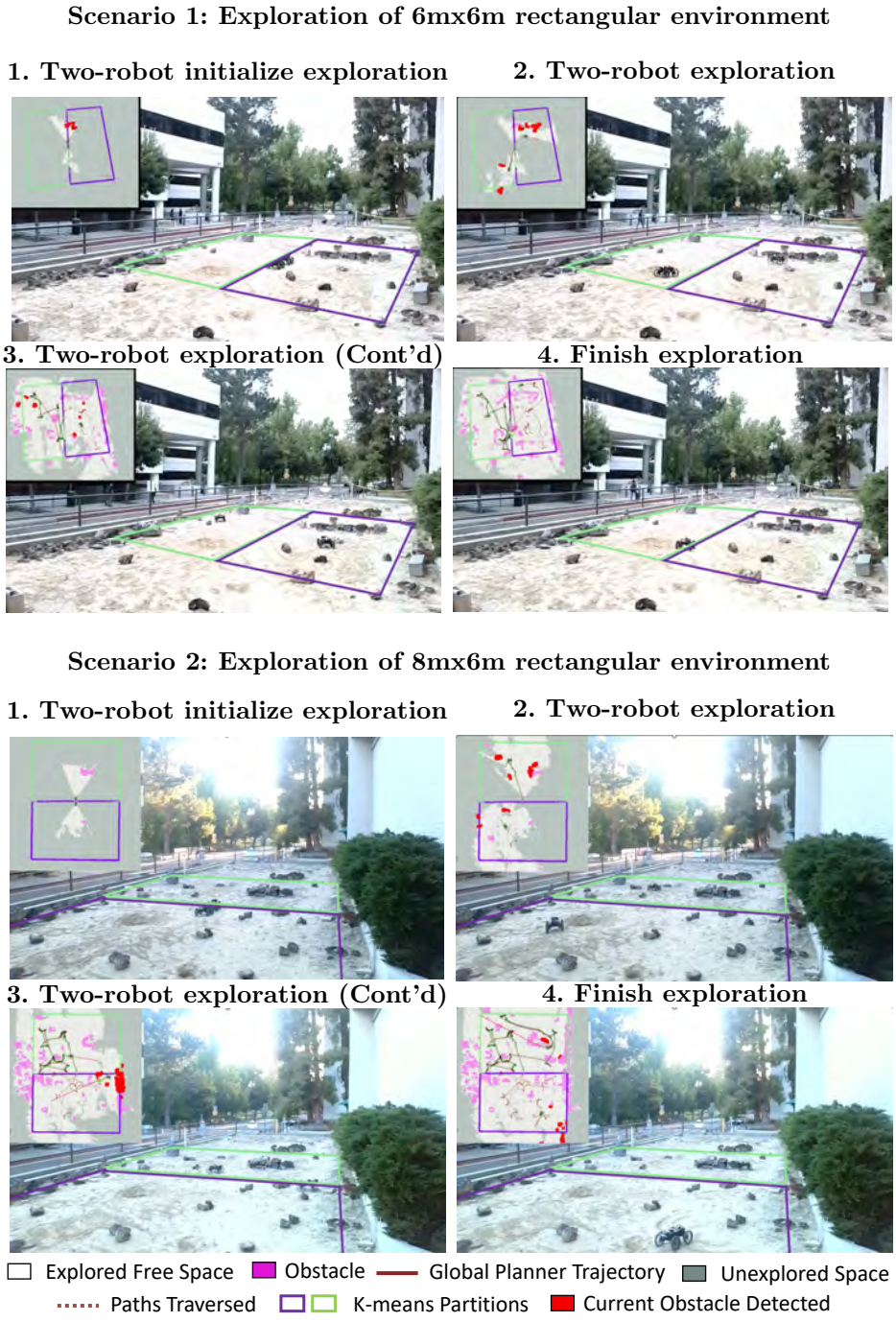

Fig. 12: Two-robots exploring 6mx6m (scenario 1) and 8mx6m (scenario 2) using our proposed algorithm

but the rest of the system was setup as it would be operated during the mission.

6.2.1 DM test environment

The experiments with the development models were executed in both an indoor test arena and the Mars Yard at JPL (Fig. 16). The campaign

