Cooperative Autonomous Distributed Robotic Exploration (CADRE)

CADRE MoonDB: Distributed Database for Multi-Robot Information-Sharing and Map-Merging for Lunar Exploration

Maíra Saboia, Federico Rossi, Viet Nguyen, Grace Lim, Dustin Aguilar, Jean-Pierre de la Croix

Jet Propulsion Laboratory, California Institute of Technology

Presented at the

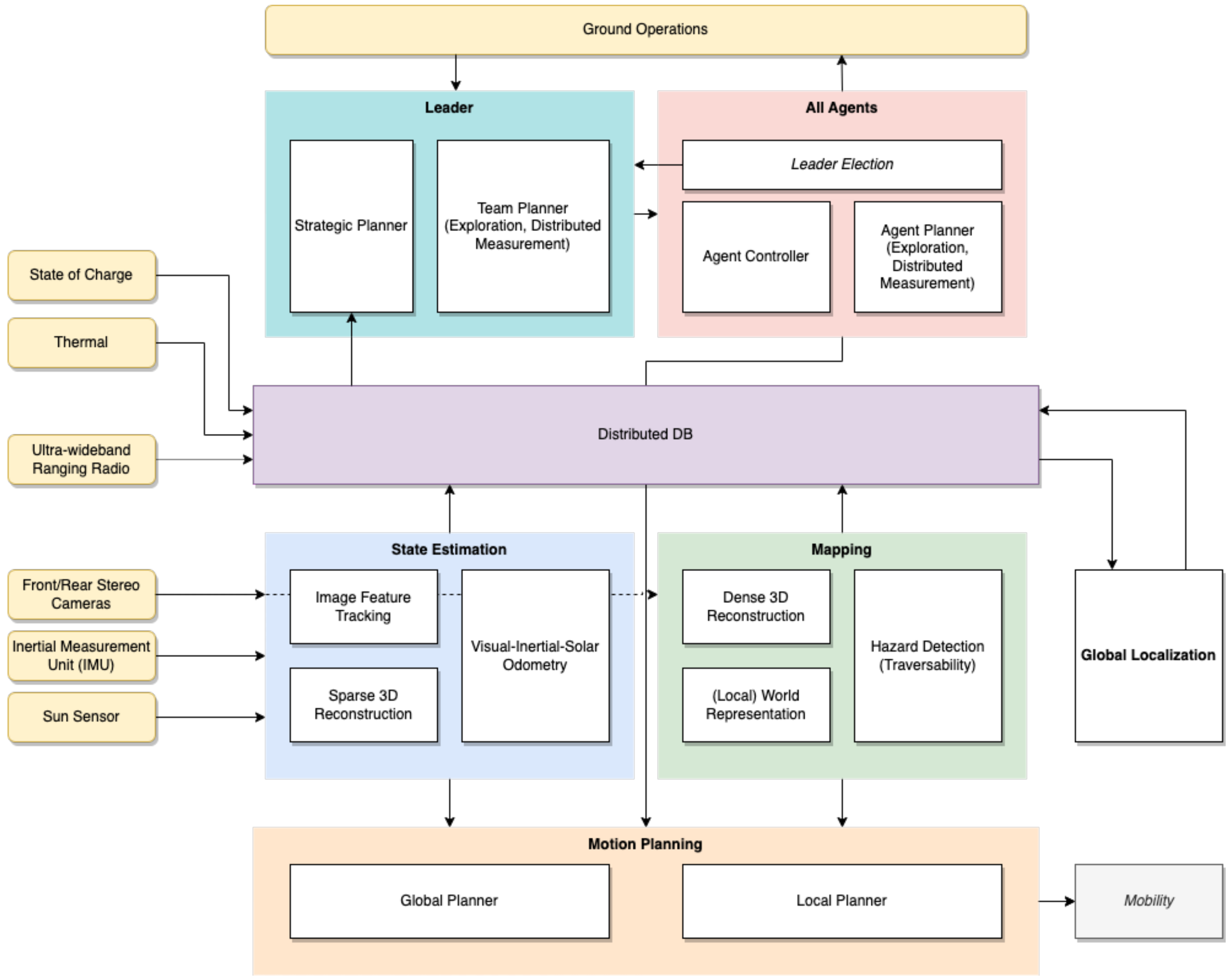
International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace ’24)

Auckland, New Zealand

May 7, 2024

Jet Propulsion Laboratory
California Institute of Technology

CADRE'S AUTONOMY



Ground

Provide goals:
Explore this region
Follow this path in formation

Leader

Decide:
What can I do with all available resources?
Who goes where?

