Individual Lab Report 8

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 9, 2025

Carnegie Mellon University
Robotics Institute

1 Individual Progress

Since the last progress review, I worked on two main tasks. The first task is on implementing the validation unit to gauge the crater's gradient after each dozing run. The second task is on a preliminary implementation of Skycam localization to localize the robot in the Moon Yard using only a stereo camera and predefined landmarks. This included a data collection pipeline, training pipeline, and deployment pipeline.

1.1 Validation Unit

The validation unit is used after each dozing run to determine the performance of the grading done to the crater. The unit takes in a boolean indicator from a topic sent by the Behaviour Executive Node (BEN) to begin validation. The unit would then use the most recent point cloud feed from the ZED camera and perform RANSAC on it to fit a plane. The points would then be discretized into grid cells. The average height of the points in each cell would be used to calculate a height map. Gradients would then be calculated using finite differences. A filter would then be applied to remove excessive gradients as they mostly come from walls and/or rocks in the Moon Yard. Once everything is processed, it sends a boolean back to the BEN to indicate whether the gradient is acceptable (i.e. below 5 degrees). Additional information would also be displayed in the terminal as seen in Figure 1.

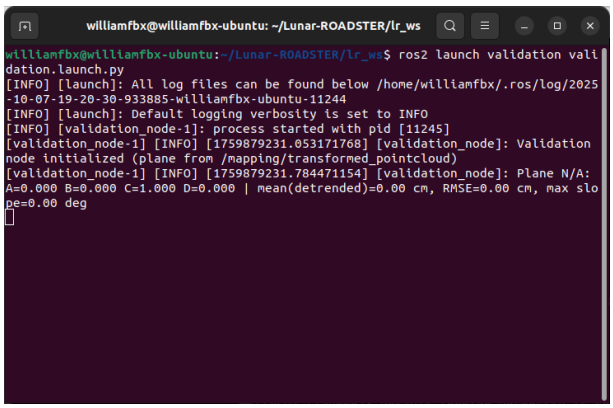

Figure 1: Terminal output of validation unit

During the implementation phase, various different gradient operators were trialed. These included finite differences, Sobel, and Scharr operators. I will be further testing and tuning the gradient operators to see which one works best experimentally in the Moon Yard. Different grid cell sizes (2cm, 5cm, 10cm), wall height thresholding, and max slope thresholding parameters were also implemented. Such parameters can be easily swapped at each runtime using a YAML config file. I will be further fine-tuning the parameters once I test the validation unit in the Moon Yard. An example visualization of the validation unit's grid map output is shown in Figure 2.

1.2 Skycam Unit

A stretch goal for our project is to remove the usage of the total station for localization and rely solely on onboard sensors for localization. The Skycam unit attempts to solve this problem by localizing itself using features above the rover (i.e. ceiling lights, piping, etc) using only a monocular camera. The current implementation does this by training a neural network to essentially overfit the environment relative to its ground truth position values. The input is a RBG image from the Skycam camera, and output

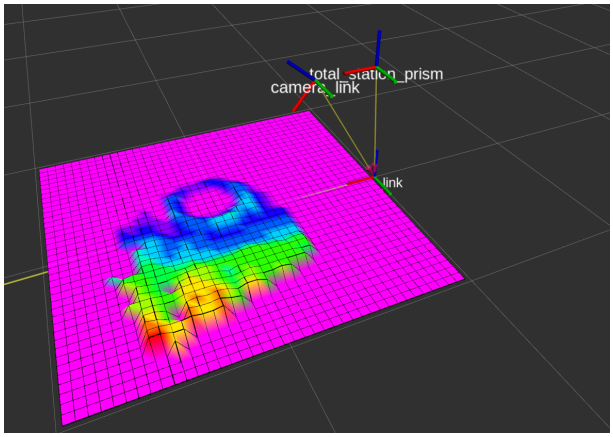

Figure 2: Visualization output of validation unit

is positioning values (x, y, z, roll, pitch, yaw). We obtain ground truth output values via our original (total station) method to perform supervised learning and fit a neural network. At runtime, the network should hopefully recover the learned output values when fed an input image stream of the learned environment. For this task, a data collection pipeline, training pipeline, and deployment pipeline were implemented. Next I will discuss each separately.