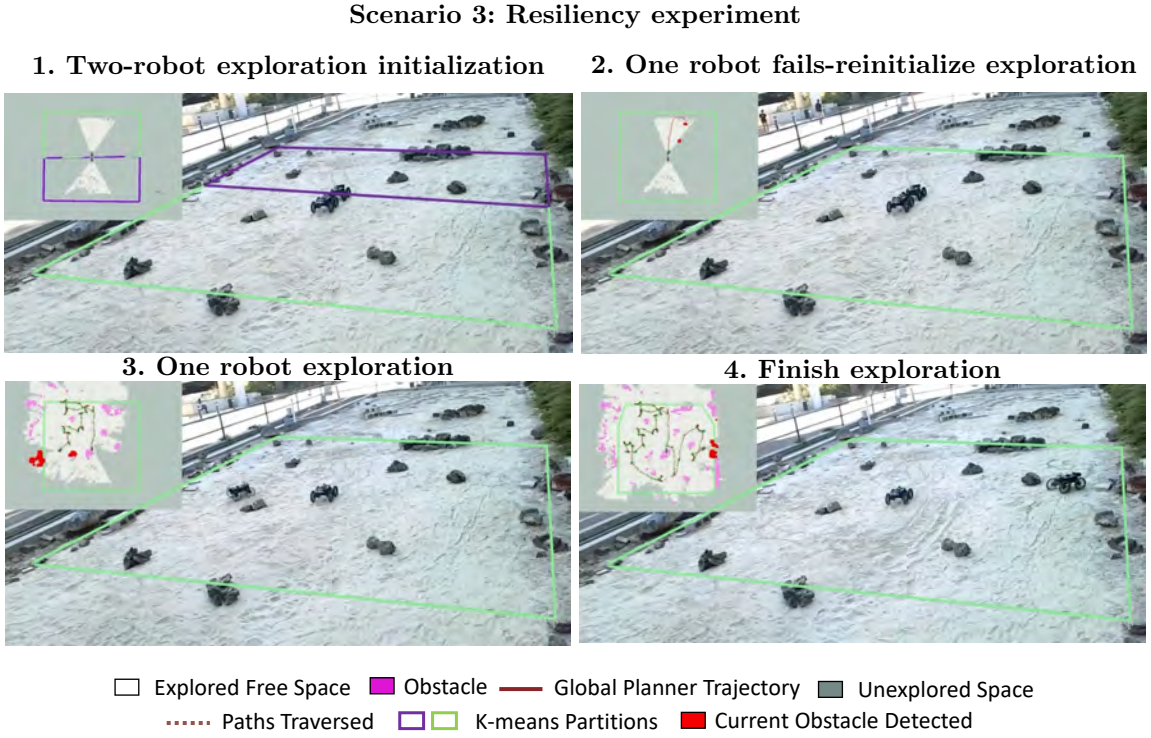

Fig. 13: Testing resilience: two robots start out (sub fig. 1), but one fails. Exploration system adapts, reassigning full exploration to the active robot.

consisted of 15 total experiments, with 2 tests in the indoor area and the remainder in the Mars Yard. Out of the 13 Mars Yard tests, 3 of the them were executed after sunset in more lunar-like lighting conditions. The indoor test arena consists of a 6.5m by 8m flat surface with sparsely-spaced rocks and small boulders. The test area in the Mars Yard is up to 18.0m by 22.5m region with the ability to configure the area similar to the lunar surface (specifically, this testbed afforded testing with craters). For the evening Mars Yard experiments, a large flood light illuminated the operational area to create shadows in effort to mimic the effect of the sun on the lunar surface.

The experiments progressed in complexity, both in terms of environmental difficulty and in terms of integration with other flight software functionality. For environmental difficulty, the initial tests were performed in the indoor test arena to provide an environment that is less challenging than the lunar surface to specifically focus on verifying the implementation of exploration algorithm. Next, the first experiments in the Mars Yard contained sparsely-spaced rocks of various sizes, and a few craters. Subsequent Mars Yard experiments had an increased number and craters to simulate more accurately the lunar surface, with an environment that included hundreds of rocks up to 0.05m diameter, rocks greater than 0.05m about every 1-2m, over 10 craters up to 1m diameter, and a large crater over 2m in diameter.

For what concerns flight software integration, throughout the experiments, additional features were included, including autonomous planning with "wake-sleep" cycles, where the rovers boot up simultaneously every 30 minutes and autonomously plan their activities including exploration, cooling down, and recharging. Use of the taboo list, described in Section 4.2.3, was also introduced in these tests to prevent the selection of a frontier point that has been deemed unreachable by the surface mobility planners multiple times (e.g., a frontier point inside of a crater) so that the rovers can continue to explore their entire region without continuously attempting to reach unreachable unknown spaces.

CADRE Flight Model rover

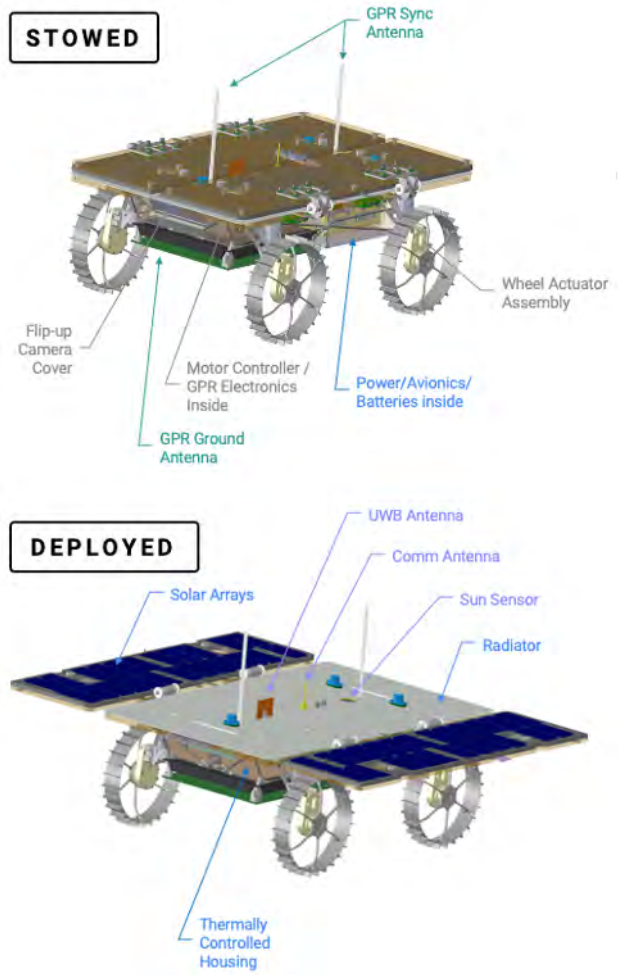

Fig. 14: CADRE flight model rovers in their stowed and deployed configurations with key hardware components highlighted

6.2.2 DM experiment results

The rovers demonstrated that they can autonomously explore larger regions within environments representative of the expected operational areas of the lunar surface, while executing the CADRE flight software on hardware that is almost identical to the flight model. Throughout 15 experiments, the rovers explored a total of 2,724 m^2 in about 25 hours. Table 3 shows the exploration rate i.e., surface area covered per second for the experiments. The experiments in the table are grouped by the following categories:

- **Indoor**: [environment] Indoor arena

CADRE Development Model rover

Fig. 15: Close up view of CADRE DM rover

- **MY**: [environment] Mars Yard with sparsely-spaced rocks and a few craters
- **LunarMY**: [environment] Mars Yard with increased number of rocks, boulders, and craters to more accurately simulate the lunar surface, significant increase of difficulty compared to the MY category
- **W-S**: [feature] Autonomous planning with wake-sleep cycles
- **Night**: [environment] After sunset with nighttime illumination to simulate lunar-like lighting conditions
- **Taboo**: [feature] Taboo search feature described in Section 4.2.3

Note that we chose only to show the results of the experiments that completed at least 20 minutes of exploration, 13 of the 15 experiments.

