The Robotics Institute, Carnegie Mellon University

Final Report
December 16, 2020

MRSD Project - Pit Navigator

Team G - The Pit Crew
Alex Withers
Awadhut Thube
Justin Morris

Sponsor - Dr. William Whittaker

ABSTRACT

Orbital imagery of the moon has identified several deep pits in the lunar surface, and there is great interest in exploring these pits to determine if they could provide access to subsurface caves where future human habitats could be constructed. A team at CMU proposes to explore one such pit by means of a small, autonomous rover that would circumnavigate the rim of the pit and capture images of the interior walls. Such a mission would require specialized planning and navigation code to keep the rover safe in the treacherous terrain near the pit and manage the process of feeding data back to Earth via a radio-equipped lander module. This project is concerned with designing the software functions necessary for the rover to operate in the pit vicinity, including brinkmanship navigation, path planning, and risk assessment. The scope of the project also includes the development of a rover surrogate and a simulated lunar environment, both used to enable testing of the software that is the focus of the project. This report details progress on the project from its inception to the date of the final demonstration of the integrated system.

Contents

1 Project Description

Studies of the moon's surface have led scientists to hypothesize the existence of a network of sublunarean tubes hundreds of feet across, extending for kilometers below the regolith. If these tubes exist, they could serve as the foundation of future human habitats on the moon. They would provide an enclosed space that would shield occupants from the radiation and extreme temperatures of the moon's surface, and could be sealed and pressurized with breathable air. Although the existence and extent of these tubes remains a theory, there is direct evidence showing that the moon's surface is studded with large pits, some dozens of meters across. These pits sink deep enough into the moon that they could potentially connect to the tube network. Because of the exciting potential of the sublunarean tube structures, these pits have become a top priority for future research and exploration.

Astrobotic Technology, in cooperation with Carnegie Mellon, proposes to send a small, fast, autonomous rover to explore these moon pits and collect data about their composition and structure[1]. This rover will have between one and two Earth weeks to travel to and collect images of a moon pit near its landing site. This process will take the robot beyond the range of wireless communication with the lander craft, which is its only connection back to Earth. Therefore, the rover must be able to operate autonomously while executing its mission, and return back to the landing site safely.

Any autonomous mission that must operate in the vicinity of a moon pit must have special routines for navigating around the pit edge in a safe and efficient manner. This project proposes to design and implement software that will generate and execute safe routes to a series of waypoints around the edge of a pre-selected pit. At each waypoint, the rover will slowly advance towards the pit edge while assessing the terrain, moving as close to the pit as possible without endangering itself. When the rover determines that it cannot safely move any further, it will collect images of the opposite wall of the pit, then back away and proceed to the next waypoint. The rover will aim to photograph as much of the pit's circumference as possible over the course of its mission duration.

2 Use Case

A small autonomous rover is exploring the surface of the moon. It arrived aboard a lander, which set down near the known location of a pit. One of the rover's directives is to gather imagery of this pit and transmit that imagery to the lander, which will then forward the collected data on to Earth. The rover was deployed from the lander, and has since been executing various mission objectives, the most recent of which was to autonomously travel to a waypoint set for it by human operators on Earth. This waypoint is in close proximity to the pit, and is outside of the communication range of the lander, meaning that the rover must operate with complete autonomy during its trek to this waypoint.

The pit's location is defined by a map of the area around the landing site that was generated before the mission launched. Prior to the start of the rover's mission, the lander provided the rover with exact coordinates of the lander's location in this map. The rover localizes itself and the pit in relation to these coordinates for the duration of its mission. Human operators will also pre-set a series of waypoints that form a circle around the pit, which will guide the rover as it circumnavigates the pit. These waypoints will be placed at locations that are guaranteed to be a safe distance from the edge of the pit.

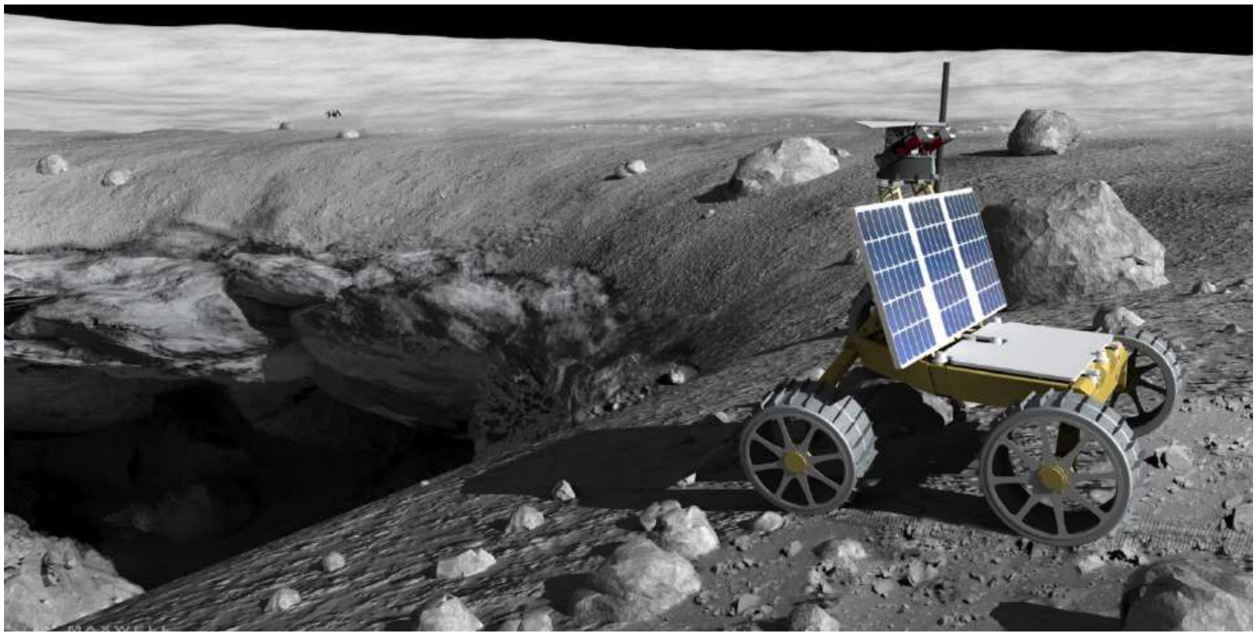


Figure 1: Rover at Pit Edge

Upon arrival at the first pit-proximate waypoint, the rover software activates the Pit Navigator routine, which employs specialized behavior designed for successful navigation in the vicinity of the pit. The rover orients itself towards the center of the pit, and constructs a mesh representation of the terrain directly in front of it. The rover then moves slowly towards the pit, monitoring the terrain mesh to detect the pit edge once it comes into view of the rover's sensors. When the rover determines that it can no longer move forward safely, it stops advancing towards the pit. A graphic that demonstrates this situation is shown in Figure 1.

From this vantage point, the rover uses the pan and tilt actuators on its pit imagery camera to align the camera such that the camera's field of view is trained on the pit wall opposite the rover's location. The camera pans from side to side, taking images at a range of angles in order to capture an arc of the pit wall. These images are combined into panoramas which represent horizontal strips of the pit interior. The panoramic images are used to construct a 3D model of the pit wall.

Once sufficient images have been captured, the rover uses its model of the pit to select a new waypoint at a different point on the pit's circumference, and plots a new path to reach that waypoint. This path takes the rover away from the pit edge before moving around the perimeter of the pit, to ensure that the rover will not fall into the pit while navigating to the next waypoint. A local planning algorithm generates a route to each new waypoint consisting of multiple small movement steps. The algorithm avoids any obstacles detected in the rover's vicinity and prevents the rover from moving too close to the pit edge, then selects the most optimal route to the destination position with these constraints applied.

The rover continues to select waypoints around the edge of the pit and take images at each associated vantage point until the sun sets and the rover runs out of battery power. As more images are captured, the 3D model of the pit wall becomes increasingly complete. The rover's goal is to collect data on as much of the pit's wall area as possible.

At all times during its mission, the rover is measuring how much data it has collected. In order to ensure that as much of this data as possible is transmitted safely to Earth, the rover will leave the vicinity of the pit at regular intervals in order to return to within communication range of the lander and deposit all collected data to the lander. When the rover leaves the vicinity of the pit, the Pit Navigator routine ends and control is returned to the standard moon operation subsystems.

The data transmitted to the lander includes rover telemetry and navigation images in addition to the pit imagery collected at each of the vantage points. Once data is stored on the lander, the rover returns to the pit once again to collect more imagery for as long as it has sufficient power to continue operating. The lander computer transmits the data to Earth, with highest priority given to the 3D model of the pit.

3 System-Level Requirements

The following requirements are derived from an objectives tree by taking into consideration the mission goals and sponsor expectations. The requirements are categorized as mandatory and desirable requirements. These categories are further divided and the requirements are classified as performance requirements which are functional requirements with an associated performance measure and non-functional requirements. Our performance requirements are mainly concerned with the amount of data captured and also with the reliability of our system.

3.1 Mandatory Performance Requirements

The system will:

M.P.1 Capture **500 MB** image data on a surface similar to the moon terrain. *(Cam-Op, Planning)*
This specifies the total data captured during the mission should be more than 500MB. We arrived at the specific number assuming that the camera has very high resolution (assumption is that we capture 4K images).

M.P.2 Capture **75 MB** image data over a single cycle in the mission. *(Cam-Op, Planning)*
A mission cycle is defined as one complete loop which consists of the data collection activity and the data dumping activity of the rover. We expect to complete 6 such cycles and collect around 75MB of data in each cycle.

M.P.3 Capture **15 MB** image data from specific coordinates on the surface of the moon. *(Cam-Op)*
Within a single cycle, the rover visits multiple waypoints and gathers data from each of these waypoints. This requirement specifies that it must record 15MB worth of data at each waypoint.

M.P.4 Calculate the relative distance to the pit edge within **2% error**. *(Brinkmanship)*
While determining the presence of an edge in the surrounding, our system must also estimate the distance to the edge with 2% of error.

M.P.5 Calculate an optimal navigation plan within **20 seconds**. *(Planning)*
Before moving to the next waypoint, our system must compute an optimal navigation plan within a time frame of 20 seconds.

M.P.6 Capture images covering **20°** angle of pit circumference from one position. *(Cam-Op)*
Once the rover reaches near the edge of a pit, it captures multiple images of its surrounding. This requirement specifies that it must cover about 20° angle of pit circumference from one position.

M.P.7 Operate such that the chance of a mission ending incident is less than **5:1**. *(Planning)*
The rover should update its plan whenever it determines that the risk of executing the current plan has exceeded this threshold. This is one of our most important requirements which emphasizes on the reliability of the system.

Apart from the above mentioned functional requirements, our system should meet the following non-functional requirements. The non-functional requirements stem from the fact that our system is a part of a larger mission and must adhere to the operational standards and interfacing capacity of the larger system.

3.2 Mandatory Non-Functional Requirements

The system shall:

M.N.1 Operate in the vicinity of a pit on the moon. *(General)*
The system is designed with a very specific use case in practice. However, it may not be feasible to test this use case given the scope of our project. Hence, we have this as a non-functional requirement.

M.N.2 Operate using hardware that meets specifications of rover design and mission. *(General)*
Along with demonstrating the proof of concept, we would like to replicate the mission as closely as possible. Thus, we plan to operate within the specifications of the overall rover design.

M.N.3 Operate within a Linux operating system environment. *(General)*
Initial work in this project was carried out in the linux operating system environment. Hence, operating with the same environment adds as one of our functional requirements.

M.N.4 Be compatible with other software systems running on the rover. *(General)*
Before we started working on the project, some basic functionalities were set up through previous work on the project. Hence, we aim to utilize those as much as possible and develop our own system accordingly.

M.N.5 Maintain a mission clock. *(General)*

Maintaining a mission clock is essential for mission planning and this is incorporated as one of our non-functional requirements.

M.N.6 Operate when rover is not experiencing any major subsystem faults. *(General)*
Finally, our system shall operate efficiently and reliably in the absence of any major subsystem faults.

In addition to the mandatory performance and non-functional requirements, we have also identified certain desirable requirements. These additions are nice to have and extend the project scope in exchange for having a more robust, reliable and valuable system. The desirable requirements are formulated to extract the greatest amount of information even when operating under different conditions.

