Individual Lab Report

Erik Sjoberg

Team C – Column Robotics

Rohan Thakker, Job Bedford, Cole Gulino

IRL 4

November 13, 2015

Individual Progress

Acquired video stream from AR.Drone2 platform

After initial difficulty with configuration, I was able to acquire the raw video stream from the AR.Drone platform and record it for further analysis along with the complete flight navigation data supplied by the ardrone_autonomy ROS package into the convenient ROS bag format.

The purpose of this video data was to enable the implementation of more advanced optical odometry as well as to manually track the ground truth during odometry experiments, as well as to enable detection of targets for identifying wellhead and dock features.

Evaluated performance of AR.Drone2 odometry

Using the captured video stream as a ground-truth reference, I was able to acquire, plot, and evaluate the amount of drift in the on-board AR.Drone odometry. The odometry data reported by the AR.Drone from of one test flight are shown below in Figure 1. This test involved takeoff followed by 2 circuits of a square with 3-meter side. The drone was manually returned to the true origin after takeoff and each circuit.

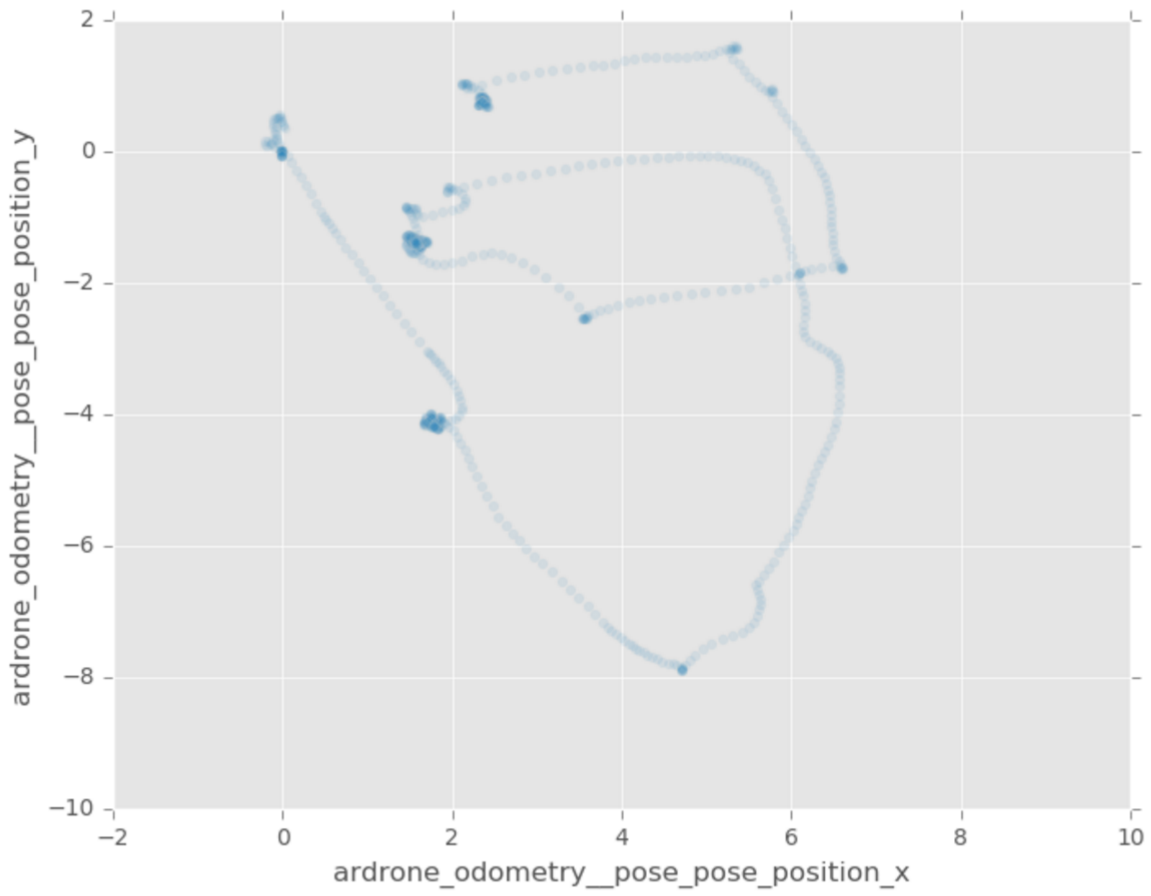

Figure 1: Recorded AR.Drone x,y pose estimates from a 3-meter square flight plan (2x)

