MRSD Project 2

Individual Lab Report #08

Parv Parkhiya

October 9th, 2019

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

Individual Progress

The first thing I worked on was improving the performance of our ROS package of combining 3 pointclouds. Since the pointclouds from all the Intel Realsense cameras comes at different times in an asynchronous fashion, we were using a ROS message filter to create a callback after all three messages are received. We were getting this output at merely 2 Hz which is not ideal for cost-map generation in the move_base planner. If process just one pointcloud even if all three cameras are publishing, the output frequency was 10 Hz. Our intuition was that the message filter that was synchronizing the incoming messages was stalling the system causing the performance decrease.

I restructured the code so that it can work with asynchronous messages. This was achieved by linking ROS subscriber callback functions to a C++ class's private member functions that store the incoming pointclouds in the private variable overwriting the previous value. The output publisher's loop runs separately which takes current values in the private variables and combines them without waiting for the incoming messages.

However, when I tested my new code, it made a very minor improvement in terms of performance. Further investigation revealed that our original intuition that stalling was the bottleneck of our pipeline is incorrect. Intel NUC simply doesn't have enough processing power to deal with 3 pointclouds at a high rate. The system can run 3 cameras in parallel and provide 10 Hz performance as long as not all three cameras are being subscribed to. ROS by default uses a lazy publishing method, where if no one is subscribing to a topic, the messages of that topic are not broadcasted in the channel to save processing. We didn't know about this earlier which lead us to check the publishing rate of pointcloud one by one instead of simultaneously.

I also worked on the improvement of the opening detection algorithm. When we tested our code outdoor, opening detection was poor. The main reason was the Realsense depth image in the outdoor environment is not as good as the indoor environment because the infrared projector doesn't work well outdoors. We were treating each depth image independently and if there are wrong detections in some of them, it could confuse the controller. We need to remove the outliers in the detection. I implemented the time filtering for window detection that removes these outliers. I restructured the code so that it can store the previous values and then applied the median filter in a computationally efficient way. Output of which can be seen in figure 1.

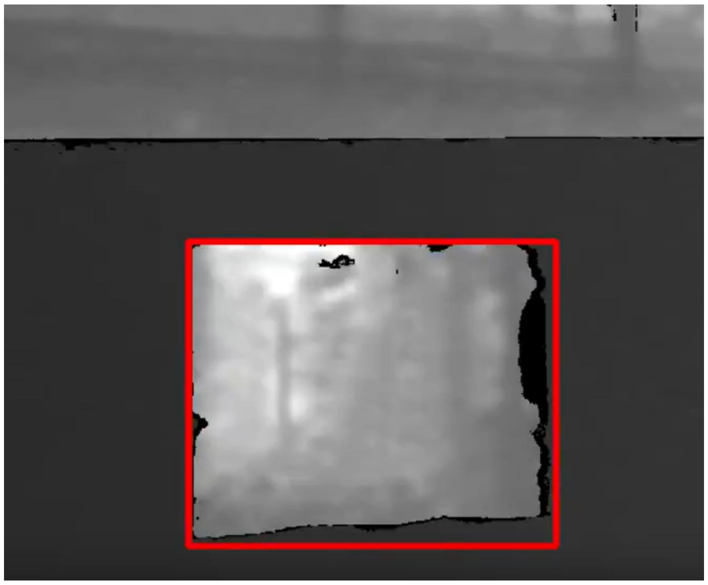
Figure 1: Opening Detection after median filtering across time

We started testing the UAV's ability to visual servo directly in front of the opening, we realized that drone is not following our velocity control signals. We checked the header, topic name, frame id, timestamp but nothing helped. I wrote a simple python ROS node to publish some constant velocity control signal for the debugging purpose. The drone started to follow those constant commands but not commands from our visual servo module.

I went through the Airlab's core stack framework that was providing the interface with DJI SDK. Velocity commands that we send in the XYZ frame are transformed into Roll Pitch Yaw thrust and the "twist" message was converted into the "joy" message that DJI can interpret. After looking through many potential issues, we decided to increase the publishing frequency from 20 Hz to 50 Hz which allowed the control of the drone using the visual servo output.