1.2.1 Data Collection Pipeline

The data collection pipeline is implemented as a ROS2 node. This node listens to the raw Skycam camera stream, total-station prism poses, and an IMU topic. At a fixed Hz rate, it saves a sample only when an image and a prism pose are both available and their timestamps differ by less than `max_time_diff_sec` (usually set to 1 second). For each valid sample it:

1. Saves the image to `output_dir` using a hash basename
2. The image filename is appended to either `train_files.txt` or `eval_files.txt` depending on the selected mode
3. Appends a line to `labels.txt` with (x, y, z, roll, pitch, yaw) and image/pose timestamps (ns)

Using this method, I saved approximately 500 images of the Moon Yard ceiling and their corresponding ground truth locations for training. The data collection apparatus is shown in Figure 3.

1.2.2 Training Pipeline

The image with its corresponding (ground truth) positions from the data collection pipeline is used to train a neural network using PyTorch. The datasets are loaded in to mini-batches and sent to device. MSE loss is calculated from the estimated position and ground truth position. The loss is back-propagated using ADAM to update the network. Vanilla neural network and Convnet neural network architectures were tested. However, the MSE loss plateaus around 1 meter from the ground truth (see Figure 4). My goal going into the next Progress Review is to reduce this loss to around 10 centimeters (so the predicted location from the neural network is less than 10 centimeters from the ground truth given by the total station). Once training is complete, it saves the network as a state dictionary ready to be used.

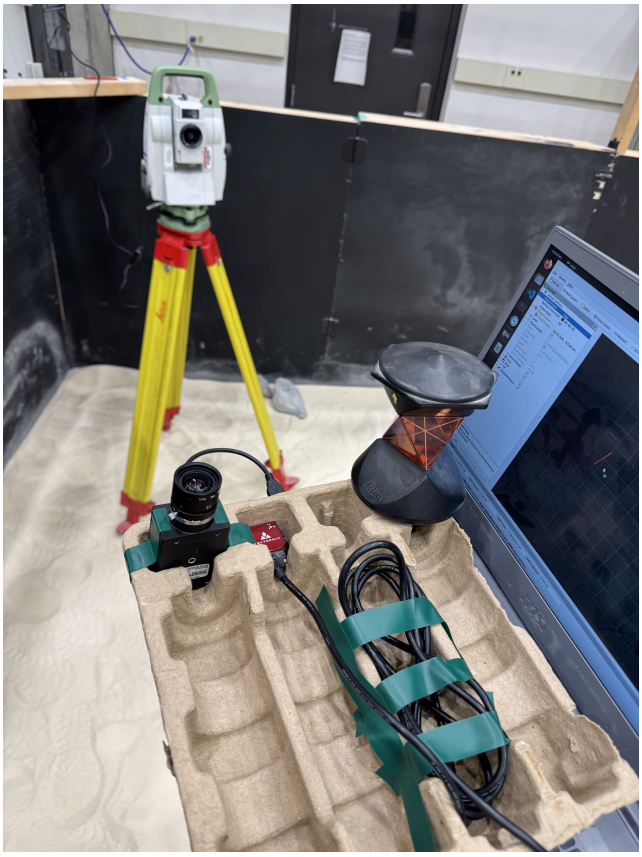

Figure 3: Skycam data collection apparatus

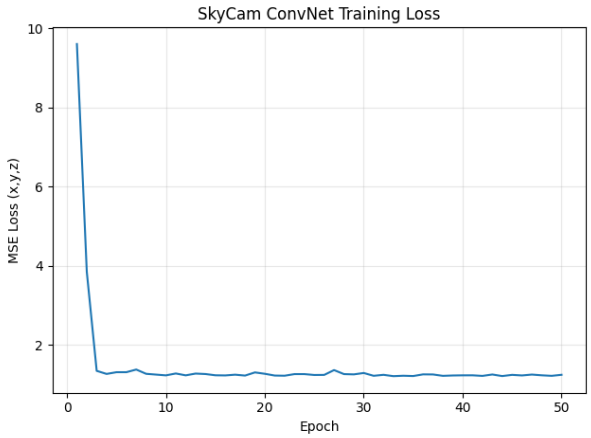

Figure 4: Training MSE loss of the Skycam network

1.2.3 Deployment Pipeline

The deployment pipeline is implemented as a ROS2 node. It loads the trained TorchScript Skycam model and subscribes to the raw Skycam image feed. It then pre-processes the image (resizing and color conversion) runs a forward pass through the network. It obtains positional data from the output of the network and publishes this to the original localization topic published by the total station. This allows the total station and the Skycam to be used interchangeably for localization without any changes to other packages.