Every Rover

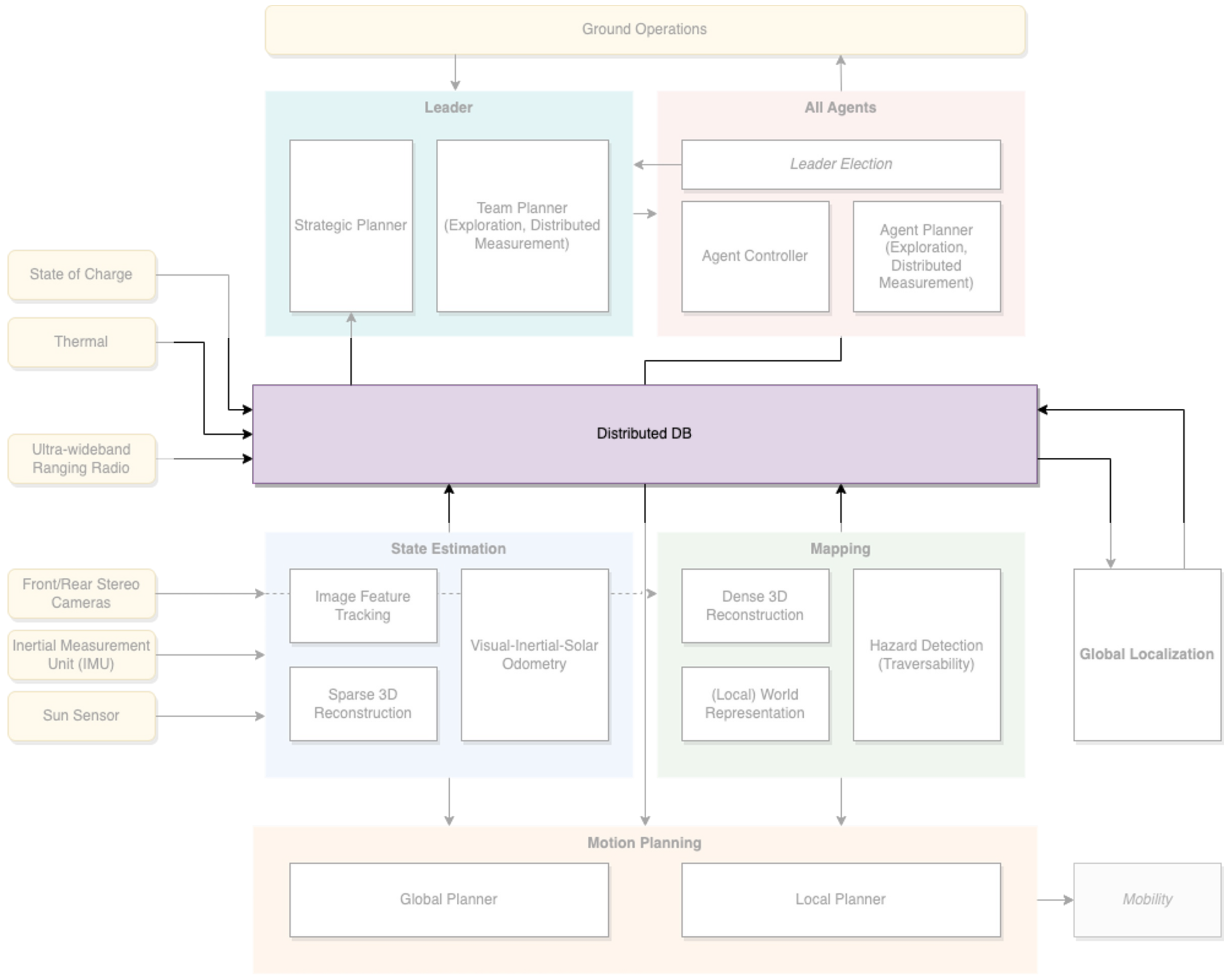
Communicate:
What do we all know?

Estimate:
Where am I?

Estimate:
What does the Moon look like?

Decide:
How to get there?

CADRE'S AUTONOMY - MoonDB



Ground
Provide goals:
Explore this region
Follow this path in formation

Leader
Decide:
What can I do with all available resources?
Who goes where?

Every Rover
Communicate:
What do we all know?

Estimate:
Where am I?

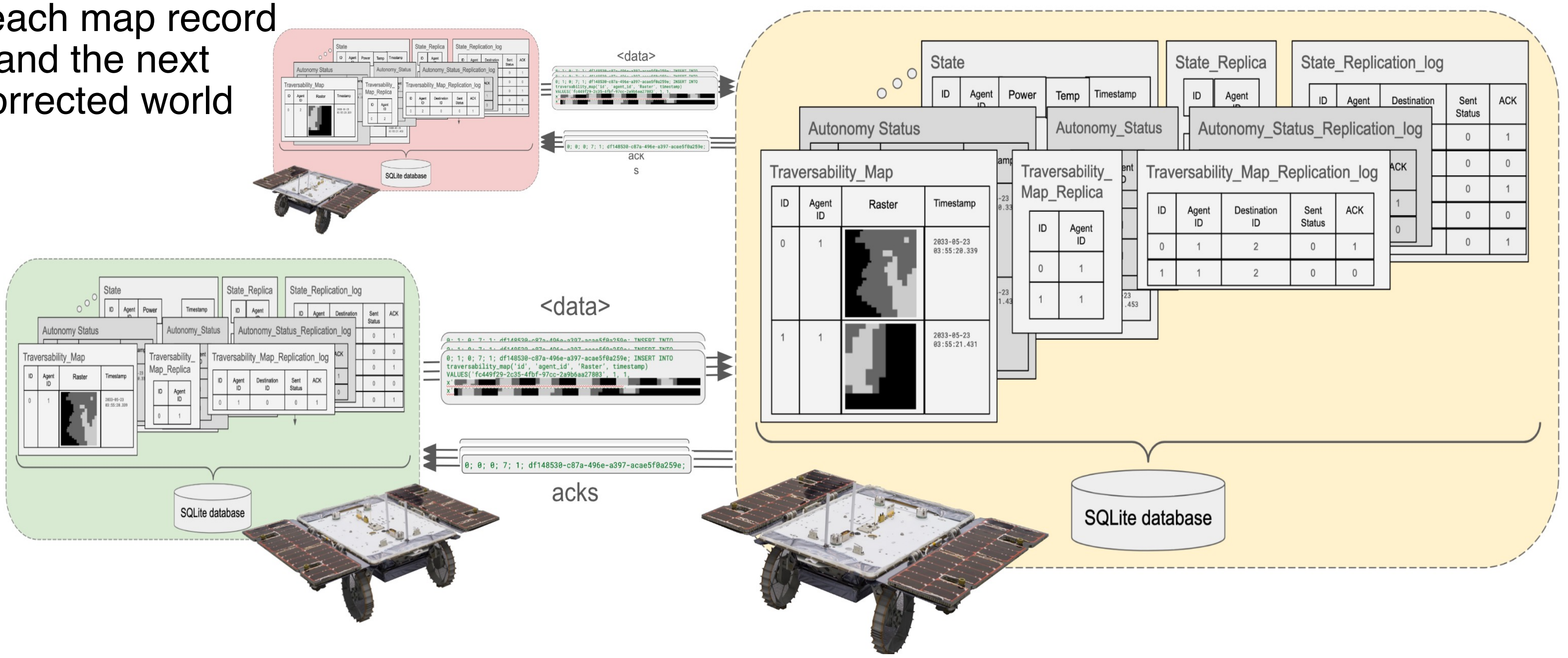
Estimate:
What does the Moon look like?

