MRSD Project 2

Individual Lab Report #07

Parv Parkhiya

September 26th, 2019

Team H:
PhoeniX
Teammates:
Shubham Garg
Akshit Gandhi
Zhihao (Steve) Zhu

Individual Progress

I worked mainly on getting the autonomy up and running on our AGV platform Husky. While Clearpath (maker of Husky Robot) provides ROS interface with various standard ROS packages from localization to planning, there are many pieces that needs to be figured out and set properly for everything to work together.

I started by looking through the Husky ROS code base trying to figure out how different ROS packages links together to achieve the autonomy. My initial plan was to generate 2D cost map using stereo camera and then provide the cost map to the planner. However, after looking through the code, I realized that the costmap is generated by a costmap_2d ROS package which publishes map in a very complicated way and has tight integration with the move_base planner package. Therefore, writing out custom costmap_2d node that is compatible with move_base planner from scratch was out of the scope. So, we decided to use costmap_2d package but that came with it's own challenge.

While costmap_2d in principle can take pointcloud as input, it is setup to use 2D laser scan. Because of weight and power constraints, we are no longer using Sick LIDAR and have moved to using Intel Realsense depth cameras. (RS-D435) So, I had to convert the pointcloud data into laser scan data. There was a package available to achieve the same. I set up various parameters through the launch file for that pointcloud to laser scan package. Certain topics need to be remapped and some parameters fine tuning was also required to get it to work.

However, for some reason, the code to convert pointcloud to laserscan was extremely slow. A single conversion was taking a couple of seconds which is simply unacceptable. Therefore, I decided to write the node to convert the pointcloud into laser scan from scratch instead of using existing ROS package. Logic for converting pointcloud into 2D laser scan is pretty straight forward. However, the challenge was to do it very efficiently. I wrote the package that can create the laser scan from pointcloud with O(n) complexity where n is the number of 3D points in the pointcloud.

Next step was to setup the gmapping package that can use output of costmap_2d to localize the robot in that map and move_base package which can use the output of gmaping and costmap_2d to publish control signal. Once everything was linked

properly, I tested the setup in the gazebo simulation was able to move husky without collision.

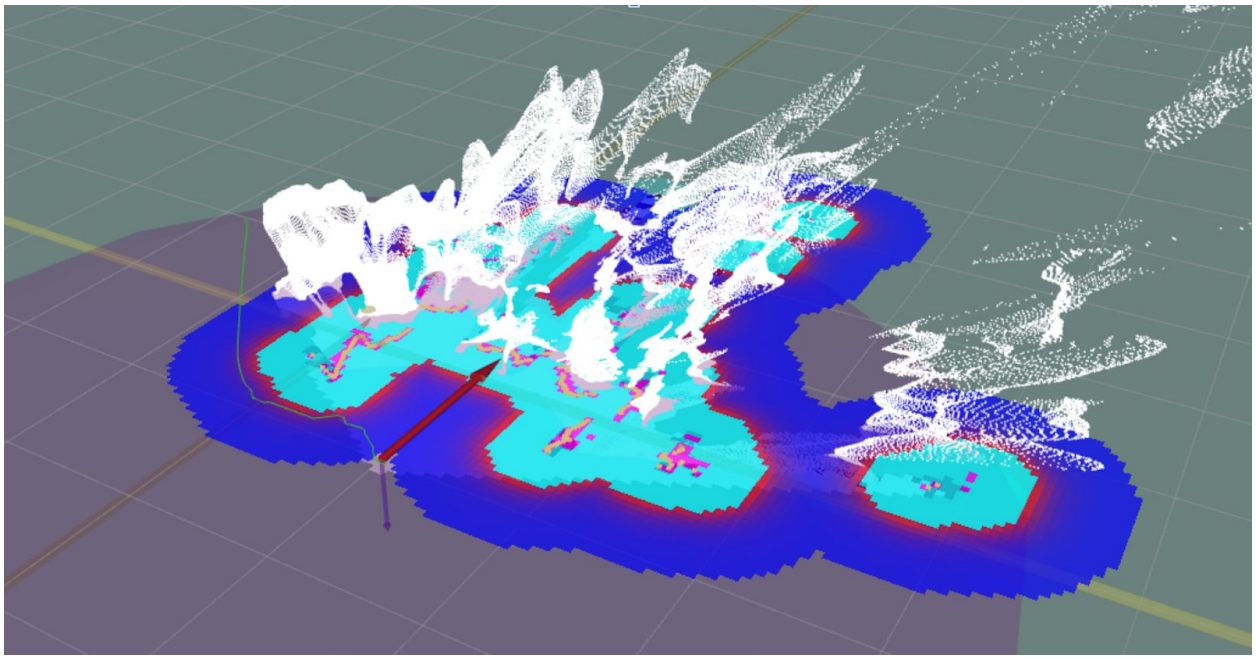

Figure 1: Cost map and pointcloud on real husky

Once everything was working in simulation, we decided to test our code on real husky. After some light debugging we were able to get everything working on real husky.

