Individual Lab Report

Erik Sjoberg

Team C – Column Robotics
Rohan Thakker, Job Bedford, Cole Gulino

IRL 7

February 10, 2016

Individual Progress

Configured RGBD-based SLAM system on Iris+ SBC

This week, the majority of my work consisted of configuring our onboard computer and sensor to perform real-time SLAM for global position estimation and obstacle detection. The purpose of this task is to enable the more advanced “stretch” goals of our project, namely obstacle avoidance and unstructured search.

Because our single-board computer uses an Arm processor (se Figure 1) instead of an Intel or AMD x86 processor, installing programs can be significantly more difficult than a single ‘apt-get install’. As expected, the RTAB-Map SLAM system did not function properly when installed following the usual instructions. It appears that the creator of the system has not yet attempted to install it on Arm, and as a result documented installation instructions were not available.

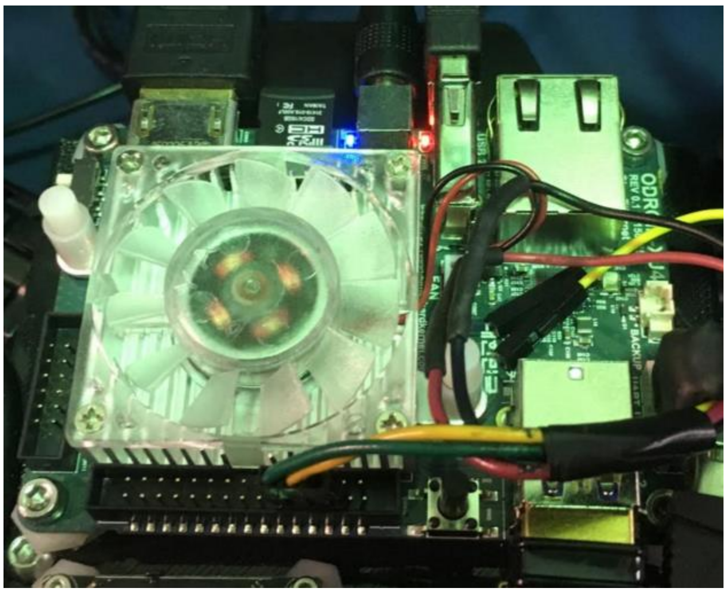

Figure 1: Odroid XU4 SBC mounted on the Iris+

Following advice of a user online, I attempted to install the package and several of its dependencies from source, including the Boost libraries and Point Cloud Library (PCL), including several modifications which would supposedly improve performance. Unfortunately, after all this work with manual dependency installation, RTAB-Map was still unable to run.

I eventually had success after using apt-get to install standard dependencies, then building RTAB-Map from source on the SBC. Apparently, the advice regarding required changes to the PCL source code was inaccurate for my system. Figure 2 below shows the working SLAM system mounted and running completely on our Iris+ quadcopter.