Decide:
How to get there?

MoonDB

Stores, shares and fuses information from multiple robots providing multi-agent planning algorithms with a consistent view of the robotic team.

- Each rover records local, multi-resolution traversability map with state estimate into local ”MoonDB” on a regular cadence (no local merging)
- Maps are synchronized ahead of planning to the leader, where a merged map can be generated based on a merging policy (occupancy type, resolution layer, etc.)
- If system corrects for drift, each map record in the database is updated and the next map query will reflect the corrected world view for planning
- Data is synchronized to a leader and not all agents; however, there’s a “designated survivor” that receives a backup. Leader election ensures that the team can continue to function. Requires mesh networking.



Data Generation - Components

Ground Operations

Strategic Planner

Team Election

Formation Sensing

Exploration Leader

Exploration Agent

GNC Global Localization

GNC localization

GNC State Estimation

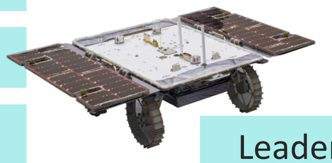
GNC Mapping

Mapper

GNC Agent

Comms

GPR Agent



Leader

"Telemetry"

Disk

SOC

Heath

Power

Temperature

Exploration Agent

GNC localization

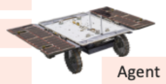
GNC State Estimation

GNC Mapping

GNC Agent

Comms

GPR Agent



Agent

Disk

SOC

Heath

Power

Temperature

"Telemetry"

"Telemetry"