As is immediately obvious from the plot, the reported odometry measurements do not resemble the true path of the drone which remained on a square trajectory returning to the origin. Drift of more than two meters was seen over the course of each 30-second circuit.

Leveraged an extended kalman filter to dramatically improve odometry drift
I subsequently acquired and built from source an open-source package (tum_ardrone) which implements an extended kalman filter tuned to the flight characteristics of the AR.Drone. This implementation, which builds upon the raw odometry data shown above, was dramatically superior, showing an order of magnitude less drift in odometry.

Scripted, tested and polished autonomous flight demo
Leveraging the accurate odometry provided by the tum_ardrone package I was able to script the simple autonomous flight demo which was shown at the progress review. This was achieved by sending a sequence of absolute position set-points to a simple PID control-loop which output target velocities into the /cmd_vel topic of the AR.Drone driver. A significant amount of time was spent understanding the quirks of the system and preparing a polished demo.

Challenges

Sloped floor causing inaccurate IMU initialization
During initial testing of the AR.Drone, I experienced difficulty in holding position as well as a significant amount of drift during takeoff and hover of the drone. This was eventually traced down to the subtly sloped floor of the B-level (presumably for drainage). The AR.Drone zeros out it's IMU after the battery is connected, and this had been taking place at an angle of several degrees. As a result, the drone listed significantly to the downward sloping direction during flight. This can be avoided in the future by a more careful choice of launch position.

Errors in autonomous flight scale estimation
When initially testing autonomous flight, I experienced a large unexpected movement of the drone which traced back to an incorrect scale estimate of the drone's internal map and the real environment. After commanding a 1m movement the drone flew more than 5 meters, although fortunately I was able to stop it with manual override. It's definitely important to keep hands on the disable button during autopilot maneuvers!

Wildly inaccurate / inconsistent ROS package installation instructions
Setup of the tum_ardrone package was significantly delayed due to the inaccurate, out-of-date installation instructions on the ROS wiki page of the driver package. Cross-referencing this with the source repository itself enabled me to get the package compiled successfully, but I'll definitely be less trusting of the first build instructions I find in the future.

Teamwork

Once again, the team worked well together and significant progress was achieved on many fronts.

Our dual-pronged development strategy has been fruitful. Cole and I have pushed forwards the capabilities of our fallback AR.Drone platform and higher level planning and software features, while Job and Rohan have been able to focus on bring-up of our Iris+ and dock hardware. We have experienced almost no work blockages due to waiting on dependent tasks, which I consider a significant success.

The use of two drone platforms has dramatically reduced the overall project risk, and proven an effective way to overcome the limitations of being unable to test on our in-development Iris+ hardware. Instead of investing a large amount of time in configuration of a simulator for development purposes, we have been able to dive straight into implementing our ideas on an easy-to-use system which on its own will be capable of fulfilling our most critical system requirements in the case of setbacks with the Iris+

Plans for Upcoming Work

Fall demo preparation and execution

For the next two weeks until the fall demo I will be focused primarily on completing and polishing our planned autonomous flight demonstration which comprises a major part of our FVE. I intend to finalize the specific test procedure and test in a variety of adversarial conditions to ensure a successful demonstration.

Spring-focused extensions

If time permits, I will also be working to extend the capabilities of our AR.Drone platform beyond the requirements of the fall demo, and assisting Cole with the design and implementation of our higher-level planning and control architecture which will apply to both the AR.Drone and Iris+ platforms.