Multi-Agent Autonomy for Space Exploration on the CADRE Lunar Technology Demonstration

Jean-Pierre de la Croix, Federico Rossi, Roland Brockers, Dustin Aguilar, Keenan Albee, Elizabeth Boroson, Abhishek Cauligi, Jeff Delaune, Robert Hewitt, Dima Kogan, Grace Lim, Benjamin Morrell, Yashwanth Nakka, Viet Nguyen, Pedro Proença, Gregg Rabideau, Joseph Russino, Maira Saboia da Silva, Guy Zohar and Subha Comandur
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
{first.last}@jpl.nasa.gov

Abstract—**CADRE (Cooperative Autonomous Distributed Robotic Exploration) is a lunar technology demonstration mission of multi-agent autonomy on a team of three rovers and a base station. The mission is slated land at the Moon's Reiner Gamma region as a CLPS (Commercial Lunar Payload Services) payload on the IM-3 mission in 2024. The goal of CADRE is to demonstrate how a team of autonomous rovers, receiving only high-level tasks from Earth, can autonomously explore a region of the Lunar surface, as well as perform a distributed measurement in coordination with a multi-static ground-penetrating radar. We envision that multi-agent autonomy will enable future missions to address hitherto-unanswered questions in planetary science on the Moon, Mars, and beyond. In this paper, we describe the autonomy architecture developed for CADRE, both for multi-agent coordination, and for single-agent driving surface mobility, and discuss the requirements and constraints that led to the selection of this architecture.**

TABLE OF CONTENTS

1. INTRODUCTION

Multi-robot systems offer great promise for planetary exploration, offering the unique ability to collect *distributed measurements* of phenomena of interest from multiple vantage points at nearly simultaneous times. Such distributed measurements are critical to address a number of unresolved questions in planetary science, ranging from subsurface composition, which can be studied through multi-static ground-penetrating radar and seismic surveys, to atmospheric circulation and the origin of trace gas emissions, which can be investigated through networked weather stations and laser spectrometers.

Mobile robotic platforms are a promising platform to collect distributed measurements; however, operating multiple, closely-cooperating robots through traditional ground-in-the-loop operations cycles is highly challenging, which has motivated significant research in the field of autonomous operations for cooperative robotic explorers.

The Cooperative Autonomous Distributed Robotic Exploration (CADRE) mission, funded by NASA's Game-Changing Development program, aims to provide a technology demonstration platform for autonomy technologies for multi-robot scientific explorers. CADRE plans to operate three autonomous robots at the Moon's Reiner Gamma region in 2024. The robots will first autonomously explore a region near the landing site, and then autonomously perform a multi-static ground-penetrating radar (GPR) survey of the region, with minimal ground intervention, paving the way for future Solar System exploration with coordinated robotic explorers.

In this paper, we will present CADRE's novel autonomy solution, and discuss the key requirements driving its design. The goal of CADRE's autonomy is to translate high-level commands from ground operators, e.g., "explore this region" and "perform a multi-static ground penetrating radar survey of this area", into commands for the rovers' individual surface navigation stacks, i.e., "go to this location by this time". Multi-agent autonomy must account for the rovers' limited resources, specifically thermal capacity and power level; and ensure that the system presents no single point of failure whose loss could jeopardize the system. CADRE's multi-agent autonomy has six key components described below:

1. A leader election module ensures that the team has one leader at all times. The leader may be one of the rovers or a base station, located on the lander. The leader mediates all coordination between the agents; non-leader agents never communicate with each other directly. If the leader is lost, another agent is immediately elected in its stead; the leader is selected based on the agents' available power and thermal capacity.
2. A strategic planner running on the leader, decides which activities should be collectively performed by the agents based on their thermal state and available power. The planner produces a sequence for each rover, including tasks such as "explore" and "collect a multi-static measurement".
3. A pair of team planners, also located on the leader, elaborate the strategic planner's high-level commands into commands for the individual agents.
4. For distributed measurements, a sampling-based motion planner is used to compute trajectories for all robots. The trajectories are obstacle-free, and steer the rovers towards satisfaction of inter-rover separation constraints required to

979-8-3503-0462-6/24/$31.00 ©2024 IEEE

perform the GPR survey. For exploration, a divide-and-conquer approach is adopted: the region to explore is divided in several subregions, one per available rover.
5. For the exploration activity, an agent planner further elaborates the team planners' instructions. The module receives as input the subregion that the rover should explore; it selects the next location that the rover should drive to with a frontier-based heuristic. Exploration ends when the subregion is sufficiently mapped.
6. The coordination functions performed by the leader require knowledge of the rovers' location, temperature, and power states, and of the traversability maps computed by individual robots. A shared state database synchronizes relevant information from all rovers to the current leader. The database also backs up all information to a designated survivor to ensure that no data is lost in case of a rover failure.

CADRE's multi-agent autonomy is enabled by a robust surface mobility autonomy stack that is capable of autonomous rover navigation, as well as, by the F' (or FPrime) flight software providing access to hardware sensors and other data relied upon by autonomy.)

2. Mission and System Overview

CADRE (Cooperative Autonomous Distributed Robotic Exploration) is a technology development project funded by the NASA STMD (Space Technology Mission Directorate) Game Changing Development (GCD) program. The objective is to further mature and demonstrate multi-agent technologies developed by NASA's Jet Propulsion Laboratory (JPL) under GCD's Autonomous PUFFER project as a flight system on the moon. CADRE's technology demonstration on the moon will demonstrate that a team of autonomous rovers can navigate and explore the lunar daytime environment with minimal human direction from the ground.

Technology Demonstration

CADRE is manifested on Intuitive Machines' (IM) IM-3 mission to an equatorial landing spot near Reiner Gamma. Reiner Gamma is Earth facing and is known for its lunar swirls–light and dark regolith mixing on the surface. CADRE is one of several payloads aboard IM's Nova-C lander. The IM-3 (and CADRE) mission will last most of the Lunar day (roughly 10 out of the 14 Earth days). The lander provides landing to lunar surface, as well as, power and communication for its payloads. CADRE will use its communication capabilities to transmit telemetry to JPL's ground data systems (GDS) throught the mission. After a ground operated deployment and commissioning of three rovers, CADRE will perform a series of experiments targeted at demonstrating various multi-agent capabilities:

- A threshold drive with the team in a specific formation, while capturing maps from on-board all rovers using their stereo cameras.
- A threshold operation of two ground penetrating radars (GPR) in a bi-static configuration.
- A baseline three-rover exploration task to fully, cooperatively map a specified region nearby the lander.
- A baseline three-rover multi-static GPR survey over tens of meters.
- A baseline resiliency experiment by disabling autonomy of specific rovers (or base station) to demonstrate autonomy's resiliency to agent loss.

Any additional mission time can be devoted to expanding on the baseline experiments.

System (Hardware) Overview

The CADRE system consists of several different components to enable the lunar technology demonstration–three (3) lunar rovers, one (1) base station, a situational awareness camera assembly (SACA), and deployers for each rover as shown in Figure 1. The deployers and SACA are mounted permanently to the lander, and so is the base station. The base station is similar in terms of its compute and wireless communication capabilities as the rovers, but lacks sensors and mobility. It is, however, hardwired for communication and power to the lander, and thus plays a key role in downlink and uplink to the system.

Rovers—The rovers shown in Figure 2 are small and light, roughly like a carry-on luggage, at less than 10kg and roughly $0.75 \times 0.5 \times 0.2\ m^3$ in volume with solar panels deployed. Solar panels are initially folded on top of the upward facing radiator, but then permanently deployed once the rovers are teleoperated to their initial "sun-bathing" locations nearby the lander after landing. The radiator is for temperature regulation, allowing the rovers to dissipate heat towards space. Each rover has four fixed wheels (skid-steer driving) with a variety of autonomy-specific sensors, such as dual stereo cameras, an inertial measurement unit (IMU), a sun sensor as a compass, and an ultra-wideband ranging radio for inter-rover distance estimation. All these sensors are used together to allow the rovers to estimate their motion in the lunar environment, as well as, understand the lunar environment itself as described in more detail in 4.

Compute is provided by ModalAI's VOXL, a Qualcomm Snapdragon 821 development board, with a quad core processor with integrated GPU. CADRE's flight software is implemented in C++ using F' [1] and takes advantage of not only the unique high and low-speed cores of the CPU, but also the GPU via OpenCL [2]. A rootfs is built using ModalAI's tools and loaded with a Debian `bookworm` chroot. This setup is consistent with CADRE's development machines, which allows the use of multi-arch and cross-building support to manage standard environments across the different prototyping and flight environments.

Each rover is equipped with a ground penetrating radar (GPR), which is self-contained hardware and software to perform a multi-static measurement with the GPRs on-board the other rovers. The GPRs antenna is mounted at the bottom front of the rover facing towards the ground with enough clearance to allow the rover to maneuver the lunar surface safely.

A key requirement for multi-agent systems is the ability to communicate with other agents. All CADRE rovers are equipped with a mesh network radios (MNR) to allow high-bandwidth communication directly or via multiple hops if necessary. The base station also has such a radio, consequently, downlink is possible from any agent to the base station and then down to the ground through the lander.

