Drone Controller in C++

Udacity Flying Car NanoDegree

Mark R. Helms

June 28th, 2018

Introduction

This project controls a drone in a simulator, which is placed under various scenarios in order to test the control logic and tuning in a progressive fashion (i.e. required actions isolate portions of the code) and culminates in a non-idealities movement test and a tilted “figure 8” trajectory-following test. In order to pass each scenario, certain thresholds must be met. The scenarios include: mass calibration, attitude control, position and heading control, non-ideal drone control and figure-8. The code is written in non-STL C++. The simulator and code skeleton were provided by Udacity.

The Drone Model

The drone model is important, because the logic and math of the controller must match that of the model in order to be effective. Figure 1 shows some information about the drone.

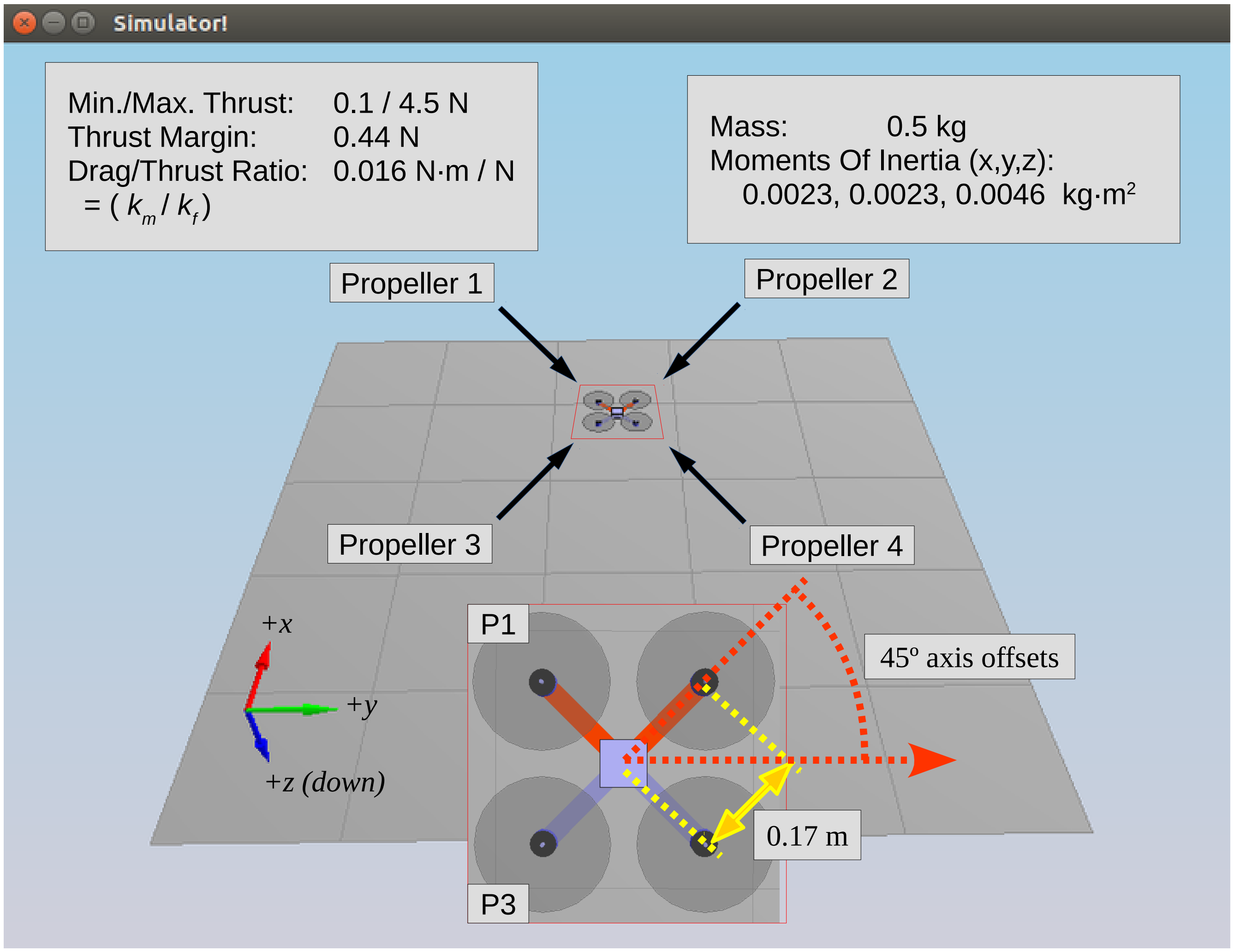

Figure 1. The drone and some of its parameters.

