Autonomous Drone Racing Project Course

Quadrotor Introduction Handout

1. Overview

Quadrotors are flying vehicles that have a rigid frame with four independently controlled motors, as shown in Figure 1. By changing the motor speeds, we can make the quadrotor perform different maneuvers such as moving towards a specified position or performing a flip. As compared to conventional fixed-wing flying vehicles, quadrotors are more compact and can perform vertical takeoff and landing, which makes them suitable for a wide range of applications. Some example applications include search and rescue, industrial inspection, and entertainment.

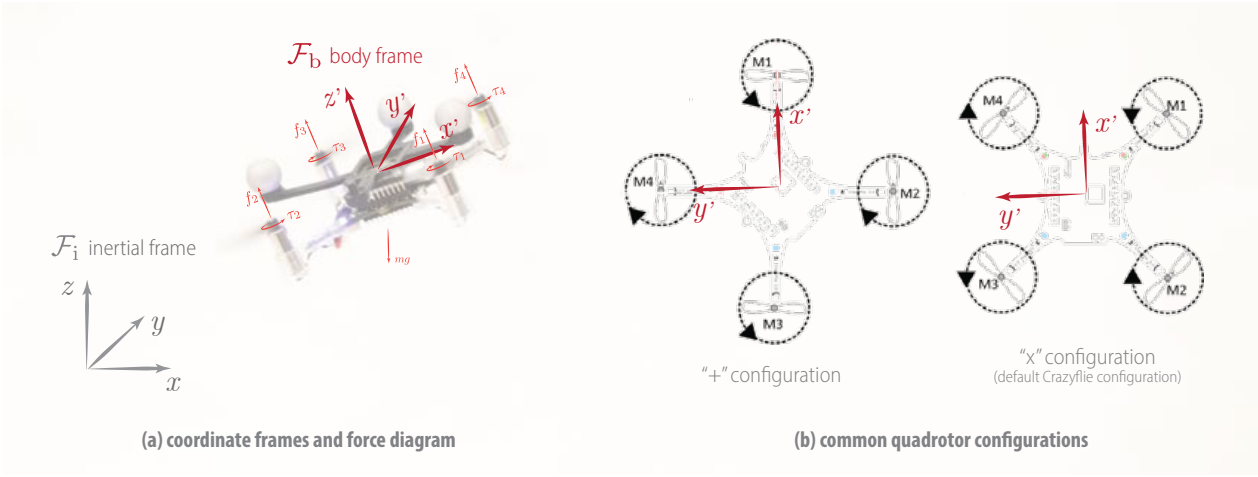

Figure 1: Illustration of a quadrotor: **(a)** A quadrotor is a flying vehicle equipped with a rigid frame and four motors that are controlled independently. Typically, two of the motors spin in the clockwise direction and two spin in the counterclockwise direction. Each motor generates a thrust f_i aligning with the axis of rotation and a torque τ_i about its axis of rotation. Different quadrotor maneuvers are realized by varying the speeds of the four independently controlled motors. As an example, hovering motion can be achieved by setting the motor speeds such that $f_i = mg/4$ for all $i = \{1, 2, 3, 4\}$, where m is the mass of the quadrotor and g is the acceleration due to gravitation. Generally, the translational acceleration of the quadrotor is determined by the attitude of the quadrotor, which is in turn controlled via the individual motors. There are two essential coordinate frames pertinent to every quadrotor estimation, planning, and control framework: *(i)* the inertial frame or the "world frame" and *(ii)* the body frame that is fixed to the quadrotor. The translation and rotation of the body frame relative to the world frame together characterize the pose of the quadrotor that is estimated based on sensor measurements and is used for planning and control. **(b)** There are two common configurations or body frame conventions for a quadrotor. The "+" configuration aligns have its x and y axes aligned with the frame, while the "×" configuration offsets by 45°. From a mathematical point of view, there are no major differences between the two configurations. Practically, the latter could be preferable for certain cases (e.g., mounting a front-facing camera below the frame. The Crazyflie platform used in this course follows the "×" configuration for its onboard estimation and control algorithm implementations.

2. Quadrotor Dynamics

The motion of a quadrotor is fully driven by the four motors—two spinning clockwise and two spinning counterclockwise, as illustrated in Figure 1. Quadrotors are underactuated systems as it has six degrees of freedom but only four independent inputs. We briefly summarize the dynamics

of a quadrotor system in this section, which are divided into two parts: *(i)* motor dynamics and *(ii)* rigid body dynamics.