As the level of complexity increased in the experiments (i.e., more difficult environment, autonomous planning, nighttime illumination), the exploration rate initially decreases. There is a notable decrease after the test environment changes to the lunar-simulated environment in the Mars Yard, as there are significantly more obstacles to navigate. After introducing the taboo list, the exploration rate goes on an upward trend, demonstrating the positive impact of the feature. By the last 5 experiments, the exploration rate was an average of 0.035 m^2 per second. At this rate, the rovers can explore a 20m by 20m region within 191 minutes, about seven 30-minute wake-sleep cycles.

Table 4 provides a more-detailed overview of selected experiments during the test campaign. For these experiments, the coefficients for the cost

CADRE Development Model (DM) rover exploration test instances

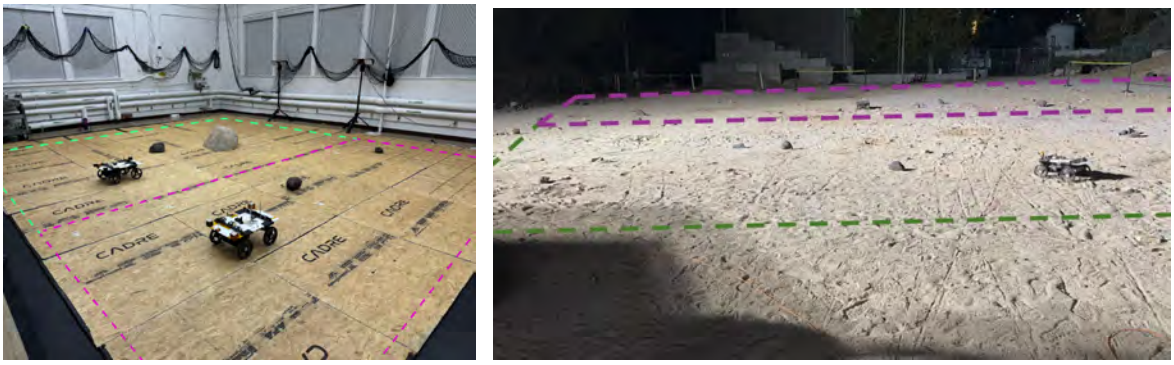

Fig. 16: Figures show CADRE DM rovers exploring JPL's indoor test arena to the left (Tests 1, 15) and outdoor Lunar-analog Mars Yard (Tests 2-14) in the right.

function of the frontier-point computation defined in equation (1) were set to $w_1 = 1.0$, $w_2 = 0.6$, and $w_3 = 0.5$. The metrics are defined as the following:

- **Explore efficiency** (m^2/m): $\frac{Explored\ Area}{Meters\ driven}$
- **Explore rate** (m^2/s): $\frac{Explored\ Area}{Duration}$
- **SMP (Surface Mobility Planner) efficiency** (s/m): $\frac{Meters\ driven}{Duration}$
- **SMP failures per** s: $\frac{SMP\ failures}{Duration}$
- **SMP failures per** m: $\frac{SMP\ failures}{Meters\ driven}$

Experiment 10 shows that the rovers are able to explore for longer periods of time at a sufficient exploration rate, with a duration of over 5.5 hours and exploration area of 491 m^2. Across the experiments that utilized the taboo search feature, the rovers were consistently able to explore over 1.3 m^2/m with the highest rate in experiment 13 at 2.4 m^2/m, demonstrating our exploration efficiency in lunar-like environments. Furthermore, test 13 also has the highest surface mobility planner failure rate of about 4 failures/meter and degraded planner efficiency of 70 seconds per meter, while test 12 has the worst exploration efficiency of 1.3 m^2/m, recording the lowest surface mobility planner failure rates of 0.4656 failures/meter and one of the best planner efficiency rates of 19 seconds per meter. This shows that the exploration efficiency is not sensitive to the efficiency of the planners, highlighting the robustness of exploration to surface mobility planner performance and failures. On the other hand, Table 4 indicates that increased rate of planner failures per meter results in poor planner efficiency and increased exploration rates, thus more power will be utilized when exploring.

Exploration supports an optional toggle to log specific data during execution. We implemented tools to ingest the data and analyze the results. Figure 17 highlights a set of products generated by our post-execution tool which visualizes the merged map with obstacles inflated, exploration region, agent location, and frontier clusters for each frontier point calculation. The graphic displays the progression of the explored area during test 10a at three states over time for each agent, wherein one rover explores the upper and the other explores the bottom region. Since we will not be able to physically observe the rovers during lunar surface operations in real-time, these tools will serve as a critical aid to evaluate exploration performance on the CADRE mission.

Overall, these experiments show that the proposed exploration architecture is able to effectively perform exploration on representative hardware in an environment representative of the Moon's Reiner Gamma — validating the design and implementation of the exploration algorithm, and clearing the way for deployment on the Lunar surface.