Scenario 1: Intro – Determining Drone Mass

The introductory scenario requires adjusting the mass until the drone is able to hover in place. This is an important step and is a fundamental parameter tune for the system. The approximate mass was found to be 0.5 kg, at which point the drone would hover in place.

Scenario 2: Motor Command, Body Rate Control, and Roll/Pitch Control

Motor Command

A collective thrust (F_{tot}, Newtons) and three moments about the x, y, and z body frame axes (M_x,M_y,M_z, Newton-meters) are used to command the motors. Both the vertical thrust (F) and torque (τ) of a motor with propeller are proportional to the square of the propeller spin rate (ω, rad/s), with thrust being proportional at a factor of k_f and torque being proportional at a factor of k_m as shown:

$$F = k_f \omega^2 \qquad \tau = k_m \omega^2 = Fl$$

In this project, we'll have to find the thrust for a motor, given a torque. It's possible to make this conversion without solving for ω, given the drag-to-thrust ratio (κ , Newton-meters / Newton).

$$\kappa = k_m / k_f$$

The moment on the drone produced by a motor about an axis is the thrust (F, Newtons) at a distance (l, meters) from the center of the drone, perpendicular to the axis of interest. In this instance, we know the distance from the center of the drone to each motor (L, meters) and the fact that each motor is at a 45' angle from the body frame axes as shown in Figure 1. Equations are as shown:

$$l = \frac{L}{\sqrt{2}} \qquad F = \frac{M}{l}$$

In order to convert the required force and torques to individual motor thrusts, the following equations are used:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \times \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{pmatrix} = \begin{pmatrix} F_{tot} \\ \frac{M_x}{l} \\ \frac{M_y}{l} \\ \frac{M_z}{\kappa} \end{pmatrix}$$

where F_1, F_2, F_3 and F_4 are the thrusts of motors 1, 2, 3, and 4, respectively, and M_x produces a roll, M_y produces a pitch, and M_z produces a yaw. Note that yaw is independent of distance to the motors, but requires a conversion from motor thrust to motor torque via κ.

The matrix equation above relating total thrust and moments to individual motor thrusts can be solved for each motor thrust (F_n), resulting in the following equations:

$$4F_1 = F_{tot} + \frac{M_x}{l} + \frac{M_y}{l} + \frac{+M_z}{\kappa},$$
$$F_2 = \frac{1}{2}\left(F_{tot} + \frac{M_y}{l} - 2F_1\right),$$
$$F_3 = \frac{1}{2}\left(F_{tot} + \frac{M_x}{l} - 2F_1\right),$$
$$F_4 = \frac{1}{2}\left(F_{tot} + \frac{M_z}{\kappa} - 2F_1\right)$$

Body Rate Control

For this scenario, body rate control includes the ability of the drone to approach and maintain a targeted roll and pitch. Given are the target body roll, pitch and yaw rates (p_{tgt}, q_{tgt}, r_{tgt}, rad/s^2) targets and the measured roll, pitch and yaw rates of the drone (p_{act}, q_{act}, r_{act}, rad/s^2). A proportional ("P") controller is used to provide a correction factor for each axis (p_{adj}, q_{adj}, r_{adj}, rad/s^2), and the drone's moments of inertia (I_x, I_y, I_z, kg-meters^2) are used to convert them into moments.

$$(p_{adj}, q_{adj}, r_{adj}) = ((p_{cmd}, q_{cmd}, r_{cmd}) - (p_{act}, q_{act}, r_{act})) \cdot (k_p, k_q, k_r)$$
$$(Mx, My, Mz) = (p_{adj}, q_{adj}, r_{adj}) \cdot (I_x, I_y, I_z)$$

Roll/Pitch Control

While the body rate controls roll, pitch and yaw rates in the body frame, another controller is necessary to control roll, pitch and yaw rates in the world frame (θ_{tgt}', φ_{tgt}', ψ_{tgt}', rad/s). This section only addresses roll and pitch. Once again, the target roll and pitch are given along with a measured roll and pitch and once again a P controller is used to produce a corrective adjustment, as follows:

$$(\dot{\theta}_{adj}, \dot{\varphi}_{adj}) = ((\dot{\theta}_{cmd}, \dot{\varphi}_{cmd}) - (\dot{\theta}_{act}, \dot{\varphi}_{act})) \cdot (k_\theta, k_\varphi)$$
$$\ldots = ((\dot{\theta}_{cmd}, \dot{\varphi}_{cmd}) - (\dot{\theta}_{act}, \dot{\varphi}_{act})) \cdot k_{bank}$$

where $k_\theta = k_\varphi = k_{bank}$ due to the x-y symmetry of the drone.