2.1 Motor Dynamics

Each motor is equipped with a propeller that is firmly attached to the motor's shaft. As the motor rotates, the propeller produces a force, known as thrust f_i, that is aligned with the motor's axis of rotation, as well as a reaction torque τ_i about the axis of rotation. The thrust and torque produced by each motor are determined by their respective characteristic curves, which are often identified empirically. A basic motor model takes the following form:

$$f_i = c_t \Omega_i^2, \tag{1}$$

$$\tau_i = c_d \Omega_i^2 + \underbrace{J_m \dot{\Omega}_i}_{\substack{\text{inertia} \\ \text{moment}}}, \tag{2}$$

where Ω_i denotes the motor rotational speed represented as revolutions per minute (RPM), c_t and c_d respectively are the lift and drag coefficients associated with the particular propeller, and J_m is the inertia coefficient. The inertia moment of the motor is often negligible; both motor thrust and torque are roughly proportional to Ω_i^2. The lift and drag coefficients of the default Crazyflie setup are 3.1582×10^{-10} N/RPM^2 and 7.9379×10^{-12} Nm/RPM^2, respectively.

How do we regulate the speed of the motor? This is controlled by the pulse-width modulation (PWM). Intuitively, the PWM controls the average power delivered to the motor by varying the duration and frequency of switching on/off the voltage supply. For the DC motor used on the Crazyflie platform, we can relate the PWM signal μ and the RMP as follows:

$$\Omega_i = \alpha \mu_i + \beta, \tag{3}$$

where the PWM signal μ_i is a 16-bit number between 0 and 65535, and $\alpha = 0.2685$ and $\beta = 4070.3$ are two constants specific to the motor.

2.2 Rigid Body Dynamics

The motor thrust and torque collectively generate thrust forces and torques that enable different quadrotor maneuvers. The collective thrust generated by the motors is always aligned with the body z'-axis:

$$F = \sum_{i=1}^{4} f_i = c_t \sum_{i=1}^{4} \Omega_i^2. \tag{4}$$

Following the "×" configuration defined in Figure 1, we can express the body torques as a function of the motor rotation rates as follows:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{x'} \\ \tau_{y'} \\ \tau_{z'} \end{bmatrix} = \begin{bmatrix} -c_{\tau_{xy}} & -c_{\tau_{xy}} & +c_{\tau_{xy}} & +c_{\tau_{xy}} \\ -c_{\tau_{xy}} & +c_{\tau_{xy}} & +c_{\tau_{xy}} & -c_{\tau_{xy}} \\ -c_{\tau_z} & +c_{\tau_z} & -c_{\tau_z} & +c_{\tau_z} \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{5}$$

where $c_{\tau_{xy}} = l c_t / \sqrt{2}$ and $c_{\tau_z} = c_d$ and l is the arm length of the quadrotor (i.e., the distance from the centre to the motor shaft). Note that the reaction torque generated by individual motors is opposite to their rotation direction (Figure 1).

The collective thrust F and the body torque $\boldsymbol{\tau}$ are the inputs to the rigid body dynamics of the quadrotor system. A continuous-time model of the quadrotor is

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{6}$$
$$m\dot{\mathbf{v}} = m\mathbf{g} + \mathbf{R}\mathbf{f}, \tag{7}$$
$$\dot{\mathbf{R}} = \mathbf{R}\mathbf{S}(\boldsymbol{\omega}), \tag{8}$$
$$\mathbf{J}\dot{\boldsymbol{\omega}} = \mathbf{J}\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}, \tag{9}$$

where $\mathbf{p} = [p_x, p_y, p_z]^T$ and $\mathbf{v} = [v_x, v_y, v_z]^T$ are the translational position and velocity of the center of mass (COM) of the quadrotor, $\mathbf{R}_b^i$ is the rotation matrix from the body frame to the inertial frame, $\boldsymbol{\omega} = [p, q, r]^T$ is the body angular velocity, m is the mass of the quadrotor, $\mathbf{J} = \text{diag}(J_{xx}, J_{yy}, J_{zz})$ is the inertia matrix, $\mathbf{S}(\cdot)$ denotes the skew-symmetric matrix mapping of a vector, and $\mathbf{g} = [0, 0, -g]^T$ is the gravitational force vector.

Note that, there are different ways of parametrizing the orientation of a quadrotor (or a rigid body in general). Euler angles and quaternions are two common parameterizations. The former is easier to interpret intuitively but has singularity issues, while the latter is easier to work with computationally but is an over-parametrization scheme. To facilitate the controller design discussion in the following section, we briefly review how the rotational dynamics can be expressed in terms of Euler angles; one could similarly express the rotational dynamics in quaternions.

