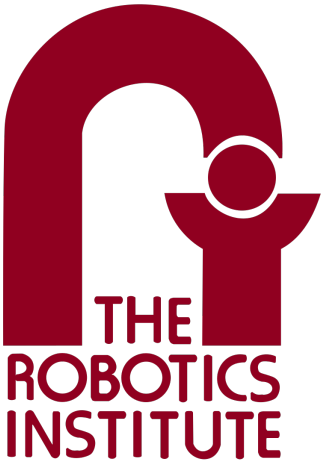
Individual Lab Report 9



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*

October 30, 2025

Carnegie Mellon University
Robotics Institute

1 Individual Progress

Since the last progress review, I worked on two main tasks. The first task was on re-implementing the validation stack using a new method that does not discretize the point cloud into grids. My second task was on a refinement of the Skycam localization unit. The previous method used a neural network and had convergence issues. My new implementation uses classical computer vision to solve the same problem.

1.1 (New) Validation Unit

The validation unit is used after each dozing run to determine the performance of the grading done to the crater. The previous implementation of the validation unit shown in Progress Review 9 discretized the point cloud feed into grid cells. A finite difference filter (e.g. Sobel, Scharr) would be applied on the grid cell to estimate gradients. However, this method was not very accurate as the grids were discrete (normally set to 5cm x 5cm). To get more accurate gradients, we will need to decrease the grid sizes. However, this would cause compute to increase drastically. Therefore, I decided to re-implement the validation stack entirely and remove the discretization step. Gradients would be calculated directly from the point cloud. The methodology is as follows:

The validation unit first obtains a point cloud from the ZED camera. It uses voxels to downsample and merge adjacent points for computational efficiency. It then calls the OpenMP library to estimate the surface normal from the point cloud. Next, excess gradients (i.e. walls and rocks) are filtered out. Slopes are then calculated from the surface normal by calculating the angle between the normal and the vertical z-axis. A smoothing operator is applied next to filter out phantom points. Finally, the process is run over a window period (normally 10 point cloud messages) and averaged to filter out any point cloud errors. Figures 1 and 2 shows the validation results on a crater and smoothed surface respectively.

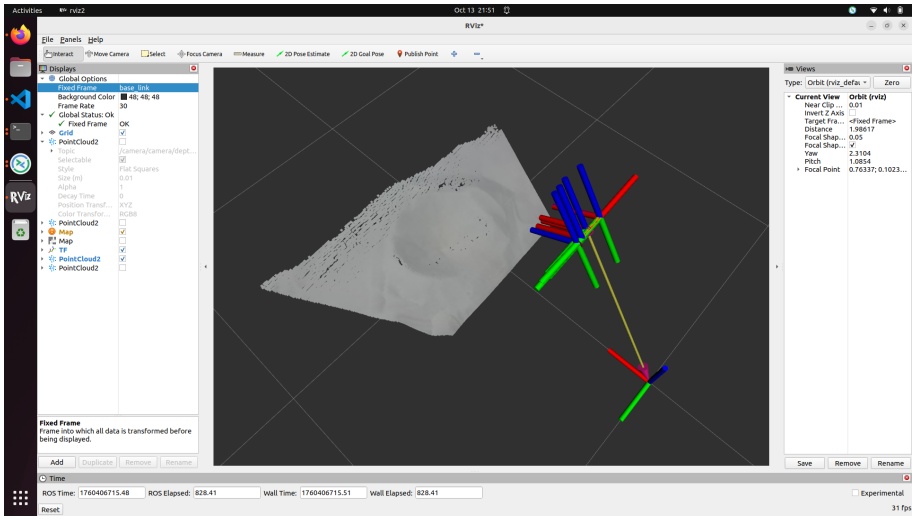


Figure 1: Validation on crater: Validation Success = False, Max Slope = 17.93 deg