6.3 Lessons learned

These experiments in multi-robot exploration have yielded promising results while also highlighting several areas for improvement. The experiments confirmed that the selected processor

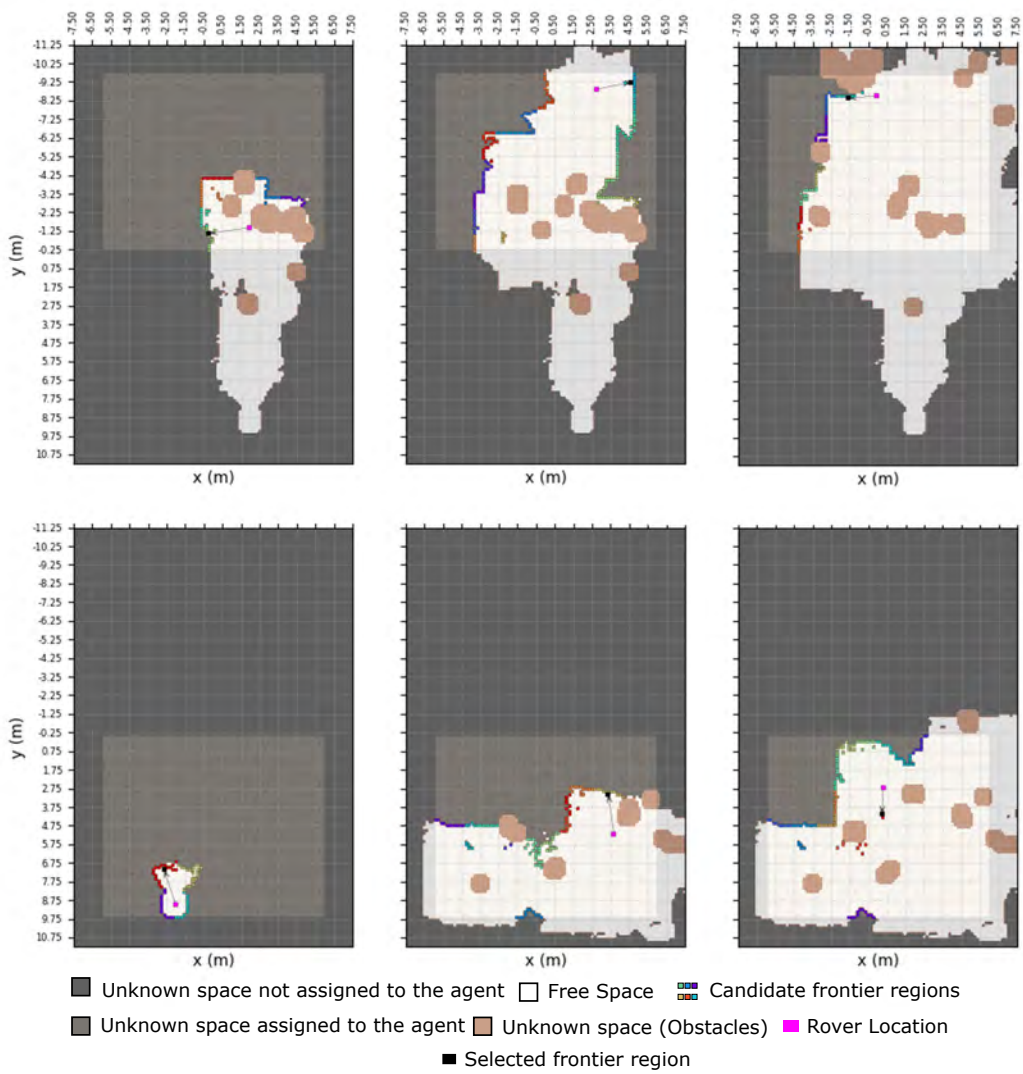

Fig. 17: A graphic displaying the progression of exploration in a lunar-simulated environment, which includes a map with inflated obstacles, region, agent locations, and frontier clusters for three frontier point calculations during Experiment 14 for two different agents, rover 1 (**top**) and rover 2 (**bottom**). Both rovers start adjacent to each other in the bottom half of the workspace.

and sensor choices are able to support multi-robot exploration operation. The experiments also allowed us to finalize algorithm choices for map-partition, agent exploration and our cost-function parameters.

We are sharing a few of the lessons learned from development and testing, as well as possible further improvements to our solutions:

Firstly, relying solely on visual odometry for localization can be insufficient, resulting in occasional increased localization errors, particularly when robots make turns during exploration. The on-board localization drift for a rover using visual-inertial-solar odometry is as low as 1.61% and as high as 3.67% (i.e., 3.67cm of drift per meter

Test No.	Exploration Area (m)	Indoor	MarsYard	LunarMY	W-S	Night	Taboo	Rate (m^2/s)
1	7.5 x 6.50	✓						0.0425
2,3	18.0 x 12.0		✓					0.0302
4	18.0 x 18.0		✓		✓			0.0276
5	18.0 x 18.0			✓		✓		0.0144
8	15.0 x 22.5			✓	✓			0.0135
9	15.0 x 22.5			✓			✓	0.0243
10a	15.0 x 22.5			✓		✓	✓	0.0232
10b	15.0 x 22.5			✓	✓	✓	✓	0.0317
11,13,14	15.0 x 22.5			✓	✓		✓	0.0306
12	15.0 x 22.5			✓	✓	✓	✓	0.0667
15	6.5 x 8.0	✓			✓		✓	0.0408

Table 3: Exploration rate (m^2/s) for development model hardware experiments.

Test No.	Explored area (m^2)	Meters driven (m)	Duration (s)	Explore efficiency (m^2/m)	Explore rate (m^2/s)	SMP efficiency (s/m)	SMP failures	SMP failures per s	SMP failures per m
8	23.084	12.6746	1710	1.8213	0.0135	134.9155	145	0.0848	11.4402
9	341.572	164.451	14046	2.0770	0.0243	85.4115	538	0.0383	3.2715
10a	339.074	220.868	14628	1.5352	0.0232	66.2296	231	0.0158	1.0459
10b	152.054	73.597	4803	2.0660	0.0317	65.2608	87	0.0181	1.1821
11	96.367	48.15	1182	2.0014	0.0815	24.5483	0	0.0000	0.0000
12	156.69	120.195	2349	1.3036	0.0667	19.5432	56	0.0238	0.4659
13	199.844	82.723	5803	2.4158	0.0344	70.1502	329	0.0567	3.9772
14	224.909	109.774	10045	2.0488	0.0224	91.5059	427	0.0425	3.8898

Table 4: Results from selected DM experiments, each with a test area of 15.0m x 22.5m. A key performance metric, the exploration efficiency (m^2/m), is satisfied across varying conditions and surface mobility planner (SMP) failure rates.

driven) in recent ground-truth instrumented localization experiments. To improve positional accuracy, we have integrated visual-inertial-solar (i.e., using a sun sensor as a compass) measurements and Ultra-wideband (UWB) ranging [59], enabling more precise inter-robot localization through pose graph optimization. This approach has been demonstrated to reduce localization error by up to 30% for two rovers and a base station. Further improvements could be made to trade localization performance for efficiency, such as coordinating visual loop closures or inducing favorable sensing geometries (e.g., for the UWB ranging) as has been demonstrated in [71].