Disk

SOC

Heath

Power

Temperature

Exploration Agent

GNC localization

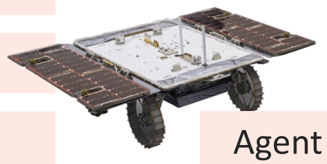
GNC State Estimation

GNC Mapping

GNC Agent

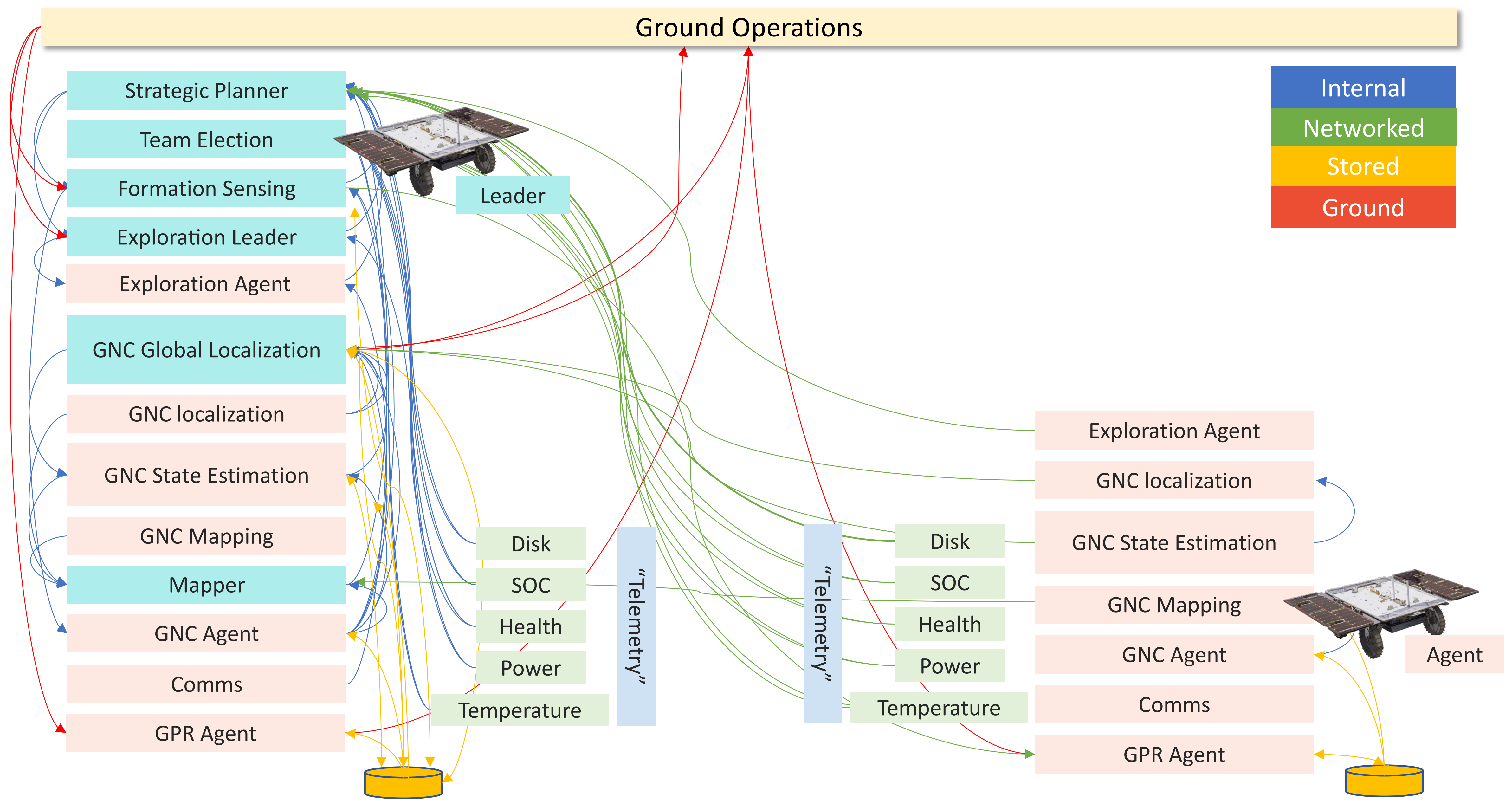
Comms

GPR Agent

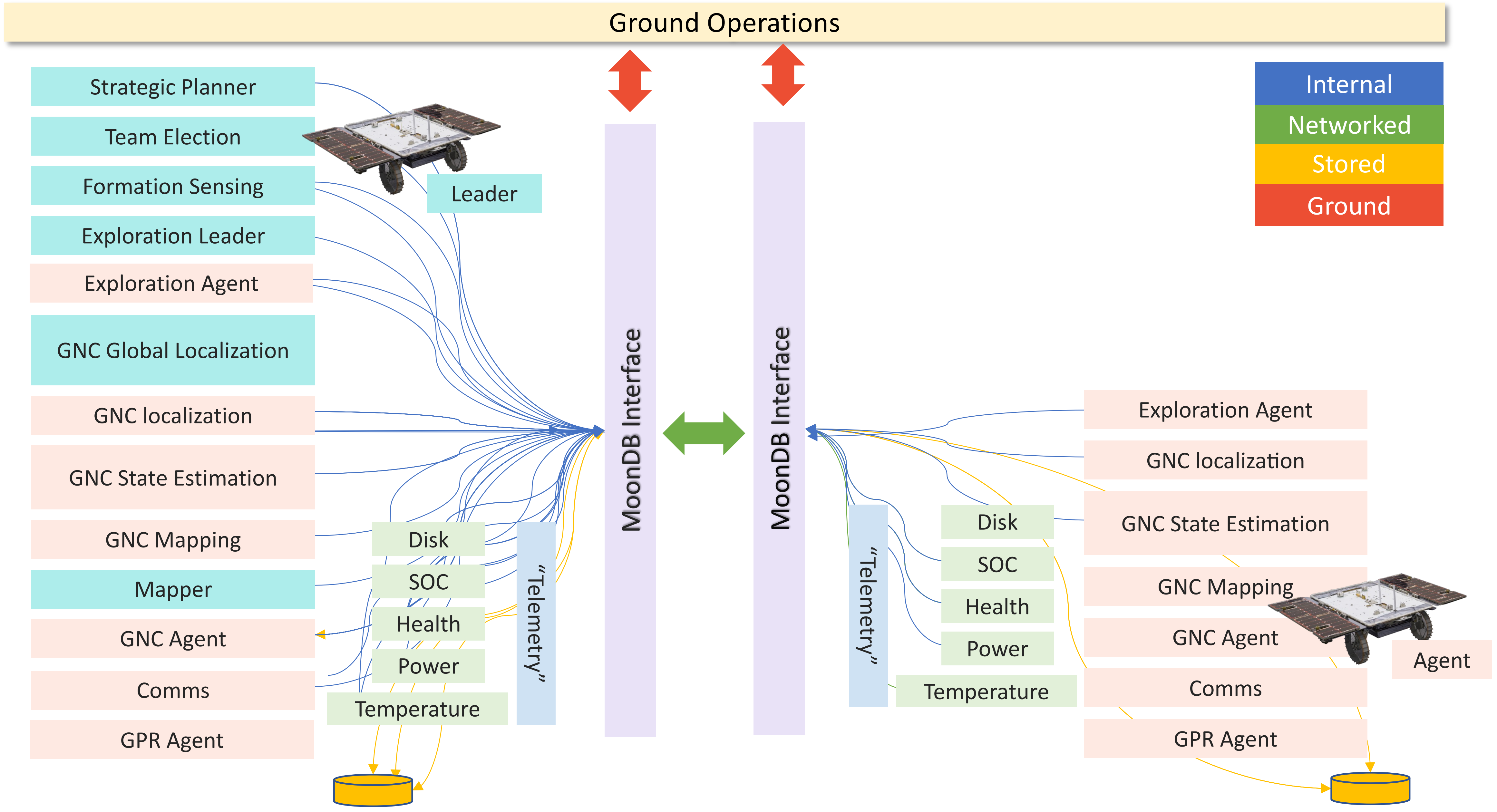


Agent

Data Flow


Ground Operations
Internal
Networked
Stored
Ground
Strategic Planner
Team Election
Formation Sensing
Exploration Leader
Exploration Agent
GNC Global Localization
GNC localization
GNC State Estimation
GNC Mapping
Mapper
GNC Agent
Comms
GPR Agent
Leader
Disk
SOC
Health
Power
Temperature
"Telemetry"
"Telemetry"
Disk
SOC
Health
Power
Temperature
Exploration Agent
GNC localization
GNC State Estimation
GNC Mapping
GNC Agent
Comms
GPR Agent
Agent

MoonDB – Distributed Data Manager



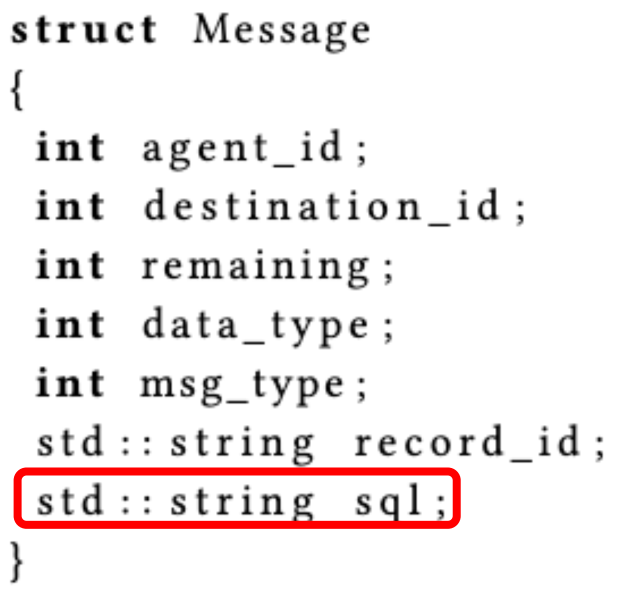
Assumptions and Constraints

- MoonDB is restricted from relying on continuous data transmission across the mesh network (no walk-and-talk).
    - This is due to electromagnetic interference (EMI) generated by motors. The interference significantly disrupts radio communication signals.
    - This limitation prevents the use of Raft-based consensus solutions like *rqlite* and *dqlite*, which demand continuous connectivity and a minimum of three nodes.
- Bandwidth is limited (1 Mbps data rate, significantly slower than typical robotics lab WiFi networks).
    - Only essential data for autonomy is transmitted; stale states are not shared with the leader.
