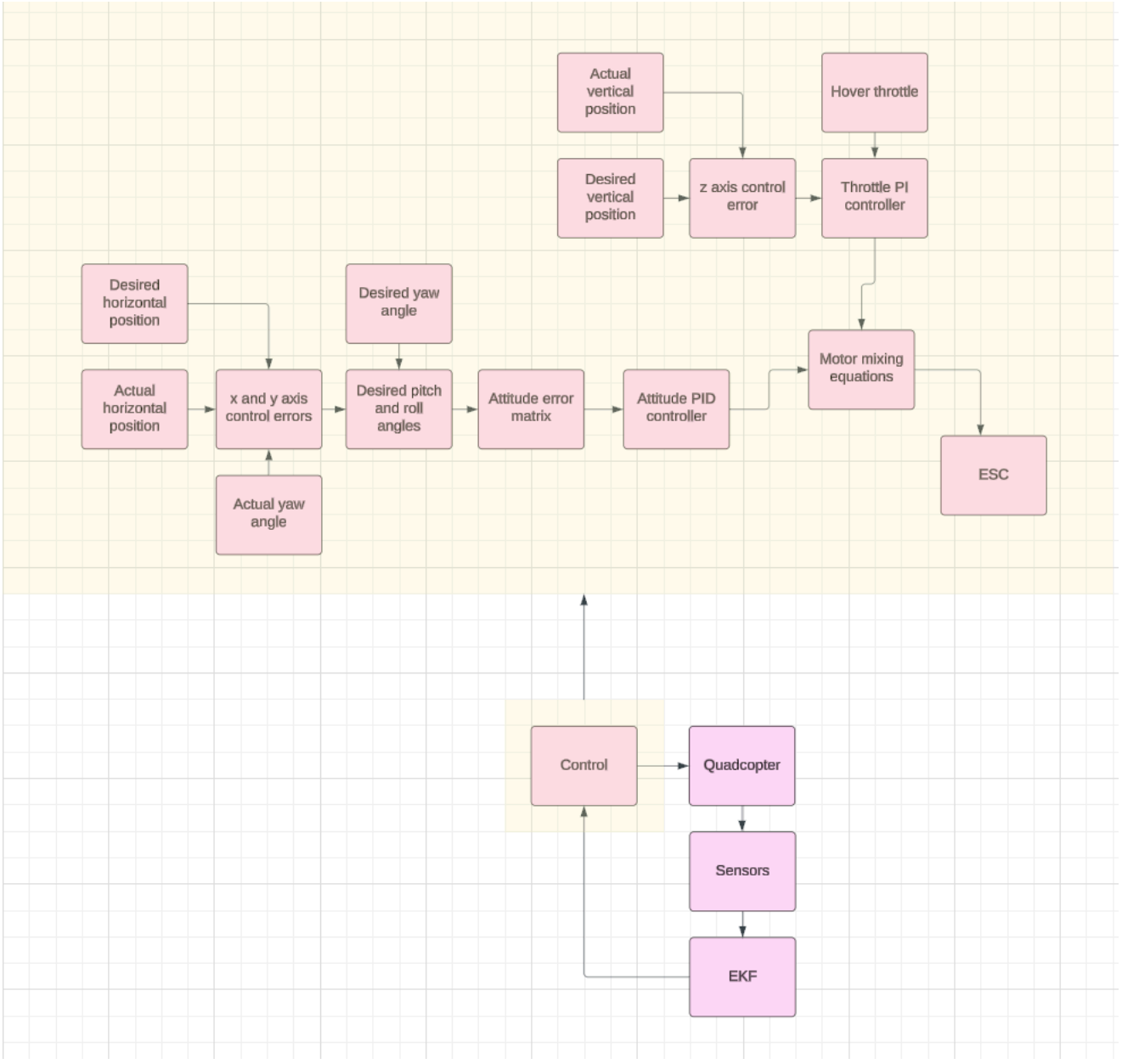
Quadcopters are underactuated systems, meaning they can't be controlled using a single controller. Manual flight only requires an attitude controller but autonomous flight requires a position controller as well.

In this document we define GUDs control architecture



Position control framework

Desired position information is inputted from a user with a remote controller or from a path planning algorithm. The actual position information is inputted from the EKF. The difference between these is the position error. The position controller produces the desired yaw, pitch and roll angles according to the position error which are treated as inputs to the attitude controller.

