Personal Progress

The past two weeks I spend most of my time working on a Gazebo simulation that is meant to (eventually) demonstrate and test the full system on the bench. I also did work to debug the obstacle avoidance code and worked on flight testing.

The Gazebo Simulation so far has two major parts to it: the Velodyne VLP-16 LIDAR model and the DJI M100 flying around. I started out with some old sample code that other people had created to model the LIDAR and DJI separately, which I needed to modify and combine in order to use for our purposes. The original code repositories are listed below for reference:

- https://github.com/caochao39/hku_m100_gazebo - This is the original Gazebo model for the DJI M100 drone, as well as some base code for getting the model to move around when subscribed to the correct topics from the DJI PC Simulator.
- https://bitbucket.org/DataspeedInc/velodyne_simulator/src/01bfb68ef647/ - This is the original code for creating the VLP-16 LIDAR model and getting data from.

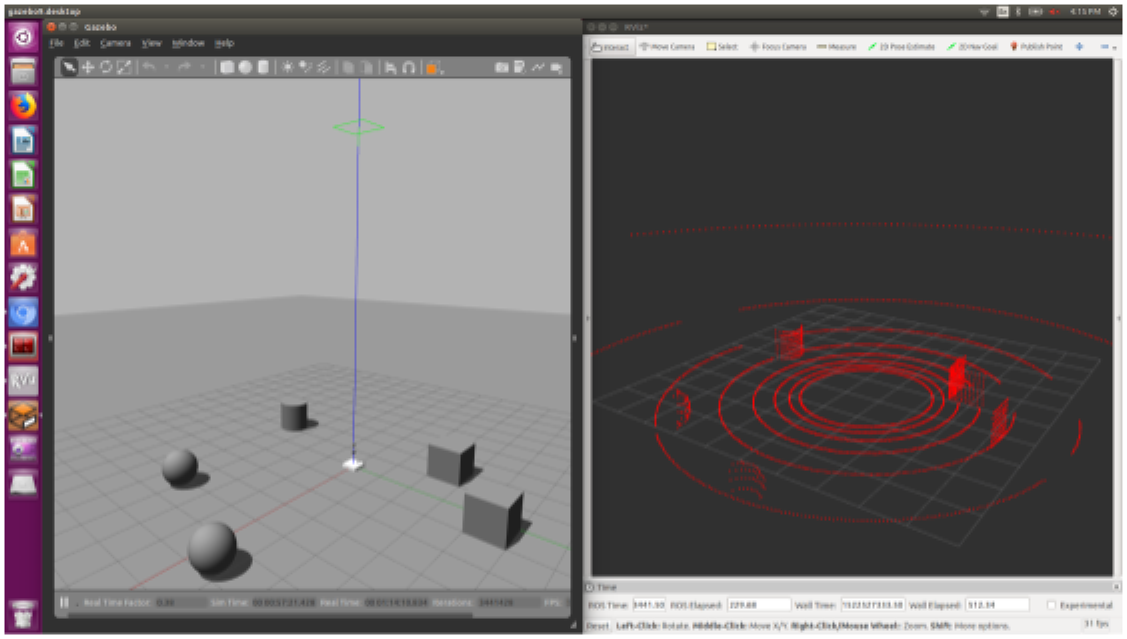

Figure 1: VLP-16 with surrounding obstacles in Gazebo (left) with the resulting point cloud (right)

One major challenge was getting all the code to build in the first place. You may recall that in my last IRL, I listed getting the code to work was one of the challenges I face, and in the absence of knowing what to do, and with limited experience in Gazebo, I decided to proceed with building my own LIDAR model to both make some forward progress as well as learn more about how the environment worked. This time around, I decided to take another stab at working with the base code, since making modifications to something that already somewhat works was in the long term going to be easier. I discovered that with respect to the LIDAR code, one issue was with the Gazebo math packages. Once I installed the correct packages and made some small modifications to some of the data structures in the code, I was able to successfully get a point cloud to display in rViz when I placed custom obstacles at various locations in the Gazebo environment. A picture of that test is shown in Figure 1.

With the LIDAR data coming through, I proceeded to get the DJI code working, Again I encountered challenges getting the code to build. This I discovered was a case of the code being old. The DJI SDK was since updated to include normal geometry_msgs data types instead of DJI specific data types. With those modifications in place, as well as adjusting the topic names for the correct data from a bag file instead of from the DJI simulator, I was able to get the code to build and run properly. I tested this by running a bag file from a previous flight on Flagstaff Hill and observing the resulting motion of the quadcopter in the Gazebo simulator.

