Executive Summary

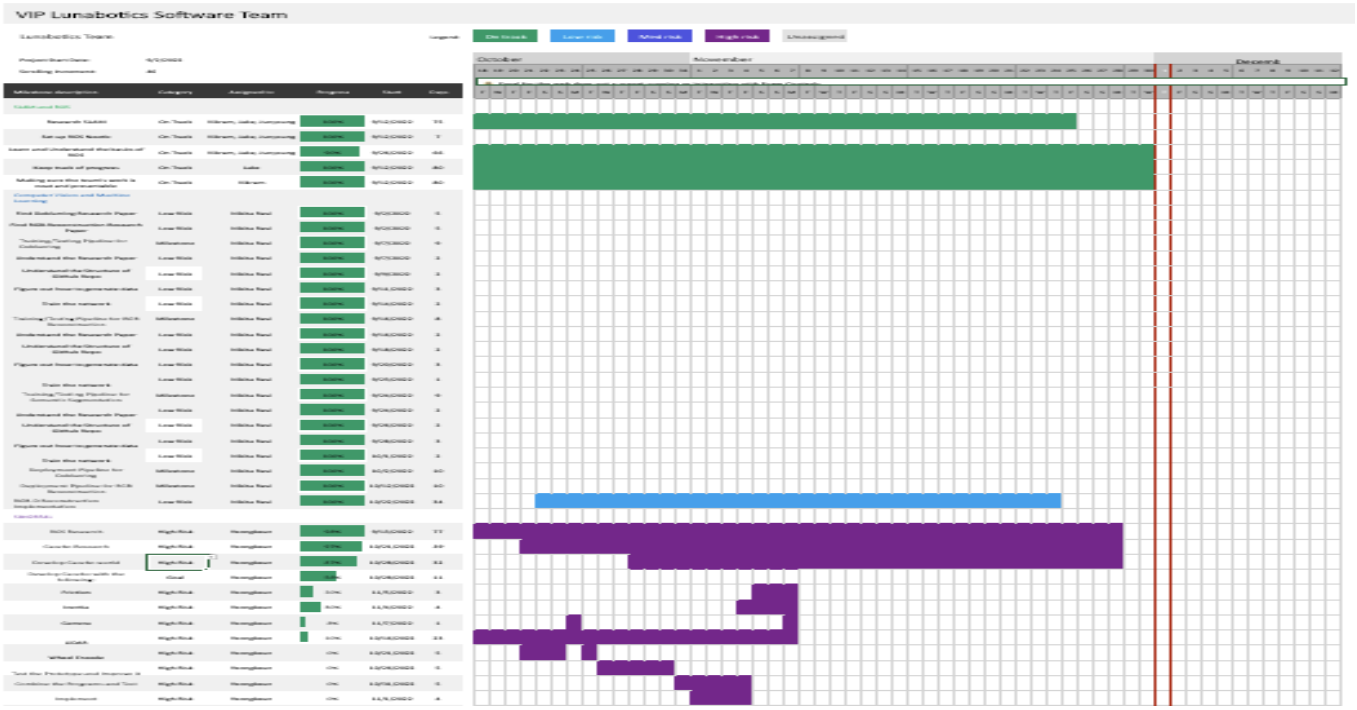
The Objective

The objective of the software subteam of Lunabotics was to develop a SLAM algorithm to localize the lunar rover in the lunar environment and build a map of the surroundings. To achieve this, we required data collected from multiple sensors, one of them being a top-down view of the lunar surface with respect to the front-facing photo the rover took and this is accomplished by designing a computer vision pipeline. We also need to design a simulator to generate data to test the localization output from SLAM.

The projects we worked on this semester are as follows:

Project	Contributors
Senior Design: Computer Vision Pipeline	Nikita Ravi
Senior Design: RGB Depth Reconstruction	Nikita Ravi
Senior Design: Gazebo Simulation	Yeongkeun Kwon
SLAM and Sensor Fusion	Jake Stall, Junyoung Kim, Vikramaditya Baid

This is our final updated Gantt chart for this semester:



LINK:
https://purdue0-my.sharepoint.com/:x:/g/personal/kim3722_purdue_edu/Eadoj7nq9h1MrlKaJZHZr1kBW_UdprNeLByK62Lhw5LSHQ?rtime=mvkTHbvR2kg

Software Projects Design Decisions

Computer Vision Pipeline

The Computer Vision pipeline consists of three different neural networks each with a different purpose. The first neural network is responsible for deblurring the images obtained from the lunar rover. The second neural network is responsible for semantically segmenting or extracting features from the deblurred images (i.e. small pebbles vs large rocks). The third neural network uses the semantically segmented image to transform the image to a birds eye view (BeV) perspective. The deblurring neural network has a validation PSNR of 24.62 and SSIM of 81%. The semantic segmentation network has a validation precision of 95%. Finally, the BeV neural network has a validation categorical accuracy of 96% and Mean IOU of 88%. The structure of both the deblurring and BeV networks were maintained so that the results would be comparable to the author's original high results. The Deblurring and BeV networks could be much better if we had more data to work and train with.

RGB Depth Reconstruction

The purpose of the RGB Depth Reconstruction project is to provide a means for which to create a custom dataset that can be used to train the computer vision pipeline with. RGB Depth reconstruction is essentially to reconstruct a 2-dimensional RGB image into a 3-dimensional structure by gathering both depth and RGB information from the Lunabotics club's preferred Intel realsense D455 camera. We chose to proceed with the same camera as the club because at the end of the day my programs will be executed on the camera the club will use for the NASA competition. Therefore a specific set of intrinsic and extrinsic camera parameters will be used for the reconstruction.

Gazebo Simulation

The purpose of my project, which is called Sim2Real, is to develop and to re-design the robot simulator in order to facilitate testing the designed robot virtually which allows for saving time and resources. My proposed product is to design and develop Gazebo simulating programs by integrating them with Robot Operating System. Through the Sim2Real project, it is expected to provide much better testing environments.

During this semester, First, I spent the most time researching how to realize real-world components and physics in the virtual world. In order to reduce gaps between the real world and the virtual testing environments, it is important to acknowledge how physics like gravity, inertia, and friction is applied to the robot in the real world, and those should be applied to the program. Then, I proceeded to research ROS and Gazebo simulation. Because I have not utilized those tools before, I encountered some issues with installing that software on Mac OS. At the time, I thought the installation process would be able to be done without allocating virtual memory on my computer, and this caused me to spend a lot of time. By researching completely regarding the software, I was able to build the environments for running the Gazebo simulation program and ROS which are necessarily required for the Sim2Real project.

In the beginning of the semester, I planned to build the robot simulator first so that it would be able to test with Nikita's project at the end of the semester, and the URDF file for the designed robot should be provided first. However, in the process of collaboration with the Lunabotics software subleader, we decided to work on the testing environment first as the robot design was not finished. So, during the last of the time allowed, I spent the most time on building the virtual

moon environment by coding in Python and parsing world files, composed of HTML-like programming language, and tested it by utilizing the Turtlebot simulator and the SLAM. After several tests, it was confirmed that the robot was navigating the map and drawing the map, but after about 5 minutes, the simulator stopped and did not respond. From the test results, I made some hypotheses that might cause this error; this might be because of the map's size, the physics applied on the Turtlebot might not be applicable, or the Ram size set to the virtual machine might not be enough to simulate the program. So, for the future project, I would spend the most time verifying those hypotheses and building our robot-simulating component by collaborating with the club.

SLAM and Sensor Fusion

The goal of this project was to develop an algorithm that performs SLAM by using sensor information that we obtain from the various sensors on the robot. Initially, we attempted to code SLAM in ROS without having a solid understanding of its fundamentals. This resulted in very little progress being made on the project, as SLAM is a very broad concept that has many different types and variances. Thus, a major design decision that we made as a team was to spend most of our effort doing extensive research on SLAM before attempting to develop the algorithms required. The rationale for this was that if there are holes in our understanding of SLAM, we would inevitably need to return to researching the topic, reducing our efficiency and overall quality of work.

The major topics that we researched were probability theory, Bayes filter, Kalman Filter, Sensor Fusion, and ROS, all of which SLAM relies upon. Probability theory helped us obtain a concrete representation for the state of the system, and it helped us factor in any uncertainties and measurement errors to give an accurate representation of the state. The Bayes filter gave us a general recursive framework for performing state estimation by simply specifying the motion and observation models for the system. We then understood how the Kalman Filter is a special case of the Bayes Filter that assumes Gaussian distributions and linearized models to give us a systematic algorithm to perform state estimation. Next, we looked into methods of sensor fusion that would take advantage of the different sensors used like the encoders, LIDAR, UWB, depth cameras, and BEV images, and combine them to give us a more accurate state estimation algorithm. Finally, once we believed that we had gained a good

understanding of SLAM, we began looking into ways to implement the algorithms. For this, we would need to learn how to use ROS - which would provide us with an efficient framework to structure our code, communicate sensor data, and prebuilt packages that help us solve subparts of our main goal. This is where we finished off this semester. And with the semester's end, the merit of our aforementioned rationale for our early-semester major design decision was confirmed, as we now have a solid understanding of the fundamentals of SLAM and are ready to begin coding next semester.

For our future design improvements, we need to start by developing an architecture for the entire algorithm. We need to specify how different nodes will interact with each other. We need to understand the specifics of how the sensor data is stored and communicated. We need to find useful ROS packages that can help us with our code. Finally, once we have developed our first algorithms, we have to find ways to test and make improvements to our code.

Individual Teammate Contributions:

Vikram

- Researched and understood the mathematics behind SLAM
- Helped fellow teammates understand SLAM
- Documented research and rewrote ideas and concepts as a tutorial to help teammates understand.

Jake

- Researched the learned the fundamentals of SLAM
- Gathered specifications (data format, data type, etc.) for the sensors that will be used in sensor fusion
- Documented SLAM and sensor fusion research

Junyoung

- Researched and understood fundamentals of SLAM
- Studied ROS basics
- Documented summary of SLAM algorithm step by step