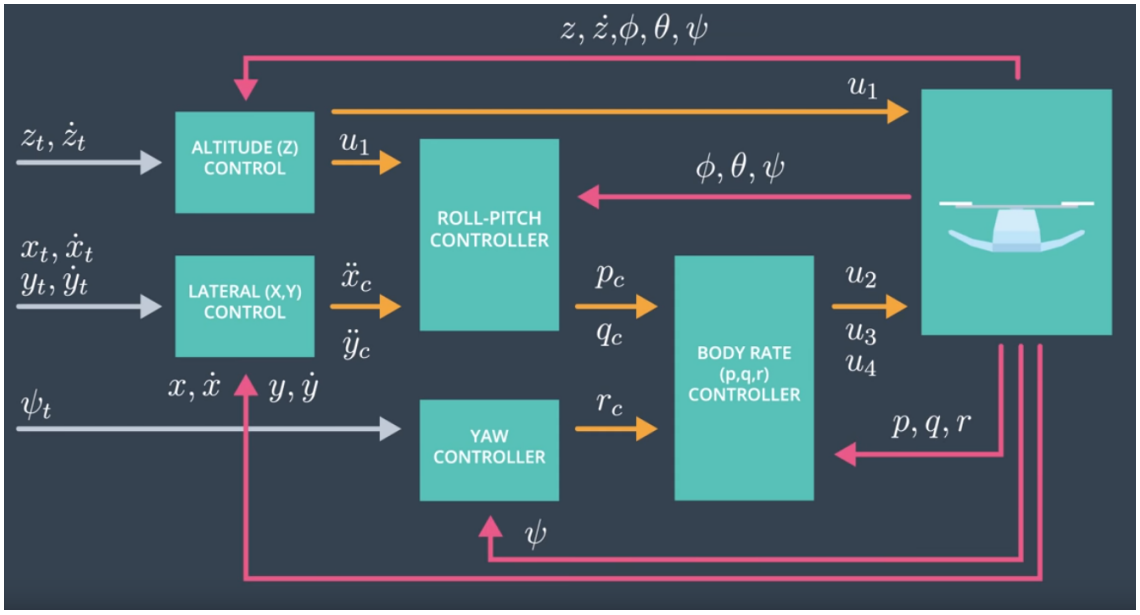
Project-3: Control of a 3D Quadrotor

1. Description

In these lessons, we have learned about the basic vehicle control, the control architecture and a full 3D cascaded control. The goal of this project is to design and build a cascade PID controller, which will control the quadrotors to fly the desired trajectory in the 3D environment. We test the C++ code controller with udacity provided simulator that provides a higher fidelity model of the real vehicle this code would be running on.

As described in the lesson, the 3D Control architecture for a quadcopter can be illustrated as the below image:



The 3D Control Architecture Diagram shows a cascade PID control system with 5 controllers, 4 actuators (u1 to u4) acting on cascade control loops. A feed-forward approach was used in the system, which adapts the motion parameters sent to the vehicle controller.

2. Project Rubric

2.1 Implemented Controller

2.1.1 Implement body rate control

The body rate control is a P controller on body rates to commanded moments. Steps of body rate control as follows,

1. compute current body rate error in all direction (r, p, and q)

2. compute angular acceleration

3. generate the rotational moment

4. constrain the desired moment within a set of bounds (-MAX_TORQUE, MAX_TORQUE)

C++: lines 97 to 117 in QuadControl.cpp

2.1.2 Implement roll pitch control

The roll-pitch control is also a P controller in the body frame, Which is to take a thrust command as well as the desired x and y accelerations and attitude pitch, roll, yaw and p, q, r. and output a target roll and pitch rate. Steps of roll-pitch control as follows,

1. get collective acceleration: c = -thrust_cmd / DRONE_MASS

2. actual portion of acceleration on x and y direction from rotation matrix

3. target portion of acceleration on x, y and z direction: b_c = acceleration_cmd / c