Key Constraints

Since CADRE's lunar technology demonstration is during the lunar daytime and near equatorial, the primary constraint are thermal limits. At peak times, the lunar surface is as hot as 120 °C, which is well above operating temperature of most consumer electronics used on CADRE. Consequently, most of the hardware is directly bolted to the radiator (which had

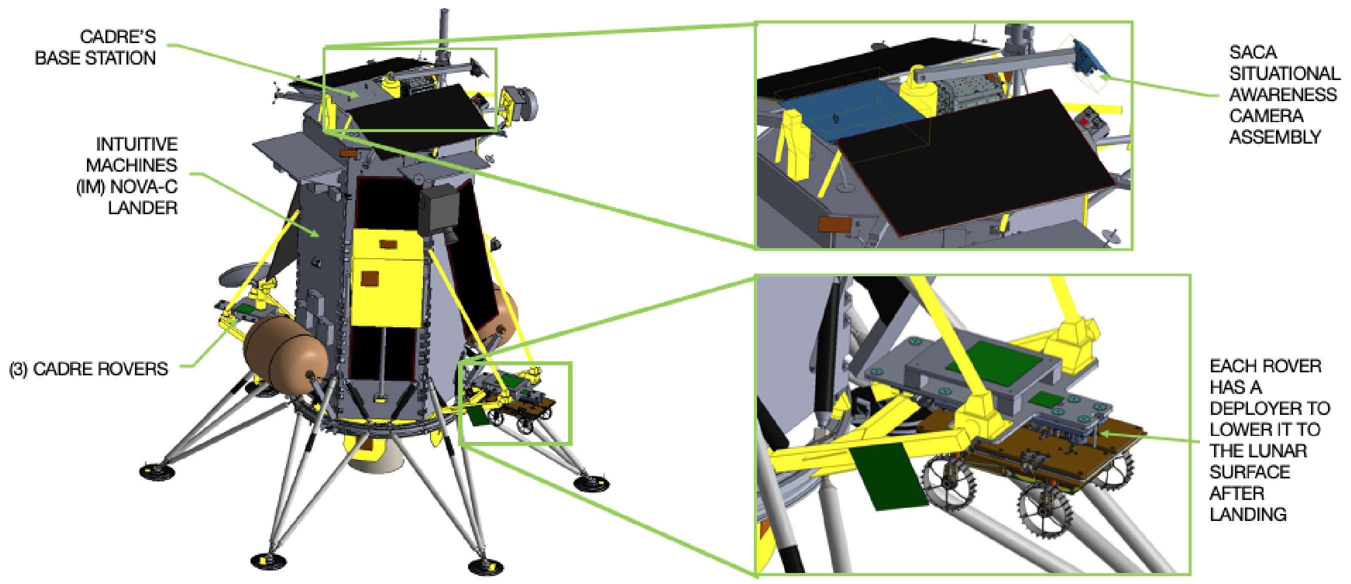

CADRE'S BASE STATION
INTUITIVE MACHINES (IM) NOVA-C LANDER
(3) CADRE ROVERS
SACA SITUATIONAL AWARENESS CAMERA ASSEMBLY
EACH ROVER HAS A DEPLOYER TO LOWER IT TO THE LUNAR SURFACE AFTER LANDING

Figure 1. Rendering of the CADRE system as it could be integrated with Intuitive Machine's lander.

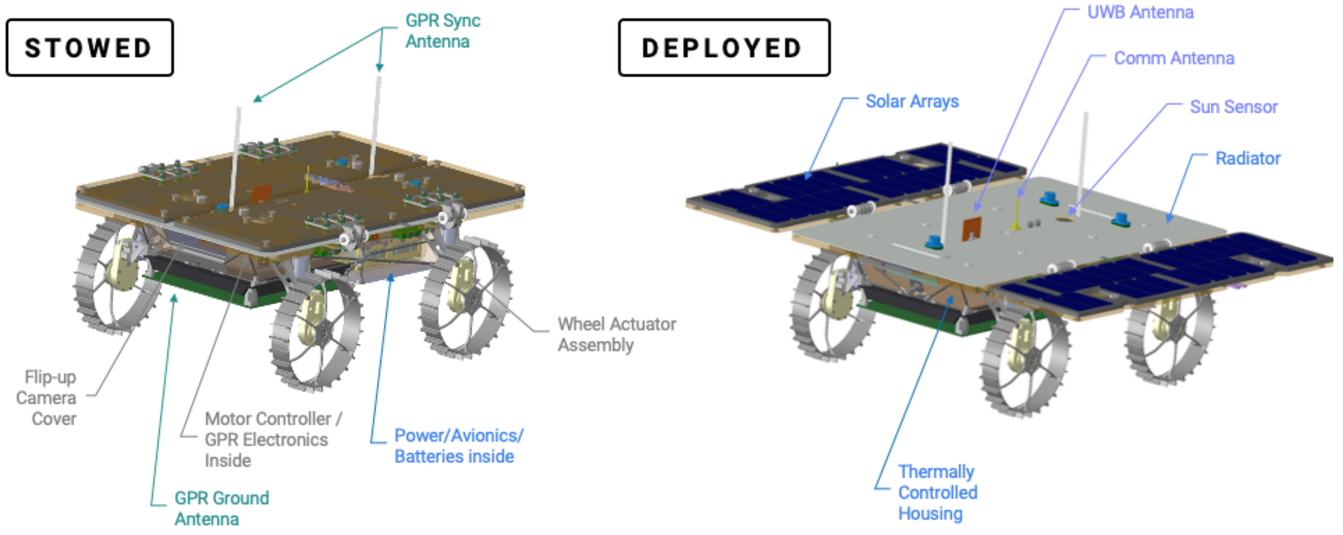


Figure 2. The rovers are illustrated in their stowed and (permanently) deployed configurations with key hardware components highlighted.

to be sufficiently sized and thus constrained the number of rovers that could be accommodated in the total payload envelope allocated to CADRE). The rovers are also operated on a 30 minute cycle, which means that Y minutes are assigned for cooling down and recharging, while $30 - Y$ minutes are available for bootup and operation. During the sleep cycle, the VOXL is completely shutdown and only reawoken by the FPGA at the 30 minute mark. Consequently, autonomy has to be capable of safely shutting down and resuming work.

Another key constraint is network bandwidth. While the MNRs (Mesh Network Radios) can allow for Mbits of communication, downlink is limited to kbits. Consequently, the rovers (and GPR) can easily transmit far more data than what can be reasonably downlinked. Ground operations has to be calculated to downlink the most important data to make timely adjustments to the system.

Although compute is more powerful than the typical $RAD750$ and slightly more powerful than Ingenuity's (the Mars Helicopter's) Qualcomm 801, computational resources in terms of CPU speed and available memory is still limited given the increased demands of autonomy in this system. Some algorithms had to be moved over to the GPU, while sensor data collection is assigned to the FPGA. The flight software is carefully allocated to specific cores to minimize resource contention.

The CADRE rovers are also much smaller than typical rovers, such that their sensors are close to the ground vs. high up on a mast. Consequently, CADRE rovers have shorter perception distances and features image-to-image move quickly necessitating a higher image processing rate.

However, at least some of these constraints can be overcome by multi-agent systems by working together to build up larger maps to plan on, to aggregate and summarize data products, to allocate compute to the least resource constrained agent, and other such cooperative capabilities.

3. Multi-Agent Autonomy

A number of multi-agent autonomy architectures were considered for CADRE. The trade study included centralized architectures where one agent makes all decisions requiring coordination between multiple assets; decentralized but globally-coordinated architectures where agents exchange messages to agree on the next course of action, either by sharing key sensor readings and computing the next course of action for all agents deterministically, in a shared-world fashion, or by bidding on activities that they should perform with an auction-based approach; and local-coordination approaches where each agent executes simple actions based on its and its immediate neighbors' states, resulting in desirable emerging behaviors. We refer the reader to [3] for a survey of coordination approaches in multi-agent systems.

For CADRE, we selected a dynamic centralized architecture with an elected leader (shown in Figure 3). All decisions requiring coordination between the agents are made by a "leader" agent; the leader is selected dynamically by the agents, ensuring resilience to loss or degradation of individual agents. This approach provides the key advantages of centralized architectures, namely, (i) lower communication overhead compared to decentralized approaches (since information must only be shared with one leader, as opposed to all agents) and (ii) greater explainability of coordination decisions compared to decentralized approaches (since all coordination decisions are made on an individual agent); at the same time, leader election mitigates the key disadvantage of centralized architecture, i.e., the presence of a single point of failure, at the cost of increased communication and complexity to ensure that the system has one, and only one, leader at all times.

A centralized architecture with a fixed leader was not selected because of the unacceptable introduction of a single point of failure. Bidding-based approaches were not selected due to increased communication overhead compared to the selected leader election architecture, and the consideration that the key advantage of bidding architectures (namely, the ability to accommodate heterogeneous agents without disclosing each individual agent's abilities and cost function to the entire team) is not highly relevant to the CADRE mission. Shared-world approaches were considered, but the possibility of scenarios where agents would take uncoordinated actions in presence of communication failures (as opposed to stopping in absence of information, as the selected approach does) was highly concerning from an operational standpoint. Finally, local-coordination approaches were not selected due to the complexity of encoding the desired behaviors in local coordination laws; the key advantage of local-coordination approches, namely, scalability to a large number of agents, was not highly relevant to CADRE's four-agent architecture.

