Sensors and Motors Lab

Rohan Thakker

Team C: Column Robotics

Teammates: Job Bedford, Cole Gulino and Erik Sjoberg

ILR03

Oct. 30, 2015

1) Individual Progress

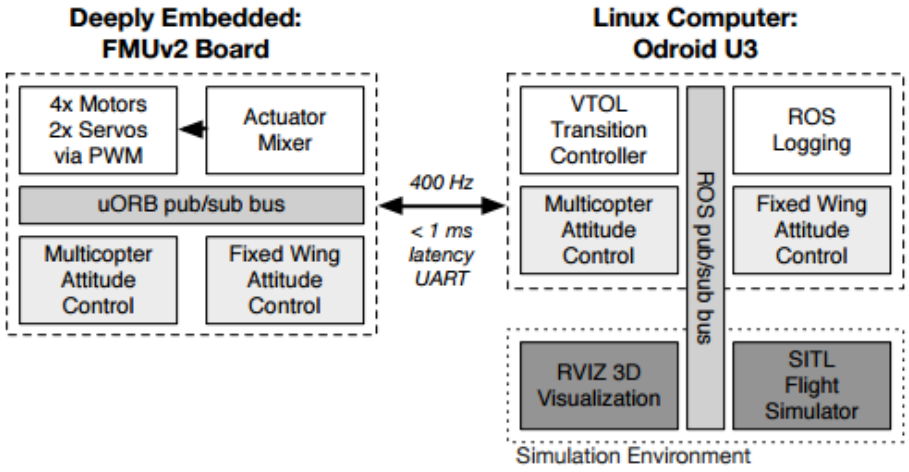


Figure 1 Software Framework for controlling the Drone using a Single Board Computer source: [1]

My main objective for this week was to get familiar with the Pixhawk firmware. I started by reviewing [1], paper of ICRA 2015 that details the software architecture of Pixhawk. The deeply embedded controller (Pixhawk) runs on NuttX operating system, shown in Figure 1. External linux companion computer (MinnowBoardMax in our case), communicates with the Pixhawk over UART using MAVROS protocol.

As shown in Figure 1, the 3DR provides Software in the Loop (SITL) Flight Simulation environment. Since there is a delay in the shipping of our IRIS+, I worked on getting the simulation environment setup. This will also help us test our control algorithms before we run them on the hardware.

Further, I also spend some time understanding the dynamics of quadcopter's and differential flatness. Appendix shows the derivation of the dynamics model a quadcopter in state space and its LQR formulation for developing a controller. Further, it also details the properties of differential flatness that makes the quadcopters controllable directly in the 4 dimensional output space (Y).

Cole and I met Shaurya Shankar (Current PhD student under Dr. Nathan Michael). He suggested that before getting into any sophisticated algorithms for state estimation, we should try to run a Lukas Kande based optical flow at high frame rate. This is generally enough for most In-door applications. Me and cole have reviewed the algorithm and will be implementing it this week. [2] shows the paper we used to review the Lukas-Kanade (LK) based optical flow. The LK algorithm uses first order Taylor series approximation and gradient descent to solve an optimization. Hence if we run at a high frame rate, the displacement between consecutive frames will be less. Hence, the Taylor's series approximation will be more accurate and the gradient descent algorithm will converge faster. We also ordered the Sony Playstation Eye camera which gives a frame rate of 120Hz.

2) Challenges

Our IRIS+ quadcoter was declared as fraud by Fedex for some unknown reason and was sent based to California. We had to get it re-ordered from 3DR and they have finally reshipped our quadcopter. We now expect to receive it by next week.

I also faced many problems while getting up the Pixhawk development environment setup. It runs on the NuttX operating system. I used [3] to setup the development environment but was getting build errors after pulling the source code from github. Then I tried [4] to setup the gazebo based simulation environment but was unsuccessful because of the same error. Finally, I resolved the issue by [5] which uses a docker container. I was able to get readings from an external joystick which can be used to control the drone in simulation.

Our team was also facing a problem of deciding the right message format in ROS to communication between Reader, Planner and Mover nodes. ROS has multiple ways to handle this: ROS Frame, TF and Odometry Message. The problem was that we have our orientation estimate in quaternions and its covariance in fixed angle representation. Erik researched each of this, and we decided to use Odometery Message:

```
# This represents an estimate of a position and velocity in free space.
std_msgs/Header header (Header.frame_id = odom frame)
        → uint32 seq   (increasing ID)
          time stamp
          string frame_id
string child_frame_id (ID of base_link = frame on quadcopter)
geometry_msgs/PoseWithCovariance pose    (USING odom GLOBAL FRAME)
        → geometry_msgs/Pose pose
                → geometry_msgs/Point position
                  geometry_msgs/Quaternion orientation
            float64[36] covariance (x, y, z, rot about X, rot about Y, rot about Z) X 6
geometry_msgs/TwistWithCovariance twist   (USING base_link QUADCOPTER FRAME)
        → geometry_msgs/Twist twist
                → geometry_msgs/Vector3 linear (x, y, z)
                  geometry_msgs/Vector3 angular (x, y, z)
            float64[36] covariance (x, y, z, rot about X, rot about Y, rot about Z) X 6
```

3) Teamwork

Table 1

Job Bedford	**Cole Gulino**	**Erik Sjoberg**	**Rohan Thakker**
Fixed bug with MOVER Node for AR.Drone in ROS	Identified requirements and design for the power board	Setup the navigation/planning framework in ROS	Studied dynamics and control of quadcoter
Tested the MOVER node on AR DRONE	Studied LK optical flow tracker	Researched and documented Odometry usage	Ordered Camera and Studied LK optical flow algorithm
	Reviewed pixhwak code and setup the development environment		Reviewed Pixhwak code and setup the simulation environment

The Table 1 shows the tasks done by each of our team members.