The Euler angles refer to the roll ϕ, pitch θ, and yaw ψ angles, which respectively correspond to rotations about the x, y, and z axes. One can uniquely define a rotation matrix through these angles following a specified sequence of rotation. As an example, a Z-Y-X sequence (i.e., first rotating about the z axis by ψ, then about the y axis by θ, and lastly about the x axis by ϕ) results in the following rotation matrix from the inertial frame to the body frame:

$$\mathbf{R}_i^b = \begin{bmatrix} \cos\theta\,\cos\psi & \cos\theta\,\sin\psi & -\sin\theta \\ \sin\phi\,\sin\theta\,\cos\psi - \cos\phi\,\sin\psi & \sin\phi\,\sin\theta\,\sin\psi + \cos\phi\,\cos\psi & \sin\phi\,\cos\theta \\ \cos\phi\,\sin\theta\,\cos\psi + \sin\phi\,\sin\psi & \cos\phi\,\sin\theta\,\sin\psi - \sin\phi\,\cos\psi & \cos\phi\,\cos\theta \end{bmatrix}. \tag{10}$$

Note that, the rotation matrix from the body frame to the inertial frame is $\mathbf{R}_b^i = (\mathbf{R}_i^b)^{-1} = (\mathbf{R}_i^b)^T$.

We can relate the body angular velocity and the time-derivatives of Euler angles as follows:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}}_{\mathbf{W}_1} \underbrace{\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}}_{\dot{\boldsymbol{\psi}}} = \mathbf{W}_1\,\dot{\boldsymbol{\psi}}, \tag{11}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}}_{\mathbf{W}_2} \underbrace{\begin{bmatrix} p \\ q \\ r \end{bmatrix}}_{\omega} = \mathbf{W}_2\,\boldsymbol{\omega} \quad \text{for } \theta \neq \frac{\pi}{2}, \tag{12}$$

The rotational dynamics can then be written as

$$\ddot{\boldsymbol{\psi}} = \dot{\mathbf{W}}_2\,\boldsymbol{\omega} + \mathbf{W}_2\,\dot{\boldsymbol{\omega}} = \dot{\mathbf{W}}_2\,\boldsymbol{\omega} + \mathbf{W}_2\,\mathbf{J}^{-1}\left(\mathbf{J}\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}\right), \tag{13}$$

where $\ddot{\boldsymbol{\psi}} = [\ddot{\phi}, \ddot{\theta}, \ddot{\psi}]^T$ and $\boldsymbol{\omega} = \mathbf{W}_1\,\dot{\boldsymbol{\psi}}$.

Let $\mathbf{u} = [F, \tau_{x'}, \tau_{y'}, \tau_{z'}]^T$ be the input and $\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \boldsymbol{\psi}^T, \dot{\boldsymbol{\psi}}^T]^T$ be the state. We can write the translational and rotational dynamics in the state-space model form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{14}$$

One could also define $\mathbf{u}$ as the motor rotation rates $[\Omega_1^2, \Omega_2^2, \Omega_3^2, \Omega_4^2]^T$ or the PWM signals $[\mu_1^2, \mu_2^2, \mu_3^2, \mu_4^2]^T$ if the lower-level motor dynamics is of interest for the particular problem.

The parameters for the Crazyflie platform are summarized in Table 1.

Table 1: Parameters of the Crazyflie platform.

mass (m)	0.27 kg
arm length (l)	42.0×10^{-3} m
moment of inertia (J_{xx}, J_{yy}, J_{zz})	$(1.395 \times 10^{-5}, 1.436 \times 10^{-5}, 2.173 \times 10^{-5})$ kg m^2

2.3 So, how do things fit together?

The PWM of individual motors regulates their rotational speeds, and the motor RPMs determine the thrust and reaction torque produced. Variations in thrust and torque among individual motors result in body torques, influencing the quadrotor's attitude. The attitude and collective thrust of the quadrotor together determine the direction and magnitude of translational acceleration, respectively, which induces translational motion.

3. Quadrotor Control

We consider a trajectory-tracking problem and present a typical controller to demonstrate the idea. The trajectory consists of $[\mathbf{p}_{\text{des}}, \psi_{\text{des}}]$ and is often given by a trajectory generator to realize a higher-level objective (e.g., navigating through an environment).

The controller design of the quadrotor system addresses the question: how do we go from the position trajectory to the low-level motor commands such that the COM of the quadrotor accurately follows the desired trajectory? One approach is to break the problem into three subproblems, resulting in a nested control architecture (Figure 2). There are three key components in the overall control system:

1. **High-level position control:** Mapping desired position (and optionally yaw) trajectories to desired attitude (i.e., roll, pitch, and yaw angles) and desired collective thrust
2. **Mid-level attitude control:** Mapping desired attitude to desired body torques
3. **Low-level motor control:** Computing motor speeds based on desired body torques and desired collective thrust

A default controller equipped onboard the Crazyflie is the Mellinger controller (Mellinger and Kumar, 2011). We summarize the key equations here; you are encouraged to check out the onboard implementation of the Crazyflie platform to better understand how it works.