Leader Election

The underlying algorithm used by CADRE's leader election process is a distributed algorithm for constructing minimum spanning graphs known as GHS, after the authors Gallager, Humblet, and Spira [4]. Upon wakeup, each agent assumes leadership of a one-node "tree" that only includes itself. By exchanging messages over the mesh radio network, the roots of each tree can discover each other and decide to merge their trees into larger trees, while leaving one of the agents (selected as the root of the tree) to lead that tree. This process is repeated up to $\log(n)$ times, until all agents are in the same tree.

This algorithm results in a minimum spanning tree (MST) for the network of connected agents (where connected means the set of agents that are connected to each other *and* the base station). The root of this MST is then assigned as "appointer", and selects a leader based on an operator-defined metric that captures a linear combination of the agent's power and thermal states, and with a strong preference for maintaining the existing leader. A second agent is selected as *designated survivor*, for increased resilience - the designated survivor is primed to become the new leader in case the selected one is lost. The primary reason for the designated survivor is to minimize the impact of having to rebuild knowledge on leader change (similar to a RAID system [**?**]) , which will be discussed in more detail in subsection 3. All agents synchronize information to both the leader and the designated survivor, allowing for handover of leadership responsibilities without a lengthy information synchronization phase. The leader selection process performed by the appointer is heavily biased towards the winner of the previous election to minimize changes in leadership and thus, mitigate the cost of leadership changes.

Strategic Planner

Leader election in a multi-agent system allows for a centralized agent to make strategic decisions about what the team should do to accomplish a specified goal. Specifically, the function of the strategic planner is to take a system-level goal from ground operations and generate a sequence of tasks to be executed by each rover on the team, accounting for the rovers' available resources (i.e., their available energy and thermal state). The underlying implementation of the strategic planner is provided by JPL's MEXEC planning, scheduling, and execution framework [5]. System-level tasks and constraints (e.g., "explore this region", "never drain the battery below 20%") are encoded in a *"task network"* (or "tasknet") that is uploaded by ground operators, along with parameters for individual activities (e.g., "drive with a maximum velocity of $\Delta m/s$"). The tasknet includes a set of tasks and their constraints, including precedence constraints (e.g., "an agent can only explore a region after regions for all agents have been computed"), resource constraints (e.g., "exploration can only be performed while the battery level is higher than 20%"), and temporal constraints (e.g., "no tasks should be scheduled at times when the rovers are scheduled to be under ground control"). The tasknet also includes predicted resource impacts (e.g., "exploring a 100 m^2 region will drain the battery by 40%"), allowing the on-board scheduler to predict the expected impact of planned activities. Two sets of tasknets have been developed for CADRE, one for exploration and one for the distributed measurements. For both activities, different tasknets are used at different times of the Lunar day, capturing variations in the predicted impact of tasks on the rovers' thermal and power state (e.g., rovers will warm up faster closer to lunar noon). Operators are able to select which set of tasknets should be used (and therefore whether exploration of a distributed measurement should be performed) by modifying the autonomy startup sequence. An on-board scheduling algorithm then processes the tasknet to generate a sequence of feasible tasks; the sequence is expected to satisfy the prescribed constraints based on the provided resource impact models. The sequence is periodically re-evaluated based on real-time measurements of available resources, in particular, the rovers's state of charge

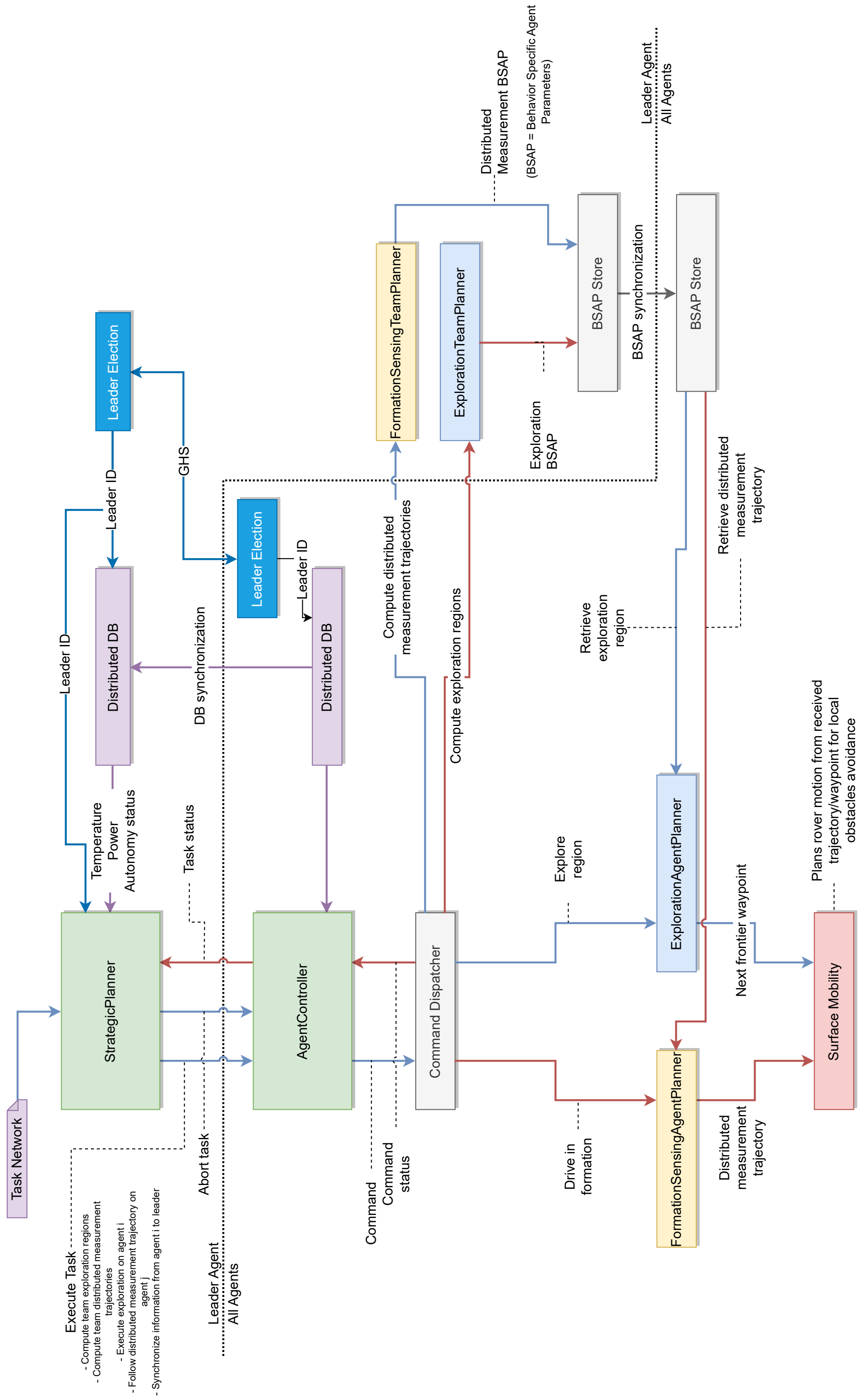

Task Network
StrategicPlanner
AgentController
Command Dispatcher
Distributed DB
Distributed DB
Leader Election
Leader Election
Leader ID
Leader ID
Leader ID
GHS
DB synchronization
Temperature
Power
Autonomy status
Task status
Execute Task
- Compute team exploration regions
- Compute team distributed measurement trajectories
- Execute exploration on agent i
- Follow distributed measurement trajectory on agent j
- Synchronize information from agent i to leader
Abort task
Leader Agent
All Agents
Command
Command status
Compute distributed measurement trajectories
Compute exploration regions
FormationSensingTeamPlanner
ExplorationTeamPlanner
Exploration BSAP
BSAP Store
BSAP Store
BSAP synchronization
Distributed Measurement BSAP
(BSAP = Behavior Specific Agent Parameters)
Leader Agent
All Agents
Retrieve distributed measurement trajectory
Retrieve exploration region
Explore region
Drive in formation
ExplorationAgentPlanner
FormationSensingAgentPlanner
Next frontier waypoint
Distributed measurement trajectory
Surface Mobility
Plans rover motion from received trajectory/waypoint for local obstacles avoidance

Figure 3. **CADRE's multi-agent autonomy architecture relies on six key components: a strategic planner, team planners and agent planners, a distributed database, a leader election module, and a Surface Mobility module.**

and temperature.

Figure 4 shows the output of the strategic planner. The top pane displays the sequence developed for all agents; the bottom pane displays the predicted evolution of key on-board resources including temperature and available energy.

While the strategic planner manages on-board resources, it does not explicitly reason about spatial tasks, e.g., where the rovers should drive to explore a region, or how they should keep in formation; rather, these decisions are delegated to "team planners" that are invoked by the strategic planner. Specifically, the task network includes tasks that capture required coordination between the agents, i.e., "compute which regions the agents should explore" and "compute a collision-free formation trajectory for the agents". From the strategic planner's standpoints, these tasks are prerequisites for the agents' individual tasks, e.g., "explore a region" and "drive in formation"; when the coordination tasks are invoked, the team planners (discussed next) ensure spatial coordination between the agents.

