Angad Sidhu

Team B Arcus

Logan Wan, Maitreya Naik, Clare Cui

ILR 10

April 5 2017

Individual Progress

For this progress review I spent my time trying to discover the exact issue with motion manager. I was finally able to figure out the exact conditions in which the motion manager bug occurred. I also devised a simple fix that, while may not completely eliminate the bug, should at least mitigate the chances of it happening. Also, in testing, the bug did not occur in simulation after 20 minutes of testing when it would usually happen in less than a minute. I'll describe the exact issue later on.

I also started looking into adding in virtual obstacles into occupancy grid map. After examining how the different sensor simulators work in the simulation, I came up with a rough design of how it would be implemented. We first start with loading in meshes of obstacles. We then want to convert these into a point cloud that can be continuously added to the grid. This point cloud is computed by finding intersections with the mesh along a voxel grid with the same resolution of the voxel grid used in our map server. These intersections then become points in our point cloud. We then transform these point clouds to the world frame and add them to one large point cloud. Finally, every update, we add these point clouds to the map server which adds it to the occupancy grid.

Challenges

I faced a lot of challenges when debugging this motion manager bug. It was so difficult because ROS nodes are all different processes and the bug was a result of improperly designed interprocess communication. Unfortunately it was impossible to use gdb (afaik) as there many processes operating in tandem. I found the best way to debug was printing out state of everything in the various processes. I was able to grasp patterns in state that led to bugs, i.e. transition to hover mode eventually led to the bug occurring. But, without transitioning to hover mode, the bug did not seem to happen. Figure 1 demonstrates the exact flow of logic which causes this bug and why the fix we made mitigates the issue.

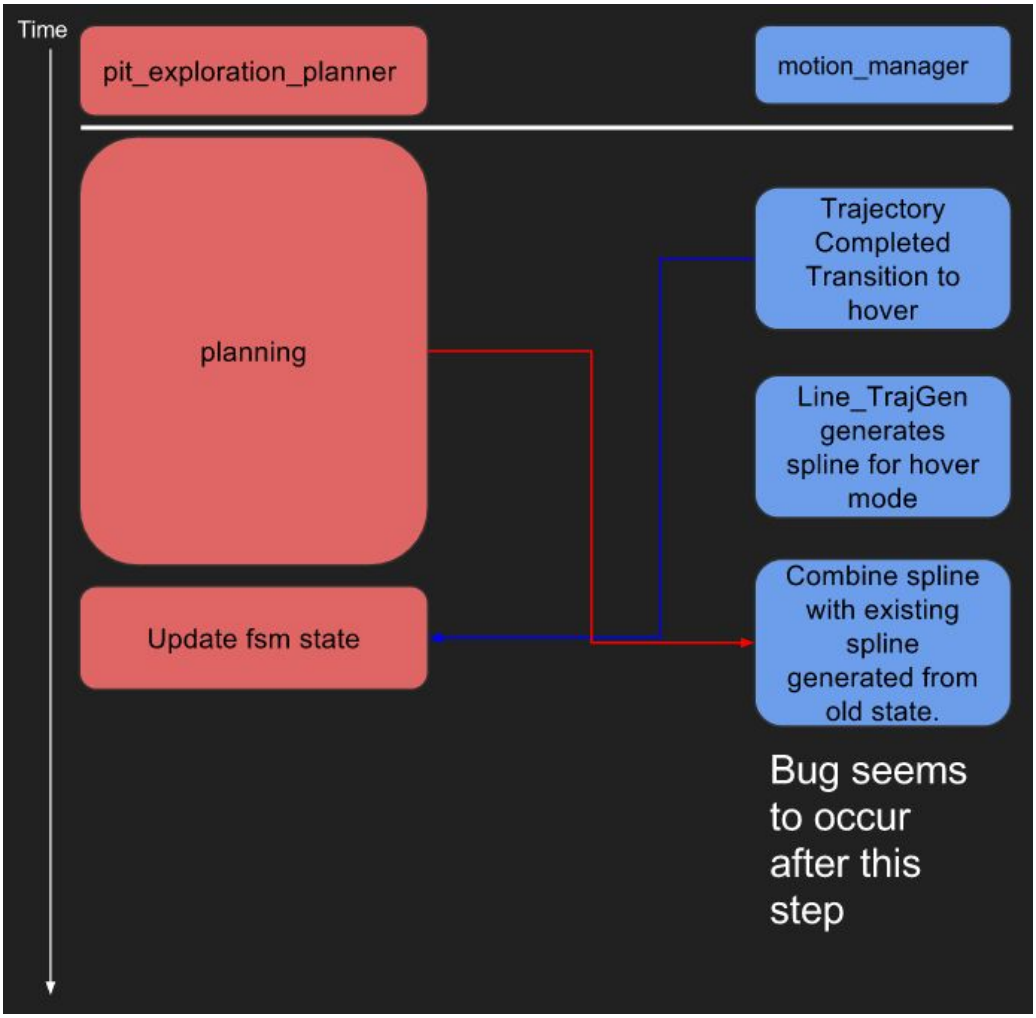

Figure 1. Motion Planning bug event trace

The issue was the pit exploration planner which generates trajectories operates for a long time (>.5 seconds). This process will attempt to send a plan to the motion manager about 1.5 seconds before it finishes. This can be very variable and can take on the order of a second. The problem is that the plan depends on the state of the fsm. So if the fsm changes its state to hover, and sends an update to the planner, it will not get it and be able to replan until after it finishes its current planning cycle. It will also send the spline it generates based on the old state to the fsm. This can corrupt the spline that was added by the line trajectory generator (which takes over when the fsm transitions to hover). I did not want to spend more time tracking down the exact method, however, of how this bug occurs so I stopped tracing at this point.

Teamwork

Logan spent time digging into the observation synchronizer and figuring out how it worked. He then jumped over to helping diagnose bugs with Clare.

Clare worked on integrating the pipeline that Logan created into the pipeline she created last PR. Towards the very end of development there were a couple of hard to diagnose bugs that I tried to help solve.
Maitreya was working with me to uncover exactly how the motion manager bug occurred.

Plans

I will hopefully be able to implement the virtual obstacle functionality in simulation based on the design I laid out above. I will also be working with Moses on getting a mini quadcopter up and running using the flight stack that the RASL lab uses. This involves updating the flight stack code in the simulation to the latest versions and compiling on the odroid.