3.1 Position Control

The desired collective thrust and attitude are computed to drive the position error of the quadrotor to zero. We define the position error as

$$\mathbf{e}_{\mathbf{p}} = \mathbf{p} - \mathbf{p}_{\text{des}}. \tag{15}$$

A possible way for computing the desired translational acceleration in the inertial frame is based on a PD control law:

$$\mathbf{f}_{\text{des}} = \underbrace{-k_p \mathbf{e}_{\mathbf{p}} - k_d \dot{\mathbf{e}}_{\mathbf{p}}}_{\text{feedback}} - \underbrace{m\mathbf{g}}_{\text{feedforward}}, \tag{16}$$

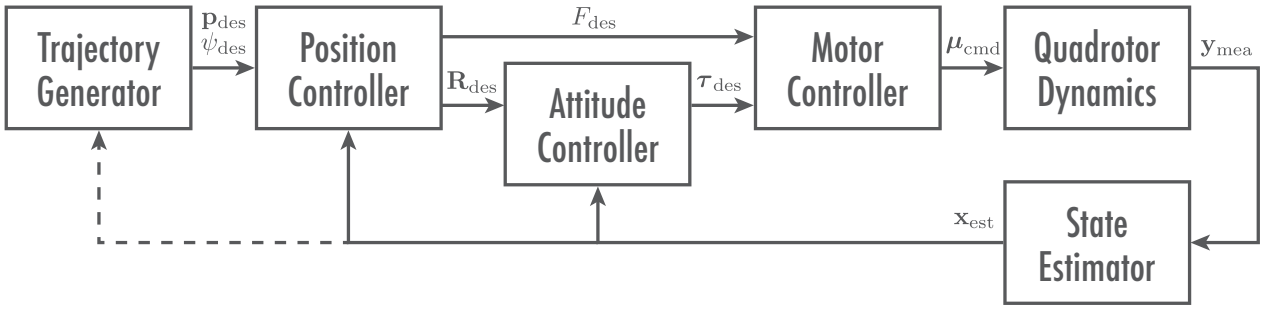

Figure 2: Block diagram of a typical quadrotor system: The trajectory generator provides the desired position and yaw trajectories to fulfill a high-level task. The controller can be decomposed into three levels: *(i)* the position controller computes the desired collective thrust and attitude to minimize the position and yaw error, *(ii)* the attitude controller computes the desired body torques based on the attitude error, and *(iii)* the motor controller then computes motor commands to realize the desired collective thrust and body torques. A state estimator is used to bookkeep the sensor measurements (e.g., IMU measurements, motion capture system measurements, ultra-wideband signals, RGB-D camera images) and provides consistent estimates of the quadrotor's full state that are fed back to the controllers and optionally the trajectory generator.

where $k_p > 0$ and $k_d > 0$ are the PD gains. The desired collective thrust (defining the desired magnitude of acceleration) is the projection of $\mathbf{f}_{\text{des}}$ to the body $\mathbf{z}'$ axis:

$$F_{\text{des}} = (\mathbf{f}_{\text{des}})^T \mathbf{z}' \tag{17}$$

The desired attitude (defining the desired direction of acceleration and accounting for the desired yaw angle) can be written as a rotation matrix as follows:

$$\mathbf{R}_{\text{des}} = \begin{bmatrix} \mathbf{x}'_{\text{des}} & \mathbf{y}'_{\text{des}} & \mathbf{z}'_{\text{des}} \end{bmatrix}, \tag{18}$$

where

$$\mathbf{z}'_{\text{des}} = \frac{\mathbf{f}_{\text{des}}}{||\mathbf{f}_{\text{des}}||}, \tag{19}$$

$$\mathbf{y}'_{\text{des}} = \frac{\mathbf{z}'_{\text{des}} \times \mathbf{n}_{\text{des}}}{||\mathbf{z}'_{\text{des}} \times \mathbf{n}_{\text{des}}||} \quad \text{with} \quad \mathbf{n}_{\text{des}} = \begin{bmatrix} \cos\psi_{\text{des}} \\ \sin\psi_{\text{des}} \\ 0 \end{bmatrix}, \text{ and} \tag{20}$$

$$\mathbf{x}'_{\text{des}} = \mathbf{y}'_{\text{des}} \times \mathbf{z}'_{\text{des}}. \tag{21}$$

3.2 Attitude Control

The attitude controller computes the desired moments to bring the actual rotation $\mathbf{R}_{\text{act}}$ close to the desired rotation $\mathbf{R}_{\text{des}}$. The orientation error is defined as

$$\boldsymbol{\varepsilon}_{\mathbf{R}} = \frac{1}{2}\left(\mathbf{R}_{\text{des}}^T \mathbf{R}_{\text{act}} - \mathbf{R}_{\text{act}}^T \mathbf{R}_{\text{des}}\right)^{\vee}, \tag{22}$$

where "$\vee$" is the vee operator that maps elements of $\mathcal{SO}(3)$ to $\mathbb{R}^3$. The angular velocity error is defined as follows:

$$\boldsymbol{\varepsilon}_{\omega} = \boldsymbol{\omega}_{\text{act}} - \boldsymbol{\omega}_{\text{des}}. \tag{23}$$