Team Planners

Team planners are responsible for solving specific cooperative problems, such as exploration or performing distributed measurements. CADRE will demonstrated both of these capabilities during the lunar demonstration, and thus, has implemented two planners. Exploration allows multiple rovers to collectively collect the data required to build up a surface map; distributed measurement uses ground-penetrating radars on-board each rover to map the subsurface by driving in a specific geometric formation.

Exploration—To decide where agents should drive to explore a region, we adopt a variant the ubiquitous frontier exploration approach proposed by Yamauchi [6].

A key design driver for the planner was to delegate significant decision-making authority to individual agents so as to enhance the system's resilience to temporary losses in communication. CADRE's predecessor, Autonomous PUFFER, solved the exploration problem using frontier based exploration combined with communication constraints [7], such that for each set of frontier locations that would be serviced by the rovers, the rovers would be within line of sight of each other, such that a connected network to the base station exists.

In contrast, in CADRE, we elected to allow individual agents to select which frontier points to explore; the leader's role was reduced to dividing the overall region to explore into N minimally-overlapping subregions, one per rover, via clustering of unknown space. Once each rover is assigned to a subregion, it performs frontier exploration in the subregion until the size of the frontier shrinks to zero. Rovers keep track of explored areas via local maps that assume three possible values: "unknown", "obstacle", and "traversable". The maps are periodically relayed to the leader, which keeps track of exploration progress, and re-computes the subregions assigned to each rover periodically; this allows the system to redirect more resources to harder-to-explore regions, e.g., regions with more obstacles cluttering the agents' perception.

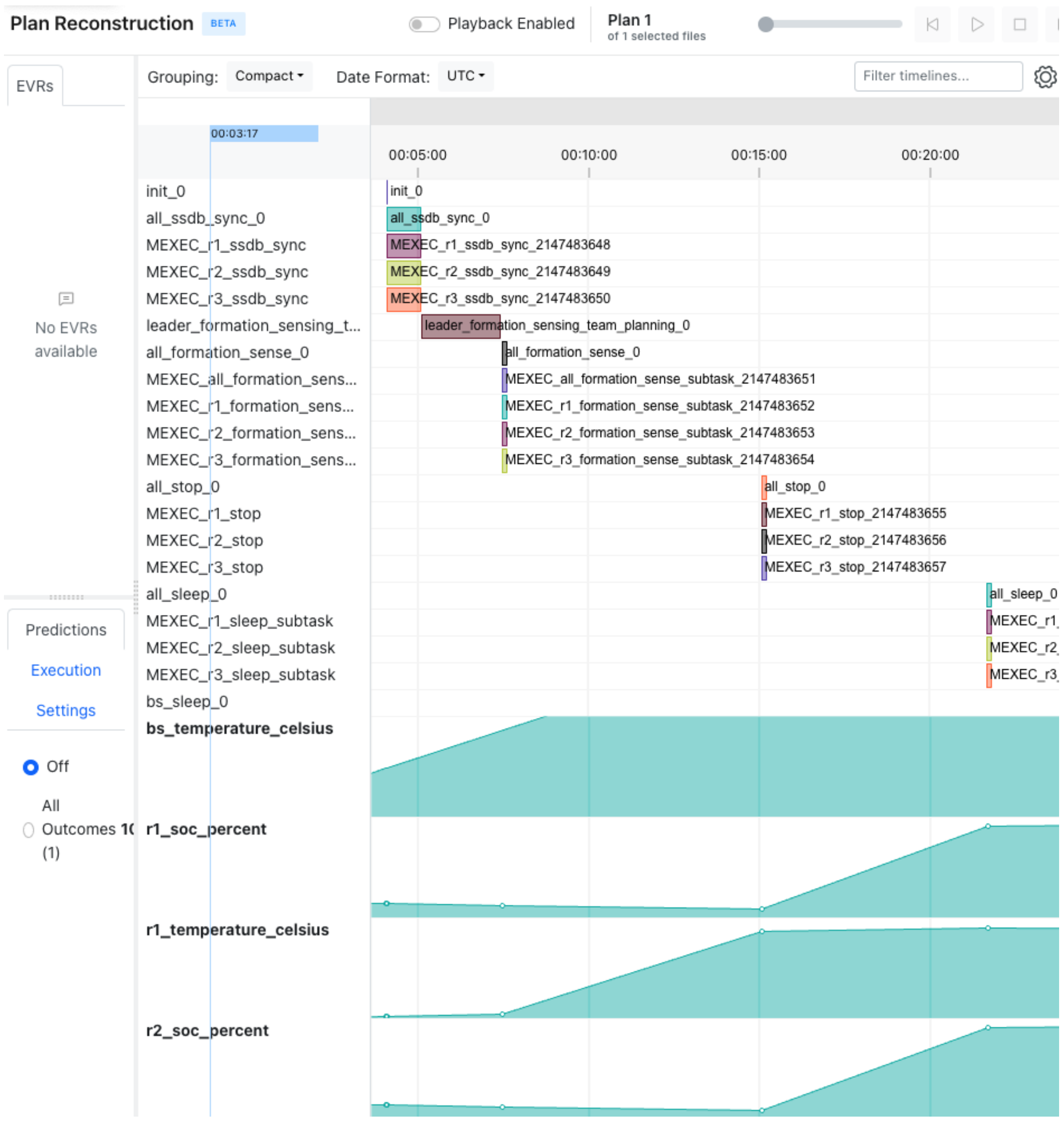
Distributed Measurement— A distributed measurement is when rovers in different physical locations use their instrument at the same time to characterize the environment from multiple viewpoints. CADRE rovers are equipped with a ground penetrating radar (GPR). The state-of-the-art, such as Preseverance's (M2020's) RIMFAX [8], is to use a single radar on a singular robotic platform to sound the subsurface by measuring its own reflections. This approach leads to the need to interpolate heavily between measurements to achieve 3D images from 2D tracks obtained through, e.g., a lawnmower pattern. In contrast, multiple, synchronized radars are able to improve on this approach by not only listening to their own reflections, but the reflection of neighboring radars at known relative positions. Such multi-static ground penetrating radar measurements allow for direct 3D subsurface measurements with lower noise due to reduced need for interpolation and additional information gained from the cross-track measurement of reflections from other robots' signals (i.e., non-self reflections).

Since a multi-static GPR is most sensitive at a specific depth with a 1:1 relationship to the separation distance between radars, the autonomy effectively needs to maintain a formation for the duration of a drive on the lunar surface. Such formation driving could be achieved with realtime control, but CADRE's approach is to break up planning and execution as is done with exploration to reduce the need for communications between agents, and in particular between non-leader agents. In this spirit, a team planner is responsible for computing collision-free, in-formation trajectories for all rovers, along with tolerances in space and time. These tolerance are selected so as to to keep the GPRs within $\pm 3dB$ of their optimal signal-to-noise ratio. The tolerances can be thought of as "tunnels" around the trajectories, and scale linearly with the distance between rovers–the larger the inter-rover distances, the larger the tolerance. A sample-based motion planning algorithm, RRT^* [9], is used to plan kinematic trajectories in the joint state space of all robots, represented as the (x, y, θ) location of a virtual leader and the (δ, ω) deviations from the nominal formation position of all robots with respect to the virtual leader. The latter deviations are important in allowing to plan trajectories samples that diverge from the nominal formation (within the prescribed tolerances) to allow the planner to find paths around obstacles in the environment. To ensure computational efficiency, this planner only generates kinematic solutions, ignoring the vehicles' dynamics: tolerances are selected so as to account for localization uncertainty as well as execution uncertainty induced by the actual vehicle dynamics.

The output of a sampling-based planner can be rather unsmooth; to mitigate this, a post-processing step smooths the trajectory, as illustrated in Figure 5. Specifically, the solution is revised iteratively to align the orientation of rovers and nudge the formation away from any close obstacle (although the original RRT solution already ensures no rover's footprint overlapping with a hazard in the map).

All points on the trajectory are then time-stamped assuming the agents will follow a conservative trapezoidal velocity profile. This way, so long as every agent can follow its own time-stamped trajectory, the agents are guaranteed to stay in formation with no communication.

The on-board planner takes the assigned trajectory and feeds it into the surface mobility. Any issues by the surface mobility subsystem in following the trajectory within the provided tolerances is reported to the leader and causes a full replan at the strategic planner level, incorporating the most recent information (e.g., obstacles that were not present in the initial map) – not immediately to the ground operations team.



Grouping: Compact

Date Format: UTC

[✓ Off]

Figure 4. Output of the strategic planner.

Distributed Database

In order to perform its duties, the leader needs to collate information from all agents. In particular, the strategic planner requires knowledge of which agents are able to participate in multi-agent autonomy (i.e., which agents have been deemed operational by ground operators and on-board fault protection), and the power and temperature of all agents; in addition, the team planners for exploration and distributed measurements require knowledge of the agents' location and, critically, of a global, up-to-date map of the environment. A *distributed database* is responsible for collecting this data from every agent and synchronizing it to the leader and designated survivor. The distributed database interfaces with power and temperature estimators provided by flight software; with fault protection; and with surface mobility, which provides each rover's location and periodically output local maps centered around the robot's location.

