Rohan Thakker

Team C: Column Robotics

Teammates: Job Bedford, Cole Gulino and Erik Sjoberg

ILR11

APRIL 12, 2016

1) Individual Progress

During the last sprint, we demonstrated the “Search for Wellhead” and “Land” sub-systems, individually. As shown in figure 1, for this sprint, we implemented the “Move to Pre-Landing Position” and integrated all the subsystems in “Tactical Planning” by implementing the “Global Planning” subsystems. In this process, we also fixed several bugs in the “Search for Wellhead” and “Land”.

The subsystems of “Tactical Planning” are implemented as functions inside a python script and “Global Planning” is implemented as a state machine to these functions. During the last sprint, we have implemented the “land” subsystem as a separate ROS node in C++. Hence, to be consistent with our architecture, I implemented the “land” subsystem in the python framework.

During testing of this subsystem, I found a bug in the Filtered Pose subsystem. As described in the previous ILR. The RANSAC filter gives the best estimate by looking at 15 most recent samples. This strategy will fail for the case when APRIL tag goes out of the field of view of the camera and then returns back. To resolve this issue, I implemented a check to process the readings only if the time difference between the oldest and the most recent sample is less than 2 seconds. For this, I had to change the message type from Pose to PoseStamped and pass the timestamp of the sample in every message. Hence I had to update the entire pipeline to be consistent with this message type.

To fuse the data of the APRIL tag, we decided to update the reference position of the position controller instead of adding a new measurement update to the state estimator running on the PIXHAWK. This saved us a lot of time because we didn’t have to look through the PIXHAWK documentation to figure out the frame conventions, message types and time synchronization problems. Hence, “Move to Pre-docking” position was implemented simply by setting the desired position to the sum of position of the APRIL tag with respect to the

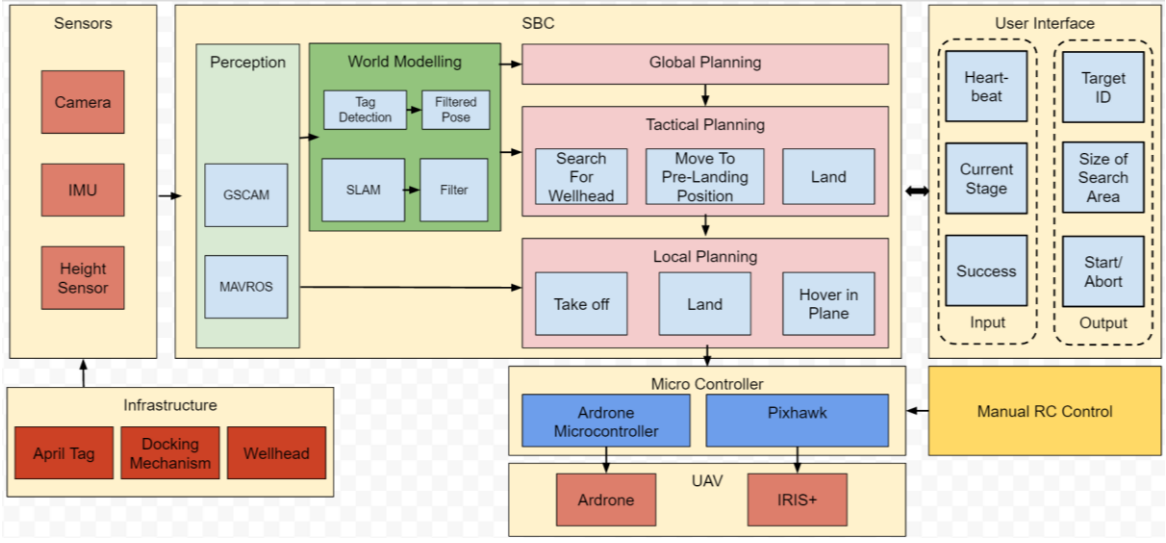

Figure 1: Cyber-physical Architecture

quadrotor (obtained from APRIL tag measurements) and the position of the quadrotor in the world. Another bug that I found in the code was that the position of the quadrotor should be taken at the time the APRIL tag update was obtained and not the current position.

Currently our system was closing the loop only on the X, Y and Z position. During this sprint I also worked on implementing yaw control. Currently, we were only sending the position updates from the APRIL tag through the "World Modelling" subsystem. Hence, I had to update the POSE messages to send the information about the yaw angle in orientation. The APRIL tag detection library was using EULER angles for the orientation. Hence, I had to ensure that the right convention of the EULER angles is followed throughout our pipeline. This functionality has been implemented but not tested with the entire system due to lack of time. However, the system does seem to work well even without running yaw control.

Finally, after implementing and unit testing these subsystems, I began working on the integration of these subsystems. I invested more time in making sure that there were no bugs in the implementation of the individual systems. Hence, the actual integration did not take more than 3 hours to code and test.