Note that, the desired angular velocity $\boldsymbol{\omega}_{\text{des}}$ should be defined such that $\boldsymbol{\varepsilon}_{\omega}$ is consistent with the rotation error $\boldsymbol{\varepsilon}_{\mathbf{R}}$. We can alternatively set it to zero, but this would typically lead to poorer tracking performance. The desired body torques can be similarly computed based on a PD scheme:

$$\boldsymbol{\tau}_{\text{des}} = -k_{\mathbf{R}}\boldsymbol{\varepsilon}_{\mathbf{R}} - k_{\omega}\boldsymbol{\varepsilon}_{\omega}, \tag{24}$$

where $k_{\mathbf{R}}$ and k_{ω} are diagonal matrices with positive gains.

3.3 Motor Control

Given F_{des} and $\boldsymbol{\tau}_{\text{des}}$, we can compute the corresponding desired motor rates. This step is really reversing the maps defined in (4) and (5) by solving for Ω_i. The desired motor rates for given F_{des} and $\boldsymbol{\tau}_{\text{des}}$ can be computed as follows:

$$\begin{bmatrix} \Omega_{1,\text{des}}^2 \\ \Omega_{2,\text{des}}^2 \\ \Omega_{3,\text{des}}^2 \\ \Omega_{4,\text{des}}^2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1/c_t & -1/c_{\tau_{xy}} & -1/c_{\tau_{xy}} & -1/c_{\tau_z} \\ 1/c_t & -1/c_{\tau_{xy}} & 1/c_{\tau_{xy}} & 1/c_{\tau_z} \\ 1/c_t & 1/c_{\tau_{xy}} & 1/c_{\tau_{xy}} & -1/c_{\tau_z} \\ 1/c_t & 1/c_{\tau_{xy}} & -1/c_{\tau_{xy}} & 1/c_{\tau_z} \end{bmatrix} \begin{bmatrix} F_{\text{des}} \\ \boldsymbol{\tau}_{\text{des}} \end{bmatrix}, \tag{25}$$

where $c_{\tau_{xy}} = lc_t/\sqrt{2}$ and $c_{\tau_z} = c_d$.

The PWMs of the individual motors are computed by inverting the map in (3):

$$\mu_{i,\text{cmd}} = \alpha^{-1}(\Omega_{i,\text{des}} - \beta). \tag{26}$$

4. Smooth Trajectory Generation

The trajectories are often required to be sufficiently smooth and satisfy a set of constraints to be feasible for a quadrotor to track. There are various methods to generate trajectories for a quadrotor system; we present one position trajectory generation formulation that is adapted based on (Mellinger and Kumar, 2011). The desired yaw angle can be often defined in a decoupled manner. For a quadrotor equipped with an onboard camera for mapping the environment, the desired yaw angle could be, for instance, designed to maximize the information gain.

The desired position trajectories $p_{x,\text{des}}$, $p_{y,\text{des}}$, and $p_{z,\text{des}}$ are respectively parametrized as splines or piecewise polynomials. The minimum-derivative trajectory optimization problem is to minimize the objective:

$$\min \quad \underbrace{\int_{t_0}^{t_1} ||T_1^{(r)}(t)||^2\,dt + \int_{t_1}^{t_2} ||T_2^{(r)}(t)||^2\,dt + \cdots + \int_{t_{S-1}}^{t_S} ||T_S^{(r)}(t)||^2\,dt}_{\triangleq J} \tag{27}$$

subjected to the spline continuity constraints (i.e., continuity of the value and higher derivatives at endpoints), where $T_s(t) = \sum_{n=0}^{N} p_{s,n} t^n$ for $s = \{1, 2, ..., S\}$ are S segments of Nth-order polynomials composing the spline, and the superscript (r) denotes the rth time-derivative. In the discussion below, following derivations similar to (Richter et al., 2016; Mellinger and Kumar, 2011), we show that this objective can be written in a quadratic form and the overall optimization can be formulated as a quadratic program (QP).