Secondly, the presence of craters that show up as unclearable unknown space necessitated the addition of the taboo list. Since the system lacks a crater detector, this approach is merely a heuristic that needs to be carefully tuned to ensure that craters are avoided, but exploration progress is still made.

Thirdly, frontier search typically starts at the robot position with the assumption that a rover is within its exploration region. If a rover has to leave this region, then by default it will try to reach the closest frontier; however, this strategy may not achieve the intended effect of leading a rover back into its own region (and out of a neighbor region, for example). Other solutions could include requiring a rover to first reach the centroid of its exploration region; however, this strategy has not been tested yet.

Lastly, rovers may observe each other and mark another rovers an obstacle in local maps that are later shared. To avoid rovers showing up as obstacles in the merged map, special care is taken to clear rover footprints at known locations of rovers. It was insufficient to clear footprints locally due to the centralized map merging policy that has no knowledge of rover footprints (i.e., it operates on raw free, occupied, or unknown map cells).We refrain from using an image-based detection algorithm for clearing footprint to reduce computational complexity, and to prevent potential misidentifications caused by varying lighting conditions and occlusions.

7 Conclusion

In conclusion, we introduced a semi-centralized, map-partition-based multi-robot exploration algorithm designed for multi-robot planetary missions. Intended for deployment in the CADRE technology demonstration mission, it can be adapted for broader planetary exploration. The algorithm features a leader agent (potentially elected) responsible for partitioning the unexplored area into non-overlapping regions, which are then assigned to individual agents. Following this allocation, each agent performs independent exploration within its designated region, requiring no ongoing coordination with the leader agent or other agents. This approach improves computational resource distribution, reduces redundant coverage, minimizes inter-agent communication, and decreases inter-robot collisions.

We evaluated the algorithm through two sets of hardware experiments and numerical simulations. Simulations showed our approach outperforms both fully centralized and semi-centralized algorithms with random frontier selection in terms of exploration time, average distance traveled, and maximum distance traveled. The first set of hardware experiments, conducted in NASA JPL's Moon Yard with prototype rovers, demonstrated resilience in navigating challenging environments and handling failure scenarios where regions were dynamically redistributed among operational robots.

In the second set, we used development model (DM) rovers closely resembling flight models to evaluate the algorithm's performance integrated into CADRE's flight software stack. These high-fidelity experiments, conducted in both an indoor arena and the Mars Yard, validated the algorithm's performance under mission-representative conditions, including lunar-like lighting challenges. Additionally, we introduced a "taboo list" feature in the frontier selection algorithm to exclude repeatedly unreachable frontier points, ensuring efficient exploration and recovery from temporary issues.

Future work will focus on optimizing region allocation and enhancing exploration efficiency. We plan to develop dynamic partitioning strategies that adapt to environmental changes and improve workload balance. Exploration efficiency will be further improved by integrating advanced cost functions for frontier selection, optimizing robot trajectories, and implementing communication-aware inter-agent coordination for collaborative decision-making. These enhancements will further strengthen the algorithm's robustness, enabling more efficient and reliable multi-robot exploration in planetary missions.

8 Declarations

Acknowledgements

The research was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

Author Contribution

Sharan Nayak, the co-primary author, authored the initial draft and conducted simulation and hardware experiments on Mercury-7 rovers. Grace Lim, the co-primary author, integrated the proposed algorithms in CADRE's flight software stack, designed and implemented taboo search, and conducted all experiments on the DMs. Federico Rossi served as an advisor to Sharan at JPL, correcting and improving the draft, and assisting in setting up simulations and hardware experiments. Michael Otte served as Sharan's Ph.D. advisor at UMD, initially funding Sharan's time at JPL as a visiting student researcher. He played a role in improving the draft and provided expertise in multi-robot exploration. JP De-La Croix, the primary investigator for CADRE, funded the

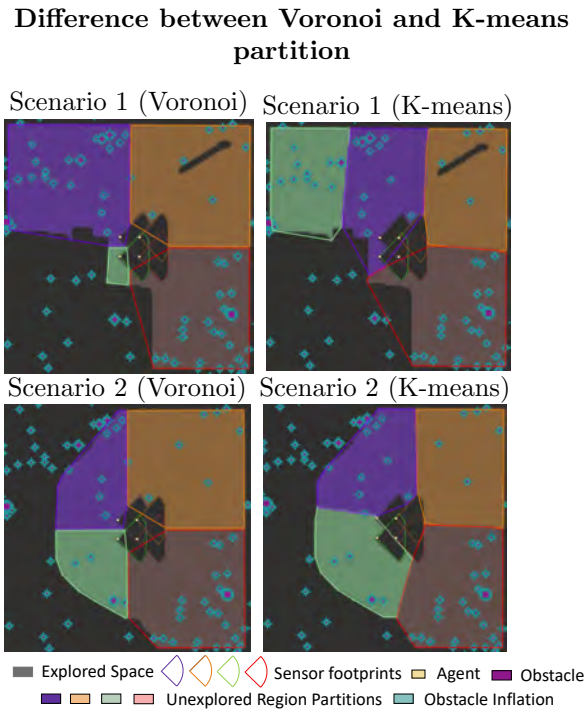

Fig. 18: The top figures show the Voronoi partition, and the bottom figures show the K-means partition for scenarios 1 and 2. K-means generates more uniform-sized partitions for the same robots' initial locations compared to the Voronoi method.

experiments, contributed to drafting the paper, and offered expertise in multi-robot exploration.