The database is built atop SQLite. A message-passing approach is used to synchronize information from agents to the leader on demand. Each value stored in the database is tagged with the originating agent's ID, and with a timestamp. For time-sensitive information such as power level, temperature, and agent location, only the most recent value is synchronized; in contrast, *all* recorded maps are synchronized to the leader, allowing reconstruction of a global map of the environment. This synchronization is triggered immediately prior to strategic and team planning activities, to ensure that the most recent data is available to the leader.

Map merging—The distributed database is responsible to store and merge local maps generated by surface mobility into a global map. Surface mobility produces maps containing one of three possible values: "traversable", "obstacle", and "unknown".

When merging multiple maps, three rules are observed:

- "traversable" and "obstacle" values always trump "unknown" values;
- higher-resolution maps are believed over lower-resolution ones (with the exception of "unknown" values); and
- at the same resolution, more recent data trumps older

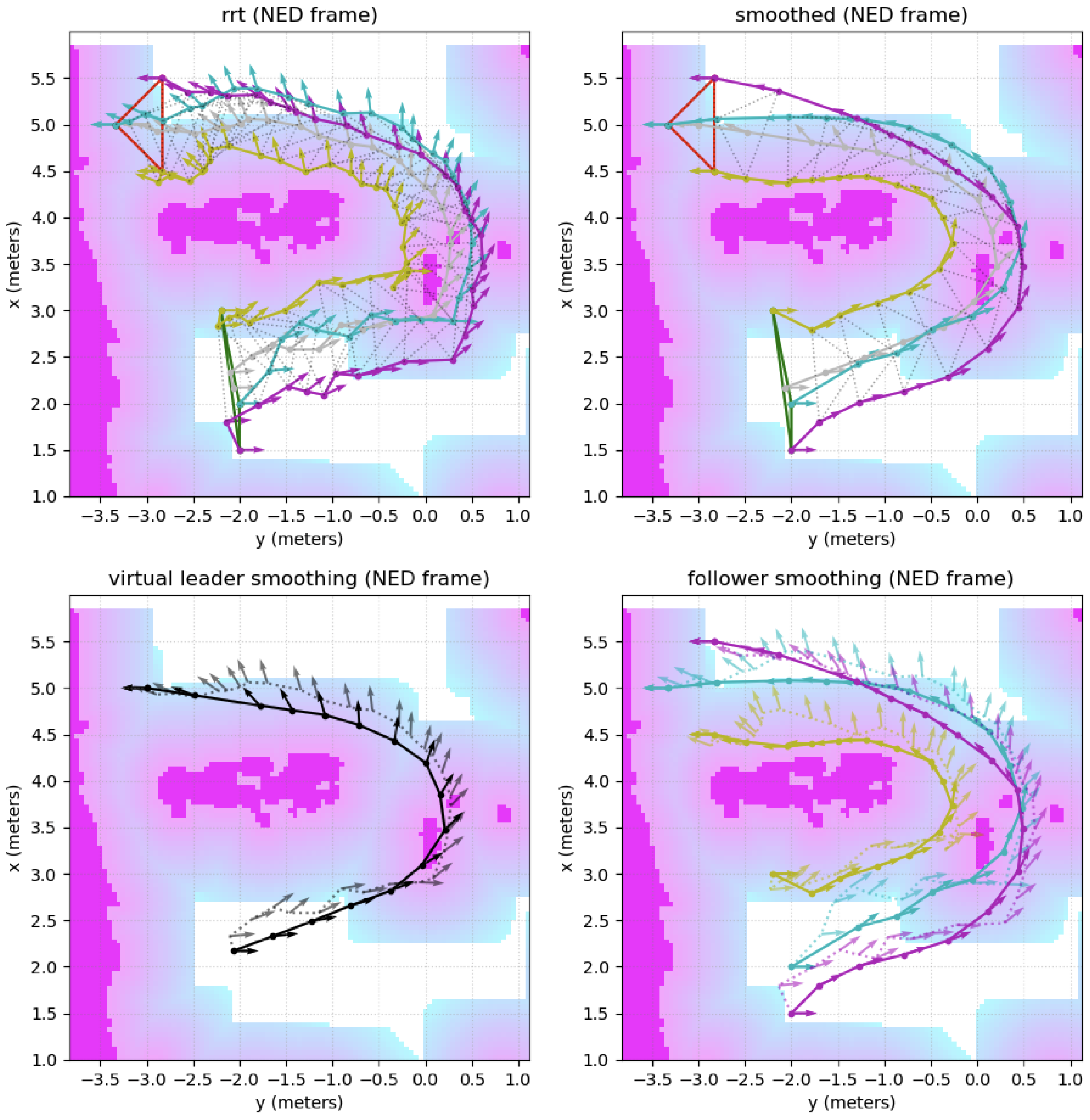


Figure 5. Multi-agent RRT^* and smoothed ouputs from the planner for the distributed measurements.

information.

Map-merging operations are implemented in OpenGL, resulting in significant parallelism; due to limitations in the Snapdragon Flight's drivers, however, the operation is performed on the Snapdragon's CPU.

The surface mobility stack includes a pose graph optimization (PGO) module (described in 4) which recomputes the most likely past and present locations of all rovers based on RF ranging information collected by radios on the agents. When PGO is executed, the location of past maps stored in the distributed database is updated accordingly.

4. Surface Mobility

A *Surface Mobility* subsystem on each rover is responsible for executing trajectories or driving to waypoints provided from the multi-agent autonomy and for providing information about the local environment for team-level planning purposes. Since a rover is executing a motion plan without any external information about its location in the world and the high-resolution structure of the surrounding environment, it uses

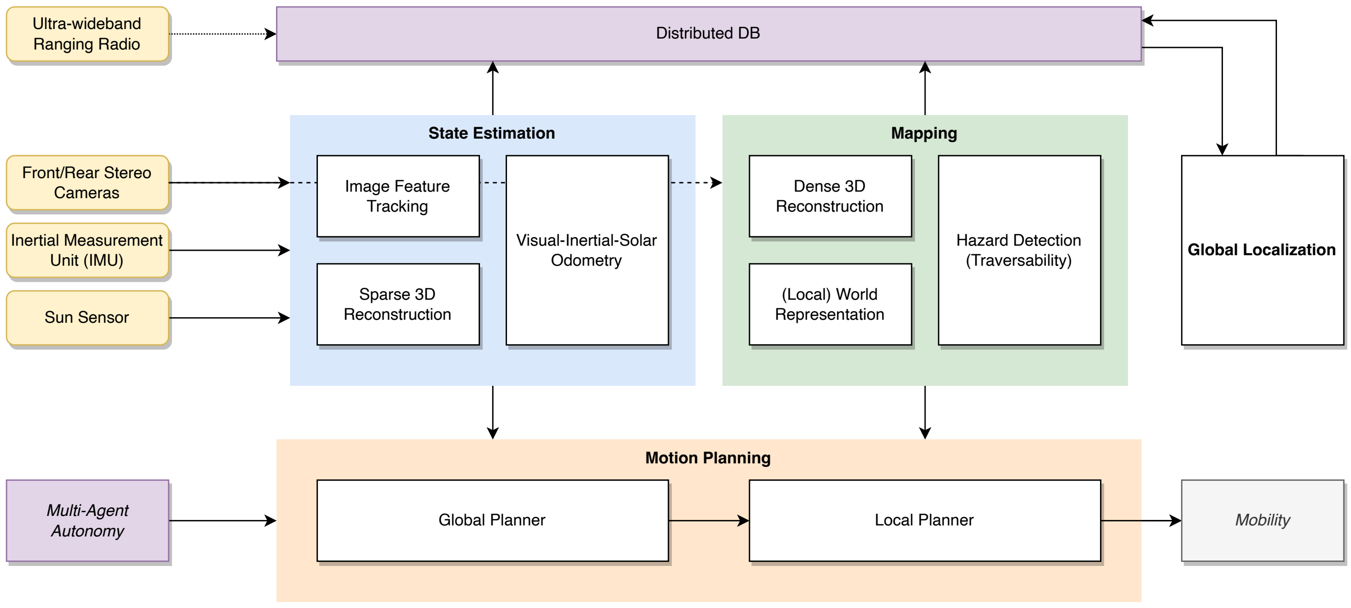


Figure 6. **Overview of the Surface Mobility subsystem's key components.**

information from its on-board navigation sensors to estimate its global pose and assemble a local, high-resolution map to enable obstacle avoidance of small-sized drive hazards below the resolution of an orbital map.

Therefore, we deploy three major subsystems (Figure 6):

- *State Estimation* estimates the pose of the rover in the world frame based on image feature tracking and inputs from the on-board IMU and sun sensor.
- *Mapping* uses inputs from the on-board camera to perform a 3D reconstruction and the assembly of a local traversability map.
- And *Motion Planning* uses the local maps and the estimated pose to execute a trajectory that is passed down from the multi-agent autonomy through the on-board agent controller.

Together, this allows the rovers to successfully navigate the lunar environment without any intervention.

State Estimation

The on-board state estimator deploys a visual-inertial-solar odometry approach that fuses visual information from the stereo cameras with inertial information from the IMU, and sun direction from the sun sensor. The odometry algorithm is designed to resolve high dynamic motion that can occur when a rover drives over rocks (the rovers have no suspension system) by integrating high-rate IMU data (accelerometers and gyroscopes at 200 Hz) and fusing the propagated pose in an Extended Kalman Filter approach with low-rate information from the vision system and the sun sensor to correct for drift and to estimate the IMU biases.

