Rohan Thakker

Team C: Column Robotics

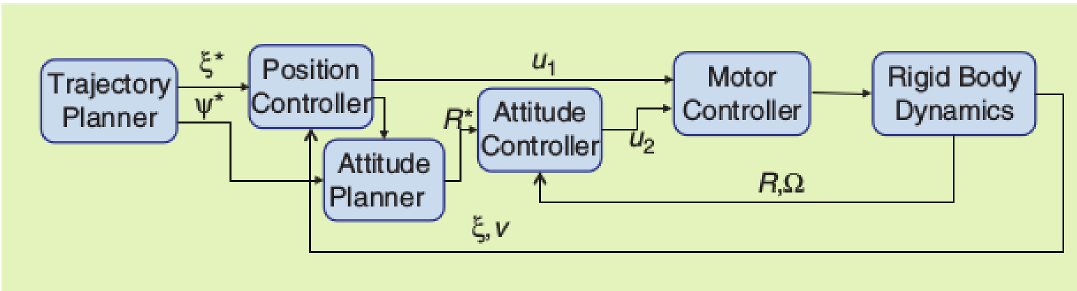
Teammates: Job Bedford, Cole Gulino and Erik Sjoberg

ILR08

Feb. 24, 2016

1) Individual Progress

For this sprint, entire team was working on autonomous docking of the IRIS+ quadrotor. As discussed in the previous ILR, we had multiple ways to implement this. Figure 1, shows the architecture of the cascaded controller which is used on most quadrotors. The reference trajectory is generated for the x, y, z position and yaw of the quadrotor. The position controller calculates the reference attitude and sends it to the attitude planner. The attitude planner generates a smooth trajectory to reach that reference attitude. The attitude controller follows this trajectory by running a PID a loop.



Mahony, Robert, Vijay Kumar, and Peter Corke. "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor." *IEEE Robotics & amp amp Automation Magazine* 19 (2012): 20-32.

Figure 1: Control Architecture

In order to accurately land on the docking platform, we needed to get the position controller working with the APRIL tag updates. Job had already worked on APRIL tag detection with the ARDRONE. Hence, he implemented the APRIL tag detection node on the Odroid using the same library. The APRIL tag detection node subscribes to the camera messages coming from the GSCAM node and publishes the pose of the APRIL TAG with respect to the camera in the camera frame.

Cole and I implemented a ROS node that subscribed to messages from the APRIL tag node and published the reference positions to PIXHAWK. The reference positions were calculated by implementing a simple proportional controller. However, the APRIL tag position updates were represented in the camera frame. Hence, we had to transform them to the body frame of the quadrotor and then to the odometry frame in which the position of the quadrotor is represented. However on testing this, we were not getting consistent readings for the APRIL tag updates. This was because of lack of proper documentation of the frame conventions.

Our main goal was to test the capability of the IRIS+ to achieve accurate landing. Hence, to validate this, we decided to work on open loop landing without the APRIL tags. We implemented this by holding the position for a couple of seconds in autonomous mode and then sending negative reference velocity along negative z-axis. As shown during the progress review. This strategy works almost as well as the ARDRONE.

2) Challenges

- Our entire work flow to test the robot was very slow. We started by running several launch files and ROS nodes. After which we would run “rosbag record” command to record all the messages. Then fly the quadrotor under manual mode and switch to autonomous mode to perform the test. After this, we transfer the data from the Odroid to our computer using “SCP” (secure copy). This would take up to 30-45 mins. Hence to speed up this process. We wrote an automated bash script to run all the nodes and launch files and copy the log files to our computer. This really help reduce our development time.
- We needed to transform the APRIL tag position coordinates from the camera frame to the odometry frame, for which we needed the positions and orientation of the body and odometry frame of the quadrotor. We found many conflicting documentations on the internet. Hence, we eventually got them experimentally.
- There was no internet on the Wifi router which we were using to SSH into Odroid. Hence, every-time we needed to google something, we had to re-connect to “CMU-Secure”. This was really more frustrating than it sounds. Thankfully, Erik got internet working on the Wifi router by using a spare Odroid and making it share the internet from an Ethernet cable.
- Small net made it very difficult to fly the quadrotor for testing. Hence, we requested Prof. Dolan to get a bigger net as soon as possible.
- During testing, we also noticed that we were not getting accurate state estimation data from the quadrotor when it was close to the ground. Initially we thought this was because of ground effects, but later we realized that the light on the quadrotor were interfering with the optical flow. We fixed this issue by coving the lights on the quadrotor using tape.

3) Teamwork

It may be evident from the individual progress section that our team worked together during this sprint to get the autonomous docking working on the IRIS+. Specifically, Job was responsible for getting the APRIL tag estimates on the Odroid. Erik was responsible for analysing the data from the ROS bags. Cole and I were responsible for implementing the position control ROS node. Apart from just following our specific responsibilities, we were all helping each other solve problems throughout the sprint.

4) Plan

This sprint we had set a very ambitious goal to achieve autonomous docking on the IRIS+ in just 2 weeks. We did not get all the deliverables for our sprint. However, we made a lot of progress in this sprint. We now have our entire software framework and workflow set up to quickly test our algorithms. This is the last sprint before the deadline of making our final decision of choosing between ARDRONE and IRIS+. Hence, we have decided that for this sprint, all the four team members will be working to complete the implementation of autonomous landing on the IRIS+. To achieve this we will divide ourselves into two teams of two members. One team will work on the problem of moving the quadrotor exactly above the APRIL tag (i.e. located on the docking station) and hold position. This will require implementing global position updates from the APRIL tag and use it to close the position controller. The other team will work on implementing autonomous land assuming that we are located exactly above the APRIL tag.

Here is a description of our demo for the next PR:

1. Set up the mock docking station on the ground (sheet of paper with APRIL Tag stuck on it)
2. Take off the IRIS+ under manual control and hover to bring the APRIL Tag in the field of view of the camera
3. Switch to autonomous mode and observe the autonomous landing of the IRIS+

5) References

[1] https://pixhawk.org/modules/px4flow

[2] http://www.hardkernel.com/main/products/prdt_info.php