Hovering Control of a Quadrotor

MEE 4411/5411 Project 2, Phase 1

“Due:” Thursday, November 16

1 System Description

1.1 Quadrotor Platform

For this project we will be using the CrazyFlie 2.0 platform made by Bitcraze, shown in Fig. 1. The CrazyFlie has a motor to motor (diagonal) distance of 92 mm, and a mass of 30 g, including a battery. An onboard microcontroller allows low-level control and estimation to be done onboard the robot. An onboard IMU provides feedback of angular velocities and accelerations.

1.2 Motion Capture Systems

Motion Capture System provides a state estimate for the quadrotor, which is nearly ground truth. The motion capture system offers three important features. First, *it is fast*: the system can be run at 100 Hz or more. Second, *it is precise*: experimental tests show that the deviations of position estimates for single static markers are on the order of 50 μm which is well beyond the requirements for flight. Lastly, *it is robust*: the software assumes that markers in models are rigidly attached which enables the system to maintain tracking even if all but one camera are occluded. Using this software, tracking of quadrotors is rarely lost, even during extreme situations such as fast maneuvers (speeds of 3.5 m/s, accelerations of 15 m/s^2 and angular speeds of 1000 °/s).

1.3 Software and Integration

Position control and other high level commands are computed in Matlab at 100 Hz and sent to the robot via the CrazyRadio (2.4GHz). Attitude control is performed onboard using the microcontroller

2 Modeling

2.1 Coordinate Systems and Reference frames

The coordinate systems and free body diagram for the quadrotor are shown in Fig. 1. The inertial frame, $\mathcal{A}$, is defined by the triad $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ with $\mathbf{a}_3$ pointing upward. The body frame, $\mathcal{B}$, is attached to the center of mass of the quadrotor with $\mathbf{b}_1$ coinciding with the preferred forward direction and $\mathbf{b}_3$ perpendicular to the plane of the rotors pointing vertically up during perfect hover (see Fig. 1). These vectors are parallel to the principal axes. The center of mass is C. Rotor 1 is a distance L away along $\mathbf{b}$, 2 is L away along $\mathbf{b}_2$, while 3 and 4 are similarly L away along the negative $\mathbf{b}_1$ and $\mathbf{b}_2$ respectively.

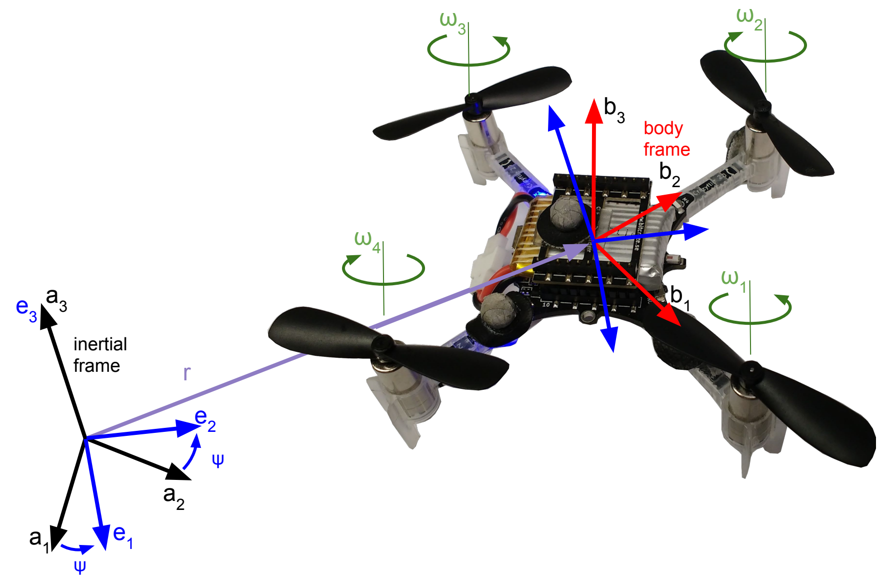


Figure 1: The CrazyFlie 2.0 robot. Note the reflective motion capture markers attached. A pair of motors spins counter clockwise while the other pair spins clockwise, such that when all propellers spin at the same speed, the net torque in the yaw direction is zero. The pitches on the corresponding propellers are reversed so that the thrust is always pointing in the $\mathbf{b_3}$ direction for all propellers. Shown also is the transformation from the inertial frame to the body-fixed frame. First a rotation by ψ around the $\mathbf{a_3}$ axis (leading to coordinate system $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$) is performed, followed by a translation $\mathbf{r}$ to the center of mass C of the robot. Subsequent rotations by ϕ and θ generate the final body-fixed coordinate system $\mathcal{B}$, where the axes $\mathbf{b_1}$ and $\mathbf{b_2}$ are aligned with the arms, and $\mathbf{b_3}$ is perpendicular to them.

