Individual Lab Report 5

Lunar ROADSTER

Team I

Author: **Boxiang (William) Fu**
Andrew ID: boxiangf
E-mail: *boxiangf@andrew.cmu.edu*

Teammate: **Deepam Ameria**
ID: dameria
E-mail: *dameria@andrew.cmu.edu*

Teammate: **Bhaswanth Ayapilla**
ID: bayapill
E-mail: *bayapill@andrew.cmu.edu*

Teammate: **Simson D'Souza**
ID: sjdsouza
E-mail: *sjdsouza@andrew.cmu.edu*

Teammate: **Ankit Aggarwal**
ID: ankitagg
E-mail: *ankitagg@andrew.cmu.edu*

Supervisor: **Dr. William "Red" Whittaker**
Department: Field Robotics Center
E-mail: *red@cmu.edu*

April 10, 2025

Carnegie Mellon University
Robotics Institute

1 Individual Progress

Since the last progress review, I worked on debugging the rover's global localization stack. This turned out to be an issue with how the TF tree was set up. Next, I mounted the sensor on the rover and finished the active mapping stack of the rover. Finally, I wrote the skeleton code for the finite state machine behavior tree of the rover's autonomous planning stack.

1.1 Global Localization Debugging

During the last ILR, I mentioned that the global localization stack was very sporadic and would fly off after around 2-3 seconds of starting the localization stack. We were able to successfully debug this on our test day on March 21st. It turned out that the TF tree of the transform of the `total_station_prism` frame was set up incorrectly. In particular, the header information of the total station data was incorrectly set to `base_link`. This meant that whenever the Jetson computer on the rover wirelessly receives an update from the total station, it thinks the `total_station_prism` frame is a distance away from `base_link`. However, since the two frames are rigidly connected, it would also cause `base_link` to move by this amount. This was resolved by setting the header information to the `map` frame. So whenever an update is received from the total station, the `total_station_prism` frame is said distance away relative to the global `map` frame of the Moon Yard.

After finding the bug, Bhaswanth and I spent some time tuning the localization parameters. We were able to find reasonable parameters where the localization worked well. With this, we decided to call it done for the localization stack and shifted our focus to work on other subsystems.

1.2 Sensor Mounting

From the finalized optimal sensor mount placement locations discussed in the previous ILR, I mounted the RealSense stereo camera onto the mast of the rover. The final mount location is $\{X, Y, Z, R, P, Y\} = \{0.5, 0, 0.6, 0, -30, 0\}$ from `base_link`. Figure 1 shows the mounted sensor on the rover.

1.3 Active Mapping

My next task since the last progress review was to work on the active mapping stack of the rover. This included both a local mapping stack and a global mapping stack. Initially, the point cloud feed from the sensor stack goes through a point cloud handler node that processes and removes any outliers in the point cloud data. The local mapping stack then uses the processed point cloud to generate a 2 meter by 2 meter elevation map (with 10 cm resolution) of the obstacles directly in front of the rover (i.e. the map is relative to the `base_link` frame). The global mapping stack uses the processed point cloud to generate a 7 meter by 7 meter elevation map (with 10 cm resolution) of the entire Moon Yard (i.e. the map is relative to the `map` frame) (see Figure 2).

To do this, a Bayes filter was initialized in each grid in the elevation map. Whenever a point cloud data is registered inside that grid, its elevation is taken. Bayes filtering is then performed on this grid that uses the variance of the new datapoint and the variance of the existing datapoints to weight the importance of the new datapoint and add it to