2) Challenges

- Last week during testing the cone-search, we noticed a large drift in the motion of the quadrotor. Our initial speculation was that it was because of shifting to the other drone. But we saw the same problem on both the drones. Further, we tried reverting back to the version of the code that was working earlier and even that did not solve the issue. Then, we noticed that one of the lights above the net was off. On turning

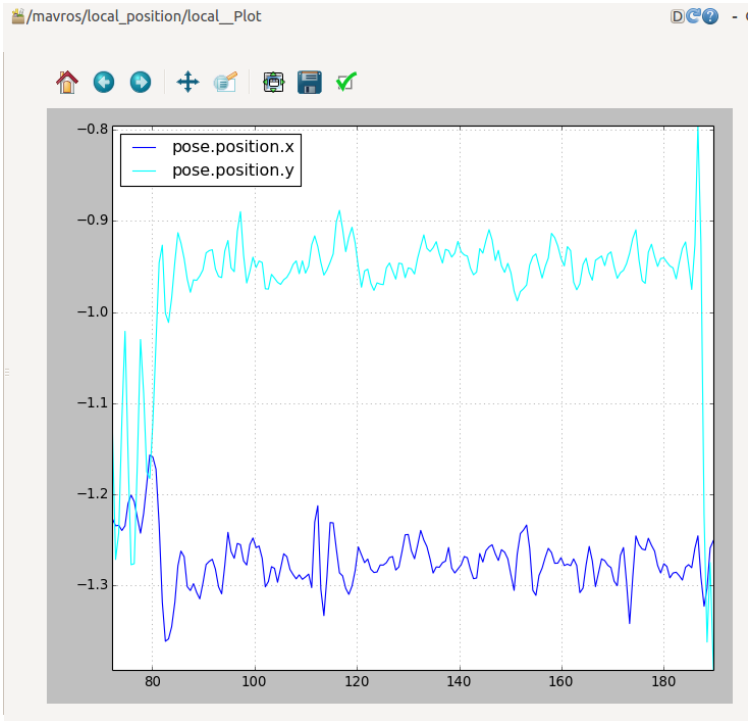

Figure 2: Position Holding Results

the light back on, we got better results and the issue was resolved. Hence, we inferred that since we are using a frame rate of 200 Hz. The flickering of the lights does not average out. Hence, having multiple lights on helps as they might have a phase difference which will reduce this effect. The figure 2 shows the results of position holding after resolving this issue.

- Another problem we faced was that we were getting updates from APRIL tag only at 1.5 Hz compared to 6-7 Hz on the other drone. We tried to debug the issue by running the same code on both the drones, but that didn't help. Finally we noticed that the GSCAM configuration was different on the two drones. The drone that was giving us updates at 6-7 Hz was using a frame rate of 100 fps instead of the default of 30 fps. Increasing the frame rate reduced the resolution of the image and hence the computation time of the APRIL tag detection algorithm.
- Another bug that took up a lot of time was using fabs() instead of abs() to find the absolution difference of ROS TIME in seconds. ROS stores time as a 32-bit integer instead of a float. Hence fabs() was interpreting the arguments as floats and returning garbage values.

3) Teamwork

During this sprint, Erik worked on refining the high level python framework for integration of the search and landing. He also spent time on testing and improving our final system and making it more robust.

Cole, Job and I worked closely on all the tasks that I described in the individual progress. We parallelized our development on the two drones. Job was working on integration of the subsystems using the existing implementation from our last sprint. The subsystem were working but they had bugs (described in section 1 and 2) which would make them fail under certain conditions.

Cole and I worked on fixing the bugs in the subsystems and unit testing them. Hence, Job was able to identify many critical bugs in our subsystems which would show up during integration and we fixed and tested them during the unit testing phase itself. One of the main learning that we have got from our project is that parallelizing development efficiently is very critical to speeding up the development time. We can fail early and identify bugs and also explore different ways to solve the same problem.

4) Plan

This sprint we have met all our requirements that are promised for the SVE. The next step is going to be about refining these results and making sure that they robust and repeatable. Hence we will be spending time on testing and tuning. We also want to work on meeting our stretch goal which is integrate SLAM updates with the on-board state estimator.

We have been facing many problems with getting the RTAB map working on the ODROID and hence we decided to shift to the MINOWBOARD-MAX which is based on x86 architecture. Hence, we can directly use the binaries and eliminate the need to compile from source. Hence, Erik will be working on getting SLAM running on this board.

Job will be working on improving the accuracy of landing and testing it with our actual docking mechanism. Cole and I will be working on integration of the SLAM updates with the on-board state estimator.

5) References

[1] RANSAC image source: http://www.3dcalifornia.com/

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

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