Spring Sprint 1: AR.Drone Risk Mitigation, Control Node, and April Tag Detection

Individual Lab Report #6

Job Bedford

Team C: Column Robotics
Eric Sjoberg, Rohan Thakker, Cole Gulino

ILR6
1/28/16

Figure 1: Ardrone tele-op test flight

Individual Progress

Welcome to Spring Semester, MRSD Project Part II! As team we decide that we need an ultimate risk mitigation strategy if development on the Iris+ drone failed meet requirements for the Spring Validation Experiment. Again the Iris+ drone is our main platform, and we have been adding onboard sensors and computation to transform it into a terrestrial analog for undersea searching and docking. With 4 months left, we have yet to get the drone to autonomously hover, let alone search and dock. If the drone becomes too much of a hurdle, the team has decided to switch back to the Ardrone. Thus development on the Ardrone to perform the SVE must commence. The Ardrone is a low-cost reliable platform, that our team has vast experience in work with throughout fall semester. My responsibility for these past two weeks was development of the Ardrone framework to be reliably controlled via our own mover node as well as establishing April tag detection from the drones onboard camera.

Controller Node:

For the fall validation experiment, our team relied on a ROS Ardrone package called tum_ardrone. Tum_ardrone utilize the dynamics of the drone, the optical flow sensor, and the IMU reading to derive robust state-estimation and control of the drone.

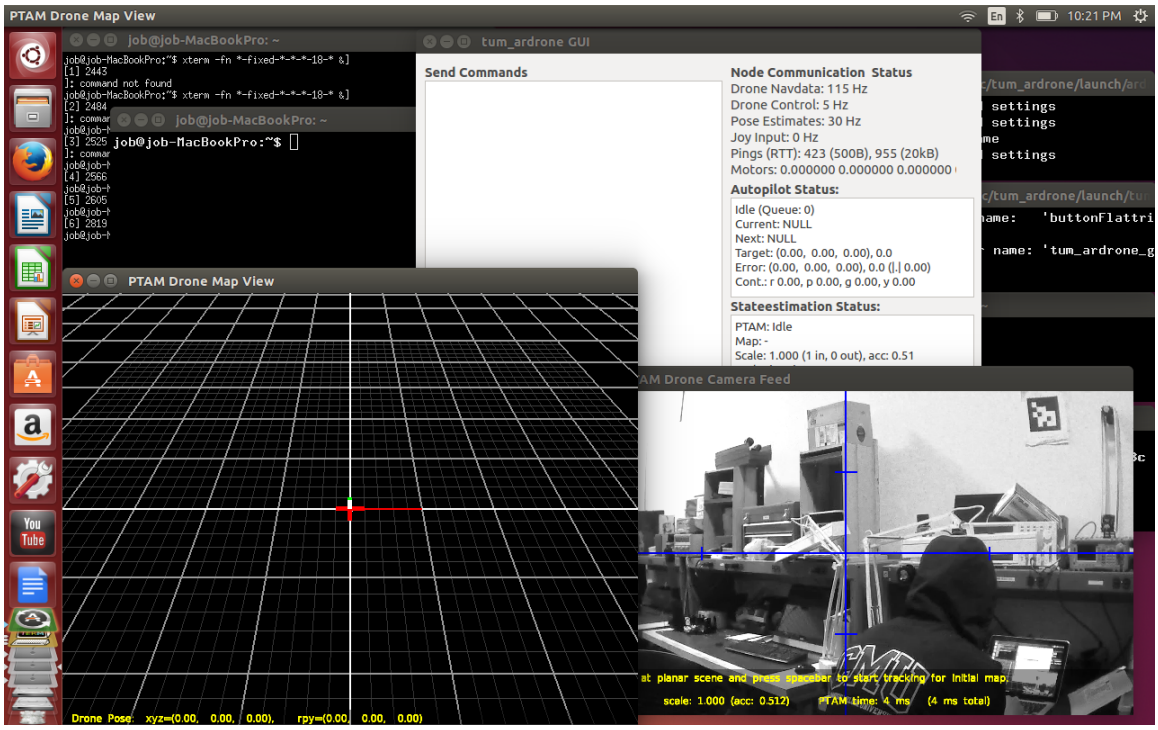

Figure 2: tum_ardrone in terminal

With this our team could coordinate the drone to desired coordinates with respect to the drone initial takeoff coordinate. Keeping the Tum_ardrone framework and control scheme, I wanted to write a node that would bypass the tum_ardrones GUI and pass the same messages to the Ardrone computer to control it. Once my own mover node was complete, a sequence of autonomous commands and instructions could orchestrate the drone's movement with the tum_ardrone package.

