ISTA 521 Final Project - Autonomous Asteroid Surface Navigation

Steven Morad

November 2017

1 Introduction

Asteroids give us insight into how the solar system was formed [1]. Low gravity and rugged terrain make asteroid exploration a risky, but rewarding endeavour. Previous missions to asteroids have generally been orbital reconnaissance or flyby [2], but surface missions are required for a better picture of asteroid structure and composition. Previous literature has suggested climbing robots as a means for low-gravity exploration [3]. A means of autonomous navigation are proposed for these systems of robots. Mimicking a mountain climber, the robot looks for climbing holds to place its microspine gripper hands and feet. Anchoring a limb to a climbing hold reduces the chance of inadvertently slipping, escaping the asteroid. Preliminary simulations have shown that this method warrants further investigation.

2 Implementation

2.1 Data Generation

A lidar system mounted on the robot will scan the surrounding terrain. Point-cloud data from the scan can be converted into a polygonal mesh (Fig. 1). With rocky and hilly terrain, scans will be missing data for obscured areas (Fig. 2). This is troublesome, as generating a path through a mesh with discontinuities is very difficult. A continuous surface can be generated by extrapolating missing data, using an approach known as screened Poisson surface reconstruction (Fig. 3)[4]. Not all the extrapolated data will be accurate, but as the robot moves closer to extrapolated terrain, it can re-scan to update the terrain map. Simultaneous localization and mapping (SLAM) will need to be integrated for map re-scanning to take place. It will be difficult to integrate SLAM without the odometry provided by wheels, but [5] shows SLAM is possible to implement using visual odometry.

Once the terrain has been converted to a continuous surface, it can be searched for climbing holds. There are many methods that can be used to