The unit was refactored as a ROS action rather than a ROS topic. This was done so that the perception stack and behavior executive node (BEN) stack is able to use it more effectively. A custom ROS action is used to enable the unit. Intermediate feedback is provided when the verbose parameter is turned on. Once the validation unit completes its computations, the final result is returned. This includes a boolean (grading success/failure), gradient information (max, mean, RMSE), and z-axis elevation

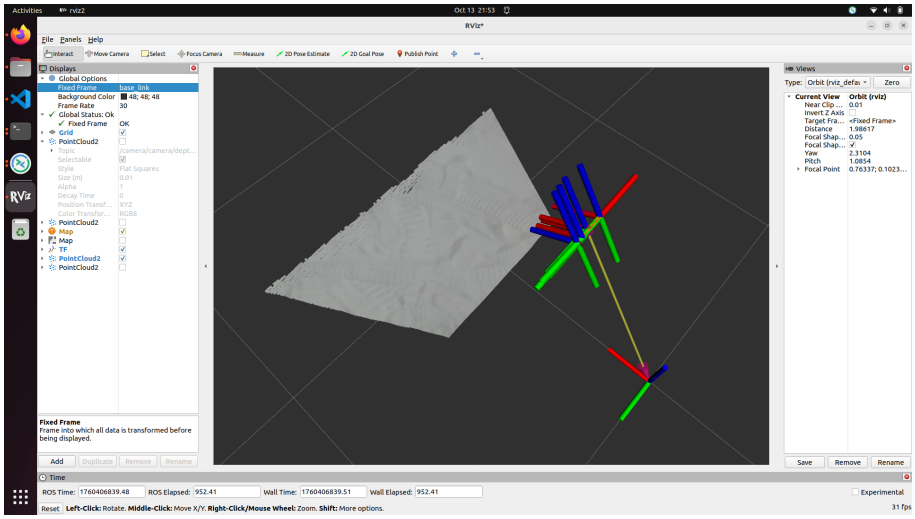


Figure 2: Validation on flat surface: Validation Success = True, Max Slope = 1.73 deg

information (min, max, average). The custom ROS action implementation is shown in Figure 3.

RunValidation.action

src > lr_msgs > action > RunValidation.action

```
# ---- Goal ----
int32 average_window
bool do_second_pass
int32 timeout_sec

---
# ---- Result ----
bool grading_success
float64 mean_gradient
float64 rmse_gradient
float64 max_gradient
float64 min_z
float64 max_z
float64 avg_z
float64 max_traversal_slope
string frame_id

---
# ---- Feedback ----
int32 collected
float64 current_mean_gradient
float64 current_rmse_gradient
float64 current_max_gradient
float64 current_min_z
float64 current_max_z
float64 current_avg_z

```

Figure 3: ROS action for validation unit

1.2 (New) Skycam Unit

My next task was implementing the Skycam unit. The pipeline was entirely revamped since the last progress review. The neural network estimator proved to be too inaccurate to be used for localization. Moreover, a new tarp covering was added to the Moon Yard. This covered a significant percentage of the field of view and caused the old data to be useless in the new environment. Motivated by these shortcomings, I decided to use classical computer vision to solve the same problem. The method is an adaptation of Andy Sloane's "Fast indoor 2D localization using ceiling lights" algorithm here. The algorithm's methodology is as follows:

The ceiling tracker skycam node is initialized and a mask is calculated so that the tarp covering the Moon Yard is filtered out. This was done to save compute and only use pixels that contain useful information. Next, the node precomputes a fisheye UV-map and run-length mask, undistorting each pixel using the calibrated lens and filtering out rays outside the valid ceiling region. During runtime, it extracts the pixels above the brightness threshold, and estimates the ceiling pose (u, v, θ) via a closed-form Levenberg–Marquardt grid alignment. In other words, the bright pixels are assumed to be the ceiling lights. This is regressed to a known ceiling light map above the Moon Yard to solve its estimated pose in camera coordinates. The result is converted to world coordinates and published.