Challenges

The biggest challenge of this progress review was the task of entering through the opening. We faced many challenges. The first challenge was getting the DJI drone to respond to our velocity commands. I had to go through a lot of code and documentation to understand what was happening and what could be potential problems. Debugging was a long process and required multiple flight tests. For the flight test, we had to collaborate with Lucas who was confident in flying the DJI drone indoors.

Once the drone started responding to our control signals, we tested our visual servoing code. It somewhat worked but not always and not robustly. Entering through the opening is a very difficult skill for a drone and we can't risk testing the same if visual servoing is not perfect. Even with the help of tags, visual servoing method was struggling to get right in front of the opening. We spend multiple days trying to improve the visual servoing and find the potential issues but we simply ran out of time and decided to try a different approach in the next progress review.

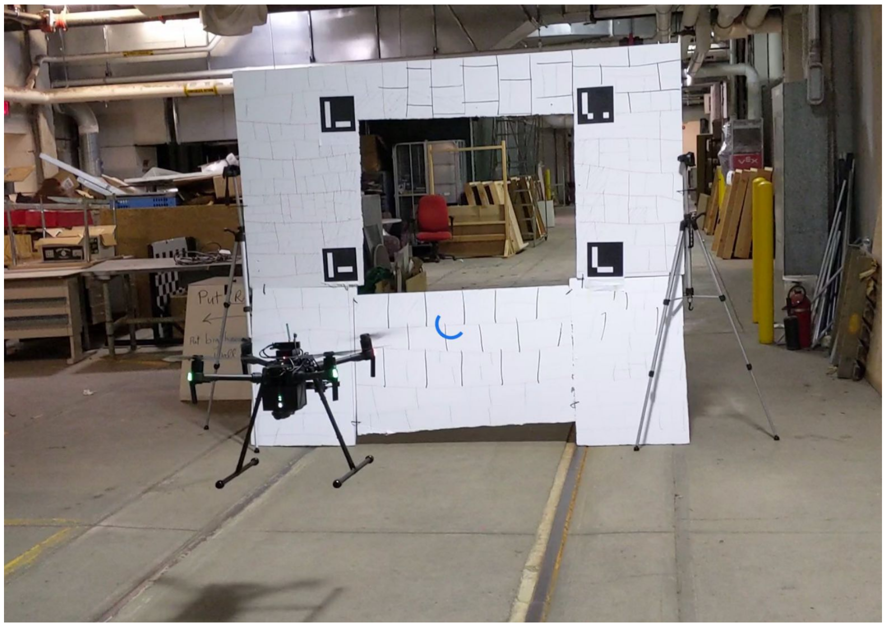
Figure 2: Visual Servoing in front of the opening

Teamwork

Akshit and I worked with doing the DJI drone flight test to align the drone in front of the opening. (As seen in figure 2) The drone test always requires a lot of pre and post work related to setting up the environment (opening) and setting up the drone. Shubham mainly worked on state estimation integration of IMU and tracking camera. I helped him in how to set up and run the "gmapping" module and move_base planner. Steve was working with door detection for husky. We brainstormed multiple ideas on how to make it work on depth images. Finally decided to move towards using pointcloud instead of depth images. I also collaborated with all the MBZIRC team members in creating the submission video and submission report which was an urgent requirement.

Future Plans

I will be working on figuring out what would be our improved deploying mechanism, especially for UAV. I will lead the effort in the procurement of the hardware and creating mounts for attaching it with UAV and AGV. I will work with Shubham in cleaning up the Husky's software so that we don't have to run 10 things separately. I will work on writing a small module that can provide the surface normal of the wall from the pointcloud. Akshit and I will explore the pose controller along with a simple mission-based approach (instead of visual servoing) of the Airlab's core stack to opening achieving entering through the opening for UAV. I will also help Shubham perform the full Husky missions. As a team, we faced a bit of a setback in terms of visual servoing not working as expected. We plan to execute an alternative approach to overcome this hurdle in the upcoming weeks.