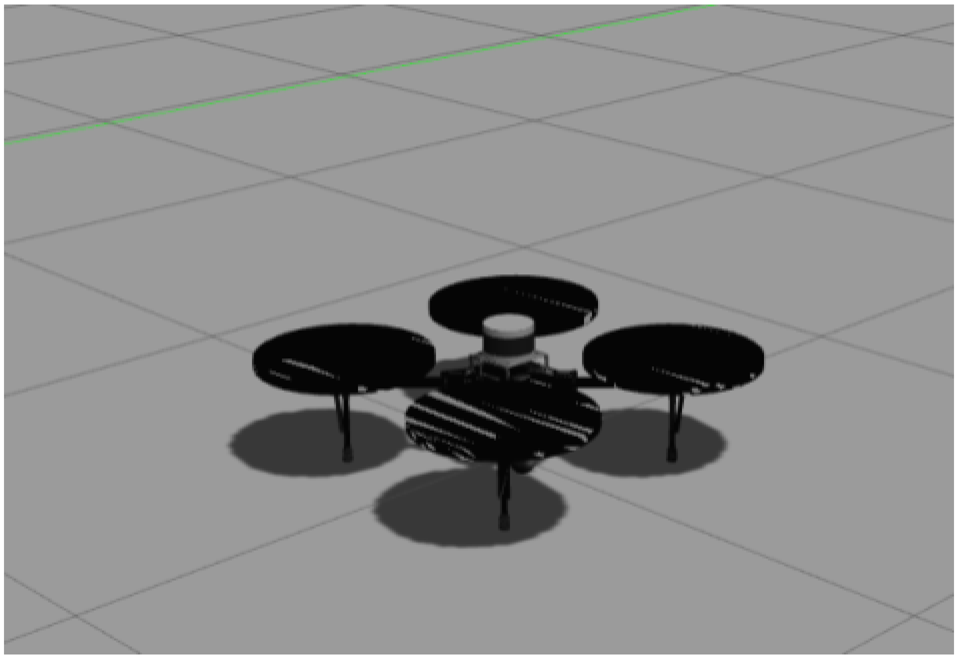

Figure 2: Combined URDF model of the DJI M100 and VLP-16

Next up was integrating the two. Here I combined the urdf files and consolidated the correct commands into a single launch file in order to get the LIDAR data coming through at the same time as the quad flying around. I again tested this with a bag file from a flight in Schenley Park. This worked to an extent-I was clearly able to see the point cloud updating, but the data in rViz would flash intermittently. I discovered this was because the rViz time kept switching back and forth in between the ROS time from the recorded bag file and the clock time from the Gazebo simulator. This caused problems because the TF buffer was constantly cleared every time this happened, resulting in the point cloud data disappearing in rViz since the pose transformation was no longer available.

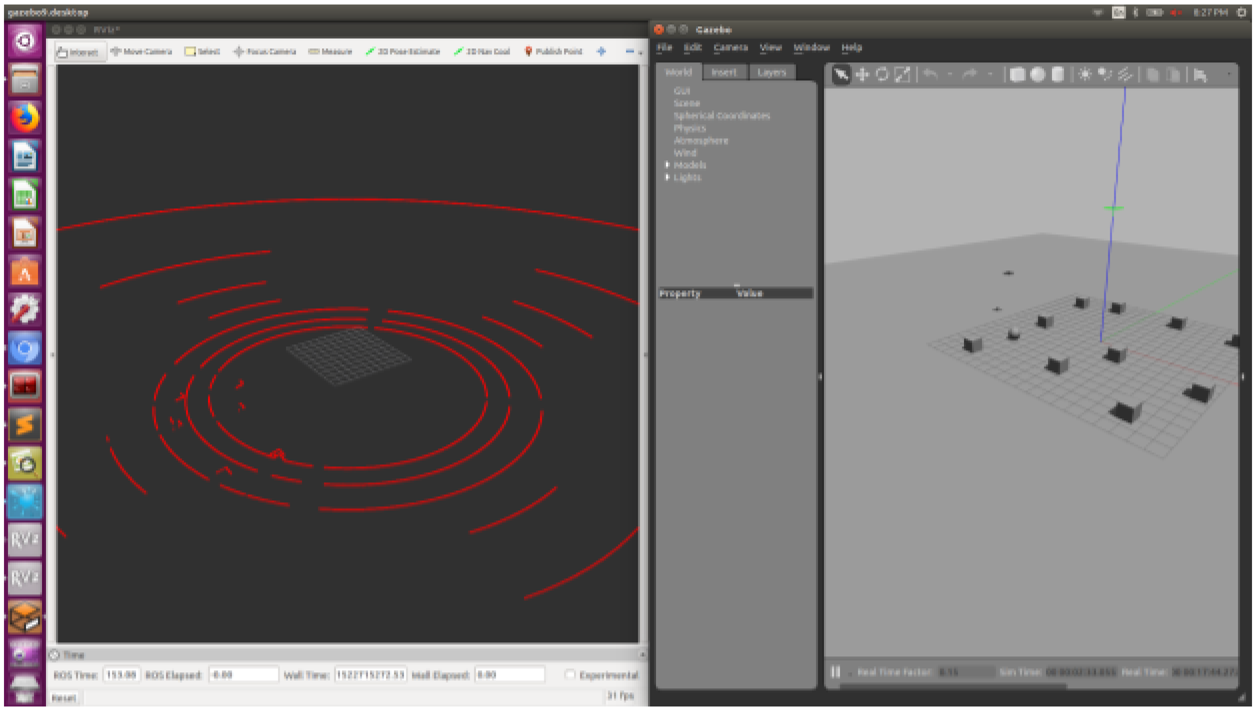
Figure 3: Quad flying according to bag file data (right) and the corresponding point cloud output (left)