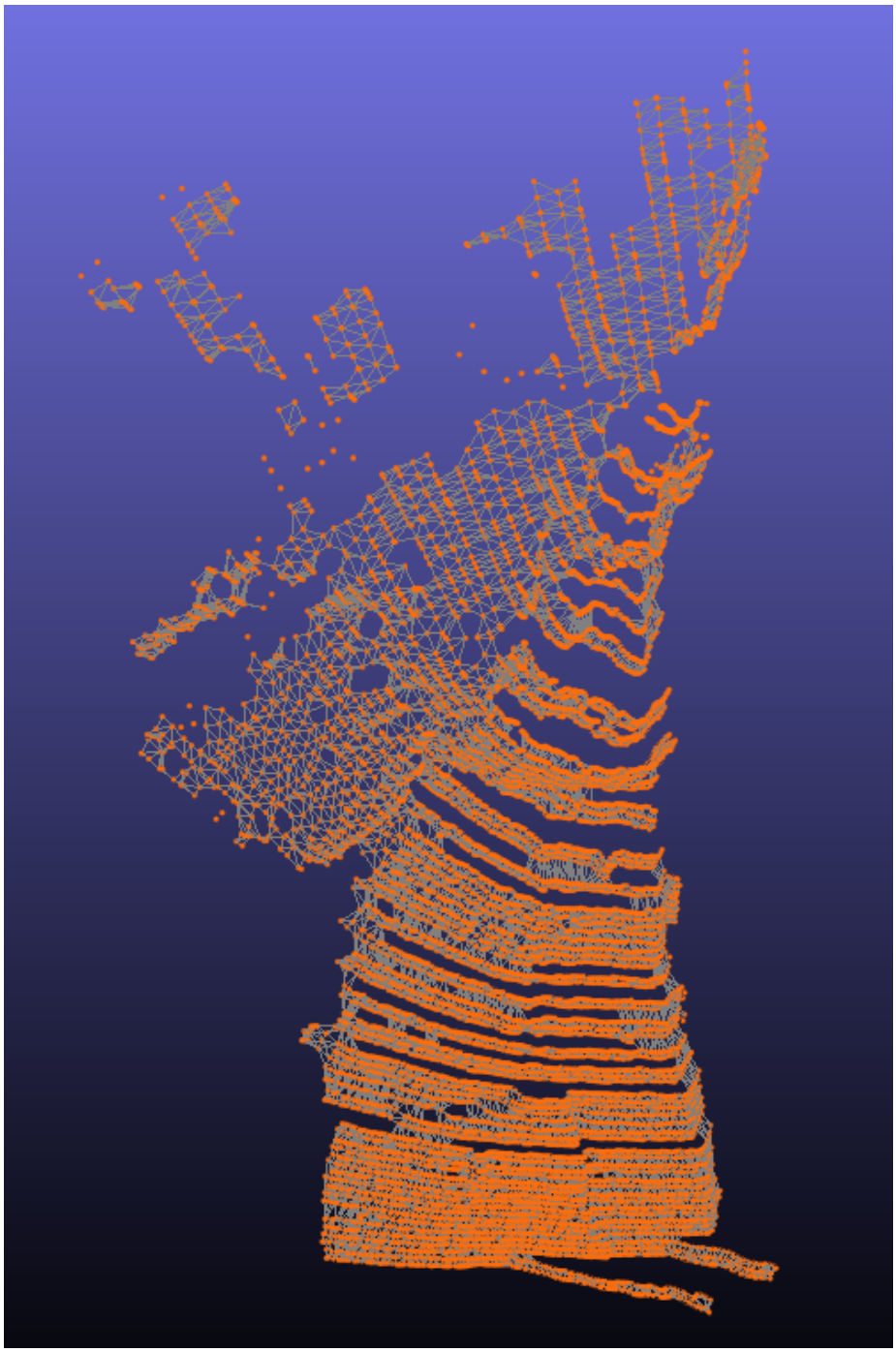

Figure 1: Bird's-eye view of the lidar point cloud in orange, and polygonal mesh in grey. Note that gaps are present in the mesh, which would make it difficult to plot a good path. Screened Poisson surface reconstruction has not been applied yet.

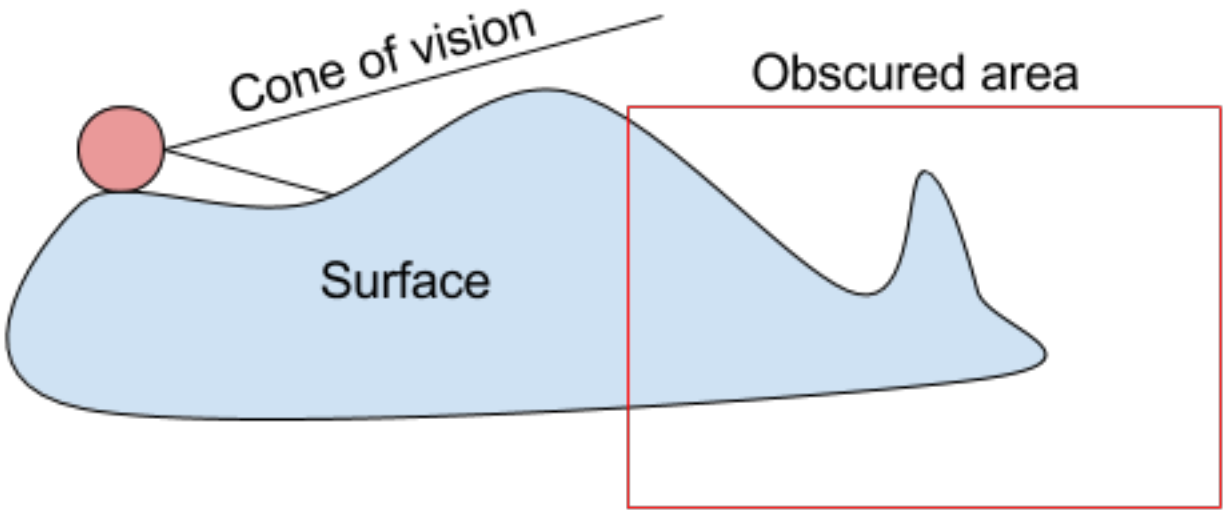

Figure 2: Occlusion caused by uneven terrain.

determine what constitutes a climbing hold. We assume a planar array of microspine hooks, attached to a robotic "wrist" with three degrees of freedom. A flat climbing surface would maximize the contact area with the gripper. To search for planar climbing surfaces in any orientation, a novel machine-learning based algorithm defined below is used to compute loss at every vertex in the surface.