Testing was performed against the ground truth localization using the total station. After calibrating and tuning, the tracker was quite accurate compared to the total station localization (see Figure 4). However, the localization drifts and the accuracy degrades when the camera experiences roll and pitch (see Figure 5). The reason is because the closed-form Levenberg–Marquardt solution assumes the camera has zero roll and pitch. My next step is to make the localization robust for non-zero roll and pitch.

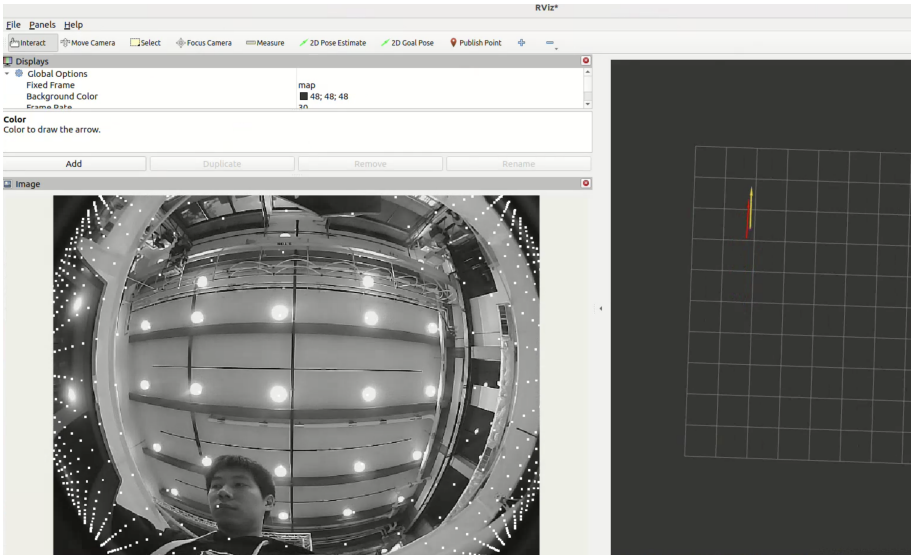


Figure 4: Ceiltracker working under no roll and pitch

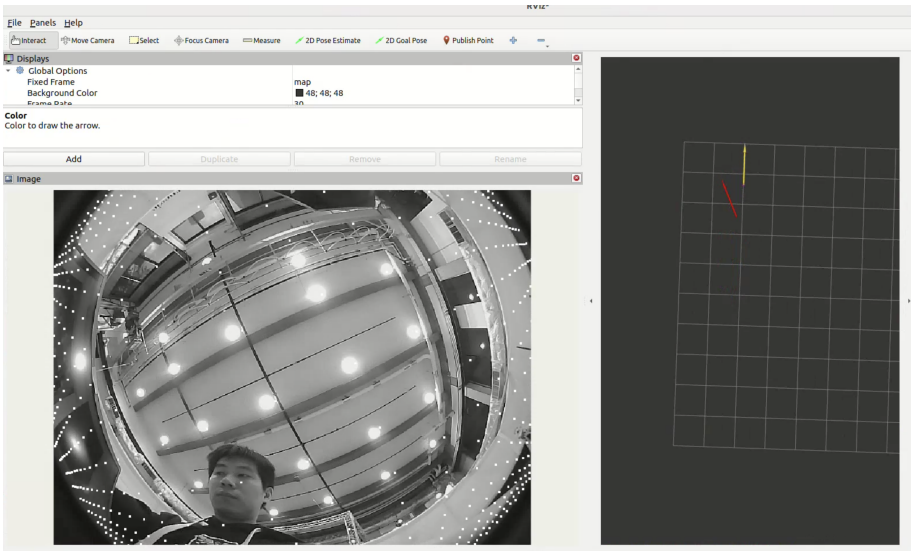


Figure 5: Ceiltracker failing under roll and pitch

2 Challenges

As discussed in the above section, one challenge currently facing the ceiltrack skycam node is that roll and pitch significantly degrades the localization accuracy. Two solutions are currently being explored. The first solution would be to solve it mechanically using a gimbal. The skycam camera would be mounted on the gimbal so that the roll and pitch experienced by the rover would not propagate to the camera. A second solution is to obtain the roll and pitch values from the IMU and post-process the image to remove roll and pitch using transformations. Both of these solutions are currently being investigated to solve the problem.