Visual information consists of image features that are tracked over time in a keyframe based approach. Initially, features are detected in the left reference view of a stereo camera, and range is computed for each feature by a sparse stereo algorithm which is based on a local block matching approach [10]. 3D features are then added as SLAM features in the Kalman filter for this initial *base frame* and tracked in subsequent *tracking frames* to calculate a vision update.

A key feature of the CADRE vision system on each rover is its front- and rear-facing stereo cameras. Since the lunar lighting environment is challenging, the rovers use two stereo camera pairs to switch whenever the lighting conditions are unfavorable in a particular direction. Lighting conditions are also addressed by an autoexposure algorithm to handle changes in scene lighting as a consequence of the rover being pointed more towards to ground (brighter) or more towards the sky (darker).

Whenever the number of tracked features drops below a threshold, the system selects the stereo camera system (front or rear) that is most feasible for feature tracking based on a image quality heuristic, and a new *base frame* is triggered using the selected stereo camera.

Finally, the state estimator uses sun angle measurements from the sun sensor to update the rover attitude. This is the only sensor that provides global information to the state estimator, reducing yaw drift substantially.

Mapping

Local mapping serves two purposes: the detection of drive hazards (rocks of a certain size and slopes beyond a certain inclination - e.g. inside craters) and the collection of 3D data of the environment to provide to the team planner. *Mapping* gets its inputs from a dense stereo algorithm [10] that uses either the front or the rear stereo cameras - depending on the drive direction - to produce sub-pixel precise disparity maps. Since the range error of stereo reconstruction scales with distance, the map representation is arranged as a multi-size, multi-resolution robocentric map to adapt the map cell size to the increasing range error with distance (Figure 7). (Figure 7, Table 1).

New measurements get fused into the map by deploying a Kalman filter based approach for level 0 (similar to [11]) and a Converging Covariance Map based approach for the upper levels ([12]). This allows for fine resolution height mapping to detect very small obstacles of 3 cm lateral size, whereas the higher levels allow for the detection of larger obstacles plus

Table 1. **Map resolution per layer**

Layer	Resolution	Size
0	1 cm	1.28 m
1	4 cm	2.56 m
2	16 cm	5.12 m
3	64 cm	10.24 m

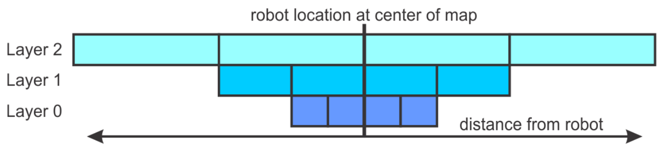


Figure 7. **Layout of multi-size, multi-resolution local map.**

the slope contained within a cell. From these local elevation maps, *Mapping* calculates a traversability map based on the roughness of the terrain and the steepness of the slope. Traversability maps are calculated in two resolutions. A level 0 resolution for the local planner, and a level 1 resolution for the global planner. Terrain is coded as *traversable*, *hazard*, and *unknown*, and the planners react differently to these values: whereas the local planner considers both, *hazard* and *unknown* as obstacles, the global planner tolerates *unknown* as traversable terrain. Before sending the traversability maps to the local and global planner, they are transformed into *signed distance field* maps which both planners use as a cost gradient map for planning.

Motion Planning

Surface Mobility motion planning uses a hierarchical approach dividing the task of finding feasible paths into two different planners with different planning horizons. Both planners formulate an optimal control problem and solve the subsequent optimization problem using a factor-graph based optimization solver (G2o [13]) considering time, heading, and collisions as major cost terms (Figure 10, 9).

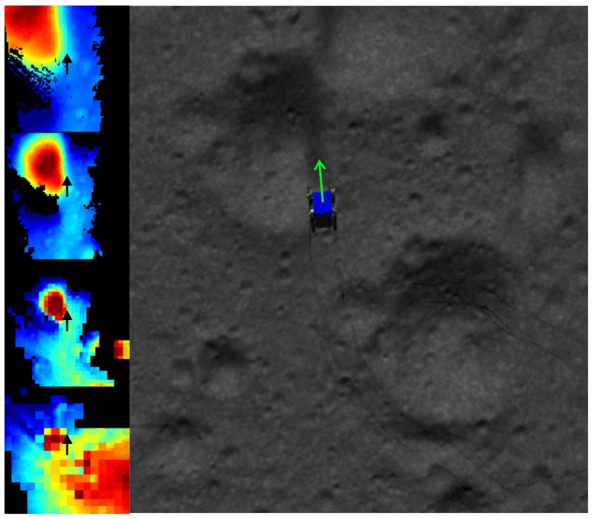


Figure 8. **Multi-resolution map in simulation: a rover is driving to the bottom of the right image encountering a crater. Left column: multi-resolution elevation map (red colors coding farther away points; rover is at the center of the map; drive direction is up). Level 0 map is at the top, Level 3 map at the bottom.**

The *Global Planner* uses a look-ahead horizon that can span outside the map extent. It receives a reference trajectory from the *Agent Planner* which consists of a sequence of waypoints in the case of formation sensing or a single waypoint in the case of exploration. The Global Planner plans a finer resolution trajectory following all waypoints within its planning horizon plus the next waypoint outside. This allows for an optimized trajectory that takes into account the traversability map were it exists, but also enables planning towards a distant waypoint (which results in a straight line trajectory outside the local map, since no hazard information is available.

The *Local Planner* receives the global planner trajectory as an input and plans a second trajectory with a much shorter planning horizon (the limit of the level 0 map) using the highest resolution (level 0) map level to avoid small-sized obstacles. Additionally, any unknown terrain is considered hazardous, since unknown map information at such close proximity to the rover indicates that there are occlusions either from positive or negative obstacles. The resulting optimized trajectory is translated into twist commands which are passed directly to the *Mobility* component that controls the motion of the rover by issuing commands to the four individual wheel motors. If necessary both planners can revert the drive direction of the rover, allowing for more flexibility in environments with denser obstacle concentrations.

As our optimization based planning approach does not provide recursive feasibility guarantees, both planners deploy a feasibility check after trajectory optimization to verify time and collision constraints on the provided solution. If a plan is infeasible, a feedback is send to the Agent Planner which ultimately triggers a re-plan at the team planner level.

State Estimation Initialization and Global Pose Optimization

Since the state estimator calculates relative poses based with respect to its starting location, it needs to be initialized with an initial pose. At the beginning of an experiment an estimate of the initial pose will be provided manually based on observations from the lander situational awareness camera (SACA), and distance measurements from an Ultrawide-band (UWB) sensor. Each rover, and the base station on the lander are equipped with this sensor, allowing for bidirectional distance measurements that are used in a *Global Pose Optimization* module to estimate an optimized pose from rover poses and UWB range measurements. The global pose optimization uses a factor graph formulation where the rover poses are graph nodes, and the edges are defined by delta poses of each rover, and range measurements to the SACA and other rovers (similar to the approach used by Autonomous PUFFER [14]).

UWB range measurements are taken periodically before and during a drive, with the option to execute a *Global Pose optimization* at each team planning cycle to reduced pose drift incurred during the previous cycle and hence to increase the precision of the team planner map that is assembled from local rover maps that are fused in the distributed database using the attached rover pose as location of the local map. This merged global map is then used for the next team planning cycle.

5. FLIGHT SOFTWARE

CADRE's multi-agent autonomy and surface mobility algorithms are implemented in C++14 with several dependencies