Once the world-frame roll, pitch, and yaw rates are known, they must be converted into the body-frame by using a rotation matrix.

$$R = R_z(\psi) \times R_y(\varphi) \times R_x(\theta),$$
$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix},$$
$$R_y(\varphi) = \begin{pmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\theta) \end{pmatrix},$$
$$R_z(\psi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Using the rotation matrix, R, and the following equations, we may determine the body-rate equivalents for roll and pitch (p, q):

$$\begin{pmatrix} p \\ q \end{pmatrix} = \frac{1}{R_{33}} \begin{pmatrix} R_{21} & -R_{11} \\ R_{22} & -R_{12} \end{pmatrix} \times (\dot{\theta}, \dot{\varphi})$$

Working backwards once more, given a collective thrust and desired world-frame x-axis and y-axis accelerations, the roll and pitch target angles (θ_{tgt}, φ_{tgt}) can be found from the trigonometric relation to the drone's total thrust, F_{tot}, being applied at the roll and pitch angles. Because sin(x)=x for very small x, and because the linearization of the algorithm is necessary for our linear controller, we will substitute x in for sin(x).

$$a_x = F_{tot} \sin(\theta_{tgt}) \approx F_{tot}\,\theta_{tgt}\,,$$
$$a_y = F_{tot} \sin(\varphi_{tgt}) \approx F_{tot}\,\varphi_{tgt}$$

The k parameters were manually tuned in order to provide suitable P controller effectiveness. Simulator results can be seen in Figure 2.

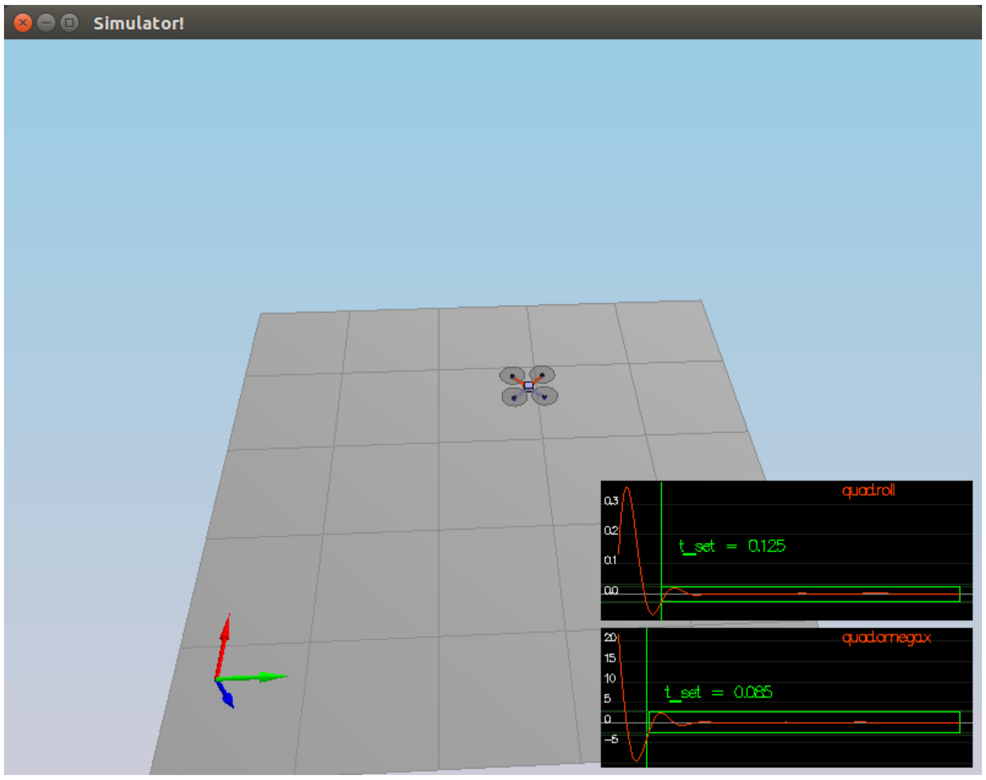

Figure 2. Successful scenario 2 run.

Scenario 3: Position Control and Yaw Control

Position Control

In order to control the lateral position of the drone, a proportional-integral-derivative (PID) controller is used, along with feed-forward (FF) acceleration. In order to be able to use a PID controller, the target for a variable and its first derivative must be known. The integral portion is necessary if a bias may exist that needs to be eliminated. An equation for a PID controller can be shown below:

$$a_{PID}=k_p(x_{target}-x_{actual})+k_d(v_{target}-v_{actual})+k_i\int(x_{target}-x_{actual})\,dt$$
$$a_{(PID,FF)}=a_{PID}+a_{FF}$$

For the cases of x and y position correction, the integral term is 0.