2.2 Core Machine-Learning Algorithm

$$L(v) = \frac{\alpha}{n}\sum_{i=1}^{n}(P_i - S_i) + \sum_{k\in S}\left\|\nabla\vec{k} \times \nabla\vec{P}\right\|$$

Where L is the loss function at vertex v. k denotes a neighbor of vertex v in surface fragment S. The SciKitLearn LinearRegression.fit method is used to calculate a parametric best-fit plane P to the surface fragment S described by the vertex v and its neighbors k. P and each neighbor k have their gradients taken, and their cross products computed and normed. The cross product of each k and P is used to calculate the difference in angle between the surface fragment normal vector and the best-fit plane normal vector. The idea is that a bumpy, hilly surface fragment S will have normal vectors pointing all over the place. This means that the sum of the cross products between the hilly surface normals $\sum_{k\in S} \nabla k \times \nabla P$ will be a large number. For the case where the surface fragment S is almost planar, the cross product between the surface fragment normals $\sum_{k\in S} \nabla k$ and best-fit planar normal ∇P will approach zero (Fig. 4).

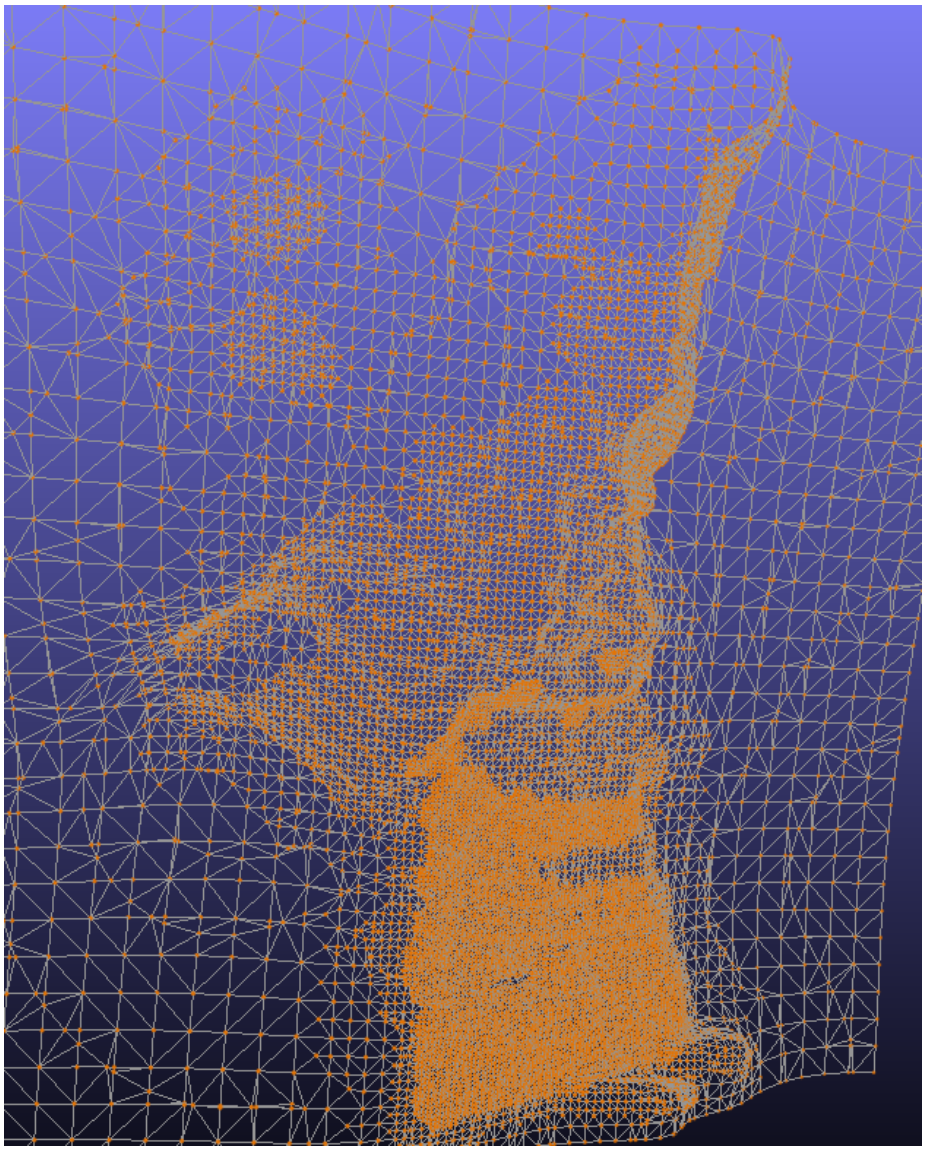

Figure 3: Poisson reconstructed mesh, with vertices shaded orange.

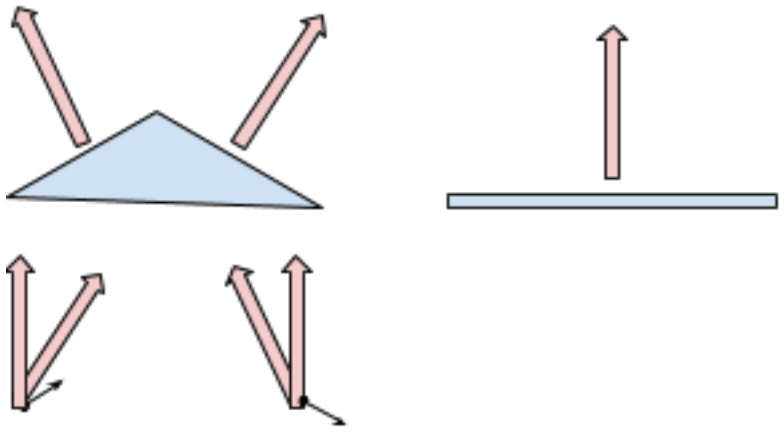

Figure 4: Surface fragment S is the blue triangle on the left, best-fit plane P is on the right. Normals of S and P are in red. The cross product is shown below, as the black arrow going into or out of the paper.