4.1 Objective Function in Quadratic Form

We first show that the minimum-derivative objective associated with each polynomial segment $T_s(t)$ can be written in the following quadratic form:

$$J_s = \int_{t_{s-1}}^{t_s} ||T_s^{(r)}(t)||^2\,dt = p_s^T Q_s p_s, \tag{28}$$

where J_s denotes the objective function associated with the sth polynomial, p_s are the coefficients of the polynomial in ascending order, and $Q_s \in \mathbb{R}^{(N+1)\times(N+1)}$ is the Hessian to be derived. Note that the objective function for the overall spline can be written as

$$J = \sum_{s=1}^{S} p_s^T Q_s p_s = P^T Q P, \tag{29}$$

where $P = [p_1^T, p_2^T, ..., p_S^T]^T$ is the stack of the coefficients for all S polynomial segments and $Q = \text{blockdiag}(Q_1, Q_2, ..., Q_S)$ is the augmentation of Hessians. In the derivations below, we focus on the objective function for one of the segments J_s.

- *Objective function:* The objective function is

$$J_s = \int_{t_{s-1}}^{t_s} ||T_s^{(r)}(t)||^2 \, dt = \int_{t_{s-1}}^{t_s} \left|\left| \frac{d^r}{dt^r} \sum_{n=0}^{N} p_{s,n} t^n \right|\right|^2 dt = \int_{t_{s-1}}^{t_s} I(t) \, dt, \tag{30}$$

  where $I(t) = \left|\left| \frac{d^r}{dt^r} \sum_{n=0}^{N} p_{s,n} t^n \right|\right|^2 = \left(\frac{d^r}{dt^r} \sum_{n=0}^{N} p_{s,n} t^n \right)^2$ denotes the integrand in the minimum-derivative objective function.

- *Derivative:* The rth derivative of the polynomial can be written as

$$\frac{d^r}{dt^r} \sum_{n=0}^{N} p_{s,n} t^n = \sum_{n=r}^{N} p_{s,n} \left[n(n-1) \cdots (n-r+1) \right] t^{n-r} = \sum_{n=r}^{N} \left[p_{s,n} \left(\prod_{m=0}^{r-1} n - m \right) t^{n-r} \right]. \tag{31}$$

  Note that the objective function drops the dependencies on the first $r-1$ coefficients; these coefficients play a role only in the constraints. In other words, the magnitudes of the higher-order coefficients are penalized with this objective, which enforces smoothness.

- *Square of polynomials:* After obtaining the rth derivative, the next operation in (30) to consider is the square of the polynomial (derivative, in fact). There are different ways of calculating the square of polynomials (e.g., in (Richter et al., 2016), it is written as a convolution and the Hessian is then calculated by taking derivatives of the convolution). Here, we try to take a different approach, where we write out the quadratic form directly (without the need to take derivatives twice afterwards).

  We first note that the square of a polynomial $\rho(\xi) = \sum_{k=0}^{K} \theta_k \xi^k$ (swapped notations to Greek letters to avoid confusion) can be written as

$$\rho(\xi) = \left(\sum_{k=0}^{K} \theta_k \xi^k \right)^2 \tag{32}$$

$$= \begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_K \end{bmatrix} \underbrace{\begin{bmatrix} \xi^0 & \xi^1 & \cdots & \xi^K \\ \xi^1 & \cdots & \xi^K & \xi^{K+1} \\ \vdots & \iddots & \iddots & \vdots \\ \xi^K & \xi^{K+1} & \cdots & \xi^{2K} \end{bmatrix}}_{\triangleq \Xi} \underbrace{\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_K \end{bmatrix}}_{\triangleq \theta} \tag{33}$$

$$= \theta^T \Xi \theta. \tag{34}$$

  For our case, where the square is raised to the derivative of the polynomial, we can apply the above quadratic form translation by only considering the elements in the bottom square block from indices r to N (since the summation is from $n = r$ to $n = N$ in (31) and the coefficient vector contains the coefficients from $n = 0$ to $n = N$).

  Here are the details. We write the square of the polynomial in (31) (which is also the integrand

of the objective function in (30)) as

$$I(t) = \left(\sum_{n=r}^{N} \left[p_{s,n} \left(\prod_{m=0}^{r-1} n - m \right) t^{n-r} \right] \right)^2 \tag{35}$$

$$= \widetilde{p}^T \underbrace{\begin{bmatrix} t^0 & t^1 & \cdots & t^{N-r} \\ t^1 & \cdots & t^{N-r} & t^{N-r+1} \\ \vdots & \iddots & \iddots & \vdots \\ t^{N-r} & t^{N-r+1} & \cdots & t^{2(N-r)} \end{bmatrix}}_{\triangleq \widetilde{M}(t)} \widetilde{p}, \tag{36}$$

where $\widetilde{p} = \left[p_{s,r} \left(\prod_{m=0}^{r-1} r - m \right), p_{s,r+1} \left(\prod_{m=0}^{r-1} r + 1 - m \right), ..., p_{s,N} \left(\prod_{m=0}^{r-1} N - m \right) \right]^T$. Note that the series multiplied by the polynomial coefficients are constants; for convenience, we define