2.2 Inertial Properties

Since $\mathbf{b}_i$ are principal axes, the inertia matrix referenced to the center of mass along the $\mathbf{b}_i$ reference triad, I, is a diagonal matrix. In practice, the three moments of inertia can be estimated by weighing individual components of the quadrotor and building a physically accurate model in SolidWorks. The key parameters for the rigid body dynamics for the CrazyFlie platform are as follows:

(a) mass: $m = 0.030\,\mathrm{kg}$;

(b) the distance from the center of mass to the axis of a motor: $L = 0.046\,\mathrm{m}$; and

(c) the components of the inertia dyadic using $\mathbf{b}_i$ as the SRT:

$$[I_C]^{\mathbf{b}_i} = \begin{bmatrix} 1.43\times10^{-5} & 0 & 0 \\ 0 & 1.43\times10^{-5} & 0 \\ 0 & 0 & 2.89\times10^{-5} \end{bmatrix}.$$

2.3 Motor Model

Each rotor has an angular speed ω_i and produces a vertical force F_i according to

$$F_i = k_F\omega_i^2. \tag{1}$$

Experimentation with a fixed rotor at steady-state shows that $k_F \approx 6.11\times10^{-8}\,\mathrm{N/rpm^2}$. The rotors also produce a moment according to

$$M_i = k_M\omega_i^2. \tag{2}$$

The constant, k_M, is determined to be about $1.5\times10^{-9}\,\mathrm{Nm/rpm^2}$ by matching the performance of the simulation to the real system.

Data obtain from system identification experiments suggest that the rotor speed is related to the commanded speed by a first-order differential equation

$$\dot{\omega}_i = k_m(\omega_i^{\mathrm{des}} - \omega_i).$$

This motor gain, k_m, is found to be about $20\,\mathrm{s}^{-1}$ by matching the performance of the simulation to the real system. The desired angular velocities, ω_i^{des}, are limited to a minimum and maximum value determined through experimentation.

However, as a first approximation, we can assume the motor controllers to be perfect and the time constant k_m associated with the motor response to be arbitrarily small. In other words, we can assume the actual motor velocities ω_i are equal to the commanded motor velocities, ω_i^{des}.

2.4 Rigid Body Dynamics

Kinematics We will use $Z-X-Y$ Euler angles to model the rotation of the quadrotor in the world frame. To get from $\mathcal{A}$ to $\mathcal{B}$, we first rotate about $\mathbf{a}_3$ through the the yaw angle, ψ, to get the triad $\mathbf{e}_i$. A rotation about the $\mathbf{e}_1$ through the roll angle, ϕ gets us to the triad $\mathbf{f}_i$ (not shown in the figure). A third pitch rotation about $\mathbf{f}_2$ through θ results in the body-fixed triad $\mathbf{b}_i$. You will need the rotation matrix for transforming components of vectors in $\mathcal{B}$ to components of vectors in $\mathcal{A}$:

$$^{\mathcal{A}}[R]_{\mathcal{B}} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}. \tag{3}$$

We will denote the components of angular velocity of the robot in the body frame by ω_1, ω_2, and ω_3:

$$^{\mathcal{A}}\omega_{\mathcal{B}} = \omega_1\mathbf{b}_1 + \omega_2\mathbf{b}_2 + \omega_3\mathbf{b}_3.$$

These values are related to the derivatives of the roll, pitch, and yaw angles according to

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{4}$$

Newton's Equations of Motion Let $\mathbf{x}$ denote the position vector of C in $\mathcal{A}$. The forces on the system are gravity, in the $-\mathbf{a}_3$ direction, and the forces from each of the rotors, F_i, in the $\mathbf{b_3}$ direction. The equations governing the acceleration of the center of mass are

$$m\ddot{\mathbf{x}} = \begin{bmatrix}0\\0\\-mg\end{bmatrix} + R\begin{bmatrix}0\\0\\F_1+F_2+F_3+F_4\end{bmatrix}. \tag{5}$$

We will define our first input u_1 to be

$$u_1 = \Sigma_{i=1}^{4} F_i.$$

Euler's Equations of Motion In addition to forces, each rotor produces a moment perpendicular to the plane of rotation of the blade, M_i. Rotors 1 and 3 rotate in the $-\mathbf{b_3}$ direction while 2 and 4 rotate in the $+\mathbf{b_3}$ direction. Since the moment produced on the quadrotor is opposite to the direction of rotation of the blades, M_1 and M_3 act in the $\mathbf{b_3}$ direction while M_2 and M_4 act in the $-\mathbf{b_3}$ direction. L is the distance from the axis of rotation of the rotors to the center of mass of the quadrotor.