The world frame to body frame conversion for z acceleration is performed at this point(x and y were performed after roll/pitch control). Using the same R as before, the equation is:

$$z_{body}=\frac{z_{world}}{R_{33}}$$

Yaw Control

Yaw rate control uses a simple P controller, as used above for body rate control. In order to avoid mathematical confusion between positive and negative radians, the terms are adjusted by factors of 2π until the absolute difference between them is between -π and π. Once again the k factor for the P controller had to be adjusted for satisfactory results. Simulator results can be seen in Figure 3.

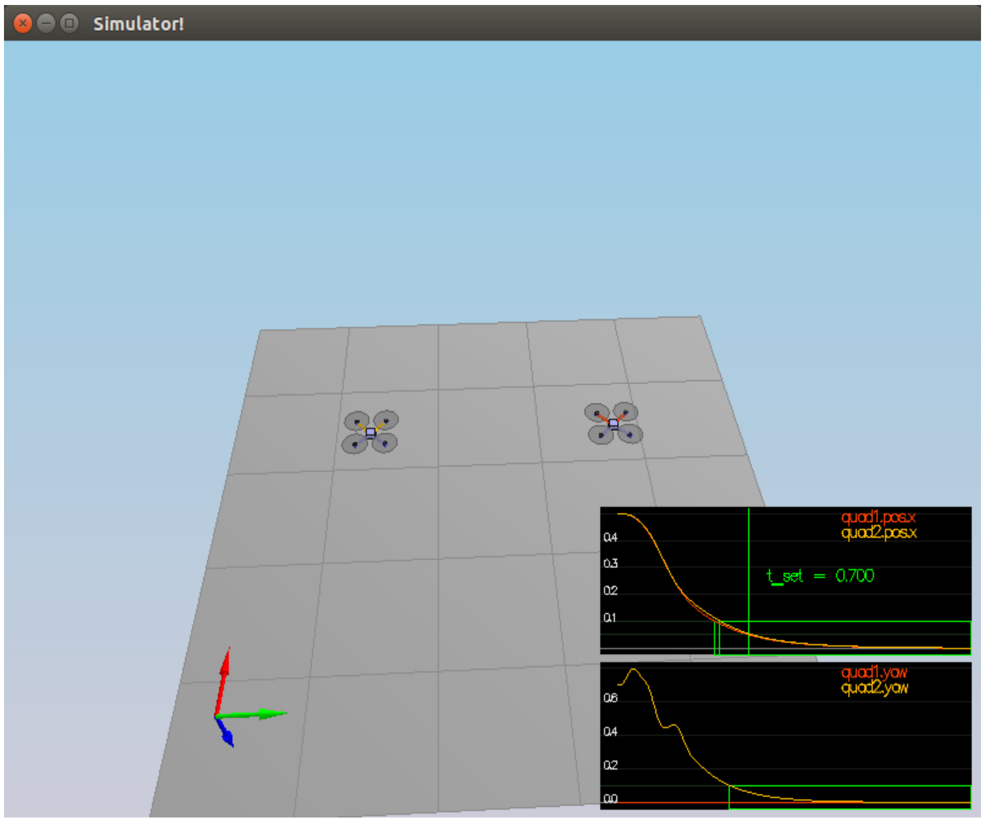

Figure 3. Successful scenario 3 run.

Scenarios 4 and 5: Non-Idealities and Trajectory-Following

Non-Idealities

In order to test the quality of the controller on non-ideal drones, scenario 4 in the simulator provides three drones. The green drone has its weight shifted back off center, the orange drone is ideal and the red drone is heavier than normal. Simulator results can be seen in Figure 4.