Attitude control framework

Actual attitude information is extracted from the EKF, and desired attitude information is provided from the position error outputted by the position controller. The error between these two is considered as attitude error and is fed into the attitude PID controller.

The PID controller then generates the thrust which should be produced by each of the quadcopters motors.

Derivation

Yaw, pitch and roll angle control

If we have x_d, y_d (desired horizontal position) and x_a, y_a (actual horizontal position), then position errors in x and y are:

$$\begin{aligned} x_e &= x_d - x_a \\ y_e &= y_d - y_a \end{aligned}$$

If position control gains for both x and y axes are P_{xp} and P_{yp} then the desired velocities $\dot{x}_d, \dot{y}_d$ can be written as

$$\begin{aligned} \dot{x}_d &= P_{xp}\, x_e \\ \dot{y}_d &= P_{yp}\, y_e \end{aligned}$$

By position control gain, we mean a tunable factor that determines how aggressively the controller responds to position error.

Thus velocity errors become:

$$\begin{aligned} \dot{x}_e &= \dot{x}_d - \dot{x}_a \\ &= P_{xp}\, x_e - \dot{x}_a \end{aligned}$$

$$\begin{aligned} \dot{y}_e &= \dot{y}_d - \dot{y}_a \\ &= P_{yp}\, y_e - \dot{y}_a \end{aligned}$$

Now, if velocity proportional gains are P_{xv} and P_{yv} and integral gains are I_{xv} and I_{yv} then control errors for the x, y axes can be written as:

$$P_{error\,x} = P_{xv}\, \dot{x}_e$$

$$I_{error\,x} = I_{xv} \int \dot{x}_e$$

$$e_x = P_{error\,x} + I_{error\,x}$$

i.e. velocity proportional gains scale the velocity error to generate a control signals thus determining how aggressively the controller reacts to velocity errors.

integral gains account for the accumulated velocity error over time. They help reduce long-term drift and corrects small persistent errors that the proportional term alone might not fix

The control error is thus a sum of these, i.e. a proportional gain term to provide immediate correction based on the current velocity error and an integral gain to correct long-term accumulated errors and ensure stability.

The same holds for the y axis:

$$P_{error\,y} = P_{yv}\,\dot{y}_e$$

$$I_{error\,y} = I_{yv}\int \dot{y}_e$$

$$e_y = P_{error\,y} + I_{error\,y}$$

If the initial heading angle of the drone is θ then the earth frame control errors e_x , e_y can be converted to the body frame desired roll and pitch angles as follows:

$$d_p = e_x\cos(\theta) + e_y\sin(\theta)$$

$$d_r = -e_x\sin(\theta) + e_y\cos(\theta)$$

i.e.

$$\begin{bmatrix} d_p \\ d_r \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix}$$

since we are converting from the global frame to the local frame

We define a desired yaw angle d_y based on user input or a path planning algorithm

Using the horizontal position controller desired outputs roll, pitch and yaw, the desired rotation matrix can be obtained:

$$R_{d_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(d_p) & \sin(d_p) \\ 0 & \sin(d_p) & \cos(d_p) \end{bmatrix}$$

$$R_{d_2} = \begin{bmatrix} \cos(d_r) & 0 & \sin(d_r) \\ 0 & 1 & 0 \\ -\sin(d_r) & 0 & \cos(d_r) \end{bmatrix}$$

$$R_{d_3} = \begin{bmatrix} \cos(d_y) & -\sin(d_y) & 0 \\ \sin(d_y) & \cos(d_y) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_d = (R_{d_1}, R_{d_2}, R_{d_3})$$

We thus define the attitude error matrix (R_e) and actual rotation matrix R_a (describes the drones real orientation) and the body frame reference attitude error e_R:

$$R_e = R_d^T R_a - (R_d^T R_a)^T$$

$$e_R = \begin{bmatrix} R_e(2,3) \\ R_e(1,3) \\ R_e(1,2) \end{bmatrix}$$

The subtraction in R_e results in a skew-symmetric matrix which is a common way of representing rotational errors. This is why in e_R we index R_e. Each indexing accesses the elements proportional to the rotational errors about the $x, y \text{ and } z$ axes

These errors are then fed into the attitude controller to generate corrective torques.