The angular acceleration determined by the Euler equations is

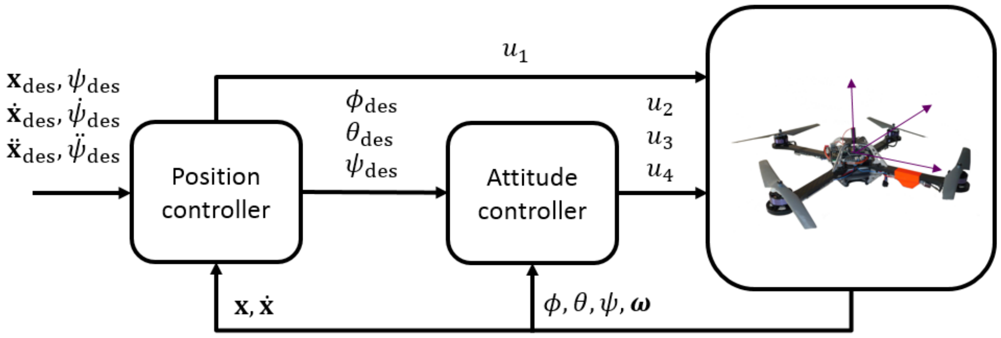
$$I\begin{bmatrix}\dot{\omega}_1\\ \dot{\omega}_2\\ \dot{\omega}_3\end{bmatrix} = \begin{bmatrix}L(F_2-F_4)\\ L(F_3-F_1)\\ M_1-M_2+M_3-M_4\end{bmatrix} - \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} \times I \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}. \tag{6}$$

We can rewrite this as:

$$I\begin{bmatrix}\dot{\omega}_1\\ \dot{\omega}_2\\ \dot{\omega}_3\end{bmatrix} = \begin{bmatrix}0 & L & 0 & -L\\ -L & 0 & L & 0\\ \gamma & -\gamma & \gamma & -\gamma\end{bmatrix}\begin{bmatrix}F_1\\F_2\\F_3\\F_4\end{bmatrix} - \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} \times I \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}. \tag{7}$$

where $\gamma = \frac{k_M}{k_F}$ is the relationship between lift and drag given by Equations (1-2). Accordingly, we will define our second set of inputs to be the vector of moments given by:

$$\begin{bmatrix}u_2\\u_3\\u_4\end{bmatrix} = \begin{bmatrix}0 & L & 0 & -L\\ -L & 0 & L & 0\\ \gamma & -\gamma & \gamma & -\gamma\end{bmatrix}\begin{bmatrix}F_1\\F_2\\F_3\\F_4\end{bmatrix}.$$



$\mathbf{x}_{\text{des}}, \psi_{\text{des}}$
$\dot{\mathbf{x}}_{\text{des}}, \dot{\psi}_{\text{des}}$
$\ddot{\mathbf{x}}_{\text{des}}, \ddot{\psi}_{\text{des}}$
Position controller
u_1
ϕ_{des}
θ_{des}
ψ_{des}
Attitude controller
u_2
u_3
u_4
$\mathbf{x}, \dot{\mathbf{x}}$
$\phi, \theta, \psi, \boldsymbol{\omega}$

Figure 2: The position and attitude control loops.

3 Robot Controllers

3.1 The Nominal State

Our controllers are derived by linearizing the equations of motion and motor models (5 – 2) at an operating point that corresponds to the nominal hover state, $\mathbf{x} = \mathbf{x}_0$, $\theta = \phi = 0$, $\psi = \psi_0$, $\dot{\mathbf{x}} = 0$, and $\dot{\phi} = \dot{\theta} = \dot{\psi} = 0$, where the roll and pitch angles are small ($c\phi \approx 1$, $c\theta \approx 1$, $s\phi \approx \phi$, and $s\theta \approx \theta$). At this state the lift from the propellers is given by:

$$F_{i,0} = \frac{mg}{4},$$

The nominal values for the inputs at hover are $u_{1,0} = mg$, $u_{2,0} = u_{3,0} = u_{4,0} = 0$.