    - Solutions like *Litestream* only replicate the entire database and risk data loss if the leader becomes unavailable, as data is flushed at intervals. It also requires a separate database file on the leader for each client agent.
- Limited computer resources (Qualcomm Snapdragon 821 SoC, 4GB RAM, 32GB flash)
    - We opted for SQLite due to its small memory footprint, reliability, and high performance ideal for embedded systems. In contrast, our prior A-PUFFER solution with Postgres, PostGIS, and Bucardo demanded more disk space, memory, and processing power.

Data Sharing

Synchronization and Replication

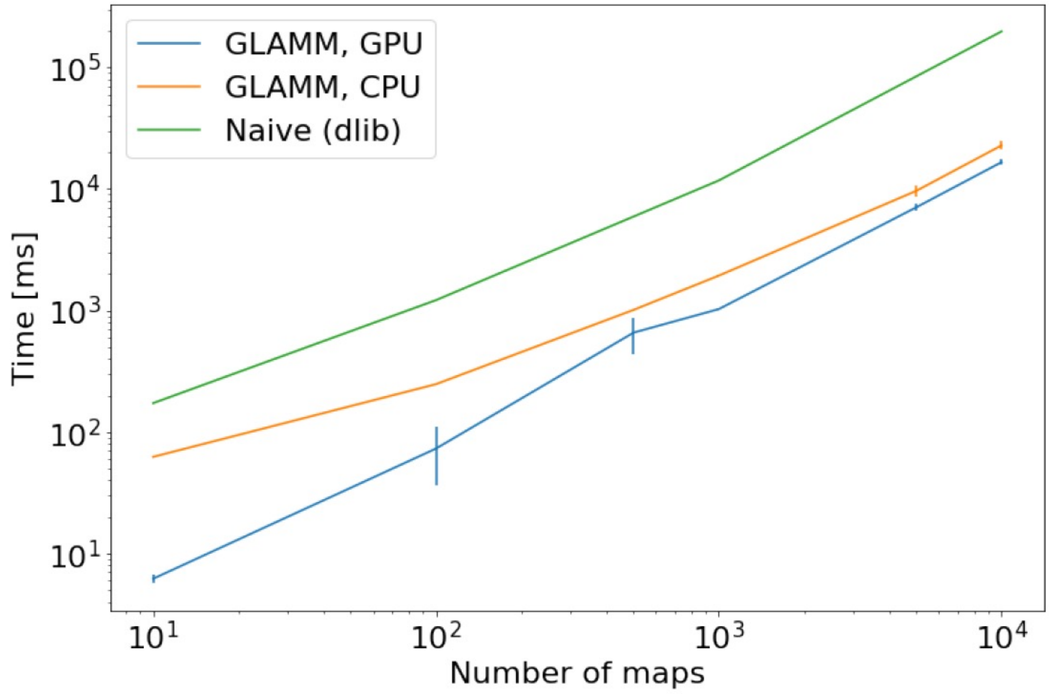
- Sync involves transferring the data of an individual agent to a specific destination, typically the leader, to keep it synchronized with all other agents.
    - Only data generated by the source agent is transferred to the destination
- Replication is utilized for backing up the leader to a designated survivor, ensuring rapid and transparent recovery.
    - All the data, not only that generated by the source agent, is transferred.
- The agent tracks acknowledgments to prevent redundant transfers and ensure complete data reception. Unacknowledged messages are retried in the next sharing cycle.
- The data is shared as a prepared SQL statement. This enables potential integrations with databases other than SQLite without requiring significant changes, promoting interoperability.



```
struct Message
{
 int agent_id;
 int destination_id;
 int remaining;
 int data_type;
 int msg_type;
 std::string record_id;
 std::string sql;
}
```