The first term in the loss function is simply the mean squared error between the plane P and surface fragment S times some weight α. This term is used to penalize surface fragments where the plane does not fit well. K-fold cross validation was considered instead of mean squared error, but was not used for a few reasons. First of all, k-fold cross validation is expensive, especially to run at every vertex in a surface. Secondly, k-fold cross validation would tell us how well our algorithm would predict new points, and prevent overfitting. We are not looking to do either of these. We do not want to predict any values that are not present in the surface, and since we are fitting only a plane (degree one polynomial) overfitting is not an issue.

2.3 Path Planning

The vertices with the smallest loss $L(v)$ are selected to be climbing holds, and form a directed graph. Euclidean distance and loss will be used to generate two separate edges in the graph, as pictured in Fig. 5. A new shortest-path algorithm we call bounded-leg A* is used to generate a constrained path between two nodes where vertex loss $L(v)$ is minimized. The algorithm will ignore any edges with a euclidean distance greater than some physical hop distance J in the case that each limb is controlled via a cold-gas thruster or hopping mechanism. Bounded-leg A* is run concurrently for each limb, generating a separate path for each limb. Note that A* will not return the absolute shortest path in this case, but the shortest path with the hop distance restriction. The algorithm will check each move, ensuring that the limbs do not diverge too far from each

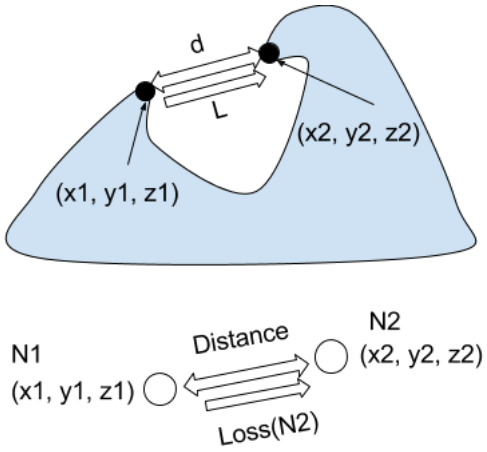

Figure 5: Graph construction from a mesh of vertices.

other.