$$\text{Torque output} = P\, e_R + I \int e_R + D\, \frac{d}{dt} e_R$$

Throttle control

Now, if we have z_d, z_a (desired and actual vertical position), then position error in z is:

$$z_e = z_d - z_a$$

If we have position control gain for the z axis as P_{zp} then the desired velocity is

$$\dot{z}_d = P_{zp}\, z_e$$

And thus we have the vertical velocity error as:

$$\dot{z}_e = \dot{z}_d - \dot{z}_a$$

With velocity control gain for the z axis as P_{zv} then the desired acceleration is:

$$\ddot{z}_d = P_{zv}\, \ddot{z}_e$$

and if acceleration proportional gain is P_{za} and integral gain is I_{za} then control error for the z axis e_z is:

$$P_{errorz} = P_{za}\, \ddot{z}_e$$

$$I_{errorz} = I_{za} \int \ddot{z}_e$$

$$e_z = P_{error\, z} + I_{error\, z}$$

Using the control error and a user defined hover throttle value d_H (baseline thrust needed to stay in place), a desired throttle d_T is defined

$$d_T = d_H + e_z$$
$$\text{Throttle ouput} = d_H + P_{za}\, \ddot{z}_e + I_{za} \int \ddot{z}_e$$

Thus, roll and pitch corrections align the drone with the desired x and y positions. The desired yaw angle is included in the attitude control and automatically corrected. Throttle control adjusts the drone vertically to align the drone with the desired z position.

Thus, we have a PI for throttle control, increases or decreasing all motor speeds equally. This doesn't result in an orientation change but in an altitude change.

Then we have a PID for torque control. This adjusts individual motor speeds differently to create rotational forces (torques). e.g. to roll right we would want to increase the speeds of the motors on the left and decrease the speed of those on the right. Pitching forward or backward means increasing or decreasing the speed of the front and rear motors. Yaw control works differently. In standard quadcopters, two motors spin clockwise and two spin counterclockwise. Yaw control involves changing the speed of the clockwise and counterclockwise motors.

Motor mixing equations

Once we have total thrust and torque outputs, we need to map these to the motors as described above.

Consider the following X-quadrotor configuration:

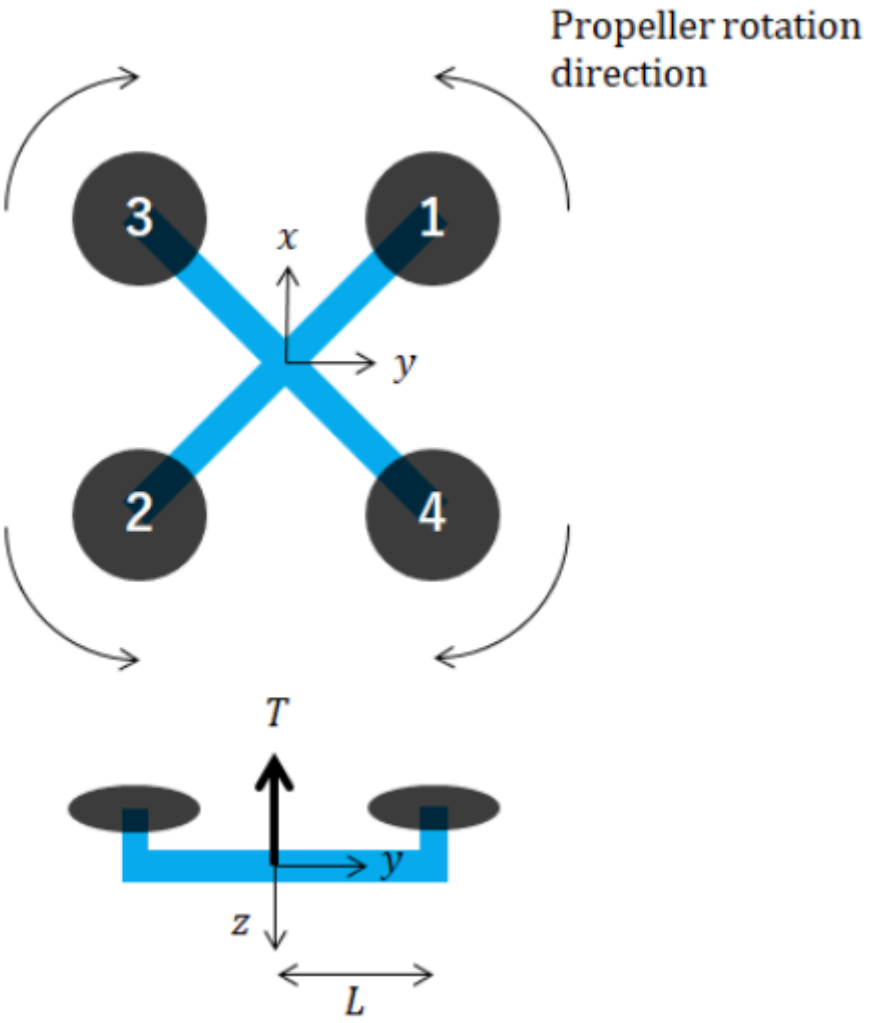
Then:

$$\begin{aligned}
\text{Motor 1 PWM} &= \text{thrust - roll} + \text{pitch} + \text{yaw} \\
\text{Motor 2 PWM} &= \text{thrust} + \text{roll - pitch} + \text{yaw} \\
\text{Motor 3 PWM} &= \text{thrust} + \text{roll} + \text{pitch - yaw} \\
\text{Motor 4 PWM} &= \text{thrust - roll - pitch - yaw}
\end{aligned}$$

$$\begin{bmatrix} \text{Motor 1 PWM} \\ \text{Motor 2 PWM} \\ \text{Motor 3 PWM} \\ \text{Motor 4 PWM} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \text{thrust} \\ \text{roll} \\ \text{pitch} \\ \text{yaw} \end{bmatrix}$$

The matrix in the middle is called a mixer table. It is an N by 4 matrix where N is the number of motors.

We can derive it as follows:



By the principle of superposition we sum the forces and torques:

$$
\begin{aligned}
T &= F_1 + F_2 + F_3 + F_4 \\
\tau_x &= -LF_1 + LF_2 + LF_3 - LF_4 \\
\tau_y &= LF_1 - LF_2 + LF_3 - LF_4 \\
\tau_z &= \gamma F_1 + \gamma F_2 - \gamma F_3 - \gamma F_4
\end{aligned}
$$

Where γ is the rotor drag to thrust coefficient

In matrix form

$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -L & L & L & -L \\ L & -L & L & -L \\ \gamma & \gamma & -\gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$$

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -L & L & L & -L \\ L & -L & L & -L \\ \gamma & \gamma & -\gamma & -\gamma \end{bmatrix}^{-1} \begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & L & 0 & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}^{-1} \right) \begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{L} & 0 & 0 \\ 0 & 0 & \frac{1}{L} & 0 \\ 0 & 0 & 0 & \frac{1}{\gamma} \end{bmatrix} \begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4L} & 0 & 0 \\ 0 & 0 & \frac{1}{4L} & 0 \\ 0 & 0 & 0 & \frac{1}{4\gamma} \end{bmatrix} \begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

If we introduce scaled torques and thrust:

$$\begin{aligned} \hat{T} &= \frac{T}{4} \\ \hat{\tau}_x &= \frac{\tau_x}{4L} \\ \hat{\tau}_y &= \frac{\tau_y}{4L} \\ \hat{\tau}_z &= \frac{\tau_z}{4\gamma} \end{aligned}$$

We then have the normalized mixer table:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} \hat{T} \\ \hat{\tau}_x \\ \hat{\tau}_y \\ \hat{\tau}_z \end{bmatrix}$$

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \hat{T} \\ \hat{\tau}_x \\ \hat{\tau}_y \\ \hat{\tau}_z \end{bmatrix}$$

The output of the mixer is fed to the ESC as PWM duty. In our calculations the output of the mixer is force, whereas the ESC accepts rotational speed of the rotor. The force needs to be divided by the thrust coefficient and square rooted.

The thrust force generated by a rotor is given by:

$$F = C_T \cdot \omega^2$$

where C_T is the thrust coefficient which depends on air density, blade shape, etc. and ω is the rotor speed in radians per second.

Solving for rotor speed:

$$\omega_i = \sqrt{\frac{F_i}{C_T}}$$

Thus each ESC input is:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \hat{T} \\ \hat{\tau}_x \\ \hat{\tau}_y \\ \hat{\tau}_z \end{bmatrix}$$

$$\omega_i = \sqrt{\frac{F_i}{C_T}} \quad \forall\, i \in \{1, 2, 3, 4\}$$

Considerations when using this framework

- C_T must be experimentally determined or provided by the manufacturer
-