Funding information

Sharan Nayak, Grace Lim, Federico Rossi, and JP De-La Croix were funded by NASA JPL under the contract 80NM0018D0004. Sharan Nayak was also funded by Minta Martin Fellowship Fund, UMD for initial part of the project. Michael Otte was funded by Minta Martin Fellowship Fund, UMD.

Ethical Statements

Not Applicable

Data Availability

Not Applicable

A Appendix

Comparison between Voronoi and K-means partitions:

Voronoi-based partition

This method [47] generates a partition based on the set of closest points in the map to each robot. The robot locations are the generator points for the Voronoi regions. This method produces even-sized partitions when the robots are symmetrically placed with respect to the unexplored cells. However, it produces uneven-sized regions if the robots' initial locations are asymmetrically located across the unexplored space (Fig. 18).

K-means based partition

This method [48] generates a partition by iteratively reducing the user-defined "distance" between the data points and the centroids of the intermediate calculated partitions. The stopping criterion for K-means varies, but the commonly used ones are a user-defined maximum number of iterations, or when the centroids stop moving beyond a given threshold distance. The performance of K-means partitioning is sensitive to the centroid initialization, and numerous works have explored techniques for its proper initialization [72–74]. The K-means method produces a more uniform partition of regions for the same input points as compared to the Voronoi method for the scenarios shown in (Fig. 18).

References

[1] Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 146–151 (1997). IEEE

[2] Yamauchi, B.: Frontier-based exploration using multiple robots. In: Proceedings of the Second International Conference on Autonomous Agents, pp. 47–53. Association for Computing Machinery, Minneapolis, Minnesota, USA (1998)

[3] Yliniemi, L., Agogino, A.K., Tumer, K.: Multirobot coordination for space exploration. AI Magazine **35**(4), 61–74 (2014)

[4] Marjovi, A., Nunes, J.G., Marques, L., De Almeida, A.: Multi-robot exploration and

fire searching. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1929–1934 (2009). IEEE

[5] Croix, J.-P., Rossi, F., Brockers, R., Aguilar, D., Albee, K., Boroson, E., Cauligi, A., Delaune, J., Hewitt, R., Kogan, D., Lim, G., Morrell, B., Nakka, Y., Nguyen, V., Proença, P., Rabideau, G., Russino, J., Silva, M.S., Zohar, G., Comandur, S.: Multi-agent autonomy for space exploration on the CADRE lunar technology demonstration. In: IEEE Aerospace Conference, Big Sky, MT (2024). IEEE

[6] NASA: Cooperative Autonomous Distributed Robotic Explorers (CADRE). https://www.nasa.gov/directorates/spacetech/game_changing_development/projects/CADRE (2023)

[7] Garrick-Bethell, I., Kelley, M.R.: Reiner gamma: A magnetized elliptical disk on the moon. Geophysical Research Letters **46**(10), 5065–5074 (2019)

[8] Solanas, A., Garcia, M.A.: Coordinated multi-robot exploration through unsupervised clustering of unknown space. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, pp. 717–721 (2004). IEEE

[9] Wurm, K.M., Stachniss, C., Burgard, W.: Coordinated multi-robot exploration using a segmentation of the environment. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1160–1165 (2008). IEEE

[10] Lopez-Perez, J.J., Hernandez-Belmonte, U.H., Ramirez-Paredes, J.-P., Contreras-Cruz, M.A., Ayala-Ramirez, V.: Distributed multi-robot exploration based on scene partitioning and frontier selection. Mathematical Problems in Engineering (2018)

[11] NASA: NASA selects Intuitive Machines for lunar science delivery. https://www.nasa.gov/press-release/nasa-selects-intuitive-machines-for-new_lunar-science-delivery (2022)

[12] Bhattacharya, S., Michael, N., Kumar, V.: Distributed coverage and exploration in unknown non-convex environments. Distributed Autonomous Robotic Systems: The 10th International Symposium, 61–75 (2013)

[13] Corah, M., Michael, N.: Efficient online multi-robot exploration via distributed sequential greedy assignment. In: Robotics: Science and Systems, vol. 13 (2017)

[14] Renzaglia, A., Martinelli, A.: Potential field based approach for coordinate exploration with a multi-robot team. In: 2010 IEEE Safety Security and Rescue Robotics, pp. 1–6 (2010). IEEE

[15] Thomas, M., Joy, A.T.: 2. Entropy, Relative Entropy, and Mutual Information, pp. 13–55. John Wiley & Sons, USA (2006)

[16] Khatib, O.: The potential field approach and operational space formulation in robot control. In: Adaptive and Learning Systems, pp. 367–377 (1986). Springer

[17] Jain, U., Tiwari, R., Godfrey, W.W.: Comparative study of frontier based exploration methods. In: 2017 Conference on Information and Communication Technology, pp. 1–5 (2017). IEEE

[18] Khawaldah, M., Al-Khedher, M., Al-Adwan, I., Al Rawashdeh, A.: An autonomous exploration strategy for cooperative mobile robots. Journal of Software Engineering and Applications **7**(3), 142–149 (2014)

[19] Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Transactions on Robotics **21**(3), 376–386 (2005)

[20] De Hoog, J., Cameron, S., Visser, A., *et al.*: Autonomous multi-robot exploration in communication-limited environments. In: Proceedings of the Conference on Towards Autonomous Robotic Systems, pp. 68–75 (2010). Citeseer

[21] Sun, Y., Zhang, C.: Efficient and safe robotic autonomous environment exploration using

integrated frontier detection and multiple path evaluation. Remote Sensing **13**(23), 4881 (2021)

