Individual Lab Report

Erik Sjoberg

Team C – Column Robotics

Rohan Thakker, Job Bedford, Cole Gulino

IRL 8

February 25, 2016

Individual Progress

Collected and analyzed flight data

In order to enable the development of closed-loop flight control, I flew test flights with our Iris+ drone and collected log data to determine the performance of the system and precise axes used on in the MAVROS flight controller interface. Because the native pixhawk hardware uses different axis conventions than ROS, there is a variety of coordinate transforms which occur automatically whenever our ROS middleware communicates with the Iris+. Due to poor documentation, I needed to actually collect data in order to figure out how exactly they had configured the various coordinate frames. Figure 1 below shows the output of one flight test where the quadcopter was flown in a square pattern.

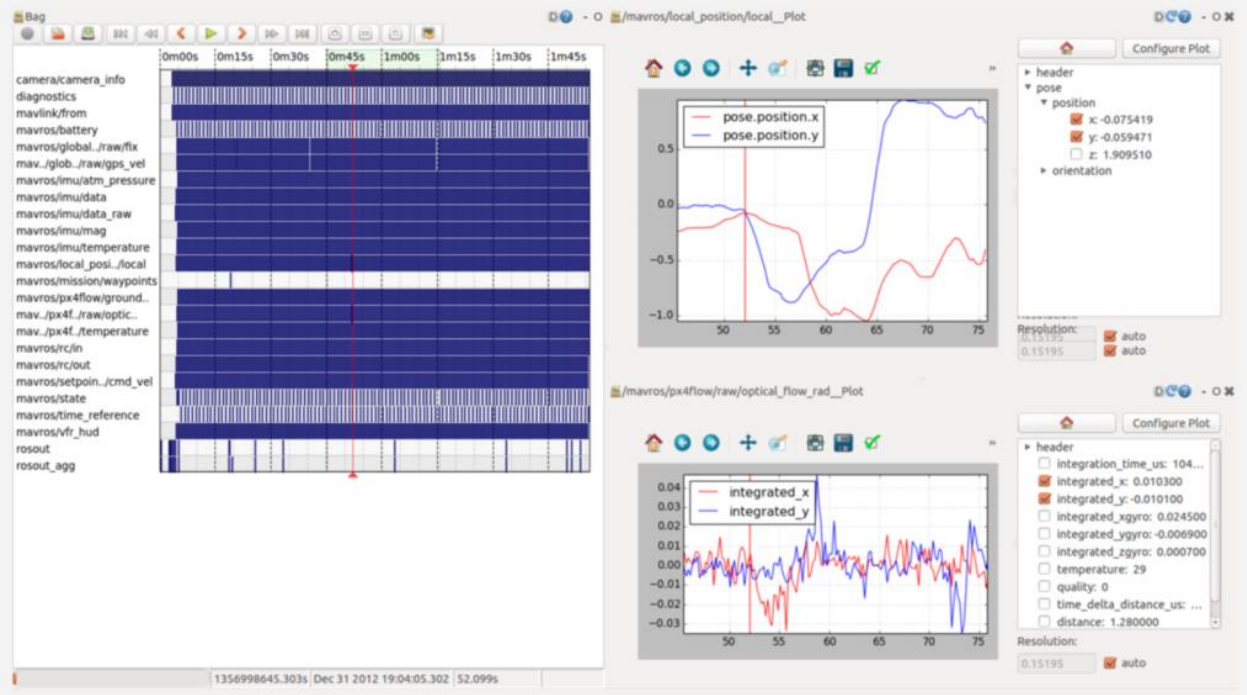

Figure 1: ROS bag log analysis to evaluate pose estimation

Wrote, tested and evaluated closed-loop april-tag hovering code

I proceeded to use the knowledge of our coordinate frames to code a simple implementation of closed-loop hovering based on time-averaged values of april-tag locations which are detected using our down-facing playstation eye camera. The basic concept is to rely primarily on the onboard station-holding and state estimation capabilities of the Iris+ firmware (including the PX4 Flow optical-flow camera) while using a globally accurate filtered measurement of the tag to correct drift between the quadcopter's local odometry frame and the actual global world coordinates. Unfortunately, due to the 20-minute code-compile-test cycle time we were facing, a variety of configuration problems kept us from successfully achieving closed-loop control.

Improved development environment

I spent around a day working to improve the efficacy of our development process, which was

quite slow due to actual coding taking place on the quadcopter's single-board computers over SSH, on a network which did not have access to the internet. I converted a spare single board computer into a wireless network bridge, providing a connection between our private wireless network and the CMU wireless network. Internet connectivity on our drone's network meant we were able to utilize source control and push our changes up to our repository on github. Testing of code on the benchtop with the quadroter blades removed also proved to be an effective strategy, as the testing in progress below in Figure 2 shows.