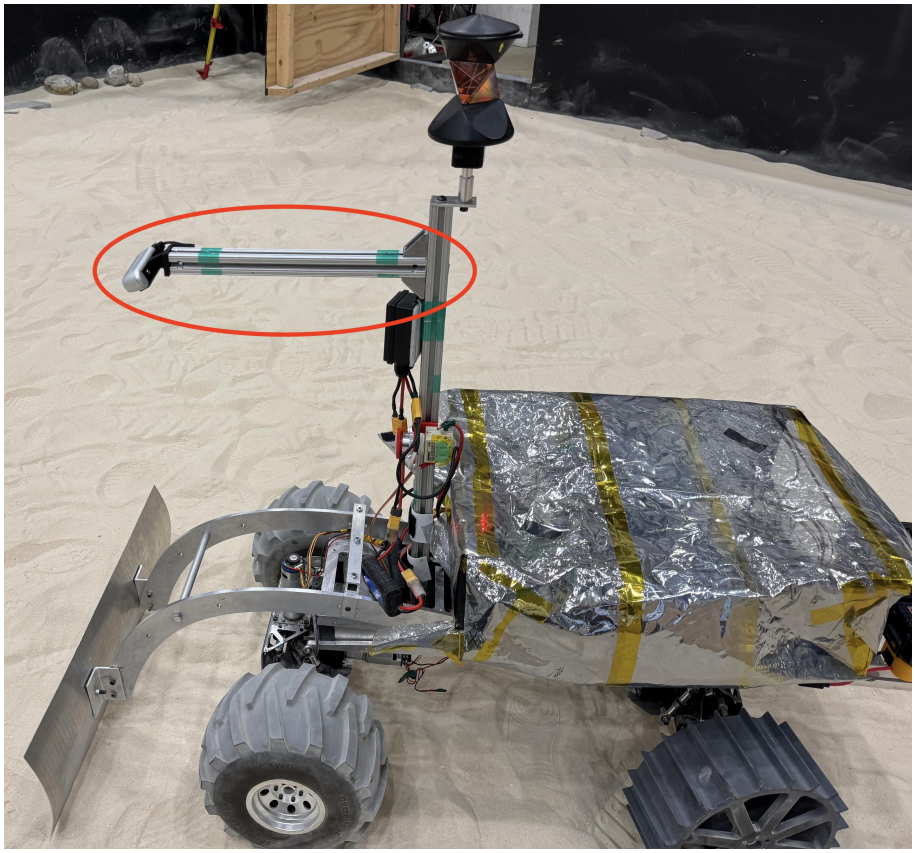

Figure 1: Sensor mount placement

the existing datapoints. Generally, if the variance of the new datapoint is high, then a low importance value is given. If the variance is low, then a high importance value is given. This is done to optimally reduce the noise present in the elevation map.

Additionally, due to the requirements of the planning stack, I also added a map initialization node to the global mapping stack. If an initial point cloud map of the Moon Yard is available (i.e. a prior FARO scan was done) and is saved to the `saved_maps` directory, then the global elevation map would first be initialized with the elevations obtained from the initial point cloud map. Figure 3 shows the initial elevation map generated from an existing point cloud map obtained from a FARO scan.