4. compute current position error of b term: b_err = b_c – b

5. compute target change rate of b term: b_dot_c = self.k_p_euler_angles[:2][::-1] * b_err

6. r = np.array([[R[1, 0], -R[0, 0]],

[R[1, 1], -R[0, 1]]],

dtype=np.float)

7. generate the target roll and pitch rate with matrix multiplication: pq_c = np.dot(r, b_dot_c) / R[2, 2]

C++: lines 125 to 168 in QuadControl.cpp

2.1.3 Implement altitude control

Steps of altitude control are as follows,

1. get actual b term on z-direction from a rotation matrix

2. compute current position error in z-direction

3. compute current vertical error in z-direction

```
4. compute target acceleration in z-direction

5. generate the thrust command

6. constrain the thrust command within a set of bounds (0.1, MAX_THRUST)
```

C++: lines 170 to 207 in QuadControl.cpp

2.1.4 Implement lateral position control

Steps of lateral position control as follows,

```
1. compute current position error in all direction in world frame

2. compute current velocity error in all direction in world frame

3. generate the target acceleration command

4. constrain the target acceleration within a set of bounds (-maxAccelXY, maxAccelXY)
```

C++: lines 210 to 251 in QuadControl.cpp

2.1.5 Implement yaw control

Steps of yaw control as follows,

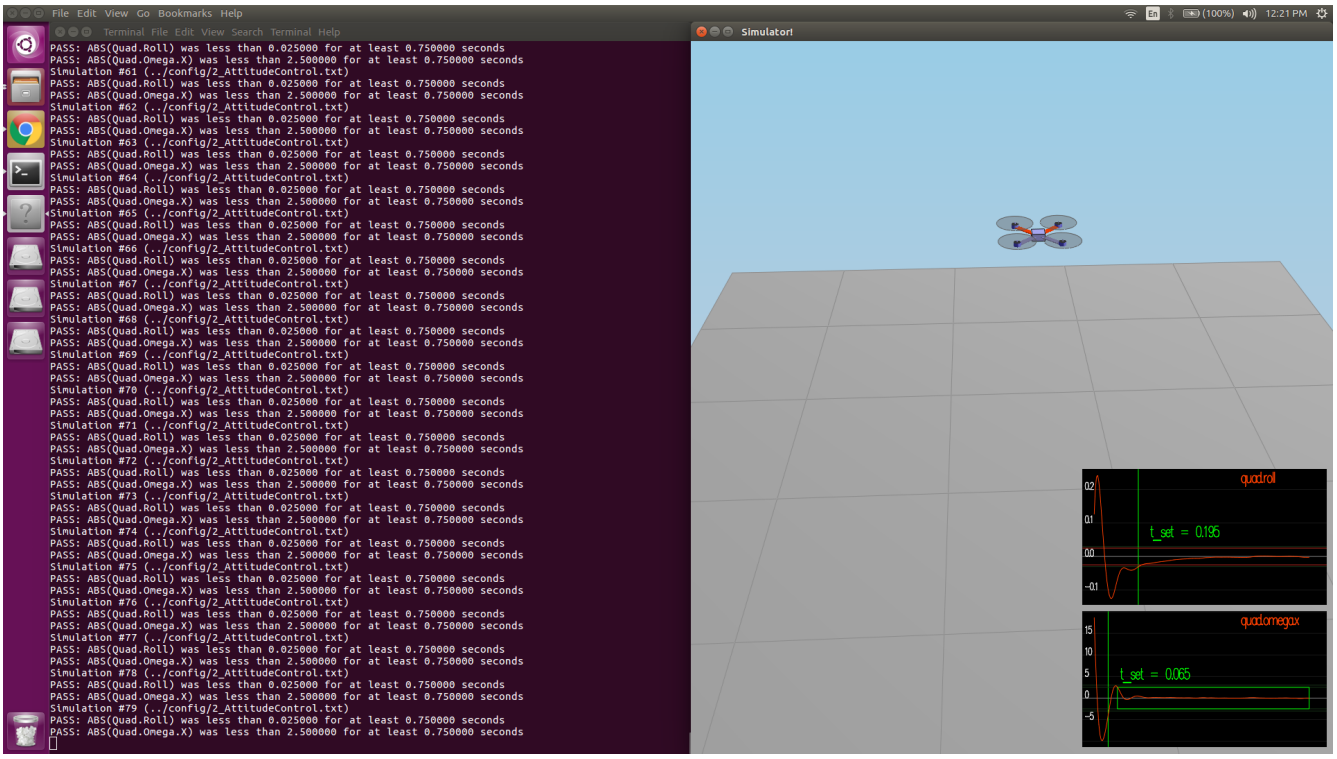
```
1. compute current yaw error

2. generate the target yaw rate command
```