3.3 Desirable Performance Requirements

The system will:

D.P.1 Operate at a distance of **0.75 meters** from the pit edge 80% of the time. *(Planning)*
This requirement is designed to strike a balance between allowing the rover to remain at a safe distance from the pit edge whenever possible, while also keeping the rover near the pit to minimize the travel time between waypoints.

D.P.2 Estimate the shape and size of the pit within **10% error**. *(Planning)*
An aspect of the mission that is being developed in parallel to our project is the creation of a 3D model of the pit based on the images of the interior that the rover captures. We hope to be able to use this model to improve the accuracy of our planning map, so that the waypoints chosen are appropriately placed around the rim of the pit.

D.P.3 Capture high resolution images of the pit with each image being **18 MB** in size. *(Cam-Op)*
The better the resolution of the images captured, the more valuable those images are to scientific research, and to the creation of the 3D model. However, the rover has limited storage, and the lander's connection to Earth has limited bandwidth, which will reduce the maximum amount of information that can be communicated.

D.P.4 Capture data such that for **80%** of the images **60%** of the image will show the pit. *(Cam-Op)*
At each waypoint location, the rover will take images at a range of pan and tilt angles. These angles could be hardcoded, but that creates a risk that some images could capture areas of the sky, particularly if the rover is at an unusual angle when it reaches the waypoint.

3.4 Desirable Non-Functional Requirements

The system shall:

D.N.1 Operate given pits of different sizes and shapes. *(Planning)*
The Pit Navigator system should be robust and flexible enough to manage the task of navigating

around pits of varying sizes and irregular shapes.

D.N.2 Take rover parameters and state into account during motion planning. *(Planning)*
The rover will continuously update its global and local plan as it proceeds with its mission. This recalculation should take into account the state of the rover as it changes. If systems are damaged or unresponsive, that should alter the rover's planned behavior to minimize risk.

4 Functional Architecture

Since the time we started working on this project, our functional architecture has evolved a lot. In some cases, we had to make changes to incorporate different functions into one block and in other cases, we divided some large operations into smaller tasks. In spite of these micro-changes, our functional architecture has covered three main subsystems at a macro level right from the beginning. The architecture outlined in Figure 2 shows these subsystems and the important functions that the system must execute to fulfill the previously mentioned requirements. The functions are derived assuming that the robot is already in the vicinity of the pit.

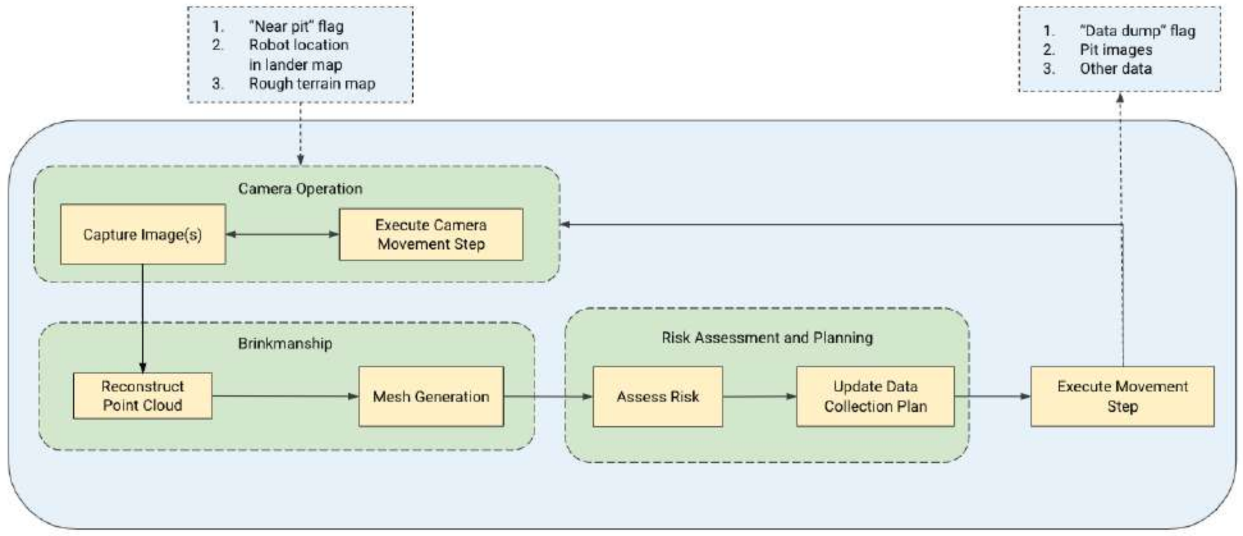


Figure 2: Functional Architure

The pit exploration phase of the mission consists of the following essential functions:

- **Capture Images:** The robot captures images of its surroundings at regular intervals. During navigation, the camera faces forward and down to capture the terrain in front of the rover. Stereo images of the camera are used to reconstruct the terrain in front of the rover with the knowledge of the camera parameters. During pit image capturing, the camera captures images at a variety of positions and exposures to maximize the amount of data collected about the pit interior.

- **Execute Camera Movement Step:** This is where the rover will turn the camera to take pictures from different viewpoints/angles. During navigation, this step will be skipped. During pit image capturing, the camera will move to preset angles or align itself with the opposite rim of the pit. The goal of moving the camera is to cover the largest percentage possible of the pit interior wall from each vantage point.

- **Reconstruct Point Cloud:** The rover generates a 3D point cloud of its immediate surroundings using the stereo images returned by the Camera Operation subsystem. The stereo images obtained from the camera are used to generate a depth map which is then used to generate a 3D point cloud with knowledge of the camera parameters. Initially, the reconstructed cloud was used directly to detect the presence of a pit edge by applying a very simple heuristic. Later, we incorporated another step which converted this cloud into a triangle mesh to obtain more information about the surrounding terrain.

- **Generate Triangle Mesh:** The raw point cloud is filtered and converted into a triangle mesh. The geometry of individual triangles in the mesh is used to determine if the rover has reached close enough to a potential brink. It is also used to determine if the terrain in front of the rover is safe for transversal. If the pit is not visible from the rover's current position and it stops due to unsafe terrain, the rover will still capture data about its surroundings.

- **Assess Risk:** As the rover proceeds with its mission, it constantly monitors multiple risk factors. These include the amount of data currently stored on the rover, the position of the sun over time, and the rover's position around the circumference of the pit. These risk factors are used to adjust the weights of the map which the rover uses to plan its routes.

- **Update Data Collection Plan:** As the planning map gets updated, the rover must recalculate its planned waypoints. While the risk remains low, the rover will continue to travel to preselected waypoints around the rim of the pit, collecting data at each point. If the risk is determined to be too high, the rover will stop circumnavigating the pit and will travel back to the lander to deposit data. After storing data on the lander, it will return to the pit to continue data collection for as long as it has sunlight to power itself.

- **Execute Movement Step:** The rover actuates its motors to execute the next movement step towards its current destination waypoint. The size and speed of this motion may vary depending on the rover's current terrain. For example, near the pit, it may be appropriate to move more slowly while approaching the pit edge. Once the rover detects the presence of a pit edge, a stop command is executed and the rover recedes after capturing images of the pit. The rover recedes at a greater speed than that used for approaching the edge.

- **Interfacing Operations:** The entirety of the Pit Navigator project will constitute one software subsystem within the PitRanger system, which comprises the whole of the functionality for the moon rover. The Pit Navigator system will control the rover during the period of its mission where it is in the vicinity of the pit, during which time it will interact with other PitRanger subsystems. Functionality like SLAM and low-level motor control, which are needed during all stages of the PitRanger mission, may be implemented by teams working outside of the scope of the Pit Navigator project, and so the Pit Navigator subsystem will communicate to other parts of the software in order to operate these functions.

5 System Level Trade Studies

Trade studies for the Pit Navigator project are primarily divided into two categories. The first category of trade studies are for various hardware functions that are necessary for the rover to complete its pit mission. Because the approximate size and power capabilities of the rover are known, we were able to assess hardware components by their ability to meet these restrictions, as well as their functional suitability for the mission.

5.1 Rover Selection Trade Study

This trade study discusses the various options for using MoonRanger[2], altering MoonRanger, or building a new rover entirely. MoonRanger is the closest surrogate to the envisioned PitRanger rover. The major considerations for choice of the rover were, the types of sensors available, the amount of data that could be captured using the various sensors and how safely can the rover operate under the required conditions. This trade study most definitely concluded that some alterations would be required on the MoonRanger. Finally, we ended up making some alterations to the MoonRanger robot and replicated a robot called Blue according to the results shown in Table 1

Criteria	**Weight**	**From Scratch**	**Mechanical Alterations**	**Electrical Alterations**	**Sensor Alterations**	**Any Alterations**	**Exact Copy**
Material Cost	10	0	2	3	4	3	5
Additional Software Effort	10	1	3	5	3	3	3
Additional Hardware Effort	10	0	2	5	5	2	5
Maneuverability	10	4	5	3	3	4	3
Sensors	15	4	2	2	4	4	2
Storage Capacity	5	4	3	5	3	4	3
Safety	20	3	2	2	4	4	2
Ease of Data Capture	5	4	2	2	4	4	2
Amount of Capturable Data	15	4	4	3	3	4	3
Total	**100**	**270**	**275**	**310**	**370**	**360**	**300**

Table 1: Rover Trade Study

5.2 Camera Motion Trade Study

Imaging the pit is an interesting challenge that can be solved in many ways. Using a skid steer rover these options compare and represent the best ways to image the pit. The alternatives for this trade study were identified through brainstorming the possible combinations for having camera motion. The panning of the camera would allow for covering a larger field of view from one particular location on the lunar surface. This means that the rover will be able to capture more information regarding the pit edge even when using a camera with a lower resolution and a lower aspect ratio. Thus, having camera motion indirectly affects the resolution requirement of the cameras. Stability of images is one of the most important considerations in this trade study along with the effect on resolution requirement. The conclusion of performing this trade study is that having dedicated motions for the cameras is best to satisfy the mission requirements. Our final system uses two Dynamixel servo motors to provide fine control of the pan and tilt angles of the camera. We arrived at this design decision using the results obtained from Table 2.

Criteria	**Weight**	**Dedicated Pan + Tilt + Zoom**	**Robot Pan + Tilt + Zoom**	**Single Picture**	**Robot Pan + No Tilt**
Keeping Camera Warm Difficulty	10	2	2	5	5
Stability of Images	20	5	4	4	2
Weight	15	2	3	4	5
Power Consumption	10	3	2	4	3
Ease of Control	10	4	2	5	2
Cost	10	3	3	3	5
Effect on Resolution Requirement	20	5	5	1	3
Size	5	2	2	3	5
Total	**100**	**360**	**325**	**345**	**350**

Table 2: Camera Movement Trade Study

The second category of trade studies are existing algorithms that will be adapted to serve the various purposes of the software. Each of these trade studies must consider the limited computer power available on the moon rover, the need for high levels of robustness on systems that will be operating in the remote and inhospitable conditions of the moon, and the level of technical knowledge and raw effort required to implement each possible solution.

5.3 Navigation Method Trade Study

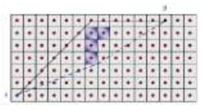
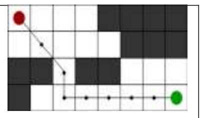
					
Category	**Criteria**	**Weight**	**Field D*[3]**	**Markov Model[4]**	**A* Algorithm**
Theoretical	**Implementation Complexity**	20	2	3	5
	Adaptability (handle more situations)	15	3	5	3
	Availability of Prior Work	7.5	2	4	5
	Code Structure Availability	7.5	2	3	5
Performance	**Performance (accuracy to human path)**	20	TBD	TBD	4
	Computational Requirement	15	5	2	5
	Prior data requirement	5	4	4	4
	Reference map resolution	10	4	2	4
Total		**100**	**250**	**257.5**	**355**

Table 3: Navigation Method Trade Study

As the rover should not fall into the pit, the act of autonomously moving around the pit can be dangerous. The options presented are compared as the first navigation strategies for the pit. Here, we perform an algorithmic trade study to identify the best choice that would satisfy the requirements of the risk assessment and planning subsystem. The requirement is to capture the risk associated with executing a certain process and to incorporate the calculated risk into future planning and navigation of the robot.