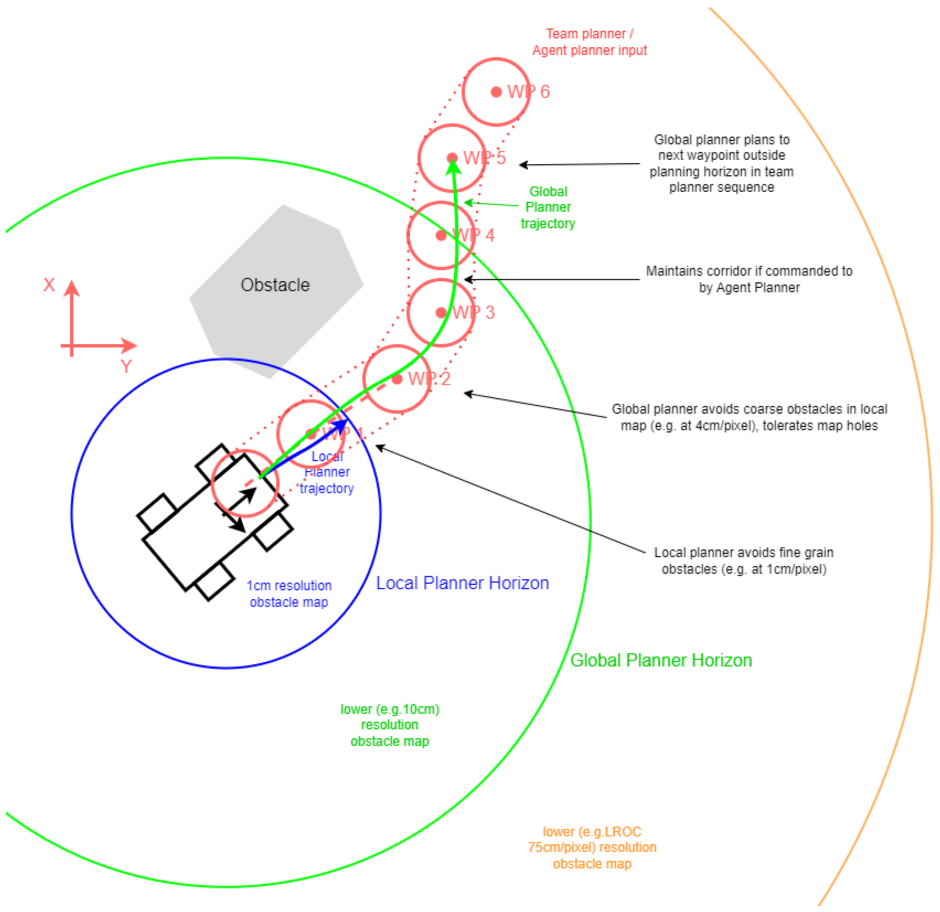

Team planner / Agent planner input
WP 6
WP 5
WP 4
WP 3
WP 2
WP 1
Global planner plans to next waypoint outside planning horizon in team planner sequence
Global Planner trajectory
Maintains corridor if commanded to by Agent Planner
Obstacle
X
Y
Global planner avoids coarse obstacles in local map (e.g. at 4cm/pixel), tolerates map holes
Local Planner trajectory
Local planner avoids fine grain obstacles (e.g. at 1cm/pixel)
1cm resolution obstacle map
Local Planner Horizon
Global Planner Horizon
lower (e.g.10cm) resolution obstacle map
lower (e.g.LROC 75cm/pixel) resolution obstacle map

Figure 9. **Cascading motion planning approach: the autonomy provides a trajectory that the global and local planner follow.**

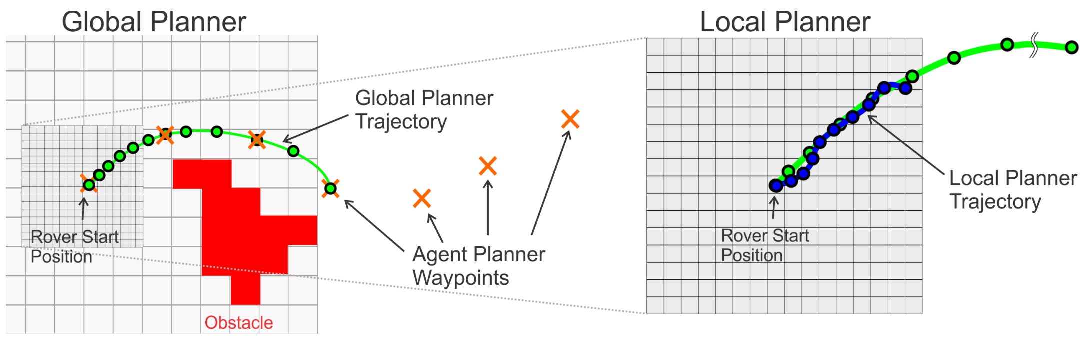


Figure 10. **Global and local planner planning along a set of team planned waypoints.**

on open-source packages, such as *g2o* [13] for the motion planner optimizer, *dlib* for minimum cost assignment, and *OpenCV* for map (image) processing functions. These algorithms are primarily built into the flight software to run on the CPU cores of the processor, while specific functions are delegate to the GPU via OpenCL [2]. Ultimately, the autonomy algorithms each map to a FPrime component that is integrated into a full deployment.

FPrime

FPrime is an open-source framework for writing a flight software system [1]. It is written in C++, and as a consequence, CADRE's flight software is written in C++14. FPrime components have been developed to provide the core functionality of the CADRE systems, such as motor controllers, communication, sensor drivers, time synchronization, and more. The FPrime deployment is run on the VOXL on a Linux OS (specifically, within a Debian bookworm 64-bit ARM chroot). A few daemons (e.g., camera driver) that interface with 32-bit drivers (provided by ModalAI) run outside of the chroot and communicate via IPC to a 64-bit component within the chroot. The output of a build is a FPrime deployment that can run on the target platforms–there are two deployments: one for the base station and one for the rovers due to differences in the hardware (including the FPGA implementation).

Algorithms as Libraries

CADRE's development of autonomy algorithms started in ROS (Robotic Operating System) due to ease of prototyping, as well as, prior implementations made available by Autonomous PUFFER. However, it was planned that the autonomy algorithms would eventually be part of the flight software, and more specifically, a FPrime deployment. Consequently, autonomy algorithms were written as libraries, which are invoked by FPrime components, or automatically generated using an autocoder, such as Navinci from the Mars Ingenuity helicopter. In either case, the FPrime component itself bridges data from ports, parameters, and other flight software inputs to the autonomy algorithms, which after some processing return outputs that can be sent back out via ports (or returned as part of a command). Such an approach also has the benefit that if CADRE's autonomy algorithms need to integrated into other frameworks, such as Space ROS or cFS (core flight software), then that is possible by integrating the libraries as-is into the new framework.

6. Preliminary Experiments

Although the actual lunar technology demonstration experiments will occur in the future, CADRE has preformed a variety of ground tests with various platforms to test its autonomy capabilities. These venues range from the Mars Yards to a cleanroom at JPL. Some experiments were preformed on 3D printed rovers (called the "Mercury 7") with the same COTS avionics and sensors, while others were performed on develoment models that are similar to the flight units aside from some missing hardware, such as solar cells on the panels. Lastly, testing was also performed on the actual flight units (rovers and base station) to ensure that the hardware and software is ready for its demonstration on the Moon. All three hardware platforms can be seen in Figure 11.

Figure 12 is a visualization of the combined autonomy data and decisions (e.g., plans) from all four flight units (three rovers and a base station) during a verification and validation tests in a cleanroom. It was demonstrated that the flight units (FMs) coperative and autonomous completed a successful formation drive, as they would need to achieve on the Moon. The base station execujted the leader functionality by planning and scheduling for the team. When driving out-of-formation, rovers reported to the leander and the system replanned and continued to drive autonomously. Three autonomous planning cycles were performed. In a second experiment, it was verified that the team stops driving when any rover violates its state-of-charge constraint (i.e., battery is too low). A third experiment inclduded presenting the team with a previously unmapped obstacle around which the team replanned. Test operators started autonmy, but did not intervene at any point.

In addition to these tests, the autonomy software has made use of a variety of simulations and emulations to test various capabilities that are not easily capture in a single testbed. For example, lunar lighting conditions were simulated and temperatures and state-of-charge emulated to test how the system would behave under various conditions. Prior to flight, ORT (operational readiness tests) will be used to test the system's effective performance with the operators in the loop.

7. Conclusions and Future Work

This paper presented the autonomy architecture of CADRE, a technology demonstration mission slated to fly to the Lunar surface in 2024. CADRE will demonstrate how a team of three robots can autonomously perform high-level tasks assigned by ground operators, including exploration and distributed measurements, while satisfying resource constraints and replanning in response to new information in the system or from the environment. We envision that lessons learned from CADRE will enable future, bolder exploration of Solar System bodies, addressing hitherto-unanswered questions in planetary science.

Future work is likely to focus on understanding the scalability of this approach to larger teams and when changes are required to allow for larger deployments. Scalability can be viewed not only in terms of the ability of the autonomous system to coordinate its resources effectively, but also for human teams on the ground to operate such a system. Another future direction is to apply this architecture to a heterogeneous team, where the roles and capabilities of each are more diverse. For example, this work could be applied to allow future rovers and helicopters to work together to cooperatively explore the surface of Mars and other planetary bodies.

Acknowledgments

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). The authors thank all the staff at the Jet Propulsion Laboratory working on CADRE for their contributions to the work described in this paper. The authors also acknowledge the contributions to CADRE by the various partners, such as Motiv Space Systems, SolAero, and Dr. Laura Remond and her students at Clemson University.

References

[1] NASA Jet Propulsion Laboratory, "F'." [Online]. Available: https://github.com/nasa/fprime

[2] J.-P. de la Croix, F. Rossi, Y. Nakka, and K. Albee, "Using opencl to speed up parallelizable autonomy algorithms on cadre," in *16th Annual Flight Software Workshop*, Pasadena, CA, 2023.

[3] F. Rossi, S. Bandyopadhyay, M. T. Wolf, and M. Pavone, "Multi-agent algorithms for collective behavior - a structural and application-focused atlas," 2021, submitted. [Online]. Available: https://arxiv.org/abs/2103.11067

[4] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and systems (TOPLAS)*, vol. 5, no. 1, pp. 66–77, 1983.

[5] M. Troesch, F. Mirza, K. Hughes, A. Rothstein-Dowden, A. Donner, R. Bocchino, M. Feather, B. Smith, L. Fesq, B. Barker, and B. Campuzano, "Mexec: An onboard integrated planning and execution approach for spacecraft commanding," in *Workshop on Integrated Execution (IntEx) / Goal Reasoning (GR), International Conference on Automated Planning and Scheduling (ICAPS IntEx/GP 2020)*, 2020.