3 Evaluation

Evaluation is a tricky subject in this case. There are no models for aggregate microspine adhesion. There are models that describe the mechanics of a single spine on a microscopic surface asperity, but there is no way to measure these asperities without a microscope [6]. Here, evaluation was done visually and logically. Two other approaches for determining flatness were attempted.

In one case, surface normals were not computed. Instead, only the the mean squared error M was used for vertex loss. This led to the best handhold points pooling in three regions, and towards the edges of the surface. This method was not used, as it did not produce results that could be used for navigation (Fig. 6). Additionally, M would score poorly in the case of having a deep divot (Fig. 7). M would be very high, but to the gripper the depth of the divot would not matter. Conversely, a surface with many microbumps would score too well, because the distance between the plane and surface would be small.

In the second case, 2D convolution was used. This method was much slower, and would timeout unless the gripper size was extremely small. Similarly, it would also fail in the same case as M when scoring deep and narrow divot, or a surface full of microbumps.

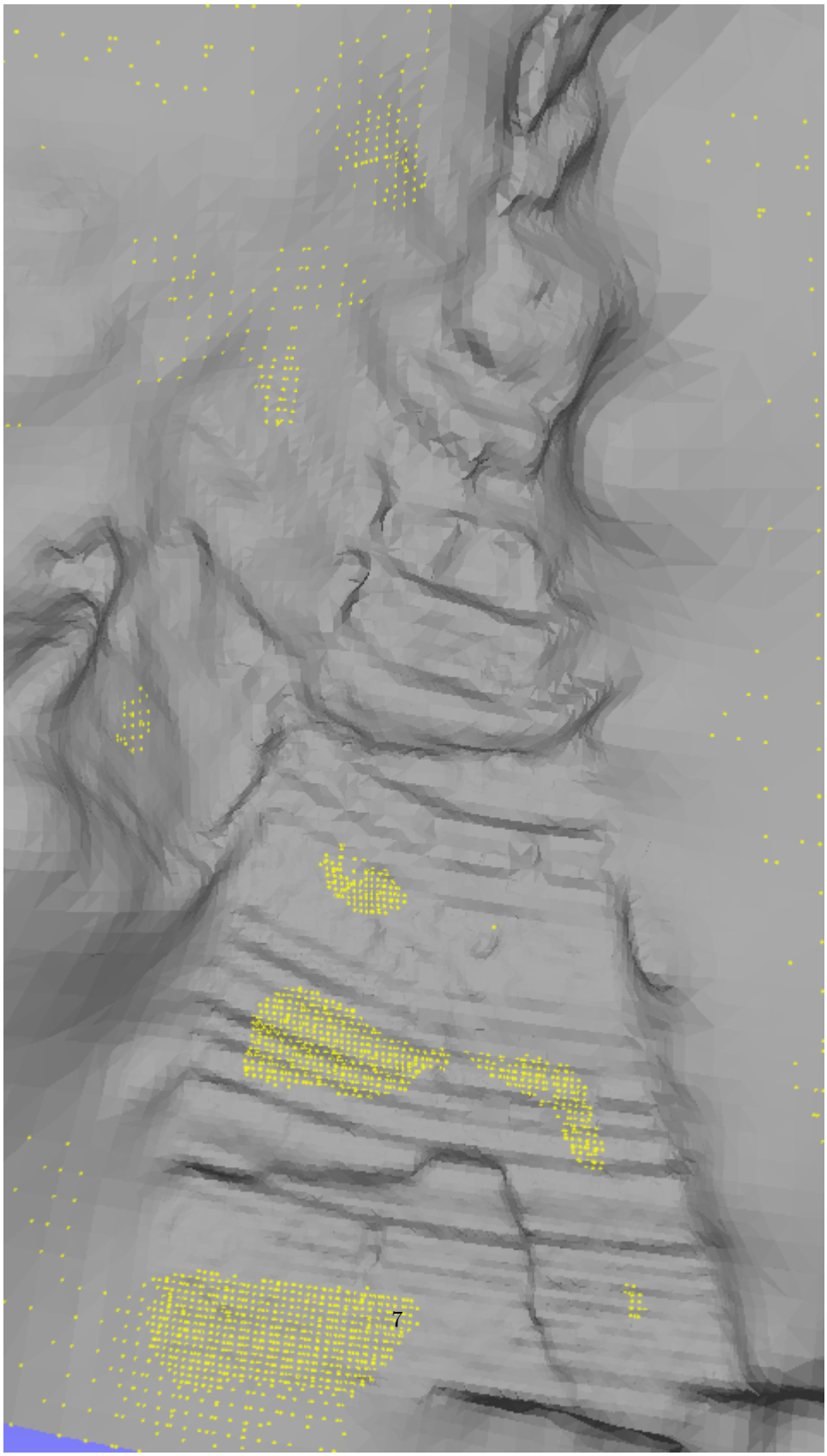

Figure 6: Holds generated using only MSE loss.

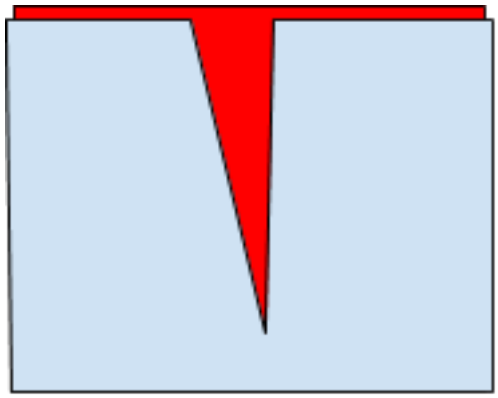

Figure 7: Visualization of convolution and mean squared error scoring with a deep divot. The red area (the difference between the blue surface and a plane) makes up a large amount of the total area, even though the blue surface is mostly flat.

4 Results

A lidar scanner was built and run using a raspberry pi. The point-cloud data was reconstructed into a mesh using MeshLab, a tool used by digital animators. The mesh was over 18,000 vertices, approximately 4000 units long, and 4000 units high. If the scanned rock wall is assumed to be 5m tall and 5m across, then each unit is 1.25mm by 1.25mm. The grippers were assumed to be circles of radius 0.75cm.

The graph generation and handhold detection were run on this mesh. A four legged robot was assumed, with a leg length of 500 units (62.5cm). The generated path is pictured in Fig. 8.

5 Results Replication

To replicate our results, please note the following: Mesh generation can take a while. On an eight-core AMD FX-8350, generation of the graph and handholds can take anywhere between five and ten minutes. To save time, a high quality graph and handhold have been provided in the data directory as python pickles. To use these, simply run:
$ pip3 install networkx scipy numpy plyfile
$ ./simulate.py

If you would like to regenerate the graph or handholds respectively, run:
$ GRAPH=true ./simulate.py
$ HOLDS=true ./simulate.py