C++: lines 255 to 274 in QuadControl.cpp

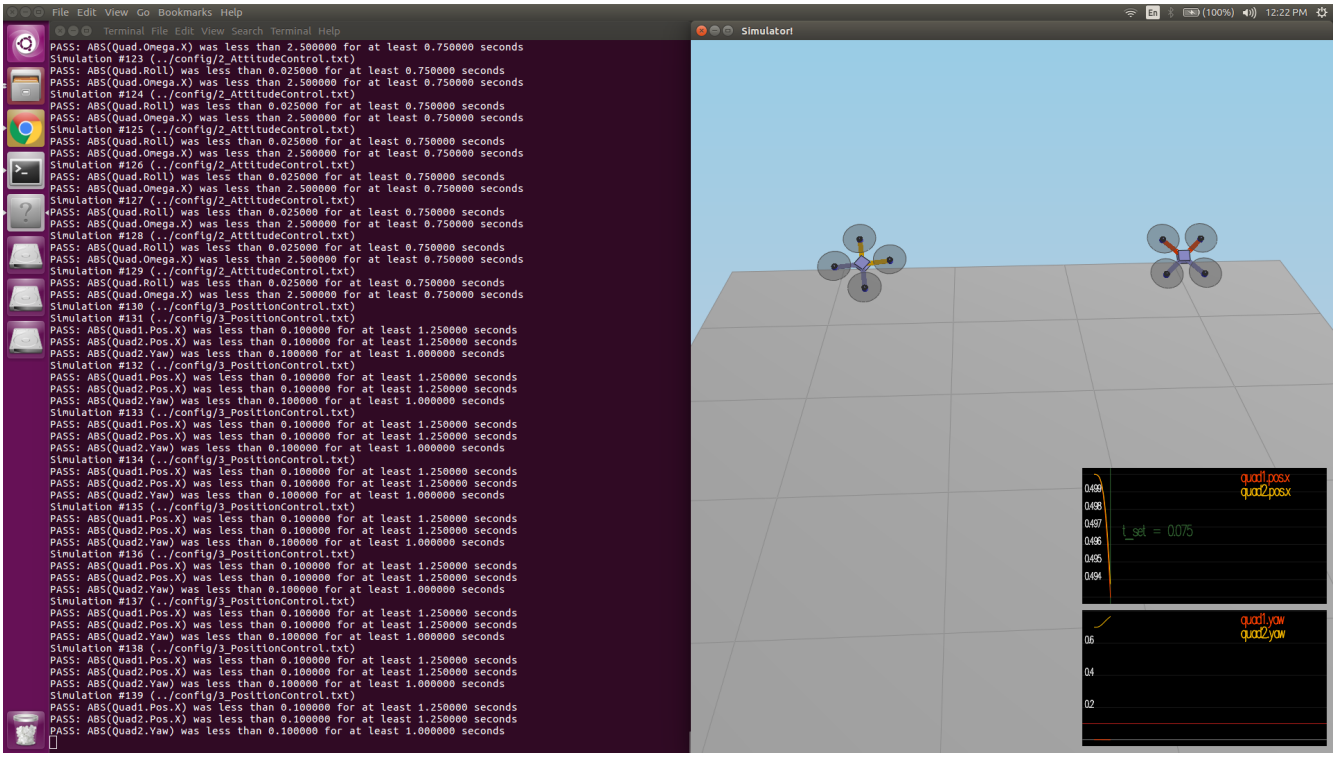
2.2 Flight Evaluation

The quadrotors controlled by the trajectory following controller successfully fly around the desired 3D trajectory within the desired limits in C++ udacity simulator environment.