We have categorized the considerations for the algorithm selection into theoretical criteria and performance criteria. Some of the performance criteria weights and scores are based off the team's understanding of the two methods. We were unable to evaluate the performance weight of the Field D* algorithm and the Markov model. The results are shown in Table 3

We were able to evaluate the performance of the A* algorithm and determined that while it did not perfectly track where a human would go, it found a sensible path to the goal that was the shortest possible. This result mathematically guaranteed that it would not be superseded by the other algorithms, and so the team chose not to evaluate the performance of the other two algorithms. The ease of implementation and availability of the code made A* a much better option than trying to re-implement a different algorithm ourselves, given that it worked suitably for the mission.

5.4 Mapping Trade Study

This trade compares the different maps and amounts of information that is useful and able to be processed by the rover. Balancing processing power needed and useful information is key. The different options included in this trade study were identified based on the most widely used techniques for mapping and the mission requirements. The criteria were decided based on the mission requirements and the amount of effort required to execute a mapping algorithm to generate the required type of the map. The exact criteria considered while performing this study and the final result of the study is shown in Table 4

Most of the options listed can be implemented using open source frameworks like Robot Operating System (ROS). The current team working on the MoonRanger project plans to use the ROS implementation of Real Time Appearance Based Mapping[5] (RTAB Map) to generate an occupancy grid map from stereo images. The most definitive conclusion of performing this trade study is that there would be no requirement of generating 3D point clouds and the mission requirements could be satisfied by generating 2D maps of the environment. Our final system uses a 2D occupancy grid map in which we set pre-defined waypoints for the rover to navigate through.

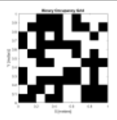
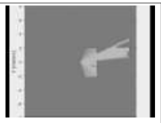
						
Criteria	**Weight**	**2D Binary Occupancy**	**2D Probabilistic Occupancy**	**3D Point Cloud**	**2.5D Map**	**Topological**
Sensor Requirement	20	4	4	5	4	4
Computation	15	5	3	2	3	5
Storage Space / Map Size	15	5	4	2	4	5
Input/Output Format	10	5	5	2	2	3
Amount of Information Captured	10	2	3	5	4	1
Reliability / Precision	20	4	5	4	5	1
Software Development Effort	10	5	5	3	4	5
Total	**100**	**430**	**415**	**340**	**385**	**340**

Table 4: Mapping Trade Study

6 Cyberphysical Architecture

The evolution of our functional architecture has a direct impact on our cyber physical architecture. Since the very beginning, our cyber physical architecture has had direct correspondence to our functional architecture. It highlights the technology options which will be potentially used to execute the individual functions. The inputs and outputs of individual functions are also included. The architecture is shown in Figure 3.

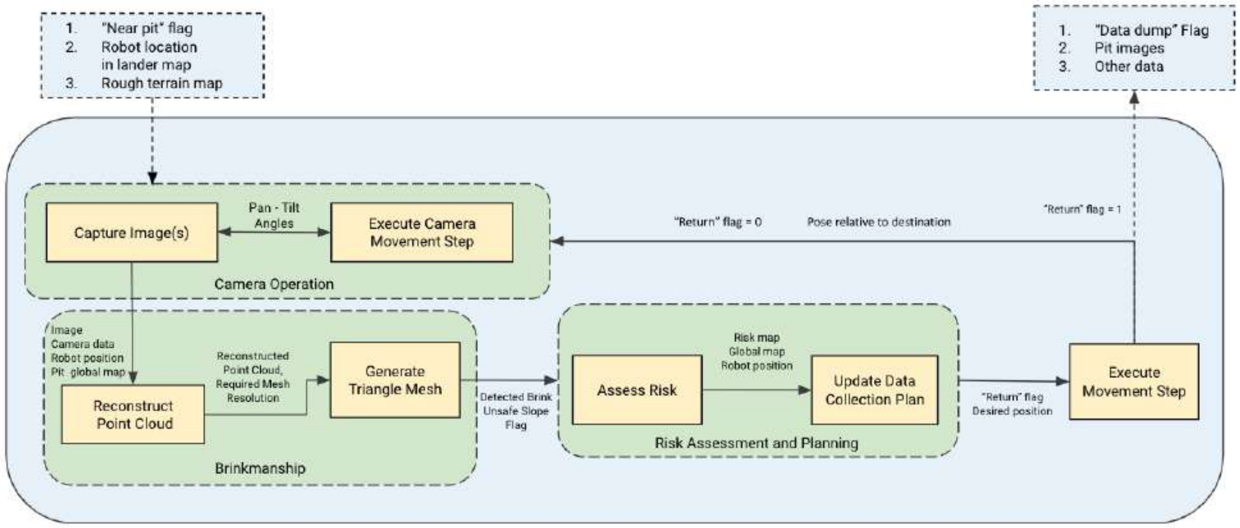


Figure 3: Cyberphysical Architecture

- **Capture Images:** An Intel RealSense camera allows to stream images of the surrounding.
- **Execute Camera Movement Step:** A camera mount having pan and tilt motions will be used. We chose 2 Dynamixel AX-12 servos, one providing pan motion and the other providing the tilt motion to the camera. The servos are controlled using an Arbotix-M controller which is connected to the rover computer via a USB interface. Software for controlling the motors is developed by utilizing the ROS API for dynamixel motors.
- **Reconstruct Point Cloud:** This will be executed through 3D reconstruction using stereo images. OpenCV's global block matching algorithm is applied to generate a depth map from the stereo images.
- **Generate Triangle Mesh:** The filtered point cloud is converted to a triangle mesh using an implementation of the Delaunay 2D algorithm from the Pyvista library.
- **Assess Risk:** A heuristic solution will be employed for estimating the risk involved in continuing the mission.
- **Update Data Collection Plan:** The TEB local planner is used to calculate an optimal path that the rover traverses.

- **Execute Movement Step:** Once the decision is made, the rover either goes back to the starting point or continues to gather more data from different positions. The rover wheels have in-built encoders which when combined with the onboard IMU of the rover, provides an estimate of the rover's position with respect to the previous waypoint.
- **Interfacing Operations:** When the rover reaches the vicinity of the pit, the Pit Navigator system will activate, taking as an input the rough terrain map and robot pose established by the overarching PitRanger system. While the Pit Navigator system runs, it will continually pass collected data including images of the pit to other subsystems within PitRanger.

During Fall 2019, our team had discussed potential options to be used as surrogate platforms. During the time, our options included the robot Blue, the AutoKrawler and the CubeRover. In the spring of 2020, we went forward with building our own rover which was a replica of rover Blue so as to have limited dependence on the MoonRanger team and have full time availability of the rover.

One loop through our cyber physical architecture describes one cycle in the pit exploration mission. The robot continuously captures stereo images of its surrounding while navigating at a slow speed. It then constructs a point cloud using the stereo images. The point cloud is converted to a triangle mesh to obtain more information about the geometry of the surface in front of the rover. The robot also tries to detect the pit edge using the mesh.

While detecting the edge the robot estimates the risk of moving close to the edge and decides whether to get closer to acquire better data. Upon getting sufficiently close, the robot captures multiple images covering a large area of the opposite wall of the pit. This will be achieved by having pan and tilt motions for the camera to capture images of the pit. When the required number of images have been captured the robot updates its current state and tracks the mission status. Based on these parameters, it decides whether to navigate to a new waypoint to capture more data or to go back to the starting location and cede control to the standard navigation system for return to the lander. The major inputs and outputs of each functional block in our system is are shown in Table 5.

Subsystem	**Function**	**Inputs**	**Outputs**
Camera Operation	Execute Camera Movement Step	1. Pose Relative to Destination	1. Camera Pose
	Capture Images	1. Capture Flag	1. Images 2. Camera Data
Brinkmanship	Reconstruct Point Cloud	1. Captured Images 2. Camera Data	1. Point Cloud
	Generate Triangle Mesh	1. Reconstructed Point Cloud 2. Required Mesh Resolution	1. Brink Detected Flag 2. Unsafe Slope Flag
Risk Assessment and Planning	Assess Risk	1. Rover/Mission Parameters 2. Potential Waypoint	1. Updated Risk Map
	Update Data Collection Plan	1. Updated Risk Map	1. New Data Collection Plan 2. Optimal Path

Table 5: Major Inputs and Outputs of Individual Functions

7 System Description & Evaluation

7.1 System Description

The system developed in this project is a specialized set of integrated software functions which activate when the flight rover reaches the portion of its mission where it must navigate the lunar pit environment. This system is organized in three parts. The first part is the planning subsystem, which develops a path for the rover to execute around the circumference of the pit. This subsystem also assesses the risk of data loss if the rover were to be damaged, and updates the path to include returning to the lander when the risk reaches unsafe levels. The second part is brinkmanship, which is a specialized navigation procedure that the rover uses when attempting to move as close as possible to the pit edge. This subsystem will activate at each of the waypoints around the pit from which the rover intends to capture images, and enables the rover to achieve the best possible view of the pit interior while remaining safe. The last part is the camera operation subsystem, which manages the conversion of stereo image data to point clouds for navigation and the capturing of images of the pit interior from each vantage point.

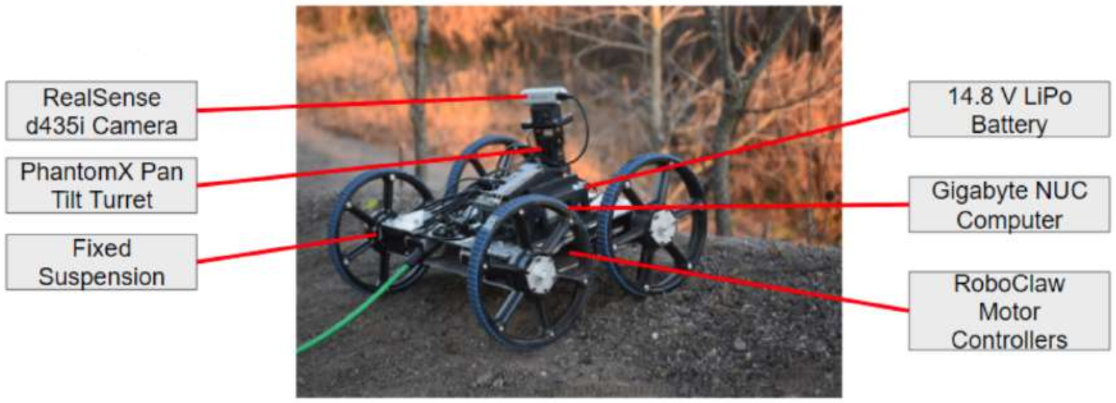


Figure 4: Surrogate Rover

In order to support and test these core functions of the system, a surrogate rover (Figure 4) was constructed and a simulation that included multiple testing environments was developed. Some additional software functions were also implemented to facilitate testing and fill in the gaps where code written by other teams would eventually go. The rover had a fixed suspension, four Vex planetary motors, two RoboClaw motor controllers, a Gigabyte NUC onboard computer, and a RealSense d435i camera mounted on a pan-tilt turret controlled by an Arbotix-M board. The simulation used the Webots simulator. The environments designed for the simulation were representations of the West Desert Sinkhole (Figure 5) and the Lacus Mortis crater, as well as a small environment with a variety of slope and brink types that we wanted our rover to be able to handle. The rover used in the simulation was similar to Blue. It had a skid-steer drive and included a stereo camera and an IMU along with the wheel encoders for performing odometry.

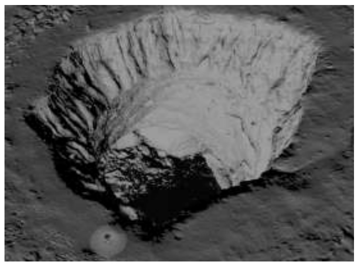


Figure 5: Sim Environment

7.2 Subsystem Descriptions

7.2.1 Camera Operation Subsystem

The camera operation subsystem is one of the simpler subsystems within the Pit Navigator project, but it is crucial to the function of many of the other components. For Blue 2, which uses a single camera for both navigation and data collection, the camera operation subsystem must ensure that the camera is constantly in the proper position and feeding images to other subsystems.

The mechanical components of the camera operation subsystem are a RealSense D435i camera, a PhantomX pan/tilt turret (shown in Figure 6), and an Arbotix-M Robocontroller (shown in Figure 7). The Arbotix-M runs an Arduino-based program that allows it to communicate with the main rover computer. Through a ROS package called arbotix-ros, the main computer can send commands to the Arbotix-M to set the angles of the pan and tilt servos. The turret is mounted to the center of the front edge of Blue 2's chassis. The RealSense camera is attached to the top of the turret using a 3D printed camera mount which is shown in Figure 8. The camera is connected directly to the main computer via a USB-C cable.

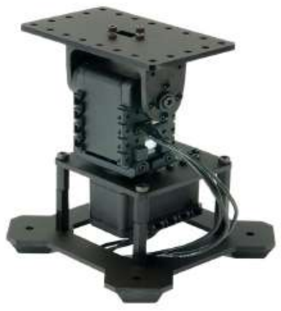


Figure 6: Pan-Tilt Turret

Figure 7: Arbotix-M Controller

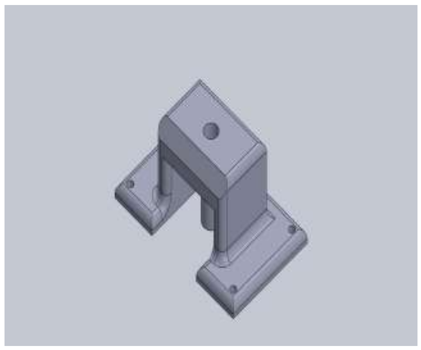


Figure 8: RealSense Mount

During navigation, the camera faces straight ahead of the rover, angled 40° downward to capture the terrain that the rover is traversing. The rover will use the camera stereo images and resultant point cloud to generate a local plan to navigate through its environment and avoid obstacles. When the rover nears the pit edge, the brinkmanship subsystem will use the point cloud to drive the rover as close to the pit edge as possible without endangering the rover.

Once the rover has reached its vantage point at the pit edge, it will use the camera to take a series of images of the pit interior. Each image will be taken with the turret at a different combination of pan and tilt angles. The rover will be aware of its own orientation and location with respect to the pit edge, and will adjust the range of angles used for image capture so that the number of images which show the pit interior is maximized. The rover will also use appropriate exposure settings for the lighting conditions within the pit to ensure that details of the pit walls are identifiable.

All images taken at a particular tilt angle will be stitched together into a single panoramic image showing a horizontal strip of the pit interior. These panoramas will then be incorporated into a 3D model of the pit, which will be constructed over the course of the mission as the rover collects images from different vantage points. In the tests performed with Blue 2, five images were taken at each of four tilt angles, for a total of 20 images and four panoramas at each vantage point. An example set of 3 images taken at one tilt angle at one particular vantage point is shown in Figure 9. The panorama obtained by stitching those images together is shown in Figure 10.

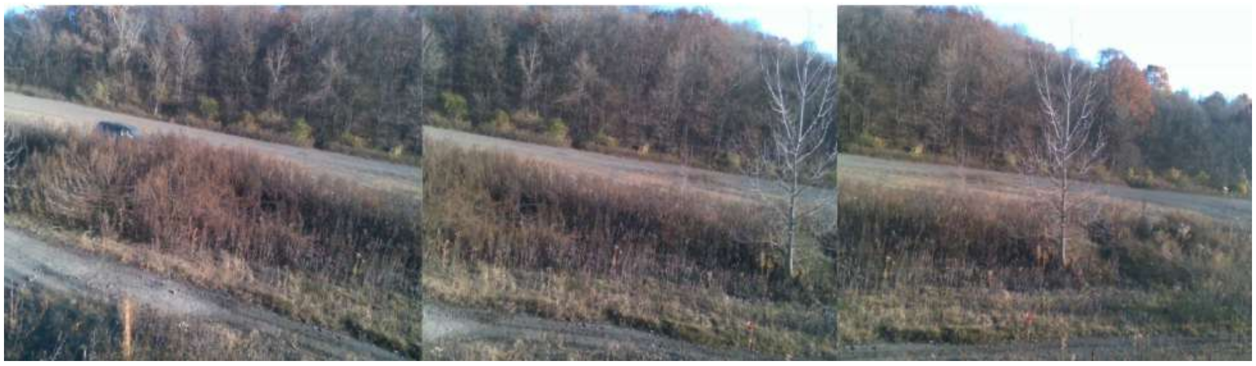


Figure 9: Three Images Captured At One Vantage Point

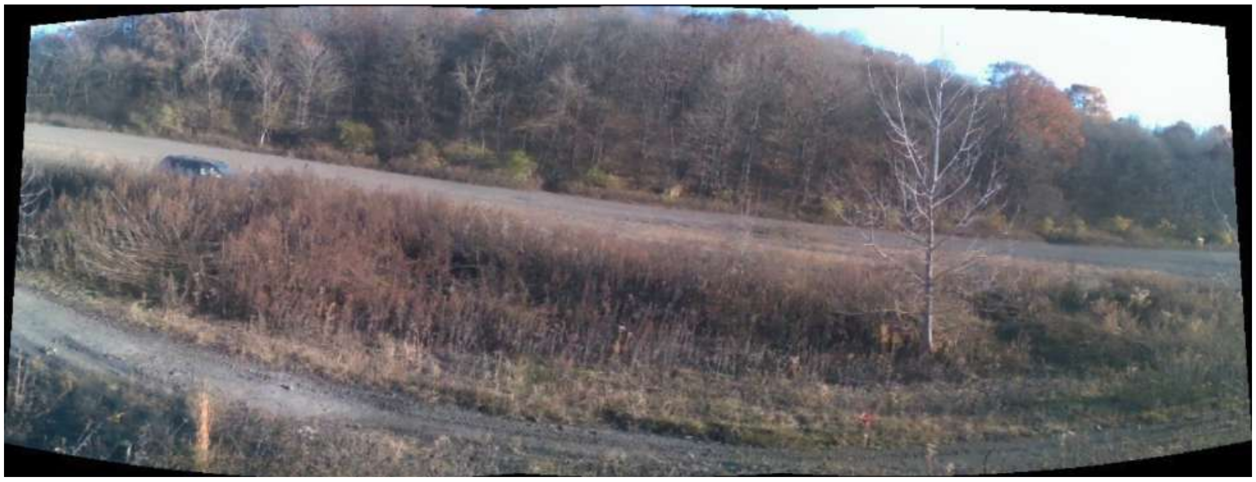


Figure 10: Panorama Output

7.2.2 Brinkmanship Subsystem

This subsystem maintains information about the robot's position with respect to the pit. This will involve local brinkmanship checking while the rover is moving to vantage points. The mission plan also includes a plan to assemble and continually update a 3D model of the pit based on images taken by the rover, so this model could be used in the localization process as well. The brinkmanship subsystem takes control of navigation from the global planner when the rover reaches a highway point near the circumference of the pit. It allows the rover to move in close to the edge of the pit to capture good quality data without actually falling in the pit. The subsystem cedes control to the global planner when it recedes back after capturing images from a vantage point which is right at the edge of the pit.

The brinkmanship system will allow the rover to localize shear brinks and unsafe slopes in the vicinity. The idea is that the rover will be able to move to a predefined location near the pit using this map and then will move close to the pit edge with the help of this subsystem. The brinkmanship system currently uses stereo depth reconstruction to get an estimate of the 3D structure of the surrounding in the form of a point cloud. Initially, we envisioned the system in such a way that it used a fairly simple heuristic. The heuristic used to detect edges only counted the total number of points in the point clouds generated from the stereo camera images. The high-level operation of the brinkmanship subsystem using the simple heuristic (number of points in reconstructed cloud) during the spring semester can be seen in the Figure 11.

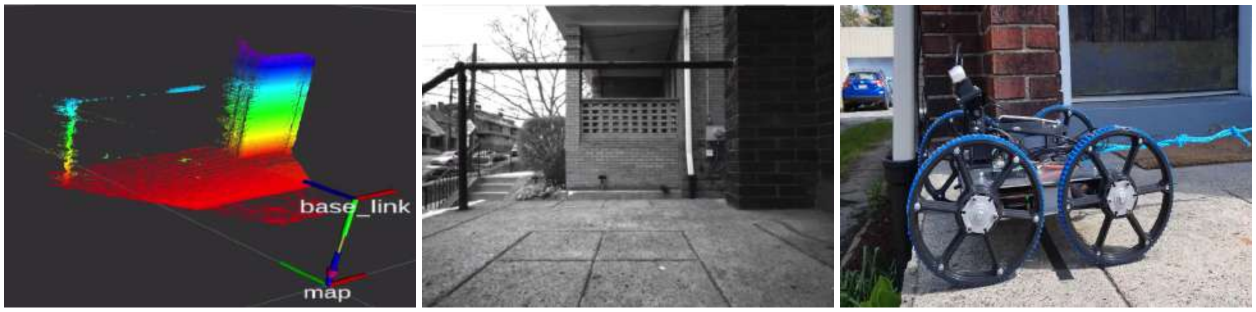


Figure 11: Left: Generated Point Cloud, Center: Rover View, Right: Final Position

The terrain on the lunar surface is varied enough that this would not be suitable for an actual mission. The team worked together to come up with a more robust heuristic that could be implemented. The solution that the team arrived at was inspired by the way 3D environments are represented in video games and other virtual depictions. These computer-generated environments are composed of meshes of many small polygons, usually, triangles, which approximate complex terrain. We reasoned that if we could convert our point clouds into triangular meshes, we could then determine the surface normals of each polygon in the mesh and use those vectors to distinguish between safe and unsafe terrain. Furthermore, gaps in the mesh could be used to indicate the presence of sheer brinks. Hence, we went forward with this idea and converted the reconstructed point cloud into a triangle mesh using Delaunay triangulation. In this particular triangulation method, the mesh generation is such as to maximize the minimum angle of individual triangles in the mesh. We used the Pyvista implementation of this algorithm in our project. Figure 12 shows the generated mesh when the rover is close to the pit edge. In this case, yellow represents unsafe terrain in the mesh.

Once the mesh is generated, the surface normals of triangles in the mesh are used to determine if there is the presence of sloped terrain near the rover. If the number of triangles for which the normal is off from the vertical axis by a fixed value is above a certain threshold, we conclude that the terrain in front of the rover has an unsafe slope. Both the angle from the vertical and the number of meshes used to make this decision can be adjusted easily to suit different conditions. The mesh is also used to locate brinks which may represent the edge of a pit. A pit edge is said to be detected when there is an absence of terrain near the rover which is reflected in the generated triangle mesh. The mesh is divided into a fixed number of cells and we conclude that the rover has reached near a shear brink when the cells which are close to the rover become empty. This subsystem will trigger an alert signal in such a case and prevent the rover from getting too close to the pit edge.

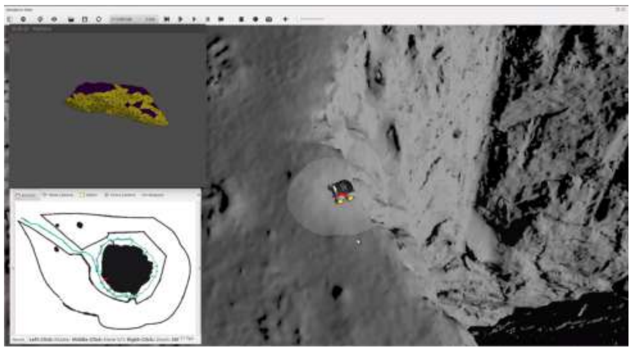
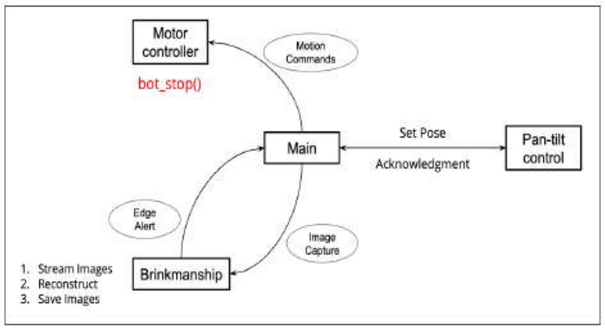



Figure 12: Left: Snapshot Of Rover In Sim, Right: Control Flow Between System Functions

Once the brinkmanship subsystem raises an edge alert, the robot stops near the edge, the camera operation subsystem controls the camera position and captures multiple images of the surrounding. The current interdependence and the simultaneous operation of the camera subsystem and the brinkmanship subsystem are shown in the Figure 12. The rectangular boxes represent the nodes in the system and the elliptical shapes represent the messages.

7.2.3 Planning Subsystem

The path planning and risk management subsystem is made of 3 parts. The global planner, the local planner, and the state machine. The global planner is responsible for using the obstacle map made from orbital imagery to find a path to the next waypoint. The local planner is responsible for sending the hardware commands to make the robot follow the path set by the global planner. The state machine is responsible for determining which waypoint the global planner plans to next. Combined, these parts tell the robot how to move its wheels to get to where it needs to go.

This subsystem runs in the onboard rover computer, the Intel Nuc. The three different parts use the ROS package, move base, to communicate with each other. Each part uses a different ROS package to operate. The local planner uses the TEB_local_planner package, the global planner uses the ROS global planner package, and the state machine uses the executive_smach package. The state machine will send a new destination to move base, which will pass the message along to the global planner. The global planner will plan the optimal path between the start and goal positions of the rover. Move base takes this path and hands it over to the local planner which will follow the path by taking the robot's position, also gathered by move base, and outputting motor commands. When the local planner has decided that the robot has reached the goal position, it flags move base, which flags the state machine to send a new goal position.

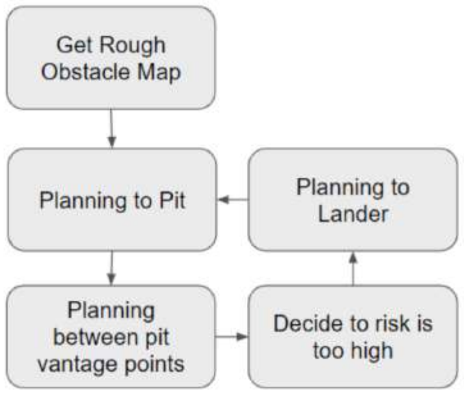

Get Rough Obstacle Map
Planning to Pit
Planning to Lander
Planning between pit vantage points
Decide to risk is too high

Figure 13: State Machine Flow Diagram

The state machine will send different waypoints when run on Blue vs when run in the simulation. This is due to the differing mission plans between the two testing platforms. In simulation, the rover can be tested through an entire mission with time constraints, so the state machine is unmodified. Figure 13 shows a flow diagram of the full subsystem, where the state machine powers the transitions. The state machine first gathers the obstacle map, then begins planning the path to the pit. Once the rover has arrived in the vicinity of the pit, it will plan to each vantage point around the pit and take pictures at each location. When the risk estimator within the state machine has decided that the rover has captured enough data and it is too risky to try approaching the pit again, it will plan back to the lander and drop off the data. An example of the rover's path can also be seen on the left side of Figure 14, where the rover visited 4 vantage points before returning to the lander. Then the state machine will take the rover back to the pit to capture more pictures at vantage points. This continues until the mission ends, when the rover has captured images at every vantage point.

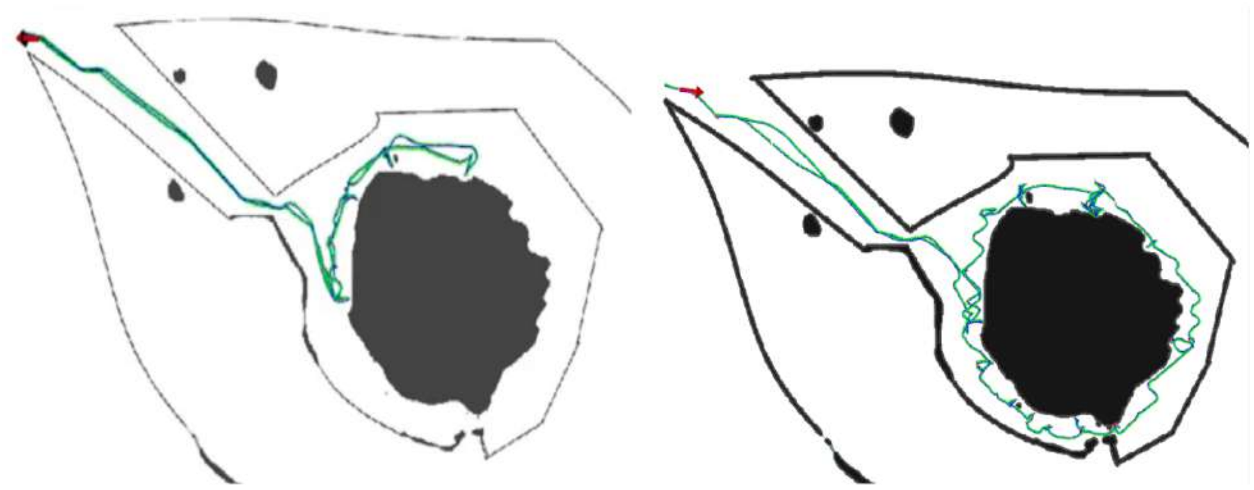


Figure 14: Path Planning and Risk Management Algorithms Demonstration

Because there is not a pit for Blue to plan and navigate around, the mission is shorter. Blue must meet each vantage point along the real brink before heading back to the lander's position. In this case the state machine will be modified to ignore the risk block and navigate to every vantage point before returning to the lander's position. This makes the state machine's plan very linear. This is the only difference between the simulation and real world planning subsystems.

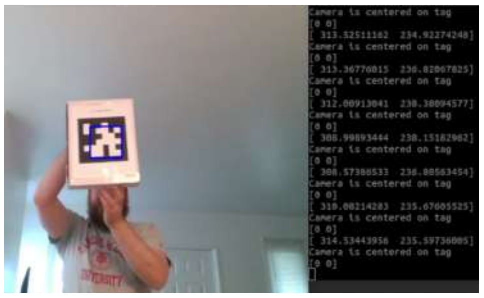
7.3 Modeling, Analysis & Testing

7.3.1 Camera Operation Subsystem

Some of the earliest work done on the project involved the RealSense camera, because it was available from the MRSD storage area. While other components were in transit, or required trade studies for selection, there was time to develop familiarity with the camera and its API. The RealSense served as an analog for the stereo camera pair that would be installed on the flight rover. Due to the lack of space-rated depth cameras or LIDAR systems, it had been determined that the rover would need to derive depth information using stereo RGB images produced by the cameras. These stereo images would need to be converted into a point cloud which could then be used by the brinkmanship subsystem. This pipeline was designed and tested by capturing images of the MRSD lab with the RealSense camera, before being incorporated into the brinkmanship routine later in the semester. Another experiment that was performed in parallel was a simple program to control the exposure of the images produced by the RealSense Camera. This was performed because the lunar rover would need to be able to handle a wide range of lighting conditions to produce the largest quantity of useful data. The lessons learned from these experiments enabled effective use of the RealSense camera in later testing.

The pan/tilt turret chosen to support the RealSense camera on the surrogate rover was the PhantomX from Trossen Robotics. This turret used two AX-12 servos, controlled by an ArbotixM Robocontroller board. The Arbotix-M could be programmed in the Arduino language, but a pre-made program that allowed the board to handle commands sent from an external computer also existed. Several programs that interfaced with the Arbotix-M board were also available. One of these, a terminal-based program that provided a limited command set to control the speed, position, and IDs of the servo motors, was used to perform initial validation testing of the assembled turret. Once the motor function had been verified, the terminal program was used as a guide for implementing position control of the turret into the rover code base.

The next step was to integrate the RealSense camera and the PhantomX turret. The Image Capture Test from the Spring Validation Demonstration was designed to showcase this integration. A marker with a visual pattern from the AprilTag library was created, and code was written to detect this tag in the RealSense camera images. The program determined the distance from the tag's position to the center of the image. This distance was then converted into a command for the pan tilt turret, so that the tag could be moved around in the camera's field of view and the turret would move the camera to center the tag. A snapshot from one of the early tests is shown in Figure 15. Early testing of this program revealed that the turret would move at high speeds when commanded to move the entire distance at once, causing it to oscillate when attempting to center the tag. This was solved by setting a maximum "velocity" in units of encoder ticks per command, and repeatedly commanding the turret to move its servos by that number of ticks until it reached the goal position. This insight was important later on in the project when the turret and camera were used to create panoramic images.


```
Camera is centered on tag
[0 0]
[ 313.32511182  234.92274248]
Camera is centered on tag
[0 0]
[ 313.36770015  236.82067825]
Camera is centered on tag
[0 0]
[ 312.00913041  238.38094577]
Camera is centered on tag
[0 0]
[ 308.99893444  238.15162982]
Camera is centered on tag
[0 0]
[ 308.57388533  236.88563454]
Camera is centered on tag
[0 0]
[ 310.00214283  235.67605525]
Camera is centered on tag
[0 0]
[ 314.53443956  235.59736005]
Camera is centered on tag
[0 0]
```
Figure 15: Testing Pan-Tilt Control

At this point, every necessary function of the camera operation subsystem had been developed. The next step was to integrate these functions with the brinkmanship subsystem. The first field test of brinkmanship capabilities also tested this integration. When the rover determined that it had reached the selected brink, it stopped moving and used the camera and turret to capture nine images at different angles. These images were stitched together into a panorama. With this final test of the camera operation subsystem, it was deemed to be complete.

7.3.2 Brinkmanship Subsystem

The development of the brinkmanship subsystem was kick-started while unit testing the functions of our RealSense camera. First, a simple program to control the exposure of the images was produced. This was performed because the lunar rover would need to be able to handle a wide range of lighting conditions to produce the largest quantity of useful data. In addition, this provided an opportunity for the team to develop familiarity with the RealSense hardware and API. The lessons learned from this experiment enabled effective use of the RealSense camera in later testing.

As the Pit Navigator project got underway, some prior work had been done to design a brinkmanship system that used the RealSense depth camera to detect edges in the rover's vicinity. Due to the lack of space-rated depth cameras or LIDAR systems, it was determined that this method would need to be replaced with one that derives similar depth information using only the stereo RGB images produced by the camera. These stereo images would need to be converted into a point cloud which could then be used by the brinkmanship subsystem. This feature was initially designed and tested by capturing images of the MRSD lab to fine-tune the stereo-to-depth pipeline.

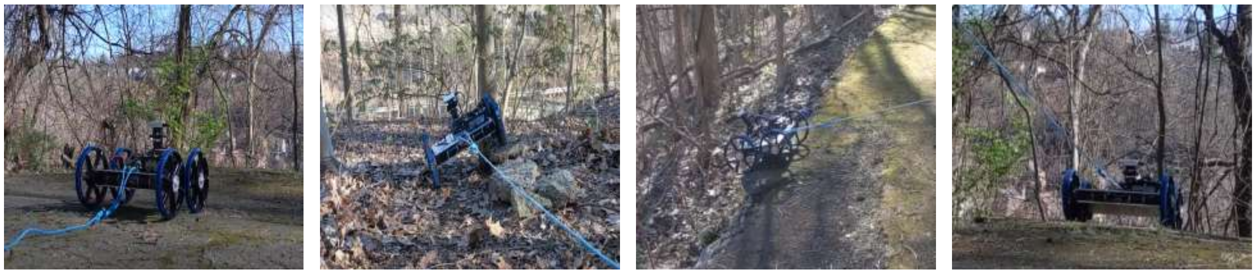


Figure 16: Rover Operation (Schenley Park Field Test)

On April 3rd, 2020, the first field test of the surrogate rover was performed. Blue was taken to a location in Schenley Park which had been identified as having ledges and slopes that would be useful for gathering data. No brinkmanship code had been implemented on Blue at this point in the project. Instead, Blue was simply teleoperated to drive forward towards each ledge or slope, while the computer saved data from the motor encoders and the RealSense camera into a rosbag file. A rope was attached to Blue's chassis to prevent it from fully falling over the brink as it approached. Tests were performed with the camera at tilt angles of 0 and 15 degrees. The stereo image data was then reconstructed into a point cloud and used to develop the brinkmanship edge detection criteria used in later tests. Images from the field test are shown in the Figure 16

Earlier in the course of the project, we relied on using a very simple heuristic that counted the number of points in the reconstructed point cloud to detect a sheer drop off. We then transitioned into using a more complex data structure to make better decisions with regards to the safety of moving forward. Almost all the testing and analysis of this particular function was carried out in our simulation environment. The brinkmanship subsystem was modeled to use a triangle mesh to output a stop signal for the rover. All the required transformations of the point cloud were carried out using a simulated IMU. A clear mesh was generated, divided into multiple parts, and processed according to our requirements.

To test whether our algorithm to detect unsafe slopes in front of the rover was functioning

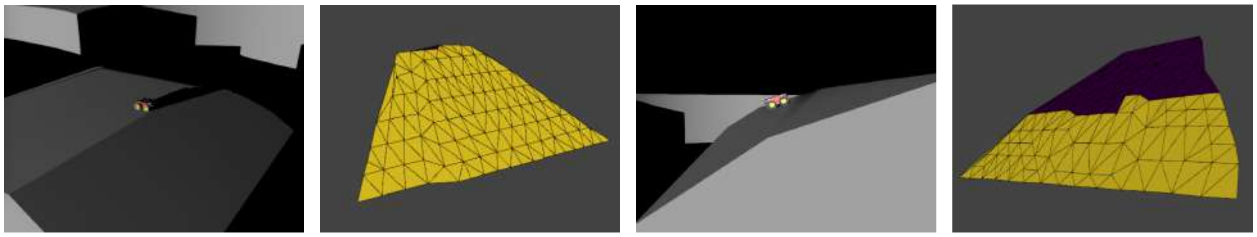


Figure 17: Custom Environment Created In Webots Simulator)

correctly, we created a test environment in the webots simulator. This environment consisted of different kinds of terrain slopes like a sheer drop off, a small and large slope, and also an area where the slope of the terrain increased gradually. The IMU was used to transform the generated point cloud into the global reference frame before the generation of the mesh. This allowed us to identify whether the rover was itself standing on sloped terrain or not. Figure 17 shows a couple of snapshots from the test environment and the corresponding meshes generated in each case. The yellow portion of the mesh corresponds to unsafe slope.

7.3.3 Planning Subsystem

The Planning subsystem was exclusively tested and designed in simulation. At the start of the project, the local planner was adopted from the previous team's code. A two state state machine was also adopted. These systems were analysed and understood by running the simulation and inputting different paths. Once understood, the systems needed refactoring before being expanded. To test that the refactoring was successful, the team ran the same paths used to analyze the algorithms. The robot's motions were inspected visually. The spring validation demonstration was done with the refactored code.

To unit test the planning subsystem during fall development, a different path was set on flat ground to confirm that the robot plans as expected, without obstacles. Another unit test path was created, starting near the pit and going around the pit to speed up navigation to vantage points. Move base also takes in localization information, which is perfect in the simulation but needed major tweaking when moved over to Blue. During this re-implementation phase, a 3 meter track was used to calibrate the ticks per revolution of the wheels, and edit the motor commands to match what was possible with the real rover. After the rover looked good on the straight track, it was brought outside to plan over a 3 meter square and return to the same location. Once it was able to return to the start position, it was tested at the Gascola testing site, where it drifted more, but within acceptable limits. A snapshot from one of the initial tests of our planning subsystem is shown in Figure 18.

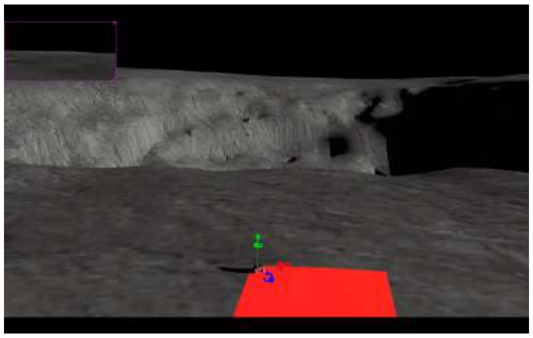


Figure 18: Planning Test In Sim

7.4 FVD Performance Evaluation

7.4.1 Simulated Mission Test

The first of the Fall Validation Demonstrations that we proposed, referred to as the Simulated Mission Test, was designed to assess the functionality of the entire system with moon-like conditions for the entirety of the mission. The proposal went through a couple of different versions as the project developed. However, the concept of what the demonstration would show always remained intact. We intended to demonstrate how our project would perform in a moon-like environment with a pit that closely resembles the lava tube skylights.

We tested our rover in the Webots simulation. We added a two meter circle around the rover so that it was easily determined when the rover had gotten within 1 meter of the brink, and could confirm that the rover had met the success criteria. Additionally, the simulation never had the occurrence of a mission failing incident during the final demonstration, even after approaching the pit edge 12 separate times at different locations on 3 separate trips to the pit. Even after running the simulation 5 times to get media videos, the rover had not created a mission failing incident, surpassing success criteria for risk incidents.

The rover's camera to take images of the pit needed an upgrade as the overall lack of textures in the simulation reduced the data collected from images. Taking an image of the sky or ground immediately in front of the rover would result in a highly compressed image that contained very little usable data (data that couldn't be compressed). Taking images of the pit would result in incompressible data as the geometries and shapes of the pit were ignored by the compression algorithms. The team decided to take the entire panorama of the pit at once that resulted in a 6400x6400 image that would contain all the data necessary as long as the camera was pointed at the pit. In general it could create an average of 45 MB of data per image, surpassing both the data in a cycle and the data in a mission requirements.

There is one success criteria that was not by our simulation test, that being the speed made good requirement. The team believes that this requirement was not met, not due to the capabilities of the rover, but due to the scale of the chosen pit. This requirement was met in the spring validation demonstration, when the rover navigated once around a pit 3.3 times the size of the current simulation pit. When the code of the spring semester is applied to the smaller scale pit, the speed made good also drops below the requirement. The team believes that this is a problem of scale because the rover has a finite number of small motions that do not scale with size, but hurt speed made good, like turning and taking images of the pit. When traveling around the smaller pit, the rover spends a larger percentage of its time turning and taking images of the pit and less time traveling to the next waypoint at max speed. This is the reverse in the larger pit. As this was not realized until too late into the project it was left as is. In future work, it would be prudent to re-examine the effect that the scale of the pit that we drive around has on the speed made good. The final evaluation for the simulated mission test is summarized in Table 6.

7.4.2 Terrestrial Pit Edge Validation Test

The second of the Fall Validation Demonstrations that we proposed, referred to as the Terrestrial Pit Edge Validation Test, was designed to assess the functionality of the entire system in

Validation Criteria	Evaluation
Average distance to pit when image capturing: $\leq 1m$	Average distance to pit when image capturing: $\leq .5m$
Amount of usable data over a single cycle: $\geq 75MB$	Amount of usable data over a single cycle: $\geq 180MB$
Total amount of usable data captured: $\geq 500MB$	Total amount of usable data captured: $\geq 540MB$
Speed made good: $\geq 0.21m/s$	Speed made good: $\geq 0.14m/s$
Risk never goes below threshold: 5:1	Risk never goes below threshold: 12:1
Mission Completion	Mission completes in each simulation

Table 6: Performance Evaluation Of Simulated Mission Test

real-life conditions for a small segment of the overall mission. The proposal went through several iterations as the project developed and world events restructured our priorities and resources. However, the core concept of the demonstration always remained intact. We intended to demonstrate an extended version of our Spring Validation Demonstration, wherein the rover would plan its path through three waypoints instead of only one.

We tested our rover at two different sites at Gascola. We conducted around 5 straight tests at the first site where we met our success criteria related to the rover's distance from the edge. Additionally, none of these tests had the occurrence of a mission failing incident. At the second site, we conducted at least 10 tests which again met the criteria related to the rover distance and mission failure.

During these tests, we were operating the rover's camera to capture images at a resolution of 640 x 480. This resulted in the system not satisfying the validation criteria related to the amount of data captured. Due to some miscommunication within our team, we failed to identify this problem earlier. However, we realized this during our Fall Validation Demonstration test that we performed live at Gascola. After that, we switched to capture images having a resolution of 1920 x 1080 which helped increase the amount of data we captured during the test. Additionally, to be on the safe side in terms of the captured data, we increased the range of the camera pan and tilt angles to capture more images at each vantage point. In this case, we captured 5 images at different pan angles for 4 different tilt angles of the rover camera. This change allowed us to satisfy all our validation criteria defined for the terrestrial pit edge validation test. Table 7 shows the final evaluation results of the terrestrial test.

Validation Criteria	Evaluation
Average distance to pit when the image capturing: $\leq 0.5m$	Average distance in each test ≤ 0.38 m (15 *inches*)
Amount of usable over a single cycle in the mission: $\geq 9MB$	Amount of usable data over single cycle $\geq 18MB$
Amount of usable data captured per location: $\geq 3MB$	Amount of usable data per location $\geq 6MB$
Produces tagged data and the stitched panorama	Images were labelled and panoramas were generated

Table 7: Performance Evaluation Of Terrestrial Test

7.5 Strong/Weak Points

Our surrogate rover met every one of our expectations. During our tests at Gascola, the rover was able to maintain traction on a range of terrains from dirt to stone. When operating on slopes, the rover was able to drive safely on slopes less than 20 degrees, the limit we set for safe slope in our brinkmanship subsystem. The rover had no suspension of any kind, but neither does the proposed design for the flight rover. Nevertheless, the rover was able to operate on terrain that varied in height by roughly the radius of the rover's wheels (5 inches).

In simulation, the planning subsystem reached a level of functionality that enabled us to execute a complete mission consisting of the circumnavigation of a pit and multiple treks to and from the lander. The rover was able to plan a path between all waypoints established for this mission, and navigate along that path successfully. The map of the pit area included some narrow gaps that the rover needed to traverse, and it was able to consistently determine a path through these challenging areas. The simulated rover was also able to adjust its path according to the risk that it continually calculated as it proceeded through the mission. However, this risk assessment was fairly simplistic, and additional work could definitely be done to improve the robustness and accuracy of that system.

In the real world, the path planning worked similarly well, but the execution was limited by the rover's ability to localize itself. The surrogate only had access to wheel odometry and IMU data for localization. This worked well enough to complete the shorter demonstration that we proposed for the FVD, but by the end of that demonstration the rover had deviated visibly from its intended course. However, this dead reckoning was capable of somewhat mitigating error from skidding and rough terrain. Overall, it served its purposes for localization even if we pushed it to its very limits.

After multiple days of field testing, we were able to improve our brinkmanship algorithm to a point where we felt confident that it could handle a variety of brink and slope conditions. We had to filter the point clouds generated from stereo feature matching to account for additional noise encountered in an outdoor environment, but were able to develop a method that preserved the important terrain information while eliminating most noise. Multiple mesh analysis methods were tested to find those which produced the best results, and then those methods were tuned further to ensure that the rover stopped at an appropriate distance from the brink. These mesh analysis methods were able to be translated back to the simulation without difficulty, and produced equivalent results in that environment.

Our team was able to achieve success in the image capture portion of the system as well. Although this subsystem was fairly simple in the scheme of the project, there were several requirements related to its outputs. Initially, we struggled to meet the data collection requirements because we overestimated the size of the image files produced by the RealSense camera. However, once we observed this issue, we were able to adjust the image resolution and take additional images in order to produce the required amount of data. The process of controlling the camera angle through the pan tilt turret was also not a problem, because the Arbotix API contained all the functions we needed to drive the servos.

Ultimately, the one requirement we struggled to meet was the simulated speed made good. Early tests in the simulator had used a different pit model, which was much larger than the West Desert Sinkhole model that we ultimately selected. This meant that the rover spent much more time driving at its top speed, and less time executing turns or moving slowly near the brink. In this model, we had no time meeting our speed made good requirement. Because of this, we treated that requirement as completed early on, and neglected to confirm that our system continued to meet it as we made additional changes and improvements. When we switched to the smaller West Desert Sinkhole model, the rover's speed made good dropped significantly, and we were unable to address this before the conclusion of our project.

8 Project Management

8.1 Schedule

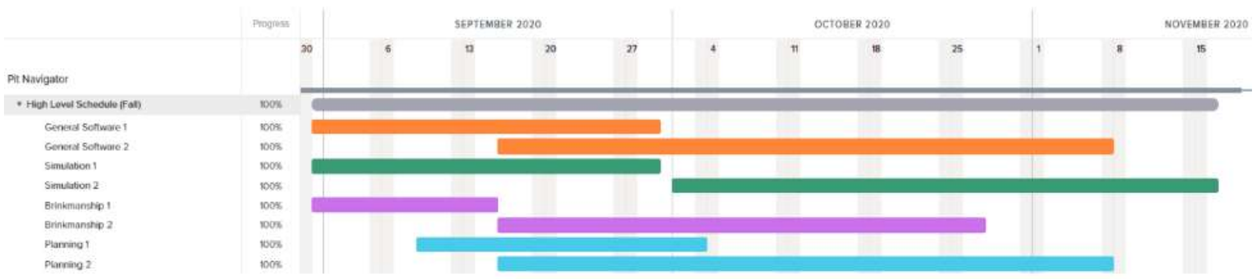


Figure 19: Fall 2020 Schedule

Figure 19 shows the final high-level schedule developed for the fall semester. Each item listed in this schedule was broken down further into individual sub-tasks. The duration of the high-level items represents the time from the beginning of the first sub-task to the end of the last sub-task contained within that item. Each high-level item was also split into a first and second half, because the project was expected to encounter a point where no work could be completed on any item until all the items had reached a certain level of sophistication.

The four categories of items were General Software, Simulation, Brinkmanship, and Planning. General Software referred to code that was not directly related to any of the project subsystems, but was still necessary to control the surrogate rover or otherwise allow the system to operate. The Simulation category was work related to the simulation and the execution of the simulated mission. The Brinkmanship and Planning categories contained action items for the corresponding subsystems. By the beginning of the fall semester, there was minimal work to be done on the Camera Operation subsystem, so no category was created for it.

In practice, the team did not adhere closely to this schedule. The process of designing the schedule provided valuable information about the dependencies of various tasks. This information was enough to guide the team's development priorities without enforcing strict deadlines. At multiple times throughout the semester, progress was compared to the schedule to provide an estimate of whether or not the team was on track. At the time of most of these comparisons, it was found that progress was about 10% behind schedule. At the end of the project, however, the team was able to make up this difference and complete all scheduled tasks.

A loose schedule was somewhat unavoidable, given the circumstances of the project. Other assignments, delivery times for off-the-shelf components, and uncertainty about the amount of testing and troubleshooting needed for various tasks meant that accurate estimates of task durations were difficult to achieve. The operating assumption when designing the schedule was that each task should be given much more time than was strictly necessary, and the tasks would be worked on intermittently during those durations. Furthermore, with only three team members it was difficult for any team member to devote a significant amount of time to enforcing the schedule.

This lax approach to scheduling had downsides. When difficulties were encountered on some task, there was no incentive to push through the trouble in order to complete the task by its deadline. Instead, that task would frequently be put by the wayside in order to work on easier tasks, unless it was truly creating a bottleneck to progress. Even tasks that were not that difficult would often be left at 80 or 90% completion, with the assumption that the last bit of work would be completed at some unspecified time in the future.

For future projects, a better approach would be to create a specific schedule and then adjust it frequently based on the realities of development. In a larger team, this would be more possible, because the time dedicated to schedule management could be split between other members of the team. Setting specific deadlines for tasks ensures that loose ends are resolved promptly and dependent tasks do not need to be postponed because their prerequisites are unmet.

8.2 Budget

Our team had a significant advantage in the budgeting of our project. Our project is contained within the umbrella of a larger development (the PitRanger project) which has significant financial resources that we can leverage. Because we expect our costs to be minimal, and to be able to outsource many expenses to the wider organization, our expected budget is fairly low.

However, our total spending until the end of the project has gone upto $2300. Most of the expenditure has been with regards to building a new robot. The complete list of items has been shown in Table 8.

Item	Quantity	Cost/ea.	Total Cost
12v 6A Regulator	2	$23	$46
Glass Standoff 1/2"x2"	1	$17	$17
Handle	1	$10	$10
USB Display	1	$100	$100
IMU	1	$276	$276
Wheels	4	$23	$92
FTDI Cable	2	$18	$36
Hubs	8	$10	$80
Motors	6	$175	$1,050
Roboclaw 2x15A	1	$90	$90
Roboclaw 2x30A	1	$125	$125
Encoder Cable 4 pack	2	$13	$26
Encoder Cable Breakout Board	8	$2	$16
Header Socket	1	$11	$11
1/8" Aluminum Sheet	2	$0	$0
Pan-Tilt Mount	1	$100	$100
Wire 1	1	$35	$35
Wire 2	1	$15	$15
Power Switch	1	$8	$8
Heat Shrink Tubes	1	$9	$9
Standoffs	1	$12.00	$12.00
Left angle USB	1	$6.50	$6.50
Right angle USB	1	$7.50	$7.50
Arbotix M board	3	$40	$120
Battery Connector	1	$7	$7
Total			$2,295

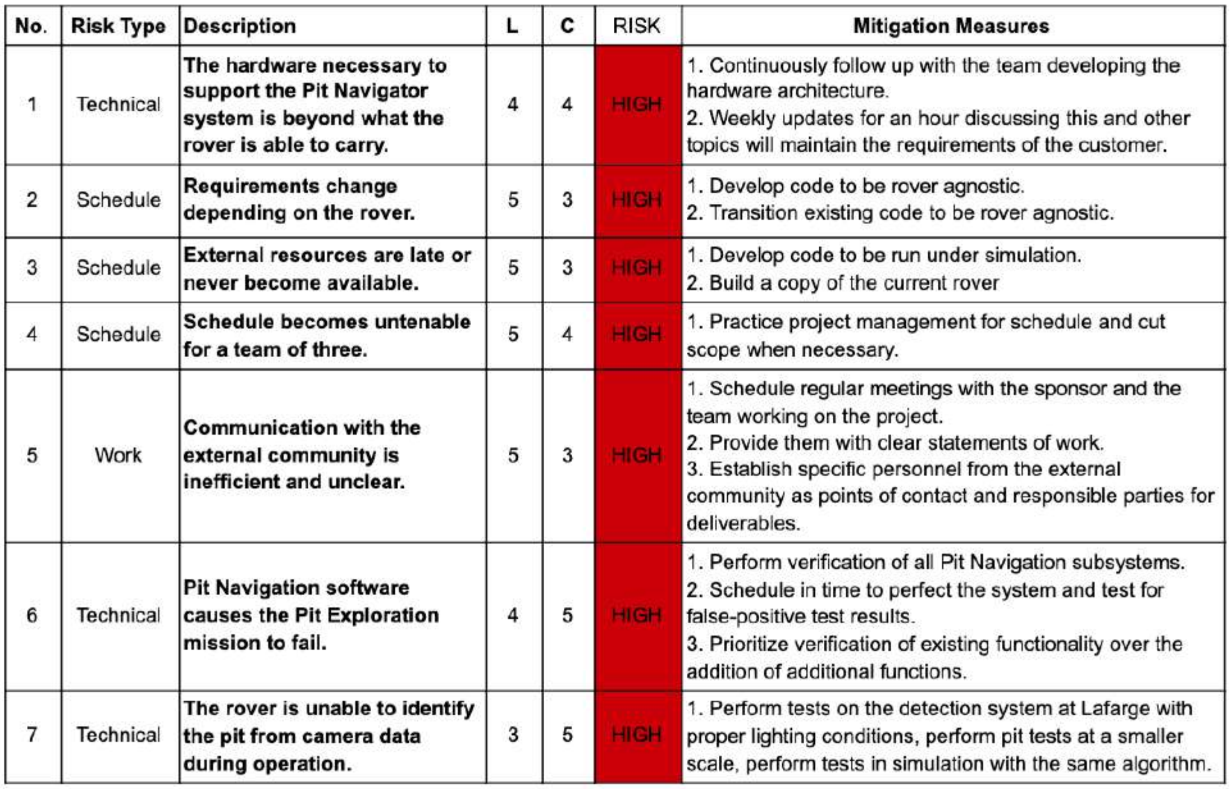
Table 8: Expense Summary

Our spending upto the spring semester was close to $1700. The remaining amount was expended on purchasing backup parts during the fall 2020 semester. We also purchased a USB display monitor during the fall semester for debugging purposes while testing out in the field.

8.3 Risk Management

Throughout the project, we kept track of risks that could cause a time delay, a work delay, a monetary increase, or an utter failure of the project. The idea was to update and check on the risks weekly so that we always knew what could go wrong and could take actions to avoid those outcomes. Then when the risk had passed the team would take it off the list and move on.

Once this was put into practice, the framing for the first set of risks was off, and caused many problems throughout the project. The first set of risks is in the table shown in Figure 20. These risks stayed exactly the same until the spring semester when schools began closing for the pandemic.

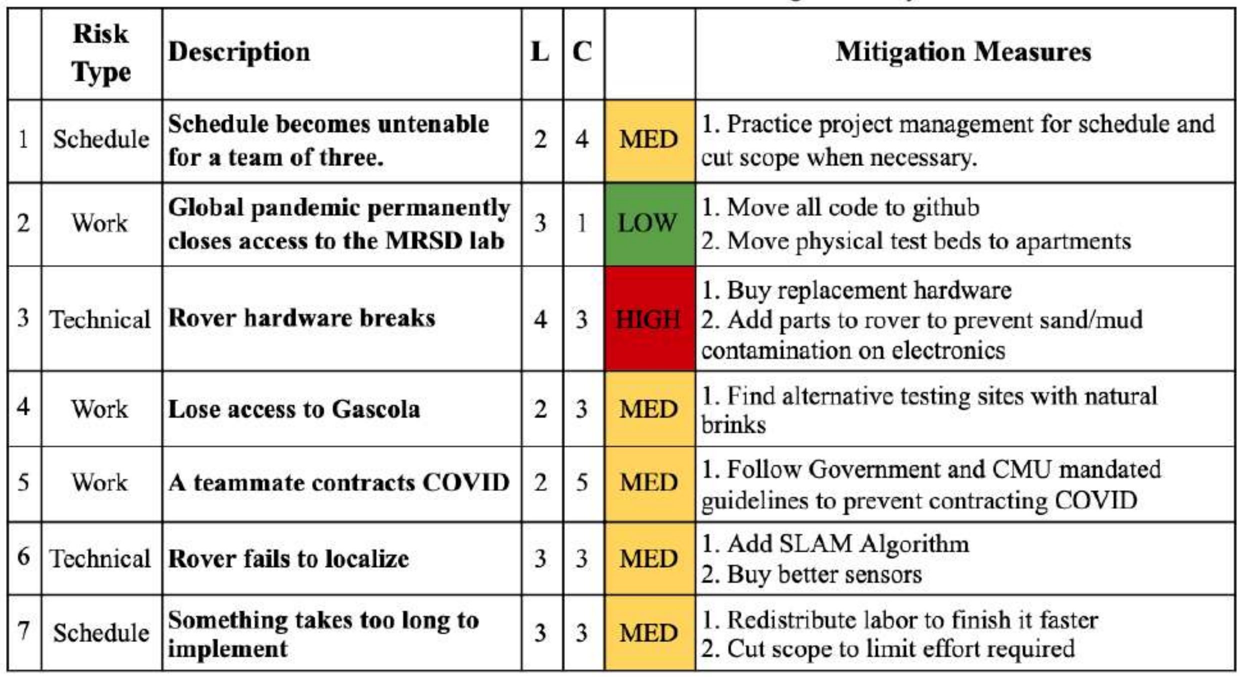


No.	Risk Type	Description	L	C	RISK	Mitigation Measures
1	Technical	**The hardware necessary to support the Pit Navigator system is beyond what the rover is able to carry.**	4	4	HIGH	1. Continuously follow up with the team developing the hardware architecture. 2. Weekly updates for an hour discussing this and other topics will maintain the requirements of the customer.
2	Schedule	**Requirements change depending on the rover.**	5	3	HIGH	1. Develop code to be rover agnostic. 2. Transition existing code to be rover agnostic.
3	Schedule	**External resources are late or never become available.**	5	3	HIGH	1. Develop code to be run under simulation. 2. Build a copy of the current rover
4	Schedule	**Schedule becomes untenable for a team of three.**	5	4	HIGH	1. Practice project management for schedule and cut scope when necessary.
5	Work	**Communication with the external community is inefficient and unclear.**	5	3	HIGH	1. Schedule regular meetings with the sponsor and the team working on the project. 2. Provide them with clear statements of work. 3. Establish specific personnel from the external community as points of contact and responsible parties for deliverables.
6	Technical	**Pit Navigation software causes the Pit Exploration mission to fail.**	4	5	HIGH	1. Perform verification of all Pit Navigation subsystems. 2. Schedule in time to perfect the system and test for false-positive test results. 3. Prioritize verification of existing functionality over the addition of additional functions.
7	Technical	**The rover is unable to identify the pit from camera data during operation.**	3	5	HIGH	1. Perform tests on the detection system at Lafarge with proper lighting conditions, perform pit tests at a smaller scale, perform tests in simulation with the same algorithm.

Figure 20: High-Level Risks For Pit-Navigator Project

While these risks are indeed risks of the project, they will not affect day to day work. Risk 7, for example, is a fundamental problem of our project that we are trying to solve. It doesn't help to track this risk on a week to week basis. The time table for this is one year before it checks in again, at the end of our project. The majority of these risks that were identified are this way, and are project killers only in months. Once these risks were evaluated every week, with no change, risk checking and updating itself was ignored. This left us open to new risks going unidentified and causing an incident.

The mistake was realized, but only once the pandemic hit. We identified that CMU might close down campus two days before it happened and used our prediction to add extra time for moving equipment out of the lab. This correct prediction brought to light the real benefit of risk tracking, which is to deal with near term project issues and act on them before it is too late. The team re-evaluated the risks of the project in the near term and found more reasonable risks that have action items in a weekly cycle. An example of these re-identified risks are exemplified in table shown in Figure 21.



	Risk Type	Description	L	C		Mitigation Measures
1	Schedule	**Schedule becomes untenable for a team of three.**	2	4	MED	1. Practice project management for schedule and cut scope when necessary.
2	Work	**Global pandemic permanently closes access to the MRSD lab**	3	1	LOW	1. Move all code to github 2. Move physical test beds to apartments
3	Technical	**Rover hardware breaks**	4	3	HIGH	1. Buy replacement hardware 2. Add parts to rover to prevent sand/mud contamination on electronics
4	Work	**Lose access to Gascola**	2	3	MED	1. Find alternative testing sites with natural brinks
5	Work	**A teammate contracts COVID**	2	5	MED	1. Follow Government and CMU mandated guidelines to prevent contracting COVID
6	Technical	**Rover fails to localize**	3	3	MED	1. Add SLAM Algorithm 2. Buy better sensors
7	Schedule	**Something takes too long to implement**	3	3	MED	1. Redistribute labor to finish it faster 2. Cut scope to limit effort required

Figure 21: Near-term Risks for Pit-Navigator Project

In this case, Risk 3 was an incident that was waiting to happen, and was completely unnoticed. During the following week we ordered several new parts and equipment only to have our equipment begin failing left and right the next day. We saved a day of shipping time on some of our items by noticing the risk a day earlier than the incident, but it could have been a lot better had we been identifying these near term risks earlier.

Risk 6 also became an incident, but we had not started the mitigation for it and it caused a several day delay which also caused Risk 7 to trigger as well. We redistributed the labor and cut the SLAM algorithm to limit the effort required to get the rover to localize. While the incidents were resolved, beginning the mitigation strategy immediately after identifying the error would have saved a few days of work on the back end.

Risk 4 actually triggered as well, but we had not found an appropriate brink substitute. While we had not lost access to Gascola exactly, the brink that we wanted to use for testing was overtaken by casual dirtbikers. We moved on from that brink and on the way out of Gascola, found another appropriate brink that was accessible, due to sheer luck. Clearly once we started identifying the near term risks, it was clear that we should have been following this method through the entire project and it could have saved us from many incidents.

9 Conclusions

9.1 Lessons Learned

This project provided many opportunities to learn and apply new skills. No member of the team had extensive or formal project management experience prior to this project, so we divided those responsibilities between us to give everyone a chance to practice. Awadhut tracked the project budget, Alex managed the risks and their mitigations, and Justin maintained a schedule for the semester.

Another new skill was the development of a comprehensive test plan. With the limited time available to us, it was crucial that we make the most of every opportunity to test our hardware and software. In particular, the field tests required significant effort to set up, so to make that effort worthwhile we needed to perform as many tests as possible. We clearly established the deliverables for the test, and used those to derive what sort of data was necessary to collect. We identified variations of the test process that would give us the most useful distribution of data in the most efficient manner. We also learned to allow time for multiple iterations of the same tests, so that poor results in the initial tests could be corrected and retested.

Assembling a complex system such as the surrogate rover required several trade studies for different components and processes. Because our work will eventually be a part of a system that operates on the moon, the conclusions we reached from these trade studies were even more important. The choices we made needed to align with the development of the larger system and needed to be robust enough to stand up to the challenging environment. We learned to assess the most relevant criteria for deciding between similar options, and how to construct our studies so that the information was clear and readable.

This project provided many opportunities to develop technical skills, as well. We learned quickly that there is a difference between breadboard circuitry of the kind we had practiced in labs before and designing a functional electrical system for a robot. We learned to interpret data sheets for controllers, motors, and other mechanical components to ensure that each component was powered safely and effectively. Proper battery safety and storage was also an important skill for us to engage with, especially after moving to working from our homes. In the fall semester, we accidentally caused multiple batteries to drop below their minimum working voltage, a state from which they could not be recharged. This taught us an important lesson about monitoring battery voltage closely, and swapping batteries at the right time.

The software side of the project also presented a great deal of complexity, and with it many opportunities for us to learn. While we had some prior experience with ROS, integrating the various components of our system into a coherent ROS network was a new level of challenge. Working with different packages required us to troubleshoot version compatibility issues and read documentation to understand the underlying mechanisms. Through practice and hard work, our familiarity with the systems and packages increased, allowing us to work more effectively and address problems when they arose.

Much of the software developed for this project was originally designed and tested in simulation. Porting this code for use on physical hardware created challenges which taught us important lessons. One issue that arose during our first field tests of the integrated system was due to the difference in coordinate frames between the simulated rover and Blue. We spent a long time trying to troubleshoot the difficulties with Blue's navigation and point cloud generation routines before discovering this issue. This taught us to maintain an awareness of our assumptions, and actively verify those assumptions at every step of the process. Another issue related to the translation from simulation to real life was the messiness of point clouds generated from feature-matching stereo images. We learned that we needed to apply additional filters to the point cloud to reduce noise that caused our brinkmanship subsystem to behave erratically. Signal processing is an important task for many systems, and nothing drives that lesson home as strongly as experiencing the consequences of noisy signals firsthand.

9.2 Future Work

Although the system in its current state meets almost all of the requirements set for this project, there are still improvements that could be made. More testing on a wider range of terrain could reveal issues and edge cases with the brinkmanship algorithm that could then be corrected. An application-specific local planner could be developed, instead of the generic local planner that the planning subsystem currently uses. Path planning using sun position, a function that one team member worked on over the summer, could be incorporated into the planning subsystem as well. More sophisticated rover surrogates, which are closer in design to the proposed flight rover, have been developed since this project began. It would be worthwhile to implement the pit navigation system on such a rover to confirm that all functionality is able to make the transition. With additional time to scout for locations, it may be possible to find a terrestrial pit that could be used to execute an entire mission with a rover surrogate.

Beyond the scope of the pit navigation system, there is a great deal of work to be done before a flight-worthy rover has been developed. To begin with, the team must successfully propose the rover as a payload for a NASA mission. This may require adjustments to the rover design or mission plan, to comply with NASA's specifications. This rover must be able to localize itself and navigate through generic lunar terrain, not just the area surrounding a pit. It must also carry cameras capable of capturing images of the pit at the proper resolutions and exposure levels. Progress in designing the hardware and software aspects of such a rover are ongoing, but there is still work to be done.

While the pit navigation project was ongoing, new information was revealed about the nature of lunar pits. Specifically, lunar pits do not generally have sheer edges surrounded by relatively flat terrain. Instead, they tend to have a conical shape at the top, with a sandy slope at the angle of repose leading down to a more vertical wall at some depth below the surface. This presents a unique challenge for a rover that must find a vantage point showing the interior walls of the pit. The rover will need to traverse these treacherous slopes in order to reach appropriate vantage points, which will require a specialized design. Because this fact became apparent when the project was already well underway, the team made the decision not to alter the design of the surrogate rover or software to compensate. However, if the pit navigation software is eventually to be executed on the flight rover it must work under these conditions.

The nature of a long-term project executed by a team composed primarily of college students is that participants will come and go. Students will contribute when their schedule permits, and usually depart for good when they graduate. Every member of this team would like to continue to be involved with this project, but will be faced with a full course load making demands on their time. If others pick up where this project has left off, the team members will attempt to ensure that the work is well-documented and that they are available to answer questions and provide guidance. The hope is that this functionality will continue to be developed, and will one day allow a rover to complete its mission on the moon. Our team would take great pride in that accomplishment.

References

[1] Whittaker, William. *Technologies Enabling the Exploration of Lunar Pits*, NIAC Proposal, Robotics Institute, Carnegie Mellon University, May 2019.

[2] Whittaker, William. *MoonRanger, Flight-Forward Moon Rover with Exploration Autonomy.*. Google Drive, Robotics Institute, Carnegie Mellon University, 27 Feb. 2019,

[3] Ferguson, D. and A. Stentz. *"The Field D * Algorithm for Improved Path Planning and Replanning in Uniform and Non-Uniform Cost Environments."* (2005).

[4] Leonardo Araujo Santos. *Policy, leonardoaraujosantos.gitbooks, gitbook, 2017*.

[5] Real-Time Appearance-Based Mapping, 2019. http://introlab.github.io/rtabmap/

[6] Figure 6: Phatom-X Pan-Tilt Mount, https://www.trossenrobotics.com/phantomx-pan-tilt

[7] Figure 7: Arbotix-M Controller, https://www.trossenrobotics.com/p/arbotix-robot-controller.aspx

APPENDIX

1. **Blue:** Surrogate rover used for the project. See Figure 4.
2. **Brink:** A location where safe, traversable terrain suddenly ends in a near-vertical edge, such as a cliff or the rim of a pit.
3. **Brinkmanship:** The act of operating safely near a brink or area of unsafe slope. Specifically, moving the rover as close as possible to the brink or slope without entering unsafe terrain.
4. **Flight Rover:** The rover which will be sent to the moon to execute the planned mission.
5. **Lander:** The spacecraft which will land on the surface of the moon. The flight rover will be a payload aboard the lander.
6. **Mesh:** A representation of a surface or object consisting of a set of connected flat surfaces that approximate a complex 3D shape.
7. **Mission:** The series of objectives that the flight rover will execute while on the moon.
8. **Panorama:** An image composed of a series of smaller images which show overlapping scenery, connected into a single image file.
9. **Point Cloud:** A set of point locations in 3D space which represent the shapes of objects or terrain in that area.
10. **RealSense:** A line of cameras produced by Intel which provide cheap computer vision functionality, including depth perception, stereo vision, and LiDAR. RealSense also provides an API of related software functions.
11. **Rover:** A wheeled robotic vehicle.
12. **Surrogate Rover:** A rover built to operate in a field environment on Earth, with qualities comparable to the eventual flight rover. Used for testing software features.
13. **Unsafe Slope:** A slope on which the rover would be unable to maintain traction, causing it to slide or roll down the slope. For this project, the threshold for safety was set at 20 degrees.
14. **Vantage Point:** A location very close to the pit edge, from which the rover will capture images of the pit interior.
15. **Waypoint:** A preset location in the map of the pit area, which the rover will use as an input to its planning subsystem when planning and navigating.