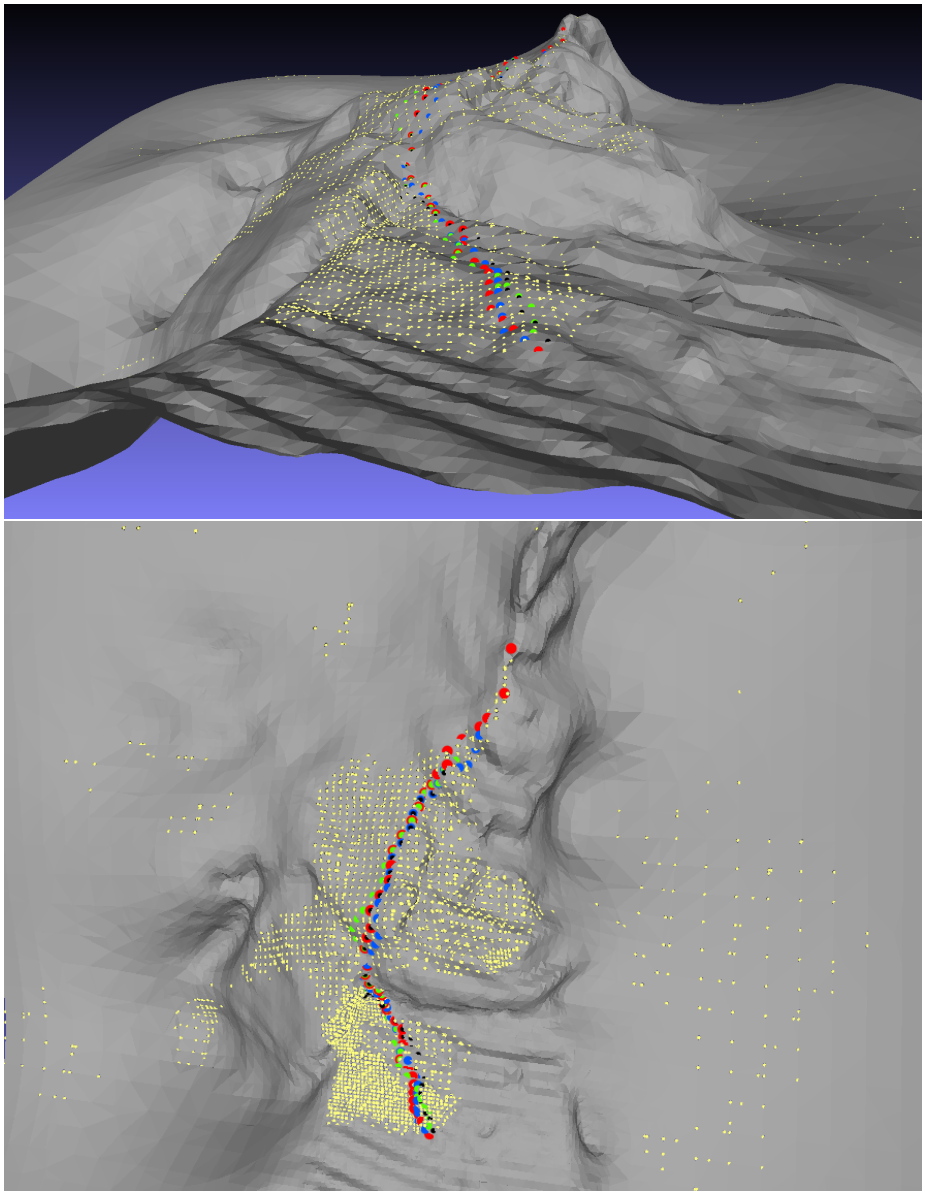

Figure 8: Computed paths overlaid on the generated mesh. Yellow vertices are available handholds. Paths for limbs zero through three are red, blue, green, and black vertices.

A decimated mesh has also been provided. If a quick regeneration is desired, add the debug flag like so:
$ TEST=true GRAPH=true ./simulate.py

Note that path planning will result in failure, as the selected start and end nodes do not exist in the decimated mesh.

Simulate will write files to the `data` directory. In this case, the files we will be interested in are in the XYZ file format:

The top 15th percentile of handholds: `handholds.xyz`

The path of each limb: `bot0-path.xyz, bot1-path.xyz, bot2-path.xyz, bot3-path.xyz`.

These can be loaded into MeshLab along with `mesh1.ply` for visualization identical to the ones presented in this paper.

References

[1] S Pfalzner, M B Davies, M Gounelle, A Johansen, C Münker, P Lacerda, S Portegies Zwart, L Testi, M Trieloff, and D Veras. The formation of the solar system. *Physica Scripta*, 90(6):068001, 2015.

[2] Williams. Asteroids. https://nssdc.gsfc.nasa.gov/planetary/planets/asteroidpage.html.

[3] Himangshu Kalita and Jekan Thangavelautham. Multirobot cliff climbing on low-gravity environments. *Proceedings of the 40th AAS Navigation and Control Conference, Breckenridge, CO*, 2017.

[4] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

[5] Jason Campbell, Rahul Sukthankar, and Illah Nourbakhsh. Techniques for evaluating optical flow for visual odometry in extreme terrain. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3704–3711. IEEE, 2004.

[6] Alan T Asbeck, Sangbae Kim, Mark R Cutkosky, William R Provancher, and Michele Lanzetta. Scaling hard vertical surfaces with compliant microspine arrays. *The International Journal of Robotics Research*, 25(12):1165–1179, 2006.