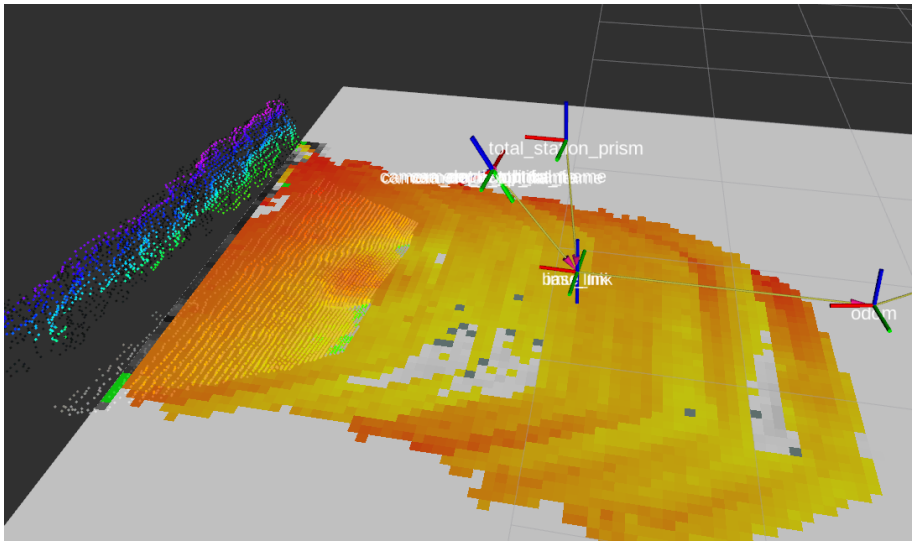

Figure 2: Global active mapping visualization

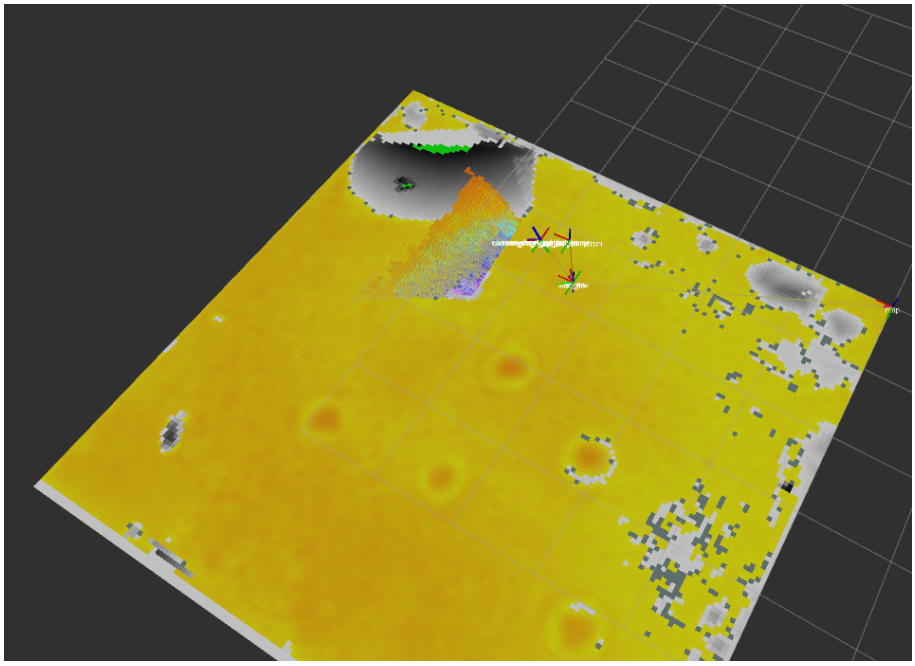

Figure 3: Global active mapping initialization

1.4 Autonomous Planning FSM Behavior Tree

My final task since the last progress review was to write up the skeleton code for the finite state machine behavior tree. The behavior tree is used to dictate the mode transitions and integration between the separate subsystems. The three most important states are the mapping state (UPDATE_MAP), tool planning state (PLAN_TRANSPORT), and the navigation state (GET_TRAJECTORY and FOLLOW_TRAJECTORY). A high-level behavior tree of our planning stack is shown in Figure 4. Whenever a state in the behavior tree is called, it is routed to call another function that exists in a separate cpp file. This allows the work to be divided as each team member is able to work in separate files and combine them once they are finished. For example, I worked on the mapping files, Ankit & Deepam worked on the tool/transport files, and Simson & Bhaswanth worked on the trajectory/navigation files.

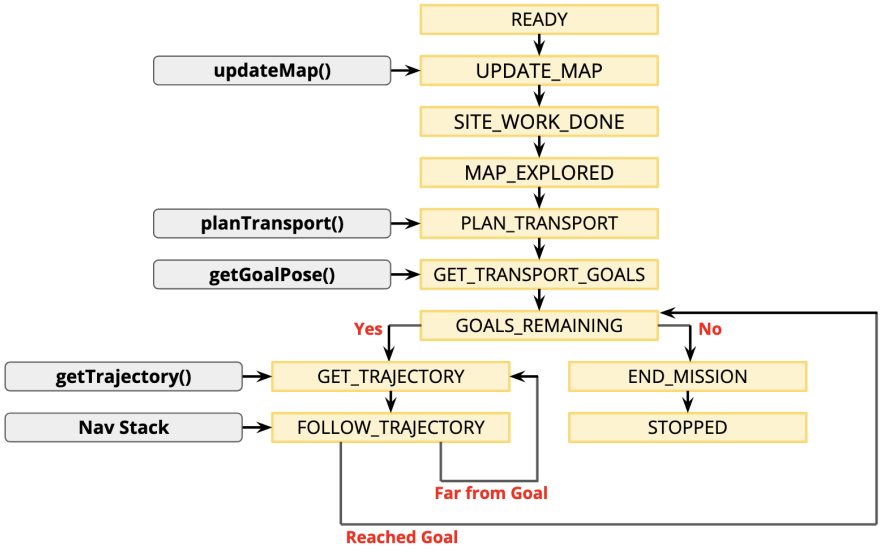

Figure 4: Planning stack behavior tree

After finishing the skeleton code of the behavior tree, I worked on integrating the

UPDATE_MAP state of the behavior tree. This involves reading in point cloud data (either from a pre-loaded map from the FARO scan or from actively mapping the Moon Yard using the stereo camera) and parsing it into an elevation map. Next, the elevation map is saved to a CSV file and saved to the `saved_maps` sub-directory. The format of the CSV is agreed upon by both Ankit and myself so that the data format that is fed into the tool planner stack optimizer is of the correct type. The baton is then passed to my teammates to use my outputs and integrate their subsystems.

2 Challenges

One significant challenge faced was that the mapping nodes keeps on being killed when they are spinning. Seemingly on a random occurrence, the following error log appears in terminal:

```
[mapping_node-1]: process has died [pid 18434, exit code -11]
```

It took some debugging and online searching to figure out the reason. It turns out that the reason is because when the environment is rich and the stereo camera registers a lot of points for the point cloud, the the entire processing stack to filter the data into an elevation map takes longer than the time interval to the next camera snapshot. This meant that the previous point cloud snapshot is deleted (to make room for the new point cloud) and attempting to de-reference the previous point cloud (to make computations) will result in a null pointer. The ROS2 wrapper will raise an error and kill the node to prevent memory access violations.

Another challenge faced was that I originally used a vector of INT8s to encode the elevation map obtained from the mapping stack. The elevations would be in centimeters and saved as integers (so 1 meter corresponded to 100 in the map). This was done so that we can directly use the `nav_msgs/OccupancyGrid.msg` message type in the navigation ROS package. However, the tool planning optimizer is tuned to take in the elevation map as an vector of floats with meters as the unit. This meant my origin elevation map was incompatible with the tool planning stack. We had to sit down and agree upon a common message type between the two subsystems. In the end, I had to change the mapping stack to output a custom message type that saved the elevations as floats to conform with the requirements of the tool planner.

3 Teamwork

A breakdown of the contributions of each team member are tabulated below:

- **Ankit Aggarwal**: My main work was implementing the tool planner and testing it using the FARO laser scan. I worked with William to integrate the Tool Planner into the FSM and he helped me visualize the planner outputs in RViz. I collaborated with Simson, Deepam and Bhaswanth to solder the PCBs. I also worked with them to debug and finalize the wiring connections in the new manufactured E-Box and general hardware debugging of the rover.
- **Deepam Ameria**: For this phase, I have been working on assisting and collaborating with everyone on the team for various tasks. I worked with Ankit, Simson and Bhaswanth to solder the PCBs and also for manufacturing, assembling, testing and debugging the E-box and the other hardware of the rover. I also worked

on understanding and charting out the software architecture, mainly to understand how the planning stack fits into the whole system. I was working on figuring out the visualisation the planning outputs. However, I was running into blockers and ultimately, William was able to complete that task.

- **Bhaswanth Ayapilla**: My initial work was in collaboration with William in debugging the global localization stack and correcting the yaw of the rover. I worked with Simson on the navigation stack and integrating it with localization. We also set up a clean environment in the Moon Yard by removing rocks, flattening the area, and remapping it. This map will serve as the final map for the tool planner to work on. I collaborated with Ankit, Deepam and Simson in soldering the PCBs, assembling the E-box, testing it and troubleshooting issues, and performing quality assurance of the entire hardware setup.
- **Simson D'Souza**: My primary responsibility was setting up the navigation stack and integrating it with the localization stack, which was done in collaboration with Bhaswanth. Additionally, the Moon Yard was scanned using a FARO scanner with Bhaswanth's assistance. I stitched the scans together and defined a new map frame origin. This new FARO scan data will be used with the navigation stack and the tool planner stack. Alongside Bhaswanth, Deepam, and Ankit, I contributed to soldering PCBs, assembling the E-box, wiring new connections based on the updated E-box design, and troubleshooting hardware issues on the rover.

4 Plans

From now until SVD, I will be mainly working on the integration of the various subsystems of the rover. In particular, I plan on mostly working on the integration between the mapping subsystem and tool planning subsystem. Since SVD is approaching on April 17, I will be wrapping up on implementing new functionalities and instead focus my effort on integration and testing out everything works as intended for the demonstration.