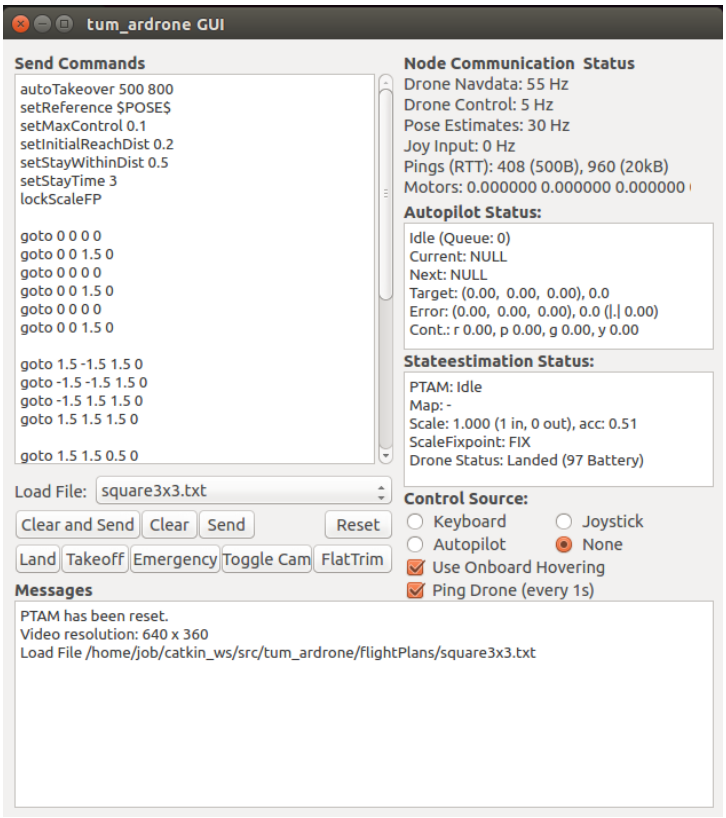

Figure 3: Tum_ardrone GUI

The command sent from the GUI are pre-defined in a 'c COMMAND' structured string. A tele-op node was written to send command in a similar fashion and control the drone to take off, land, goto left and goto right. This node will now function as a foundation for control of the Ardrone

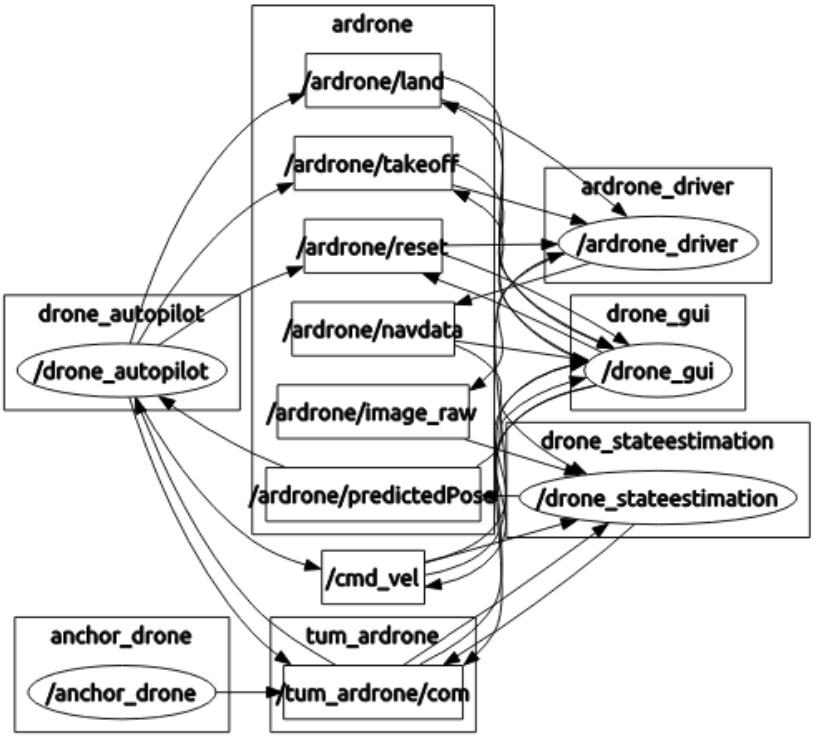

Figure 4: RQT node Graph of Tum_Ardrone with TeleOp node (anchor_drone) passing messages to the Tum_Ardrone computer