[22] Cordes, F., Bindel, D., Lange, C., Kirchner, F.: Towards a modular reconfigurable heterogenous multi-robot exploration system. In: Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space, pp. 38–45 (2010)

[23] Deng, D., Duan, R., Liu, J., Sheng, K., Shimada, K.: Robotic exploration of unknown 2d environment using a frontier-based automatic-differentiable information gain measure. In: 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 1497–1503 (2020). IEEE

[24] Keidar, M., Kaminka, G.A.: Efficient frontier detection for robot exploration. The International Journal of Robotics Research **33**(2), 215–236 (2014)

[25] Quin, P., Alempijevic, A., Paul, G., Liu, D.: Expanding wavefront frontier detection: An approach for efficiently detecting frontier cells. In: Australian Conference on Robotics and Automation (2014)

[26] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms, (2022). MIT press

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

[28] Sun, Z., Wu, B., Xu, C., Kong, H.: Adadetector: Adaptive frontier detector for rapid exploration. In: 2022 International Conference on Robotics and Automation, pp. 3706–3712 (2022). IEEE

[29] LaValle, S.M., Kuffner Jr, J.J.: Randomized kinodynamic planning. The International Journal of Robotics Research **20**(5), 378–400 (2001)

[30] Silva Lubanco, D.L., Pichler-Scheder, M., Schlechter, T., Scherhäufl, M., Kastl, C.: A review of utility and cost functions used in frontier-based exploration algorithms. In: 2020 International Conference on Robotics and Automation Engineering, pp. 187–191 (2020). IEEE

[31] Gomez, C., Hernandez, A.C., Barber, R.: Topological frontier-based exploration and map-building using semantic information. Sensors **19**(20), 4595 (2019)

[32] Colares, R.G., Chaimowicz, L.: 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, pp. 268–274 (2016)

[33] Banfi, J., Quattrini Li, A., Rekleitis, I., Amigoni, F., Basilico, N.: Strategies for coordinated multi-robot exploration with recurrent connectivity constraints. Autonomous Robots **42**(4), 875–894 (2018)

[34] Bautin, A., Simonin, O., Charpillet, F.: Minpos: A novel frontier allocation algorithm for multi-robot exploration. In: International Conference on Intelligent Robotics and Applications, pp. 496–508 (2012). Springer

[35] Zlot, R.M., Stentz, A.T., Dias, M.B., Thayer, S.: Market-driven multi-robot exploration. Technical Report CMU-RI-TR-02-02, Carnegie Mellon University, Pittsburgh, PA (January 2002)

[36] Xiong, G., Gong, J., Chen, H., Su, Z.: Multi-robot exploration based on market approach and immune optimizing strategy. In: Third International Conference on Autonomic and Autonomous Systems (ICAS'07), pp. 29–29 (2007). IEEE

[37] De Hoog, J., Cameron, S., Visser, A.: Role-based autonomous multi-robot exploration. In: 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, pp. 482–487 (2009). IEEE

[38] Wu, L., García García, M.Á., Puig Valls, D., Solé Ribalta, A.: Voronoi-based space partitioning for coordinated multi-robot exploration. Journal of Physical Agents **1** (2007)

[39] Albee, K., Bhamidipati, S., Rossi, F., Vander Hook, J., Croix, J.-P.: Lunar leader: Persistent, optimal leader election for multi-agent exploration teams. In: Int. Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace), Auckland, NZ (2024)

[40] Amigoni, F., Banfi, J., Basilico, N.: Multirobot exploration of communication-restricted environments: A survey. IEEE Intelligent Systems **32**(6), 48–57 (2017)

[41] Hollinger, G.A., Singh, S.: Multirobot coordination with periodic connectivity: Theory and experiments. IEEE Transactions on Robotics **28**(4), 967–973 (2012)

[42] Cordes, F., Ahrns, I., Bartsch, S., Birnschein, T., Dettmann, A., Estable, S., Haase, S., Hilljegerdes, J., Koebel, D., Planthaber, S., *et al.*: Lunares: Lunar crater exploration with heterogeneous multi robot systems. Intelligent Service Robotics **4**, 61–89 (2011)

[43] Martinez Rocamora Jr, B., Kilic, C., Tatsch, C., Pereira, G.A., Gross, J.N.: Multi-robot cooperation for lunar in-situ resource utilization. Frontiers in Robotics and AI **10**, 1149080 (2023)

[44] Thomas, G., Howard, A.M., Williams, A.B., Moore-Alston, A.: Multirobot task allocation in lunar mission construction scenarios. In: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 1, pp. 518–523 (2005). IEEE

[45] Mankins, J.C.: technology readiness levels. Technical report, Advanced Concepts Office, Office of Space Access and Technology, NASA (1995)

[46] Nayak, S.: fast feasible motion planning without two-point boundary value solution. PhD thesis, University of Maryland, College Park (2023)

[47] Aurenhammer, F.: Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Computing Surveys **23**(3), 345–405 (1991)

[48] Lloyd, S.: least squares quantization in pcm. IEEE Transactions on Information Theory **28**(2), 129–137 (1982)

[49] Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1-2), 83–97 (1955)

[50] Lamrous, S., Taileb, M.: Divisive hierarchical k-means. In: 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce, pp. 18–18 (2006). IEEE

[51] Haines, E.: Point in polygon strategies. Graphics Gems **4**, 24–46 (1994)

[52] Shimrat, M.: Algorithm 112: position of point relative to polygon. Communications of the ACM **5**(8), 434 (1962)

[53] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)

[54] Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)

[55] Feurer, M., Hutter, F.: Hyperparameter optimization. Automated machine learning: Methods, systems, challenges, 3–33 (2019)

[56] Zheng, K.: ROS navigation tuning guide, pp. 197–226 (2021). Springer

[57] Rosmann, C., Feiten, W., Wösch, T., Hoffmann, F., Bertram, T.: Efficient trajectory optimization using a sparse model. In: 2013 European Conference on Mobile Robots, pp. 138–143 (2013). IEEE

[58] Golombek, M., Rapp, D.: Size-frequency distributions of rocks on mars and earth analog sites: Implications for future landed missions. Journal of Geophysical Research: Planets **102**(E2), 4117–4129 (1997)

[59] Boroson, E.R., Hewitt, R., Ayanian, N., Croix, J.-P.: Inter-robot range measurements in pose graph optimization. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4806–4813 (2020). IEEE

[60] Hörner, J.: Map-merging for multi-robot system. Bachelor's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Prague (2016). https://is.cuni.cz/webapps/zzp/detail/174125/

[61] Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, Berlin, Heidelberg (2010)

[62] Datta, A., Kim, J.-S., Kanade, T.: Accurate camera calibration using iterative refinement of control points. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, pp. 1201–1208 (2009). IEEE

[63] Rehder, J., Nikolic, J., Schneider, T., Hinzmann, T., Siegwart, R.: Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In: 2016 IEEE International Conference on Robotics and Automation, pp. 4304–4311 (2016). IEEE

[64] Rusu, R.B., Cousins, S.: 3d is here: Point cloud library (pcl). In: 2011 IEEE International Conference on Robotics and Automation, pp. 1–4 (2011). IEEE

[65] Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry. In: 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, vol. 1, p. (2004). IEEE

[66] Fankhauser, P., Bloesch, M., Hutter, M.: Probabilistic terrain mapping for mobile robots with uncertain localization. IEEE Robotics and Automation Letters **3**(4), 3019–3026 (2018)

[67] Wermelinger, M., Fankhauser, P., Diethelm, R., Krüsi, P., Siegwart, R., Hutter, M.: Navigation planning for legged robots in challenging terrain. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1184–1189 (2016). IEEE

[68] Zhi, L., Xuesong, M.: Navigation and control system of mobile robot based on ros. In: 2018 IEEE Advanced Information Technology, Electronic and Automation Control Conference, pp. 368–372 (2018). IEEE

[69] Saboia, M., Rossi, F., Nguyen, V., Lim, G., Aguilar, D., Croix, J.-P.: Cadre moondb: Distributed database for multi-robot information-sharing and map-merging for lunar exploration. In: Int. Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace), Auckland, NZ (2024). https://www.federico.io/pdf/Saboia.Rossi.ea.MASSPACE24.pdf

[70] Bocchino, R., Canham, T., Watney, G., Reder, L., Levison, J.: F prime: an open-source framework for small-scale flight software systems. In: Proceedings of the AIAA/USU Conference on Small Satellites, vol. SSC-18-XII-04 (2018)

[71] Knowles, D.: Leveraging relative ranging geometry for fault detection and multi-robot coordination. Phd thesis, Stanford University (2024). Available at https://searchworks.stanford.edu/view/in00000106620

[72] Pena, J.M., Lozano, J.A., Larranaga, P.: An empirical comparison of four initialization methods for the k-means algorithm. Pattern Recognition Letters **20**(10), 1027–1040 (1999)

[73] Fränti, P., Sieranoja, S.: How much can k-means be improved by using better initialization and repeats? Pattern Recognition **93**, 95–112 (2019)

[74] Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Systems with Applications **40**(1), 200–210 (2013)

Sharan Nayak received his Bachelor of Engineering (B.E.) degree in electronics and communications engineering from Visveswaraya Technological University, India, in 2009; his M.S. degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, GA, USA, in 2011 and M.S. degree in aerospace engineering from University of Maryland (UMD), College Park, MD in 2020, and the Ph.D. degree in Aerospace Engineering from UMD in 2023. His research interests are in motion planning of single and multi-agent autonomous systems. He was a visiting student researcher at NASA JPL and worked on various projects related to multi-robot exploration and motion planning.

Grace Lim received her B.S. in Mathematics from California State Polytechnic University, Pomona, and M.S in Computer Science from Georgia Institute of Technology. She is currently a Robotics Systems Engineer in the Maritime and Multi-Agent Autonomy group at Jet Propulsion Laboratory, Pasadena, CA.

Federico Rossi received the M.Sc. degree in space engineering from Politecnico di Milano, Milan, Italy, in 2013, and the Ph.D. degree in aeronautics and astronautics from Stanford University, Stanford, CA, USA, in 2018. He is currently a Robotics Technologist with Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA, where he leads the planning, scheduling, and execution team for CADRE, an upcoming, NASA-funded Lunar multi-agent autonomy technology demonstration. His research focuses on optimal control and distributed decision-making in multiagent robotic systems, with applications to robotic planetary exploration.

Michael Otte (M'07) received the B.S. degrees in aeronautical engineering and computer science from Clarkson University, Potsdam, New York, USA, in 2005, and the M.S. and Ph.D. degrees in computer science from the University of Colorado Boulder, Boulder, CO, USA, in 2007 and 2011, respectively. From 2011 to 2014, he was a Postdoctoral Associate at the Massachusetts Institute of Technology. From 2014 to 2015, he was a Visiting Scholar at the U.S. Air Force Research Lab. From 2016 to 2018, he was a National Research Council RAP Postdoctoral Associate at the U.S. Naval Research Lab. He has been with the Department of Aerospace Engineering, at the University of Maryland, College Park, MD, USA, since 2018. He is the author of over 50 articles, and his research interests include autonomous robotics, motion planning, and multi-agent systems.

Jean-Pierre de la Croix is a Robotics Systems Engineer in the Maritime and Multi-Agent Autonomy group. He joined JPL after completing a Ph.D. in Electrical & Computer Engineering at the Georgia Institute of Technology in 2015. His research focused on new control techniques for large-scale robotic systems, such that humans can easily and effectively interact with these complex systems. At JPL, he continues to work on multi-agent robotics for new and challenging applications. He is the principal investigator of CADRE, an upcoming, NASA-funded Lunar multi-agent autonomy technology demonstration.