$$c(q) = \prod_{m=0}^{r-1} q - m, \tag{37}$$

where r is the order of the derivative we are taking in (31). We define a matrix $C \in \mathbb{R}^{(N-r+1)\times(N+1)}$

$$C = \left[\begin{array}{ccc|ccc} 0 & \cdots & 0 & c(r) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & c(r+1) & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & c(N) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{0}_{(N-r+1)\times r} & \widetilde{C} \end{array}\right], \tag{38}$$

and rewrite (36) as

$$I(t) = p_s^T C^T \widetilde{M}(t) C p_s \tag{39}$$

$$= p_s^T M(t) p_s, \tag{40}$$

where the substitution $\widetilde{p} = C p_s$ is made and $M(t) = C^T \widetilde{M}(t) C$.

▷ *Integration:* Given the quadratic form of the integrand in (40), we can then evaluate the integration in the minimum-derivative objective function in (30) between two given times t_{s-1} and t_s. We note that, in the quadratic form in (40), the only time-dependent terms come from $M(t)$, which are summarized in the matrix $\widetilde{M}(t)$. The operations of the quadratic form are equivalent to a set of summations over the non-zero entries of $M(t)$ which comes down to the elements of $\widetilde{M}(t)$. The integration of the quadratic expression $I(t)$ can be then evaluated and written as below

$$J_s = \int_{t_{s-1}}^{t_s} I(t)\, dt \tag{41}$$

$$= \int_{t_{s-1}}^{t_s} p_s^T C^T \widetilde{M}(t) C p_s \, dt \tag{42}$$

$$= p_s^T Q_s p_s, \tag{43}$$

where

$$Q_s = C^T \widetilde{Q}_s C = \left[\begin{array}{c|c} \mathbf{0}_{r\times r} & \mathbf{0}_{r\times(N-r+1)} \\ \hline \mathbf{0}_{(N-r+1)\times r} & \widetilde{C}^T \widetilde{Q}_s \widetilde{C} \end{array}\right], \tag{44}$$

0 are zero matrices with dimensions indicated by the corresponding subscripts, and $\widetilde{Q}_s$ contains the element-wise integrations of $\widetilde{M}(t)$ from t_{s-1} to t_s

$$\widetilde{Q}_s = \begin{bmatrix} t_s - t_{s-1} & \frac{t_s^2 - t_{s-1}^2}{2} & \cdots & \frac{t_s^{N-r+1} - t_{s-1}^{N-r+1}}{N-r+1} \\ \frac{t_s^2 - t_{s-1}^2}{2} & \cdots & \frac{t_s^{N-r+1} - t_{s-1}^{N-r+1}}{N-r+1} & \frac{t_s^{N-r+2} - t_{s-1}^{N-r+2}}{N-r+2} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{t_s^{N-r+1} - t_{s-1}^{N-r+1}}{N-r+1} & \frac{t_s^{N-r+2} - t_{s-1}^{N-r+2}}{N-r+2} & \cdots & \frac{t_s^{2(N-r)+1} - t_{s-1}^{2(N-r)+1}}{2(N-r)+1} \end{bmatrix}. \tag{45}$$

Continuity as Equality Constraints. The position trajectories are splines and C^r continuities can be enforced at the endpoints to ensure feasibility for tracking. Similar to the objective function derivation, we consider one polynomial segment indexed with s. The rth derivative of the polynomial at the endpoints can be written as

$$A_{srt} p_s = \begin{bmatrix} 0 & \cdots & 0 & c(r) & c(r+1)t & \cdots & c(N)t^{N-r} \end{bmatrix} p_s, \tag{46}$$

where $t = \{t_{s-1}, t_s\}$ is the time at the end point. A side note on the notation: the subscripts of the matrix A are s denoting the segment index, r denoting the rth derivative of the polynomial, and t denoting the starting or the terminal endpoint. For each polynomial segment s, we can encode the continuity constraints as follows:

$$A_s p_s = b_s, \tag{47}$$

where

$$A_s = \begin{bmatrix} A_{s0t_{s-1}} \\ A_{s1t_{s-1}} \\ \vdots \\ A_{sr_{\max}t_{s-1}} \\ A_{s0t_s} \\ A_{s1t_s} \\ \vdots \\ A_{sr_{\max}t_s} \end{bmatrix} \text{ and } b_s = \begin{bmatrix} T_s^{(0)}(t_{s-1}) \\ T_s^{(1)}(t_{s-1}) \\ \vdots \\ T_s^{(r_{\max})}(t_{s-1}) \\ T_s^{(0)}(t_s) \\ T_s^{(1)}(t_s) \\ \vdots \\ T_s^{(r_{\max})}(t_s) \end{bmatrix}. \tag{48}$$

4.2 The Smooth Trajectory Generation QP

The minimum-derivative trajectory optimization problem is overall a QP that can be solved efficiently:

$$\begin{aligned} \min_{p} \quad & \sum_{s=1}^{S} p_s^T Q_s p_s \\ \text{subject to} \quad & A_s p_s = b_s \quad \text{for } s = \{1, 2, ..., S\}, \end{aligned} \tag{49}$$

where p_s contains the coefficients of the sth polynomial segment (in the ascending order), Q_s is the Hessian matrix defined in (44), and the pairs (A_s, b_s) set continuity constraints on the endpoints of the polynomials composing the spline. One may optionally introduce additional constraints on the spline trajectories or their higher derivatives to account for the physical constraints of the quadrotor and the operating environment.