[6] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE In-*

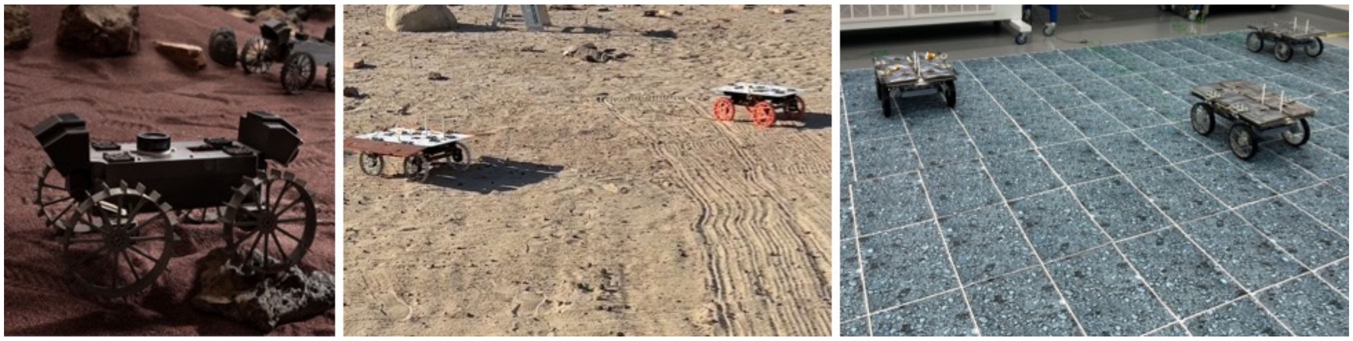


Figure 11. **The Mercury 7 prototype rovers (left), develoment models (middle), and flight units (right) in their various testbeds.**

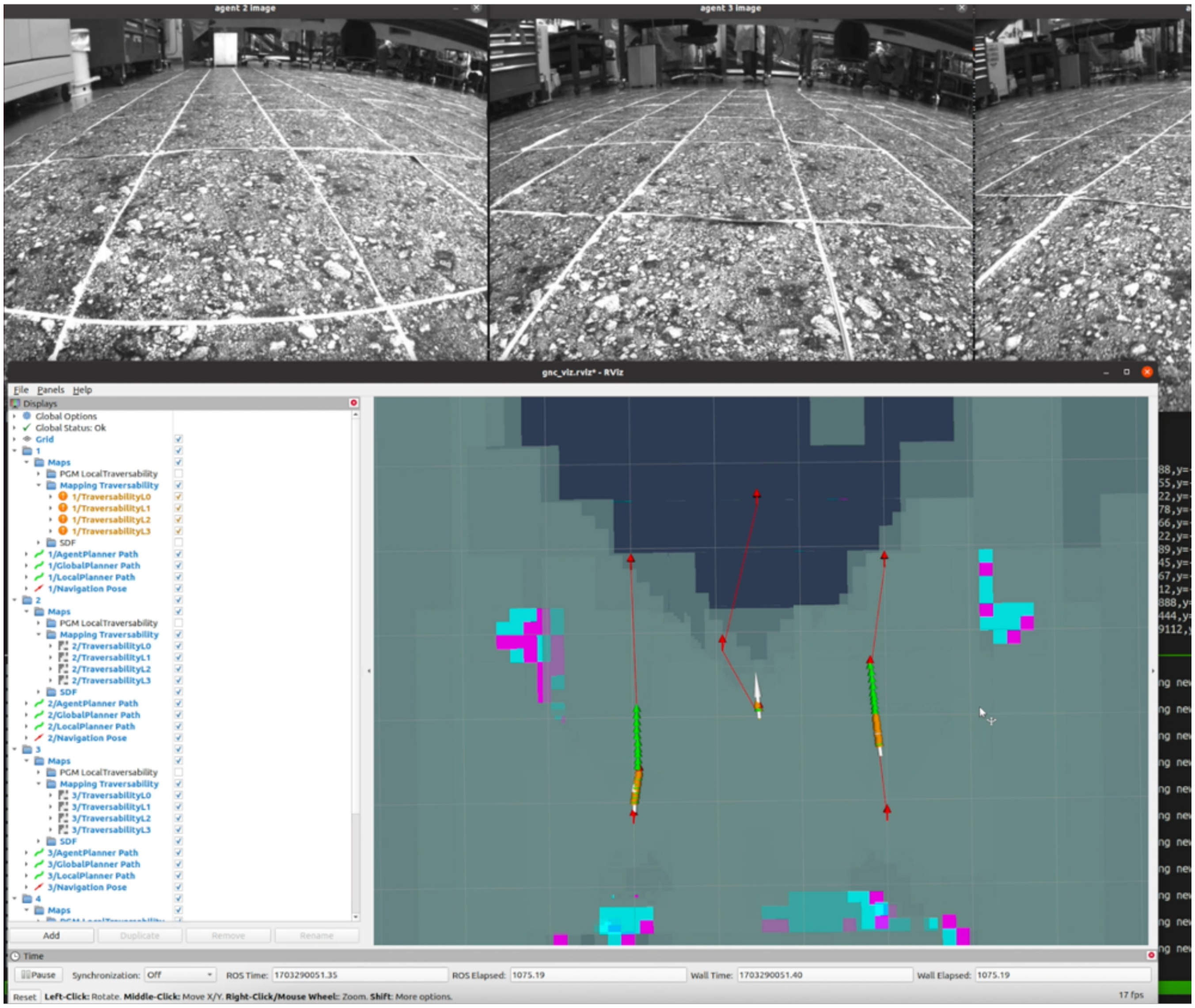


Figure 12. **Visualization of autonomous operation during a cleanroom test on the flight units. Red are the formation trajectories computed by the team leader (base station), while green are the on-board trajectories. Cyan and magenta are hazards, each seen by one or more of the rovers.**

ternational Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'. IEEE, 1997, pp. 146–151.

[7] J. Banfi, A. Quattrini Li, I. Rekleitis, F. Amigoni, and N. Basilico, "Strategies for coordinated multirobot exploration with recurrent connectivity constraints," *Autonomous Robots*, vol. 42, pp. 875–894, 2018.

[8] S.-E. Hamran, D. A. Paige, H. E. Amundsen, T. Berger, S. Brovoll, L. Carter, L. Damsgård, H. Dypvik, J. Eide, S. Eide *et al.*, "Radar imager for mars' subsurface experiment—rimfax," *Space Science Reviews*, vol. 216, pp. 1–39, 2020.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[10] S. Goldberg, M. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," in *IEEE Aerospace Conference*, 2002.

[11] P. Schoppmann, P. F. Proença, J. Delaune, M. Pantic, T. Hinzmann, L. Matthies, R. Siegwart, and R. Brockers, "Multi-resolution elevation mapping and safe landing site detection with applications to planetary rotorcraft," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1990–1997.

[12] L. Werner, P. F. Proença, A. Nüchter, and R. Brockers, "Covariance based terrain mapping for autonomous mobile robots," in *Submitted to: 2024 IEEE International Conference on Robotics and Automation*, 2024.

[13] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

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

BIOGRAPHY

Dr. 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 and 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 currently CADRE's principal investigator.*

Dr. Federico Rossi *is a Robotics Technologist with the Maritime and Multi-Agent Autonomy Group at the Jet Propulsion Laboratory, California Institute of Technology. He earned a Ph.D. in Aeronautics and Astronautics from Stanford University in 2018, a M.Sc. in Space Engineering from Politecnico di Milano and the Diploma from the Alta Scuola Politecnica in 2013. In his free time, Federico enjoys astronomy and photography. He is currently CADRE's multi-agent autonomy lead.*

Dr. Roland Brockers *is a Robotics Technologist at the Jet Propulsion Laboratory. He received his Ph.D. in Electrical Engineering from the University of Paderborn (Germany) in 2005, and has been conducting research in autonomous navigation of unmanned robotic systems for more than 23 years. Roland is the Surface Mobility lead on the CADRE mission. His current research interests include 3D perception systems for small mobile robotic platforms, and autonomous robotic systems with applications in earth science and planetary exploration.*

Guy Zohar *received a B.S. and M.S. in Aerospace Engineering from California Polytechnic State University, San Luis Obispo in 2013. He has been with JPL for 10 years and is currently the System Manager on CADRE. He was the lead Payload System Engineer on Sentinel-6 and the Mars Sample Return Lander projects.*

Subha Comandur *is CADRE's Project Manager at NASA JPL. Subha has over twenty years of engineering and leadership experience at JPL and industry on various spacecraft missions including Mars Curiosity Rover, JUNO, Soil Moisture Active Passive. During her time as Lead, Technical Group Supervisor and Assistant Section Manager in the Spacecraft Mechanical Engineering Section, Subha earned various awards including a NASA honor medal for Early Career Achievement in 2014, JPL award for Exceptional People Leadership in 2017 and a Principal Designation in 2019. Prior to CADRE, she led a large, multidisciplinary team as the Product Delivery Manager for Mechanical and Harness Subsystems on Europa Clipper spacecraft. Subha started her career at Boeing as an electrical engineer, where she worked on numerous projects including demonstration of on-orbit autonomous servicing spacecraft. She has a Masters in Computer Science from University of California, Irvine and a Bachelors in Electrical and Electronics Engineering from India.*