April Tag Detection:

In order for our system to find the wellhead and position itself to dock, it will need of feature to localize itself with. April tags are specially designed black and white targets that computers can discern the pose and orientation of in three space. These tags are frequently used in robotics application and make for robust, low-maintenance beacon detection. There exist three main casts of April tag, Tag36h11, Tag25h9, and Tag16h5. Each of these casts has about 16 unique ID's.

Team A and the MRSD Wiki pointed me to the AprilTags C++ Library develop at MIT, http://people.csail.mit.edu/kaess/apriltags/. This was my starting point. The library off the bat grabs camera feed from the labtop and starts searching for predefined April tags. The software worked well, though I had to change some default setting to specify the tag cast. Now I needed a ROS wrapper for this C++ in order to integrate with the ardrone drivers and controls. The goal was to post the April tag poses and orientations on a rostopic for the other nodes to utilize.

Github revealed a few ROS wrappers from the C++ MIT library. Two of them were dead ends. I spent a good amount of time tying to repurpose their publishing, but that ultimately was to no avail. Eventually I consulted Team D, who recommended third source that I ultimately ended up using. In no time at all I had April tag detection up and running.

Figure 5: April Tag Testing, Drone pointed at April tag

Challenges

The most challenging issue this past week was debugging April Tag packages. As mentioned before the tag detection softwares I initial worked with were not fully developed or required unique msg passing in ROS that was difficult to reverse engineer. This particular package, http://wiki.ros.org/apriltags_ros, I spent most of my time trying to get working. In the process, I gain some good practice with writing subscribers and classes in C++. I was seeking to work more with software this semester, so this was a good learning experience.

Other than that, everything else was pretty straight forward, thou it took me a while to figure out one need to specify to tag class the April tag detector should look for.

Teamwork

The team divided up into three groups each with their own drone subproject. I was working with the AR.Drone as an ultimate failback plan for the SVE. This included setting up an architecture and sensing scheme that will be used for the final demo. Cole and Rohan worked on debugging the fall drone. And Eric focused on establishing the second spring drone.

This week Erik worked on establishing everything we developed on the first Iris+ on the second Iris+. This included assembly of the drone, construction and mounting of any and all hardware and sensors, adapting the power system, update and installing the proper firmware, and calibrating the system as a whole. Eric performed all these tasks at a breakneck pace and by the end of the sprint the second drone was further along than the first drone. Part of the speed was due to understanding the team built over the past semester and utilizing the right Pixhawk4 firmware from the get go.

This week Rohan and Cole work as a team to debug and fix the fall Iris+ drone. The drone's biggest issue was its UART communication for the Pixhawk hardware to the Odroid. Have struggles with a faulty level shifter and unreliable micro USB ports, they spent most of the past two weeks debugging and research hackish solutions to solve it. They eventually found a round about way of repurposing the RF communications port to the computer to act as a makeshift UART port. Overall the issue was using the MAVROS firmware as oppose to the Pixhawk 4 firmware on the Pixhawk hardware. As sound as Rohan and Cole realized Erik had more success with the Pixhawk4 firmware, the team ultimately choose to go that route for the remainder of the project.

One of the greatest takeaways this week was learning the advantages of having two identical hardware platforms. With two platforms, work exponentially increases. If one sub-team is hung up on a debugging issue, the other sub-team can try a different route and if either one works, it short-circuits the other. Also with two platforms the team has more freedom to test the drones.

Upcoming Week

With the control node and April tags detection, these next two weeks I will program the drone to perform the barebones of SVE subtasks. I will first make the Ardrone autonomously hover over an April tag for an extend period of time using it's downward facing camera. Using the control node, the Ardrone will be orchestrated to perform a hardcoded 'tornado' search for a well tag. Using the drones forward facing camera, will also program to drone to home on a wall -mount April tag, from an interesting starting position with the April tag in initial sight.