Linearizing (5), we get:

$$\begin{aligned} \ddot{x} &= g(\theta \cos\psi_0 + \phi \sin\psi_0) \\ \ddot{y} &= g(\theta \sin\psi_0 - \phi \cos\psi_0) \\ \ddot{z} &= \frac{1}{m} u_1 - g \end{aligned} \tag{8}$$

Linearizing (6), we get:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = I^{-1} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{9}$$

If we assume the rotor craft is symmetric so $I_{xx} = I_{yy}$, we get:

$$\begin{aligned} \ddot{\phi} &= \frac{u_2}{I_{xx}} = \frac{L}{I_{xx}}(F_2 - F_4) \\ \ddot{\theta} &= \frac{u_3}{I_{yy}} = \frac{L}{I_{yy}}(F_3 - F_1), \\ \ddot{\psi} &= \frac{u_4}{I_{zz}} = \frac{\gamma}{I_{zz}}(F_1 - F_2 + F_3 - F_4). \end{aligned}$$

In other words, the equations of motion are decoupled in terms of angular accelerations. Each component of angular acceleration depends only on the appropriate component of $\mathbf{u}_2$.

3.2 Position and Attitude Control

The control problem is to determine the four inputs, $\{u_1, u_2, u_3, u_4\}$ required to hover or to follow a desired trajectory, $\mathbf{z}^{des}$. As shown in Figure 2, we will use errors in the robot's position to drive a position controller from (8) which directly determines u_1. The model in (8) also allows us to derive a desired orientation. The attitude controller for this orientation is derived from the model in (9).

The inner attitude control loop uses onboard accelerometers and gyros to control the roll, pitch, and yaw and runs at approximately $500\,\text{Hz}$, while the outer position control loop uses estimates of position and velocity of the center of mass to control the trajectory in three dimensions at 100-200 Hz.

Attitude Control We now present a proportional plus derivative (PD) attitude controller to track a trajectory in $SO(3)$ specified in terms of a desired roll, pitch and yaw angle. Since our development of the controller will be based on linearized equations of motion, the attitude must be close to the nominal hover state where the roll and pitch angles are small.

Near the nominal hover state the proportional plus derivative control laws take the form:

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = I \begin{bmatrix} k_{p,\phi}(\phi^{\text{des}} - \phi) + k_{d,\phi}(\dot{\phi}^{\text{des}} - \dot{\phi}) \\ k_{p,\theta}(\theta^{\text{des}} - \theta) + k_{d,\theta}(\dot{\theta}^{\text{des}} - \dot{\theta}) \\ k_{p,\psi}(\psi^{\text{des}} - \psi) + k_{d,\psi}(\dot{\psi}^{\text{des}} - \dot{\psi}) \end{bmatrix} \tag{10}$$

Position Control In the next subsection, we will present a position control method that uses the roll and pitch angles as inputs to drive the position of the quadrotor. The position control algorithm will determine the desired roll and pitch angles, θ^{des} and ϕ^{des}, which can be used to compute the desired speeds from (10). The hover controller is used for station-keeping or maintaining the position at a desired position vector, $\mathbf{r}_0$. The desired yaw angle, is specified independently and we will assume it to be a constant, ψ_0.

3.3 Hover Controller

For hovering, $\mathbf{x}^{\text{des}}(t) = \mathbf{x}_0$ and $\psi^{\text{des}}(t) = \psi_0$. The command accelerations, $\ddot{\mathbf{x}}^{\text{des}}$, are calculated from a proportional plus derivative (PD) controller. Define the position error in terms of components using the standard reference triad $\mathbf{a}_i$ by:

$$e_x(t) = (x^{\text{des}}(t) - x(t)).$$

Note that $x^{\text{des}}(t) = x_0$ is constant. The error for y and z is defined analogously. In order to guarantee that this error goes exponentially to zero, we require

$$(\ddot{x}^{\text{des}} - \ddot{x}) + k_{d,i}(\dot{x}^{\text{des}} - \dot{x}) + k_{p,i}(x^{\text{des}} - x) = 0, \tag{11}$$

where $\dot{x}^{\text{des}} = \ddot{x}^{\text{des}} = 0$ for hover.

From (8) we can obtain the relationship between the desired accelerations and roll and pitch angles

$$\begin{aligned} \ddot{x} &= g(\theta^{\text{des}} \cos\psi_0 + \phi^{\text{des}} \sin\psi_0) \\ \ddot{y} &= g(\theta^{\text{des}} \sin\psi_0 - \phi^{\text{des}} \cos\psi_0) \\ \ddot{z} &= \frac{1}{m} u_1 - g. \end{aligned}$$

For hover, the last equation yields:

$$u_1 = mg + m\ddot{z} = mg - m\left(k_{d,3}\dot{z} + k_{p,3}(z - z_0)\right) \tag{13}$$