GPU-*accelerated* Map Merging

- GLAMM – *(Open)GL Accelerated Map Merging* takes advantage of OpenGL (ES) shaders to offload the map merging operation of thousands of maps to the GPU.
    - Implemented by using OpenGL ES 3.0 for shader programming and EGL for X11-less (headless / off-screen) rendering on an embedded platform
    - Note: due to a driver issue on the VOXL, it is running in software emulation mode, but is still faster than the original CPU implementation!

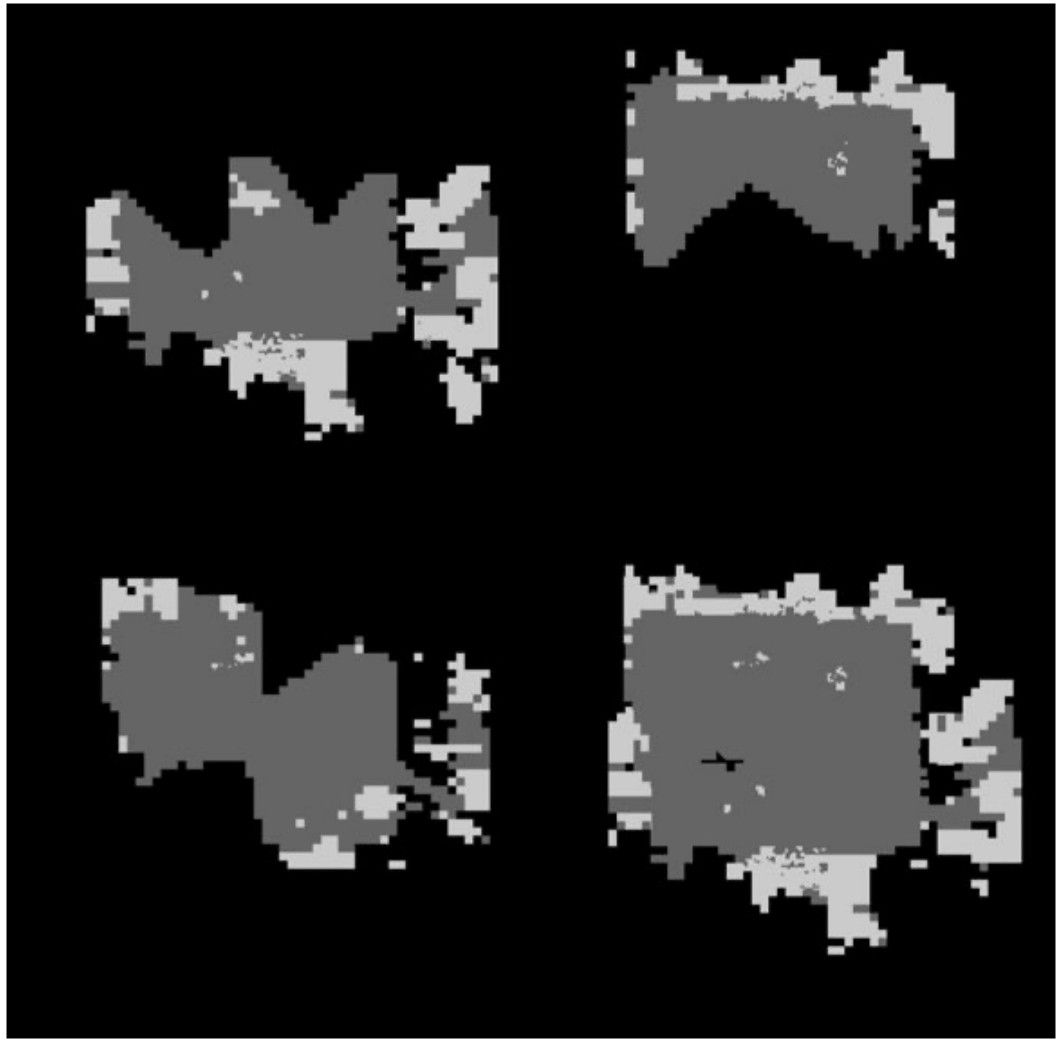


Shader Map Merging Policy

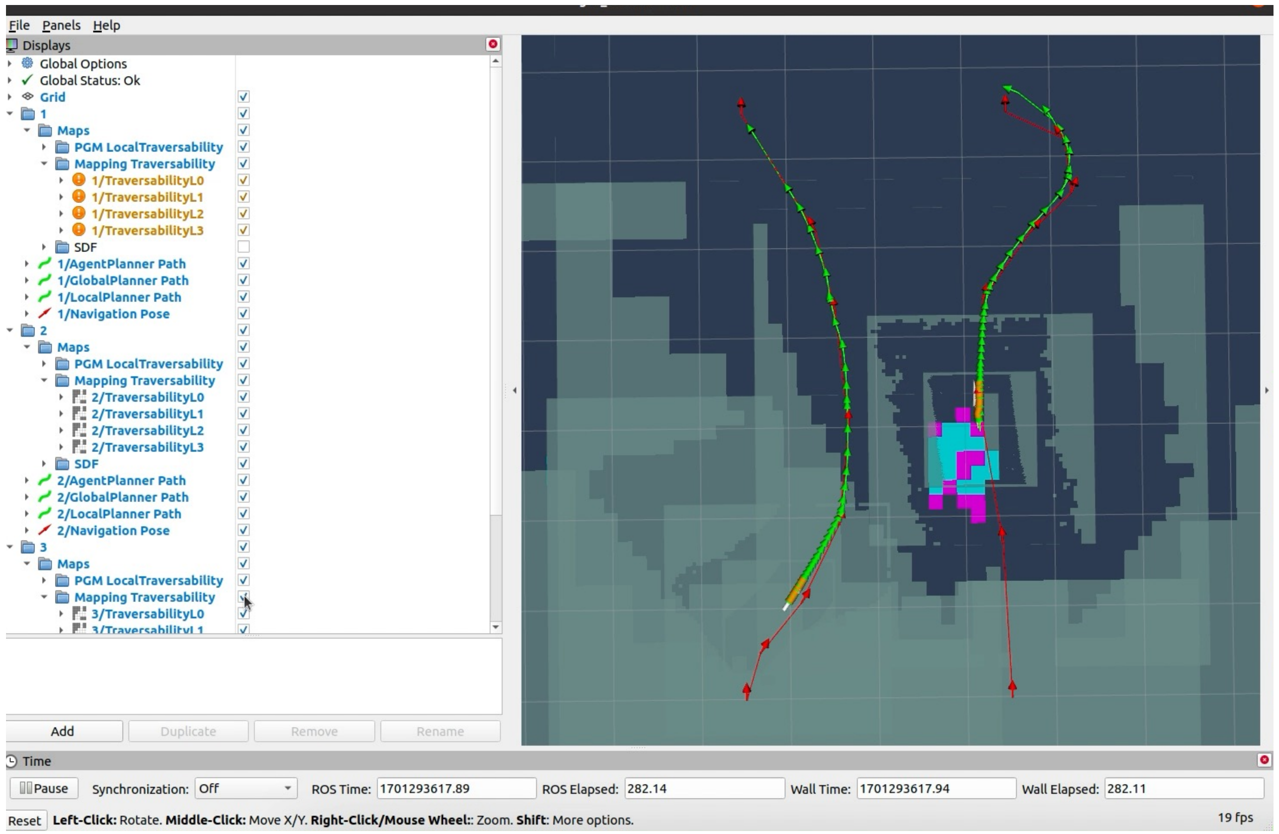
- GLAMM loads in a map as a texture and uses a shader to apply a pixel-by-pixel merge policy from this texture to an existing merged texture (i.e., iteratively)
  - Maps are passed in ordered by resolution (assuming higher resolution is better than lower resolution), and by measurement time (assuming that newer information at the same resolution is more accurate than older data).
- The following merge policy is implemented as the fragment shader, while geometry (e.g., rotation) is left to the vertex shader:
  - If a region has been marked as an obstacle at time t but is free at time $t' > t$ at the same or higher resolution, we mark it as free.
  - If a region has been marked as free at a lower resolution, but it is marked as an obstacle at a higher resolution (even at an older time), we conservatively mark it as an obstacle.

Distributed Mapping

- MoonDB integrates with a Pose Graph Optimization (PGO) module.
    - The PGO module constructs a graph where rover poses are represented as nodes.
    - Visual inertial odometry (VIO) measurements connect poses collected by the same robot.
    - Range measurements from ultra-wide-band (UWB) radios connect poses collected by different robots simultaneously.
- VIO measurements and UWB ranges are collected on each rover, stored in MoonDB, and synchronized to the leader.
- Whenever a local map is stored in MoonDB, a corresponding PGO node is created.
- The PGO module periodically recomputes the set of past and current robot poses to maximize the likelihood of the observed VIO and UWB measurements.
- Modified rover pose nodes are updated in MoonDB, adjusting the corresponding map location if necessary.
- This approach reconciles maps collected by multiple agents, utilizing UWB range measurements to counterpose drift and prevent redundant obstacle copies.