2 Challenges

A challenge faced during the implementation of the validation unit was that the ZED camera had a slight roll angle relative to the rover's base link. This meant that the point cloud for a flat surface would be slightly tilted off-axis and one side of the flat surface would appear higher in the grid map when in reality the entire region should be the same height. This caused a slight residual gradient (of approx. 4-6 degrees) due to the tilting even though the ground is flat. This took some time to figure out the issue. Eventually, RANSAC is used to first fit a plane before elevations were calculated to offset this roll angle error.

Another challenge is that the Skycam data collection pipeline is not very accurate. The ground truth locations are sent from the total station to a TX2 chip, and then relayed via WiFi to my computer. The relay via WiFi sometimes causes the total station data to be clustered together so that in one instance four or five location updates are received at once. I am still currently working on solving this problem. A potential solution is to have a long connector wire connecting the total station to the laptop directly. This foregoes the latency issues of the TX2 chip and WiFi module.

3 Teamwork

For this progress review, my progress was mainly siloed and did not involve much collaborative work with teammates. However, I did collaborate with team members by lending a hand on hardware issues and solicited their requests for what outputs they required for the validation unit and Skycam unit. For example, the validation unit outputs both a boolean (done/redo) and the gradient in degrees. This will be used later by Deepam and Ankit as an input to their planning stack. The Skycam unit outputs to the same topic as the original total station did. This will be used later by Simson and Bhaswanth for their navigation and controller units. A breakdown of the contributions of each team member are tabulated below:

- **Ankit Aggarwal**: Ankit collaborated with Deepam on implementation of the baseline perception stack. This involved data collection, annotation and training the model. He also collected more data for the YOLO model with the entire team. Ankit collaborated with Simson, Deepam and Bhaswanth to debug and refine hardware. He was also responsible for the wiki proposal.
- **Deepam Ameria**: Deepam's primary work involved implementation of the perception stack. He collaborated with Ankit on this task and trained the YOLOv8 model on the custom crater dataset that Simson, Bhaswanth and William helped create. He also worked with Bhaswant, Simson, and Ankit to debug and refine hardware. Moving forward, he will work on extracting the position of the crater from the detected bounding boxes, and work with the team to test and integrate the subsystem with the existing stack and robustify the online inference.
- **Bhaswanth Ayapilla**: Bhaswanth's primary work involved working on the global planner. The planner has been tested on a preliminary occupancy grid map. He also worked with Simson in brainstorming ideas for the global navigation controller, which is Pure Pursuit. Bhaswanth worked with Simson, Deepam and William in making the test bed and mapping the Moon Yard. He also helped Ankit and Deepam in making their dataset for the YOLOv8 by creating craters and taking pictures. Finally, Bhaswanth worked with the entire team in debugging hardware issues and finally get the rover moving.

- **Simson D'Souza**: Simson worked on implementing the global navigation Pure Pursuit controller, and the coding for it has been completed. Navigation tuning and testing with the global planner are pending and will be carried out in collaboration with Bhaswanth. In addition, he prepared the Moon Yard by creating craters of various sizes to closely replicate the expected layout of the Fall Validation Demo environment and performed FARO scanning to generate a 2D costmap. He also contributed to collecting crater data for the YOLO model along with the entire team and collaborated with Deepam, Bhaswanth, and Ankit to debug and refine the rover's hardware.

4 Plans

From now until progress review 10, I will be mainly working on fine-tuning the validation unit parameters in the Moon Yard. I'm hoping to get more accurate gradient results from the validation node so that it is robust to the movement of the rover. Additionally, I'm also hoping to finish the Skycam unit and output localization to within 10 centimeter accuracy. This will include tasks such as changing the data collection pipeline and tuning the network architecture.