A second challenge was that the ZED camera would sometimes give out phantom points for the point cloud. This is most likely because the lunar regolith is relatively featureless, so the camera is unable to pinpoint its location using only stereo cameras. This meant that the gradients calculated from the validation stack would sometimes be inaccurate. To solve this, a bilateral pass smoothing operator is passed through the raw (down-sampled) point cloud so that the phantom points are smoothed against the points in its neighborhood. This operator proved to be sufficient to solve the phantom points issue for the validation stack.

3 Teamwork

For this progress review, I collaborated with Ankit and Deepam that were implementing the planning/perception pipeline and solicited their requests for what outputs are required for the validation unit. The validation unit was originally implemented as a ROS topic. It was changed to a ROS action at their request. Additionally, the z-axis elevation information was added as an output of the validation stack. This will be used in the planning pipeline to adjust the tool height dynamically. I also collaborated with Simson and Bhaswanth that were implementing the navigation stack. They provided me with accuracy requirements for my Skycam unit. The failure to meet these accuracy requirements was what necessitated me to re-implement the entire Skycam unit. A breakdown of the contributions of each team member are tabulated below:

- **Ankit Aggarwal**: Ankit primarily focused on parsing robot poses using crater geometry from Perception and integrating it with the previous planning methodology. He worked with Deepam and Bhashwanth to ideate how the navigation stack will use the robot poses for manipulation and debugging teleoperation issues. Ankit worked with Simson to fix the ongoing steering slip issue. Deepam and Ankit also worked with William to finalize how the validation and perception stacks will interact. Finally, he worked with the whole team to finalize the Behavior Executive Node for complete subsystem integration.

- **Deepam Ameria**: Deepam's work since last PR was focused on implementing the Perception Stack online on the Jetson Orin. He ported the code to the Orin, and implemented confidence threshold for robust detections. He used camera intrinsics to determine the centroid of the crater, and the bounding box edges to determine the crater diameter. These values will be published on a topic, and can be used by the planning stack to plan robot poses for the mission. Deepam also worked with Ankit and Bhaswanth to ideate how the planning stack will be used by the navigation stack. Deepam also took ownership of the Standards and Regulations task, researching and compiling information and the application of the standards chosen by the team. He collaborated with the entire team to finalize

the workflow of the Behavior Executive Node, which is crucial for the integration of all the subsystems.

- **Bhaswanth Ayapilla**: Bhaswanth's primary work involved working on the navigation stack. The global planner has been tested and tuned on the final map of the environment and deployed on the rover. He worked with Simson in the global navigation controller. They collectively tested and tuned the complete navigation stack, along with William's help. Bhaswanth helped Ankit and Deepam by fixing teleoperation issues. Ankit and Simson helped in fixing the steer issues, which was crucial for navigation. Finally, he worked with Ankit and Deepam ideate how the planning stack will be used by the navigation stack.
- **Simson D'Souza**: Simson primarily focused on the navigation stack and the generation of the final global costmap. The global navigation controller was modified to enhance navigation performance. Bhaswanth and Simson collaborated on tuning the navigation stack to improve overall accuracy. Additionally, Ankit and Simson worked together to resolve the steering slippage issue. As a team, we also collaborated on planning the workflow for the Behavior Executive Node, which is crucial for integrating the various modules and addressing potential integration issues in advance.

4 Plans

From now until progress review 11, I will be mainly working on solving the Skycam localization issue under non-zero roll and pitch. However, since Skycam localization is a FVD Encore goal, I will be prioritizing my time in implementing the behavior executive node (BEN) stack and integrating the subsystems. I will also be testing and performing quality assurance on the localization and validation packages so that the subsystems operate as intended for FVD and FVD Encore.