Map Merging



File Panels Help

Displays

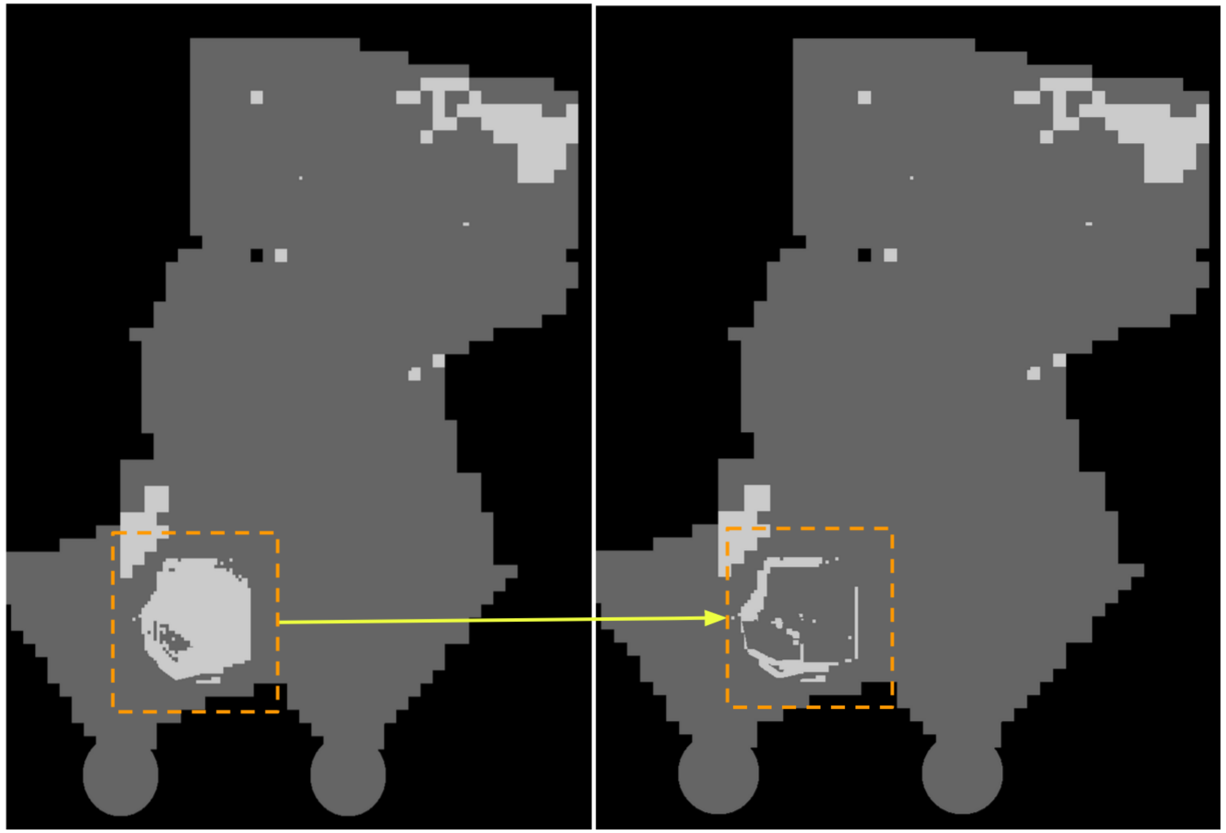
- Global Options
- Global Status: Ok
- Grid ✓
- 1 ✓
  - Maps ✓
    - PGM LocalTraversability ✓
    - Mapping Traversability ✓
      - 1/TraversabilityL0 ✓
      - 1/TraversabilityL1 ✓
      - 1/TraversabilityL2 ✓
      - 1/TraversabilityL3 ✓
    - SDF
  - 1/AgentPlanner Path ✓
  - 1/GlobalPlanner Path ✓
  - 1/LocalPlanner Path ✓
  - 1/Navigation Pose ✓
- 2 ✓
  - Maps ✓
    - PGM LocalTraversability ✓
    - Mapping Traversability ✓
      - 2/TraversabilityL0 ✓
      - 2/TraversabilityL1 ✓
      - 2/TraversabilityL2 ✓
      - 2/TraversabilityL3 ✓
    - SDF ✓
  - 2/AgentPlanner Path ✓
  - 2/GlobalPlanner Path ✓
  - 2/LocalPlanner Path ✓
  - 2/Navigation Pose ✓
- 3 ✓
  - Maps ✓
    - PGM LocalTraversability ✓
    - Mapping Traversability ✓
      - 3/TraversabilityL0 ✓
      - 3/TraversabilityL1 ✓

Add Duplicate Remove Rename

Time

Pause Synchronization: Off ROS Time: 1701293617.89 ROS Elapsed: 282.14 Wall Time: 1701293617.94 Wall Elapsed: 282.11

Reset **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift:** More options. 19 fps



Conclusions and Future Work

MoonDB serves as a storage, sharing, and data fusion solution for multi-robot systems. While this version is highly optimized for CADRE, accommodating mission constraints and limitations, its design allows for expansion to other systems and applications without necessitating substantial redesign.

Future:

- Enhance System Versatility
  - ROS integration
    - Support all standard ROS message types to ensure compatibility across various robotic applications.
  - Develop an Object Relational Mapper (ORM)
    - To enhance adaptability and simplify interactions with databases.
- Improve Scalability and Data Management
  - Utilize batch operations to efficiently handle large volumes of data processing
  - Perform onboard data summarization to reduce bandwidth usage and enhance real-time processing capabilities.
  - Prioritize data transmission based on importance and urgency to optimize resource utilization and response times.

Cooperative Autonomous Distributed Robotic Exploration (CADRE)

CADRE MoonDB: Distributed Database for Multi-Robot Information-Sharing and Map-Merging for Lunar Exploration

Maíra Saboia, Federico Rossi, Viet Nguyen, Grace Lim, Dustin Aguilar, Jean-Pierre de la Croix

Jet Propulsion Laboratory, California Institute of Technology

Presented at the

International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace '24)

Auckland, New Zealand

May 7, 2024

Jet Propulsion Laboratory
California Institute of Technology