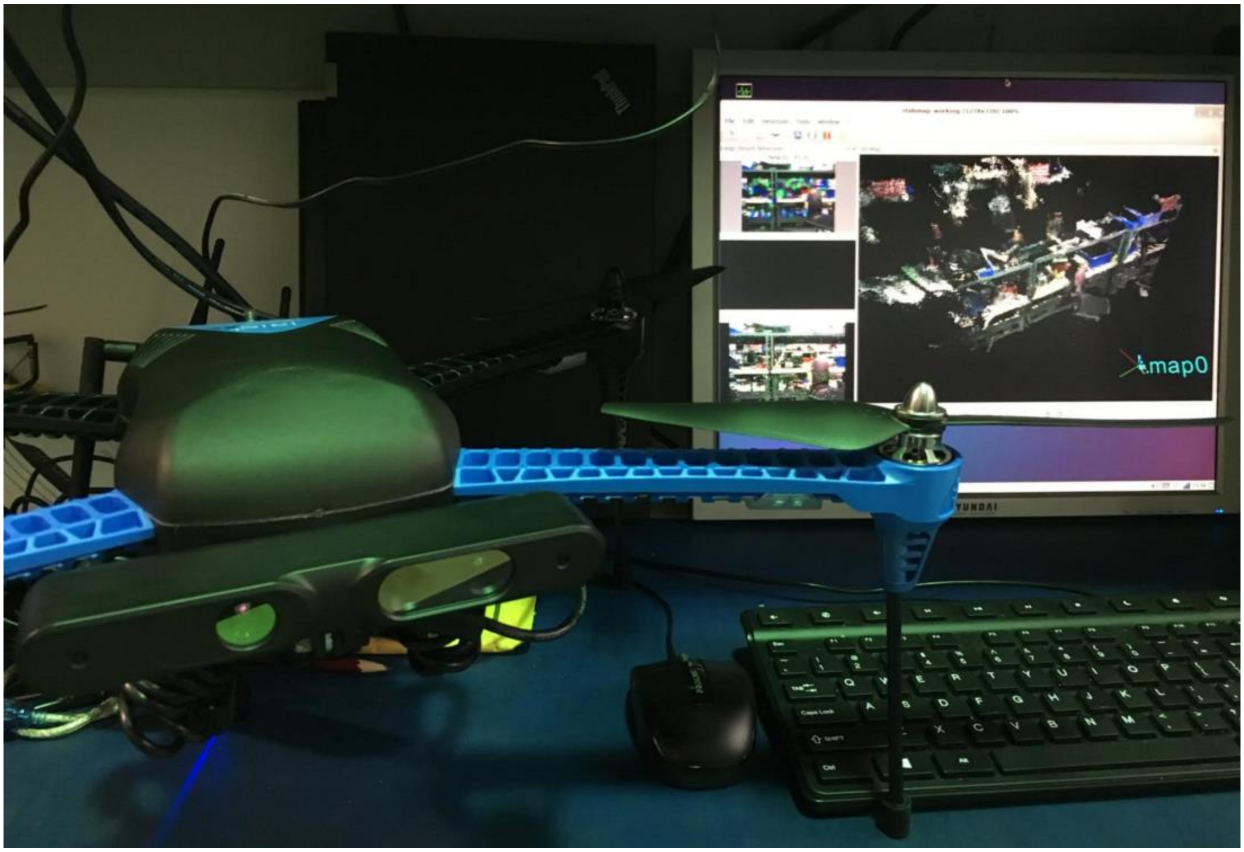

Figure 2: Iris+ Quadcopter and RGBD sensor with onboard SLAM system running

The map in the background of the quadcopter in Figure 2 exceeds the field of view of the sensor, and consists from several stitched sections of depth and color data. With default settings the RTAB-Map system is able to perform 6-DOF odometry as well as loop closure at a rate of approximately 1-3 HZ. Further tuning (including shifting from VGA to QVGA and reducing the rate of loop-closure detection) should enable odometry at 2-4 times this speed, meeting our requirements for mapping during conservative flight of the quadcopter.

Challenges

Slow build times on the Odroid XU4

Our Odroid XU4 SBC only contains an SD card as a filesystem, and as a result reading and writing many small files takes a very long time. The limited filesystem combined with the weaker processor meant that building PCL alone took several hours for each attempt. One takeaway is that having a fast filesystem, in addition to the processor, is critical of you intend to build large software dependency chains.

Incorrect and missing documentation
When building software products on Arm processors, it is often necessary to debug compiler problems yourself. This was definitely the case for getting RTAB-Map working on the Odroid, and troubleshooting incompatibilities took up the majority of my time during this week. Although I was able to find tips and advice online, in the end this advice ended up causing more work and taking more time than would have been required if I had debugged the system myself without attempting to save time by following their recommendations.

Teamwork

Once again splitting up the work into three teams was successful. Job was able to continue to reduce our programmatic risk by integrating the capabilities of our AR.Drone backup platform, while Cole and Rohan continued their work on the critical flight integration. Meanwhile I was able to set up the capabilities which will enable obstacle avoidance and more advanced search.

The focus on delivering demos after each two-week sprints has continued to be successful.

Plans for Upcoming Work

Validating Autonomous Landing
The next chunk of work I will be tackling is the most critical piece – getting the autonomous landing working within the requirements of our dock. We will be testing the built-in capabilities of the drone platform to land under its current position, as well as integrating the tag-based localization in order to achieve precision landings.