Beyond this, I also contributed to the debugging process for implementing the obstacle avoidance code in the DJI simulator. We were having some initial problems getting it to work, I helped Shivnag and Joao look over the code and test for a solution. One of the main problems turned out to be a reference frame problem, though we still have work to do in order to make the avoidance/emergency brake functionality smoother and more intuitive.

Figure 4: Testing the stop functionality in the DJI simulator

We also had a major flight in which we tested the integrated system on Thursday the 5th (right after the PR, when the weather finally cooperated). Shivang, Hari and I went out to Flagstaff Hill and flew the quad with the Epson to test the integration of the Bird's Eye View and FPV in a live environment.

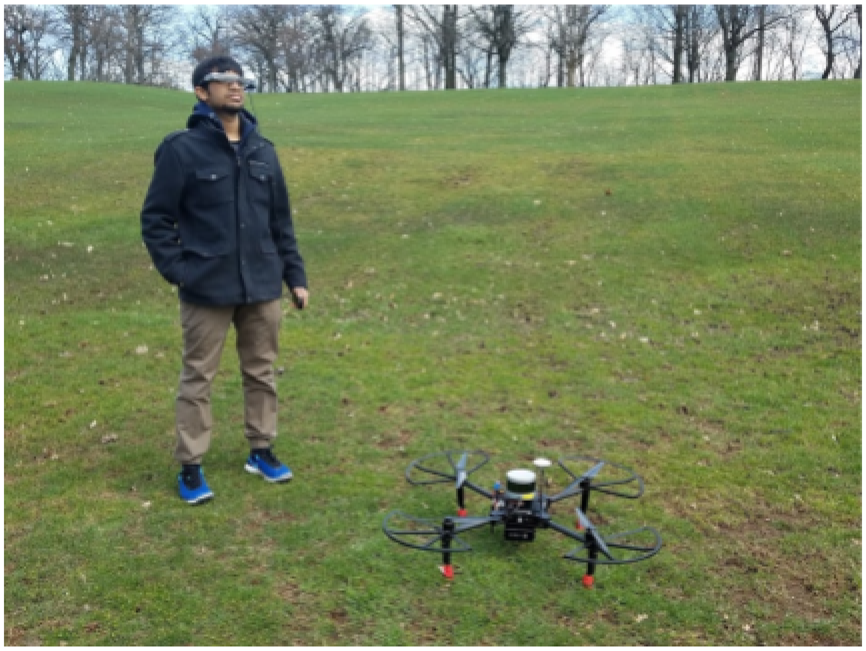

Figure 5: Testing the Headset and Quad live

Challenges Faced
One challenge we faced as a team that directly impacted my work was the weather throwing a wrench into the testing schedule. We have had a lot of rain recently (and even snow), so it's been hard to schedule tests.

For me personally, I had a lot of trouble getting the base Gazebo simulator working, and had to fight through a lot of small issues while learning how Gazebo works. I finally did get it working, and I'm planning on making my code (and documentation) publicly available so other MRSD teams in the future can benefit from my experience.

Teamwork

Teamwork
Joao: Joao worked on integrating the sounds warnings with Hari and worked on integrating the obstacle avoidance with Shivang.

Shivang: Shivang worked implementing Joao's Matlab code into C++ code for the obstacle avoidance, as well as cleaning up the visuals of the coloring and Bird's Eye View.

Hari: Hari worked on integrating the sound warning code into our stack, as well as improvements to the Bird's Eye View.

Nihar: Nihar has been working on refining the user interface and improving the FPV camera view based on the input we received from David Murphy at the last pilot workshop (last week).

Future Plans

The team is scheduled to do a dry run/integration test with NEA at Nardo on Friday (weather permitting, next week as a backup). We will continue to make modifications to the user interface as we refine the integration and do more extensive testing.

My primary work is centered around getting the simulation ready. I have to finish integrating with the DJI simulator (we didn't end up having time to test this before the last PR), as well as integrate a FPV camera onto the quad. We will then be able to test our code on the simulated environment. This is what we are hoping to show for the SVE dry run at the next PR as well as our demo for National Robotics Week.