4) Plan

Cole and I will be working together on low-level controls of the IRIS+. This includes implementing the LK based optical flow on the MinnowBoardMax and optimizing the code to get maximum frame-rate. We will also setup the simulation environment of the IRIS+ in gazebo and controlling it from a ROS node.

Erik will be working on implementing visual odometry using available open-source packages and comparing them. Job will be using Erik's work to see it's performance on the ARDrone and also set up the framework to publish images received from the ARDrone to the planner node.

5) References

[1] http://people.inf.ethz.ch/dominiho/publications/ICRA_2015_px4_autopilot.pdf

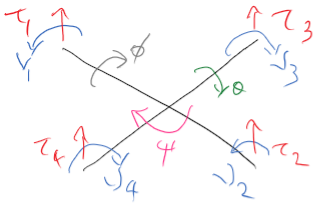
[2] https://www.cs.cmu.edu/afs/cs/academic/class/15385-s12/www/lec_slides/Baker&Matthews.pdf

[3] https://pixhawk.org/dev/minimal_build_env

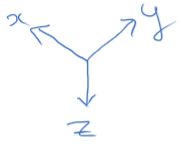
[4] https://pixhawk.org/dev/ros/sitl

[5] https://pixhawk.org/dev/ros/automated_sitl

Appendix:



$$u_1 = (\tau_1 + \tau_2 + \tau_3 + \tau_4)$$
$$u_2 = \lambda(\tau_1 - \tau_3) = \dot{\phi}$$
$$u_3 = \lambda(\tau_1 - \tau_2) = \dot{\theta}$$
$$u_4 = v_3 + v_4 - v_1 - v_2 = \dot{\psi}$$
$$\ddot{x} = u_1 \cos\phi \sin\theta = u_2$$
$$\ddot{y} = u_1 \sin\phi = u_3$$
$$\ddot{z} = g - \frac{u_1}{m}\cos\phi\cos\theta = u_4$$



Euler angles $[\phi, \theta, \psi]$
roll, pitch, yaw

$$u = \begin{bmatrix} \sum F_i \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$\dot{x} = \frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ u_1\cos\phi\sin\theta \\ u_1\sin\phi \\ g - u_1\cos\phi\cos\theta \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \lambda(\tau_4 - \tau_3) \\ \lambda(\tau_1 - \tau_2) \\ v_3 + v_4 - v_1 + v_2 \end{bmatrix} \quad \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ u_2 \\ u_3 \\ u_4 \end{matrix} \qquad (1)$$

$$y = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix} = \begin{bmatrix} I_{3\times3} & 0 & 0 & 0 \\ 0 & 0_{1\times5} & | & 0_{1\times5} \end{bmatrix} x = Cx \qquad (2)$$

16-10-2015 22:50 - Screen Clipping

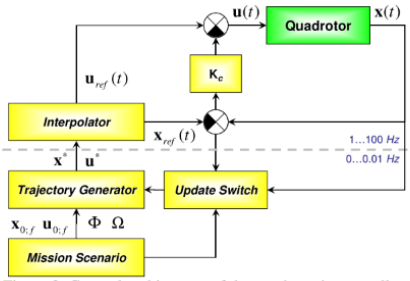


Figure 2. General architecture of the quadrotor's controller.

LQR Problem formulation:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$u(t) = -K_c\, x(t)$$
$$\dot{x}(t) = (A - BK_c)\, x(t)$$
$$\rightarrow J = \int_0^\infty (x^T Q x + u^T R u)\, dt \quad \text{is minimized}$$

Assuming $x_{ref}(t) \rightarrow$ desired trajectory

$$u(t) = u_{ref}(t) - K_c\,(x(t) - x_{ref}(t))$$

$$Q = 10^{-5} I_{12\times12} \qquad R = \text{diag}(10^{-5}, 10^8, 10^8, 10^8)$$

NOTE: A, B are linearized about hover posⁿ.
LTI Valid within set $\xi_{sf} = \{\theta, \phi : \theta^2 + \phi^2 \le \gamma^2, \gamma = 45^\circ\}$
obtained by Analyzing for envelope of operation.
This Linear model can be used as initial guess in the non-linear optimizaⁿ.

Trajectory optimizaⁿ :-

eg:. $\min_{u(t) \in U \subset \mathbb{R}^t} \phi \quad \forall \; t \in [0, t_f]$

S.T. $\dot{y} - C g(x, u) = 0$

$y_0 - C g(x_0, u_0) = 0$ — from ① & ②

$y_f - C g(x_f, y_f) = 0$

$C(x, u) \leq 0$

dynamic inequality constraint on trajectory (for obstacle avoidance) & on the state & input (to avoid singularities & constrains on u)

$$\phi = (1-w) \frac{1}{t_f} \int_0^t \sqrt{p_1 \dot{x} + p_2 \dot{y} + p_3 \dot{z}}\, dt + w (t_f - T)^2$$

ϕ: cost to measure optimality of trajectory

$\sqrt{p_1 \dot{x} + p_2 \dot{y} + p_3 \dot{z}}$: running cost (fuel)

$(t_f - T)^2$: terminal cost

$w = 1, T = 0$ minimum time problem.

$w = 0$ $p_1 = p_2 = p_3$ minimum fuel

Requirement for differential flatness:

① components of y are not differentially related over R

② every system variable may be expressed as a funcⁿ of output y

③ every component of y may be expressed as a funcⁿ of system variables & a finite no. of their time derivatives

The state vector x and control input u can be both expressed via derivatives of the output vector y ie.

$x = h_1(y) \quad u = h_2(y)$