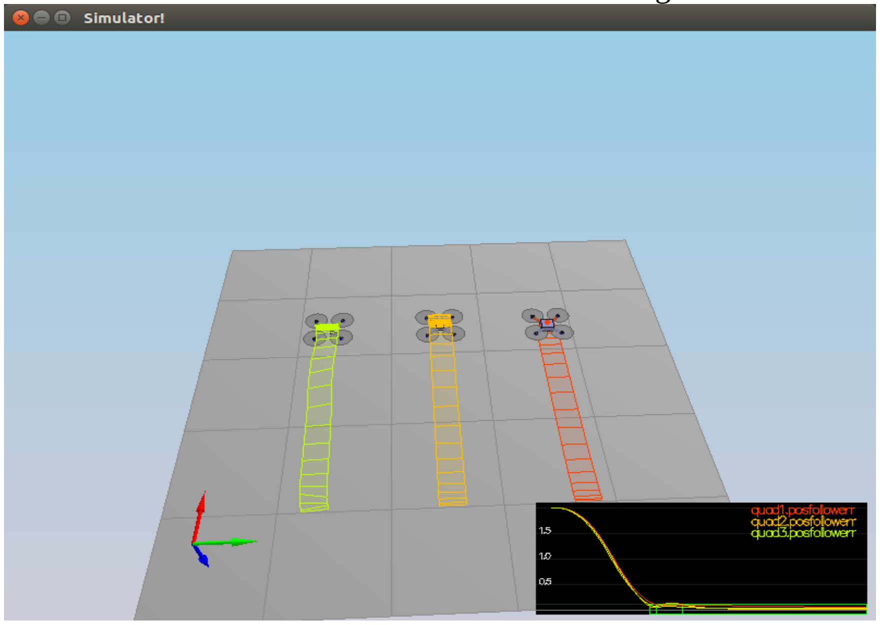

Figure 4. Successful scenario 4 run.

Trajectory Following

In order to further test the quality of control parameters chosen, the drone is asked to fly in a tilted figure-8 pattern and deviation from this pattern is measured. Simulator results can be seen in Figure 5.

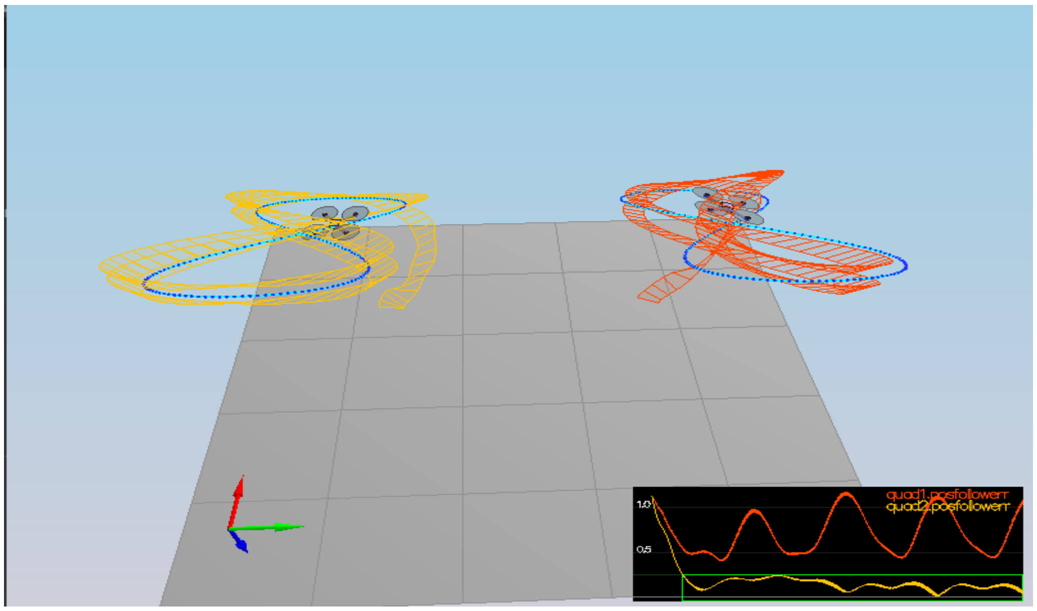

Figure 5. Successful scenario 5 run.

PID Controller Factors

Suitable PID controller factors found were as follows:

Variable	kp	ki	kd
position x / y	40	-	13
altitude z	45	30	30
world pitch / roll	17	-	-
world yaw	2	-	-
body pitch / roll	65		
body yaw	8		

Code Sections

Job	Code Function (QuadControl.cpp)
Conversion from desired thrust/moments to individual control	GenerateMotorCommands
P controller for body rates and conversion to moments	BodyRateControl
Conversion from desired collective thrust and x, y accelerations to desired roll and pitch angles. P controller for roll and pitch angles. Conversion of roll and pitch angles to body rates.	RollPitchControl
PID-FF controller for vertical position. Conversion from of vertical thrust from world-frame body-frame.	Altitude Control
PD-FF controller for horizontal position.	LateralPositionControl
P controller for yaw.	YawControl

Improvements

While the controller factors successfully pass the tests in each scenario, further tuning could be useful to allow for improved results.