The other two equations can be used to compute the desired roll and pitch angles for the attitude controller:

$$\phi^{\text{des}} = \frac{1}{g}(\ddot{x}\sin\psi_T - \ddot{y}\cos\psi_T) \tag{14a}$$

$$\theta^{\text{des}} = \frac{1}{g}(\ddot{x}\cos\psi_T + \ddot{y}\sin\psi_T) \tag{14b}$$

The desired roll and pitch velocities are taken to be zero.

$$\dot{\phi}^{\text{des}} = 0 \tag{15a}$$

$$\dot{\theta}^{\text{des}} = 0 \tag{15b}$$

Since the yaw, $\psi(t)$ is prescribed independently by the trajectory generator, we get:

$$\psi^{\text{des}} = \psi_0 \tag{16a}$$

$$\dot{\psi}^{\text{des}} = 0 \tag{16b}$$

These equations provide the setpoints for the attitude controller in (10).

Note that the attitude controller must run an order of magnitude faster than the position control loop in order for this to work. In practice as discussed in [1], the position controller runs at $100\,\text{Hz}$, while the inner attitude control loop runs at $500\,\text{Hz}$.

The controller design above was based on the decoupling of the position and attitude control subsystems as motivated in Fig. 2. This is only valid when the attitude control loop is much faster than the position control loop permitting us to use, what is called in the control theory literature (see [2]), a *backstepping* approach to controller design. This is roughly the approach that justifies the development above.

4 Project Work

4.1 Tasks

You will simulate the quadrotor dynamics and control using the matlab simulator posted on the meam620 wiki. The simulator relies on the numerical solver `ode45`, details about this solver can be easily found online. You don't need to be an expert in numerical methods, but it would be beneficial if you know the basics of how ode solvers work. Your tasks include:

1. **Trajectory Generator**
   You will use the `step.m` trajectory function in this Phase of the project. This sets the desired position (and yaw angle) of the robot to a constant value. You should go through and change these values as you tune your control gains (see below).

2. **Controller**
   The main task is to implement a controller in file `controller.m` that makes sure that your quadrotor settles at the desired position. This controller will be used again in the following phases to follow moving trajectories.

The controller takes in a struct `qd`, current time `t`, and quadrotor parameters `params` and outputs thrust `F` and moments `M`.

As you can tell from above, there are many different control gains that you need to tune. In total, there are 12 gains (a pair of P and D gains for each of the six states $x, y, z, \phi, \theta, \psi$). One strategy for this is to try to isolate the gains as much as possible. For example, you can tune $k_{p,z}$ and $k_{d,z}$ without considering any other gains. To do this, just set the desired point to have $x_0 = y_0 = \psi_0 = 0$ and set $z_0 > 0$. This will cause the set point to be above the robot so it will have to move up to get there. Similarly, if you set $x_0 = y_0 = z_0 = 0$ you can tune the ψ gains on their own.

Unfortunately, since the robot needs to turn in order to translate in the x and y directions, the rotational gains will be coupled to the planar translation gains. However, if you set $y_0 = z_0 = \psi_0 = 0$ and set $x_0 > 0$ then you can tune only the x and θ gains. Similarly, you can then tune the y and ϕ gains (hint: think about the symmetry of the quadrotor!).

I highly recommend running the simulator many times with different gains and different set points to make sure your gains are robust. Take small steps, take big steps, step in a single direction, step in multiple directions, etc. Having a good set of gains for the hover controller will make following a trajectory in Phase 3 much easier!

4.2 Simulator

The quadrotor simulator comes with the student code. Before implementing your own functions, you should first try running `runsim.m` in your matlab. If you see a quadrotor falling from position $(0, 0, 0)$ then the simulator works on your computer and you may continue with other tasks. This is because the outputs of `controller.m` are all zeros, thus no thrust generated.

When you have the basic version of your trajectory generator and controller done, you will be able to see the quadrotor flying in the space with proper roll, pitch, and yaw,leaving trails of desired and actual position behind. The desired position is color-coded blue and the actual position is red. After the simulation finishes, two plots of position and velocity will be generated to give you an overview of how well your trajectory generator and controller are doing. Note that you will not be graded on these plots but by the automatic grader.

4.3 Submission

When you are finished you may submit your code by putting it all into a single `.zip` file and emailing it to pdames@temple.edu with subject line "MEE4411 Project 2.1" (clicking the previous link will automatically start a correctly formatted email for you). Prof. Dames will test your submission and send you a list of items to fix. You are encouraged to update your code to fix any bugs, however each student is limited to 3 submissions per phase of the project.

References

[1] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro uav testbed," *IEEE Robotics and Automation Magazine*, 2010.

[2] H. Khalil, *Nonlinear Systems*. Prentice Hall, 1996.