ILR08 - Progress Review 9

Justin Morris
Teammates: Awadhut Thube, Alex Withers
Team G: The Pit Crew

October 15, 2020

1 Individual Progress

Team G's acquisition of one of the lab computers has been a massive boon to my productivity in the past two weeks. With remote access to this computer, I have been able to access the simulation environment and begin writing and testing code without needing to rely on my teammates for help testing and debugging. Awadhut and I have agreed to split our time with the computer, so that I have priority before 2 pm and Awadhut takes over after that. This has provided motivation for me to begin being productive early in the day, in order to make the most of my time.

Our major goals for Progress Review 9 all revolved around being able to transform point clouds into triangular meshes. Awadhut identified the PyVista library as a useful method of generating triangular meshes, and implemented some basic functionality into a ROS node that could be run alongside the simulation [1]. This node subscribed to a channel which published point clouds of the rover's perspective, converted them into point clouds, and produced a visualization of the resultant mesh. Building off of Awadhut's work, and with reference to the PyVista documentation, I was able to improve the robustness of this mesh generation process and color the visualization to differentiate between safe and unsafe terrain. Further explanation of my contributions can be found in section 2. The output of this node can be seen in Figure 1.

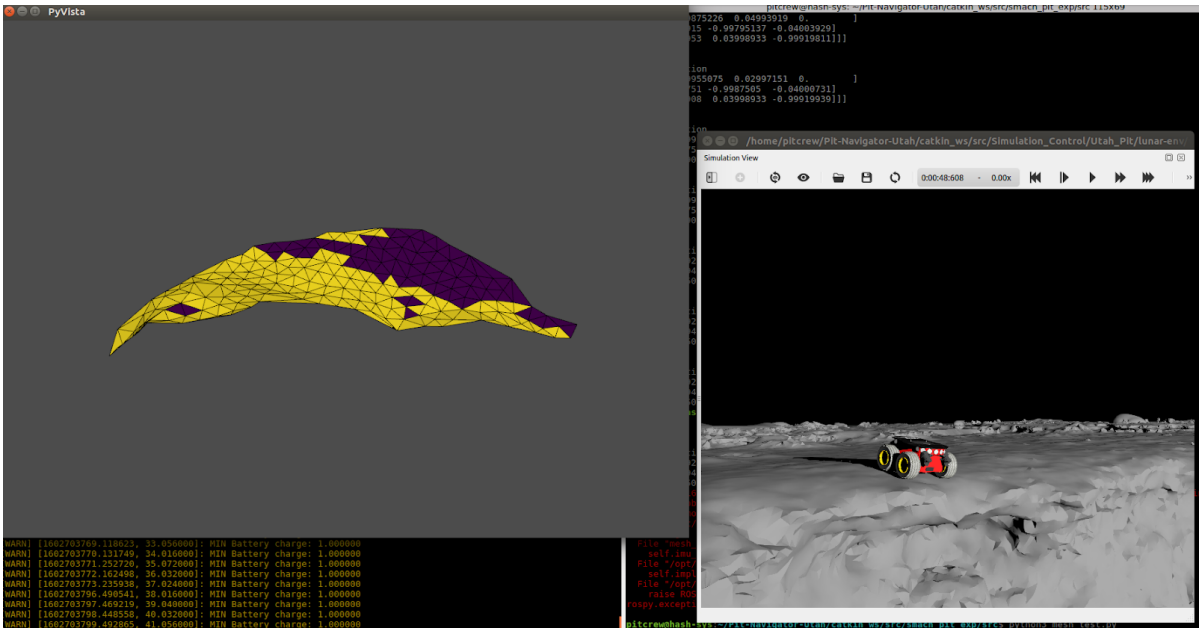

Figure 1: An example of a mesh generated while the rover is positioned facing the pit edge.

Terrain safety, or traversability, is measured according to the magnitude of the vertical component of the unit normal vector for each cell in the triangular mesh. If the vertical component is less than 0.93, that indicates that the angle between the normal vector and the vertical axis is greater than 20 degrees. To create the mesh visualization, I created a boolean array with values of True if the corresponding mesh cell was unsafe, and False if it was safe according to this metric. This array was provided to PyVista's plotting function to use as the basis for coloring the mesh. The end result was a plot of the mesh in which safe cells were colored purple, and unsafe cells were colored yellow (this color scheme could be changed by providing the PyVista function with a different color map).

2 Challenges

In the lead-up to Progress Review 8, it had become clear that we needed access to more computers that were capable of running the Webots simulator if we wanted to make progress at an appropriate rate. We obtained permission to collect one of the lab computers from the MRSD lab, so Awadhut and I headed to campus to move the computer to Awadhut's apartment. From there, Awadhut set up remote access so that I could also use the computer. Configuring remote access to allow me to connect to the computer without Awadhut being present on the other end to assist was a process that involved some challenges of its own, but we eventually got it running. The connection has some small amount of lag, but overall it works well enough for me to contribute to the project in ways I could not before.

When I first began to work on the mesh code, Awadhut warned me that the meshes being produced were of poor quality when the rover was near a steep slope or brink. I confirmed this myself and began reading the PyVista documentation and tweaking the code to find a solution. I eventually determined that the issue was with the 2D Delaunay triangulation algorithm that we were using to generate the triangular meshes. The algorithm attempted to find a best-fit plane to use as a basis for mesh generation, and when the rover was close to an edge the best-fit plane for the resultant point cloud was more vertical than horizontal, even though the terrain itself was still mostly horizontal. This resulted in the algorithm attempting to build a horizontal mesh out of vertically oriented triangles, and produced the strange surfaces that Awadhut and I observed. At its root, this issue turned out to be related to the way that we were thinning our point cloud to speed up mesh generation. I was able to find a function within PyVista that cleaned up point clouds, and using that function instead of our previous method resolved the issue with mesh generation near brinks and slopes.

3 Teamwork

This period since acquiring the lab computer has contained a lot of teamwork. Because Awadhut and I are sharing access to the computer, we naturally have to trade off and keep each other up to date on our progress. Implementing the mesh generation and heuristic was a team process. Awadhut discovered the PyVista library and wrote the first prototype of the ROS node. We worked together to develop and debug our original method of thinning the point cloud. When we discovered the issue with mesh generation near brinks, I found the solution in PyVista and reworked the code. I also figured out how to color the visualization that Awadhut had initially incorporated into the ROS node. Awadhut used the rover IMU data to transform the point cloud so that the resultant mesh normals were in the world frame.

This was also a payoff for Alex as the simulation expert. Since Awadhut and I had not had easy access to the simulation until now, we benefited greatly from his guidance on how to use the simulator. Alex also figured out how to extract IMU data from the simulated rover, which allowed Awadhut to use that data to transform the point clouds into the proper coordinate system.

4 Plans

Our most immediate plan is to take this mesh-based terrain assessment and incorporate it into the brinkmanship routine that the rover runs when navigating to a vantage point. The rover should be able to determine from the mesh how far ahead it can travel safely, and where the pit edge is located. To do this, we will need to find a way to interpret the negative space in the mesh as being empty space, in addition to assessing cells in the mesh for traversability.

In addition to this, we want to test the mesh generation on real-life environments. We will collect stereo imagery from Blue, and apply the same procedure. This will be the first step in porting our entire codebase to

Blue, so that we can test all functionality on a physical rover surrogate. One of our goals for Progress Review 10 is to have Blue perform navigation to multiple waypoints, so that will also be a priority. Once Blue can navigate to waypoints and perform the brinkmanship routine, it will be ready for field operation of the type that we will demonstrate in our Fall Validation Demonstration.

After those tasks, it will then be a process of integration and validation to reach a final product. In both the simulation and in real life, we need to make sure that every aspect of our code works together without error, and is robust to the variety of environments and terrains that we might expect to encounter over the course of a lunar mission. Our project will ultimately be critical to keeping the rover safe as it explores the moon's surface, so we need to do everything we can to make sure that it is up to the challenge.

5 References

1. https://docs.pyvista.org/