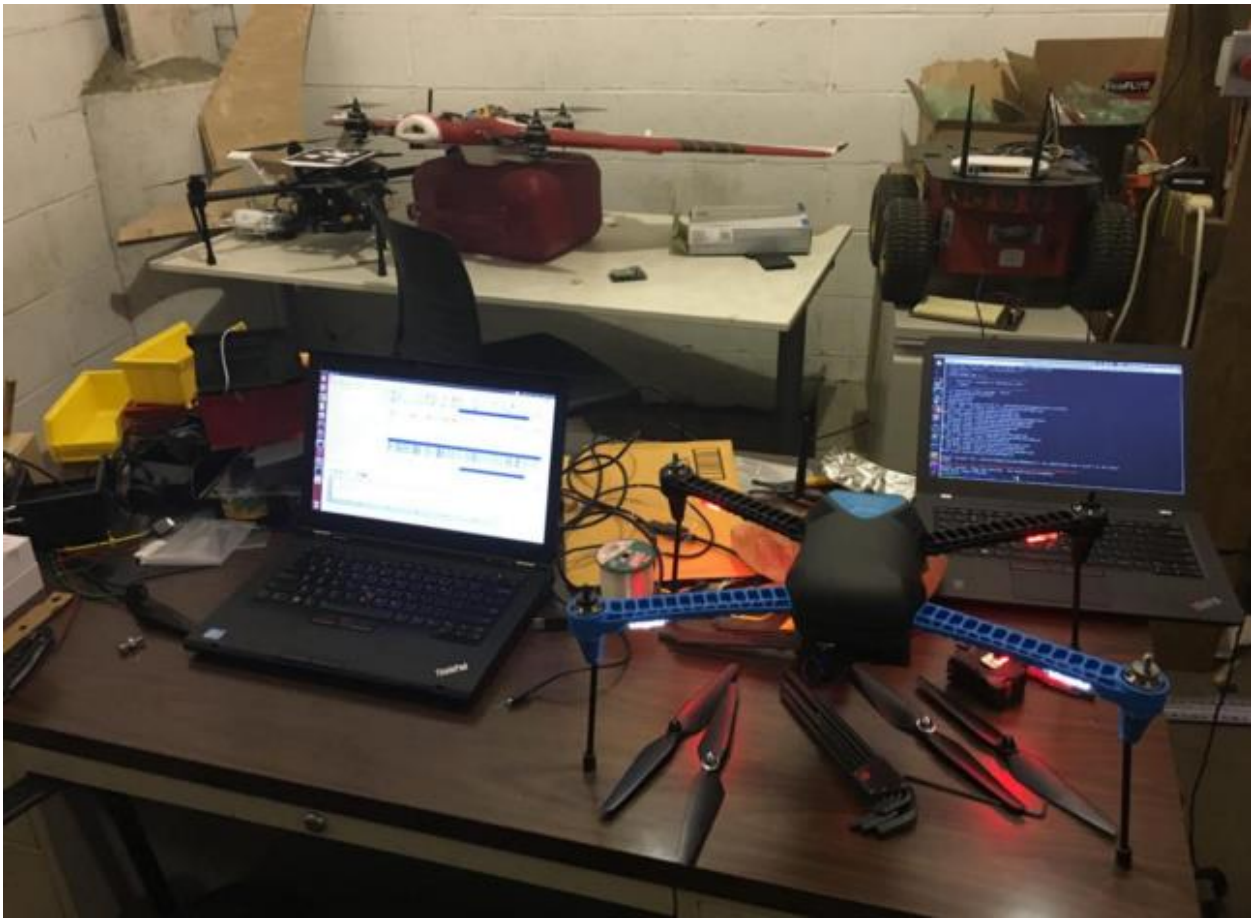

Figure 2: Quick sanity checks on desktop with propellers removed

Implemented velocity-based landing sequence

I also implemented a simple automated landing sequence based on command velocities. This was a simpler solution as compared to the location setpoint approach used in the april-tag and ended up being more successful as well.

Challenges

Slow development cycles

The need to actually perform flight testing in order to test our control code, coupled with the poor development environment, conspired to impede our progress more than expected. It took several hours to perform the type of basic debugging which would usually only take a few

minutes when writing a more traditional desktop application. The value of automated testing for sanity checks (in the form of a simulator or otherwise) became much more clear to me. It's a shame to have silly mistakes waste 20 minutes because you had to fly the drone and then grab the ladder to fish it out of the net when it flies off in an unexpected direction!

Coordinate frame ambiguities

The inconsistency of the way x, y, and z are defined between different levels of the stack and in different contexts created many hours of debugging that we could have avoided if we'd first verified the most basic level of functionality. The desire to jump straight to a solution ended up costing us quite a bit of time.

Teamwork

In order to focus on the most important goal of automated landing, our entire team focused our efforts on getting this functionality working with our two quadcopters. Job got our april tag library installed, while Cole and Rohan developed our offboard-mode control code framework.

The availability of two drone platforms was especially critical this week, since we often had two team members working at the same time on different versions of code.

Plans for Upcoming Work

In the next two school weeks I will be enabling point-to-point setpoint-based navigation in order to enable the sort of functionality we demonstrated during our fall validation experiment on the Iris+. Once this is working, I will shift my focus to assisting the rest of the team with ensuring robust automated docking is working consistently.