Challenges

To move the husky to a desired goal location without colliding is a non-trivial task and requires many things to work properly. Clearly creating everything from scratch from local planner to controller is out of the scope. I still had to create a couple of packages in the large pipeline. The major challenge was of course pipelining data from various nodes in an appropriate way. Base frame of the messages and setting up tf tree are involved tasks and required some intensive debugging to get it write.

Code working in simulation of course didn't work directly on real husky. Some parameter tuning for pointcloud was needed but even after that we were not able to identify the problem. Finally, I realized that rosbag record was somehow crashing the realsense node and without pointcloud data, everything was going haywire. Running 3 realsense camera together might be way too much processing for the onboard

computer. Once I stopped the rosbag record, husky was able to avoid obstacles and find paths through the opening.

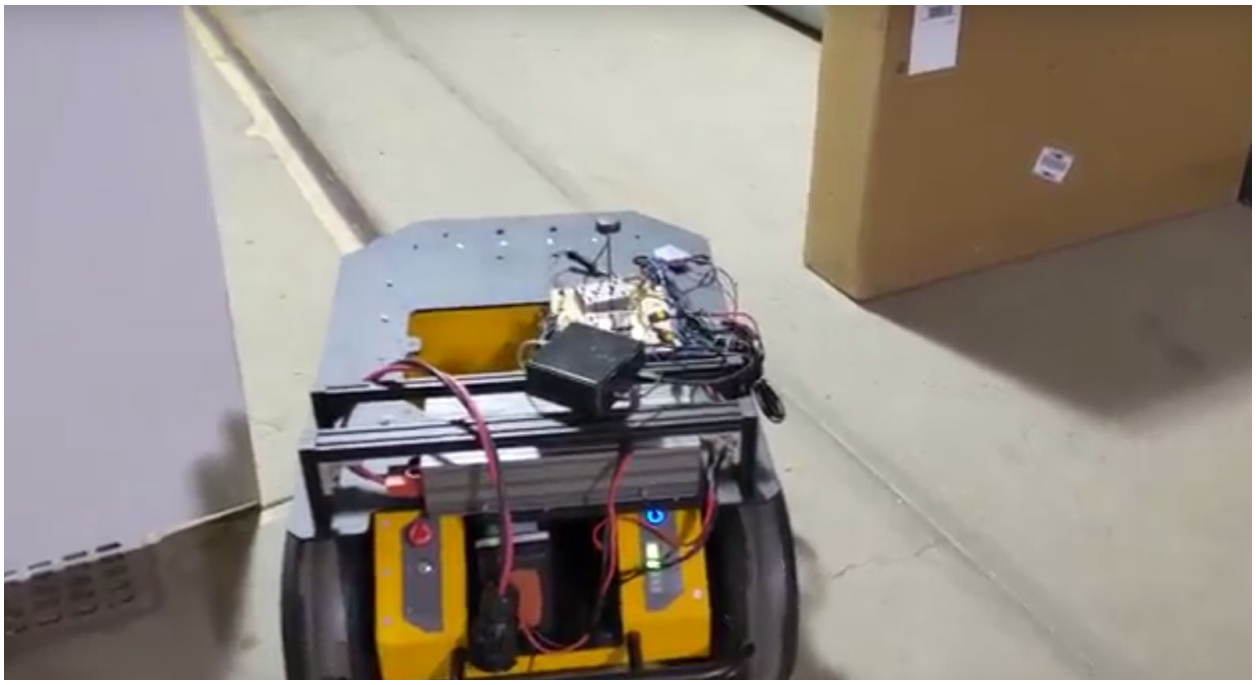

Figure 2: Husky autonomously entering through the opening

Teamwork

Shubham was working on setting up transform for all three realsense cameras and getting combined pointcloud. However, he was facing difficulties in finding bugs in his code. We worked together to fix various minor issues in transforming point cloud so that everything can be with respect to the base_link frame of the husky. I was able to correct mistakes in the code and get the combined pointcloud. Akshit also helped in using the message filters to combined three interrupts into single callback function.

Steve was working on door detection using depth images. Initially, we thought that window detection code would also work for door detection but for door we can only find three edges. Edge of the floor is invisible in the depth image. Since I wrote the original opening detection code for the window, I helped Steve in understanding that code and takeaways from my experience.

Akshit helped me in setting up the environment for testing the code on husky. He also helped in recording the video of the attempts and debugging minor issues that we faced during the tests.

Future Plans

Currently the code that combines the three pointcloud to one is very slow and creates a bottleneck for the pipeline (2 Hz). I will restructure the code to make it more efficient. Other task is to fuse the IMU data and Intel tracking camera data into robot_localization package to improve the localization. Currently husky uses just the wheel encoder data for odometry. Improving robot_localization should result in smoother planning execution. I would handing the fusing part. I will also help Akshit at the drone aspect of the project in conducting flight test. I would also contribute in software part of establishing communication across systems.

As a team, we would like to improve our performance of local planning on husky and make drone enter through the opening. We will also work on getting common database and communication channel for collaboration across systems. Extended goal would be to get improvement in the water deploying mechanism.