5. Further Reading

Drone Racing

[1] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023. [pdf]

[2] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," 2023. [pdf]

General Quadrotor Modeling, Planning, and Control

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525. [pdf]

[2] C. Powers, D. Mellinger, and V. Kumar, "Quadrotor kinematics and dynamics," in *Handbook of Unmanned Aerial Vehicles*. Springer, 2015, ch. 16, pp. 307–327. [pdf]

Crazyflie Modeling, Planning, and Control

[1] C. Luis and J. L. Ny, "Design of a trajectory tracking controller for a nanoquadcopter," *arXiv preprint arXiv:1608.05786*, 2016, Technical Report, Mobile Robotics and Autonomous Systems Laboratory, Polytechnique Montreal. [pdf]

[2] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 604–611, 2020. [pdf]

[3] S. Teetaert, W. Zhao, N. Xinyuan, H. Zahir, H. Leong, M. Hidalgo, G. Puga, T. Lorente, N. Espinosa, J. A. D. Carrasco *et al.*, "A remote sim2real aerial competition: Fostering reproducibility and solutions' diversity in robotics challenges," *arXiv preprint arXiv:2308.16743*, 2023, IROS 2022 Safe Robot Learning Competition. [pdf]

Learning and Adaptation

[1] K. Pereida and A. P. Schoellig, "Adaptive model predictive control for high-accuracy trajectory tracking in changing conditions," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 7831–7837. [pdf]

[2] Z. Wu, S. Cheng, P. Zhao, A. Gahlawat, K. A. Ackerman, A. Lakshmanan, C. Yang, J. Yu, and N. Hovakimyan, "L1 quad: L1 adaptive augmentation of geometric control for agile quadrotors with performance guarantees," *arXiv preprint arXiv:2302.07208*, 2023. [pdf]

[3] A. P. Schoellig, F. L. Mueller, and R. D'andrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, pp. 103–127, 2012. [pdf]

[4] M. Greeff and A. P. Schoellig, "Flatness-based model predictive control for quadrotor trajectory tracking," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6740–6745. [pdf]

[5] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Deep neural networks as add-on modules for enhancing robot performance in impromptu trajectory tracking," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1397–1418, 2020. [pdf]

References

D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.

C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," 2023.

C. Powers, D. Mellinger, and V. Kumar, "Quadrotor kinematics and dynamics," in *Handbook of Unmanned Aerial Vehicles*. Springer, 2015, ch. 16, pp. 307–327.

C. Luis and J. L. Ny, "Design of a trajectory tracking controller for a nanoquadcopter," *arXiv preprint arXiv:1608.05786*, 2016, Technical Report, Mobile Robotics and Autonomous Systems Laboratory, Polytechnique Montreal.

C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 604–611, 2020.

S. Teetaert, W. Zhao, N. Xinyuan, H. Zahir, H. Leong, M. Hidalgo, G. Puga, T. Lorente, N. Espinosa, J. A. D. Carrasco *et al.*, "A remote sim2real aerial competition: Fostering reproducibility and solutions' diversity in robotics challenges," *arXiv preprint arXiv:2308.16743*, 2023, IROS 2022 Safe Robot Learning Competition.

K. Pereida and A. P. Schoellig, "Adaptive model predictive control for high-accuracy trajectory tracking in changing conditions," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 7831–7837.

Z. Wu, S. Cheng, P. Zhao, A. Gahlawat, K. A. Ackerman, A. Lakshmanan, C. Yang, J. Yu, and N. Hovakimyan, "L1 quad: L1 adaptive augmentation of geometric control for agile quadrotors with performance guarantees," *arXiv preprint arXiv:2302.07208*, 2023.

A. P. Schoellig, F. L. Mueller, and R. D'andrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, pp. 103–127, 2012.

M. Greeff and A. P. Schoellig, "Flatness-based model predictive control for quadrotor trajectory tracking," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6740–6745.

S. Zhou, M. K. Helwa, and A. P. Schoellig, "Deep neural networks as add-on modules for enhancing robot performance in impromptu trajectory tracking," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1397–1418, 2020.