Individual Lab Report 9

By Clare Cui

Team B: Arcus

Clare Cui

Maitreya Naik

Angad Sidhu

Logan Wan

23 March 2017

Individual Progress

For the past two weeks, I worked primarily on the visualization stack. My goal was to understand the different packages and enable the occupancy grid map to accept RGB data. To be able to do this, I first worked on understanding the dense_voxel_map package and documented summaries of the functions implemented. The dense_voxel_map package essentially processes queued point clouds and creates an occupancy grid map based on the locations of the points. Since visualization must occur with either a sensor_msgs::PointCloud2 type or pcl::PointCloud type, voxels are determined to be occupied, free, or unknown from the original point clouds and are returned as these same types for visualization. However, the difference here is that the "point cloud" is in the form of evenly spaced points in the center of each voxel instead of scattered based on individual LiDAR scans. Once the point clouds are processed and the grid cell states have been determined, the grid is visualized.

I also took some time to understand the map_utils package, which made a lot of calls to the log_odds probability of each cell associated with a point cloud point. We wanted to convert what was originally a vector of float values into a vector of structs that contained both the log-odds float values and uint8_t RGB information. Since Logan and my pipelines have not yet been connected, I wrote a constructor that would automatically set the color of any new instance that was created. After creating this and trying to compile, I then discovered a lot of places where there were issues with the new struct in place.

A lot of the modifications I made were based on the error output, mainly making sure that any function that tried to access a log-odds was accessing it correctly in the struct. There was also some small issues converting from sensor_msgs::PointCloud to pcl::PointCloud, and the reason we went with the pcl library was because it was able to handle RGB and was more user-friendly than sensor_msgs::PointCloud2. Converting between the two was not a trivial task; the structure of pcl::PointCloud was slightly different, requiring me to include some new libraries to handle the sensor_msgs header and removing any references to "channels," which did not exist for the pcl::PointCloud type. Additionally, using pcl briefly introduced me to templating, where I learned pcl::PointCloud used a templating paramater, struct PointXYZRGB, that determined what type of point it held. The struct had an RGB member that needed to be of type uint32_t, so this also required conversion from uint8_t.

In the end, I ended up making changes to header files in three different packages: dense_voxel_map for initializing an RGB occupancy grid publisher node and visualizing it; map_utils for the new struct, point cloud type conversions, and log-odds accessibilility; and the information_utils package which also accessed log-odds probabilities of cells. The output of this visualization can be seen in Figure 1 below. All the cells are colorized randomly so as to show that the cells are capable of taking on any color. A video of the occupancy grid map populating the simulation environment can be found at: https://youtu.be/RK0FCifyR44.

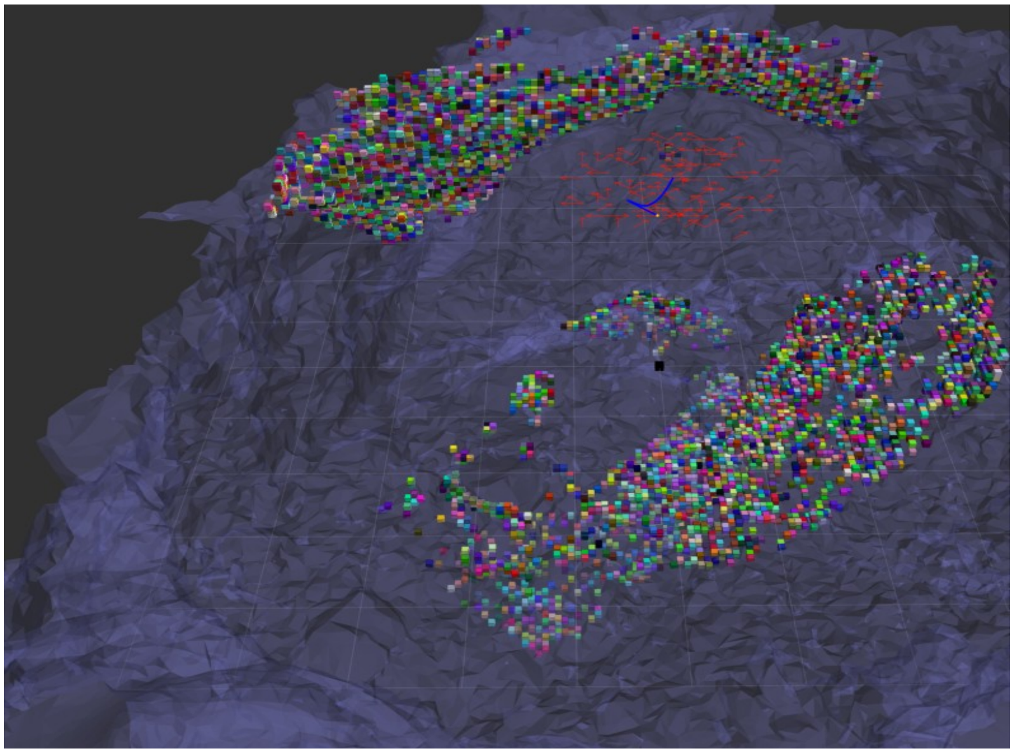

Figure 1: *Occupancy grid map with RGB-colored voxels.*

Challenges

The challenges this week were all related to learning and understanding the code. For example, originally, I did not understand the purpose of creating different functions for different point clouds, such the functions toPointCloud, toUnknownPointCloud, and toIncrementalPointCloud. I also was not sure how to handle resizing of a struct and got stuck trying to interpret the documentation when I looked it up. There were also issues with using new libraries, where I had to modify the CMakeLists and package.xml files. Luckily, I was able to get help from Angad and my PhD advisor, Wennie Tabib, in resolving a lot of these issues. In addition to their help with the exact problem, I tried to understand the issue as much as I could by asking them additional questions so that I know what to do and look for in the future if I ever encountered similar problems.

Teamwork

Logan Wan: Logan worked on bringing up the map_server so that it was capable of using the RGB camera in simulation to extract color information corresponding to points in the Velodyne point cloud.

Angad Sidhu: Angad worked on understanding the motion planning and trajectory generation.

Maitreya Naik: Maitreya tuned the flight controller and worked on understanding the motion planning and trajectory generation's architecture at a high level.

Future Plans

I plan on working with Logan to combine our two pipelines and provide a complete visualization of the simulated cave in correctly colorized voxel grid map form. Afterwards, I will be working towards synchronizing the sensor output – I paused on this because I was reassigned to work on map_utils. Eventually I hope to work with Logan on visualization of a colorized dense voxel map with previously bagged data from Lafarge.