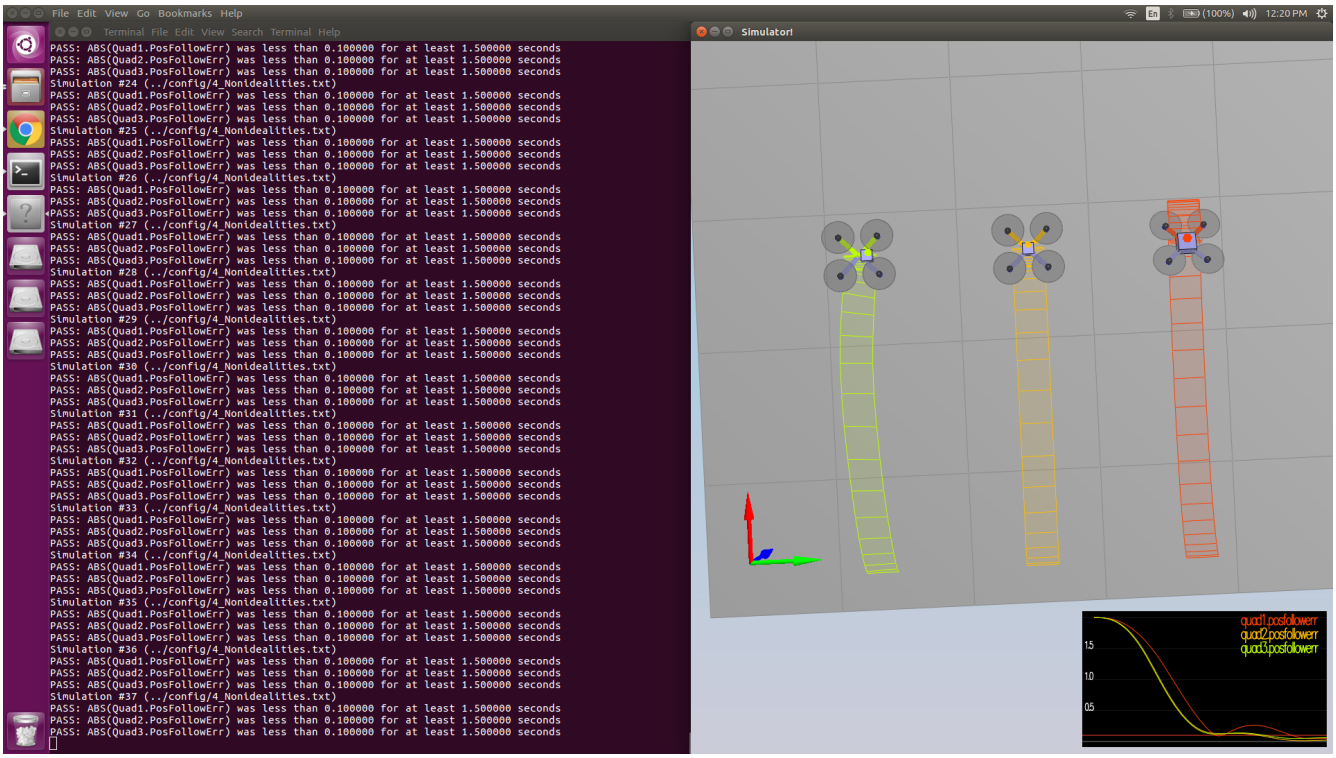
Scenerio 2:



Scenerio 3:



Scenerio 4:



Scenerio 5: