Math Modeling: Coordinate Transforms and Movement Calculations

Tello drones localize relative to specific mission pads. To create a larger space where drones can remain within global bounds requires knowing the location between mission pads, as a predefined map.

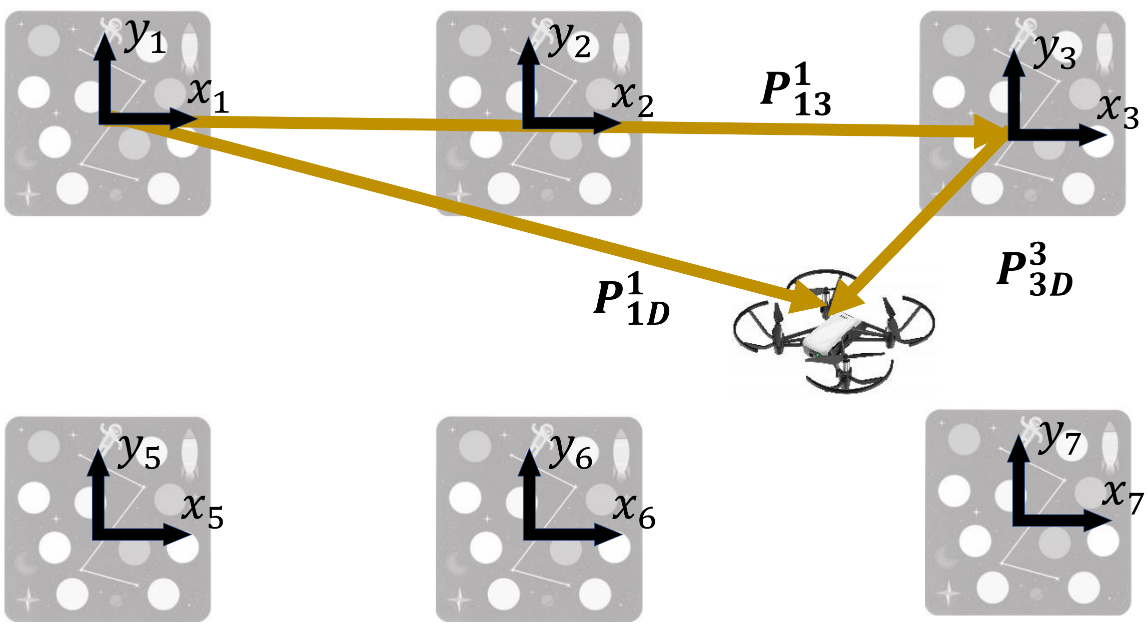

Figure 1: Mission pad

We can define these vectors as the following:

P^1_{1D}: Position in global coordinate frame

P^3_{3D}: Position of drone in local mission pad frame

P^1_{13}: Pre-determined location of mission pad

Here we can see that if we want to find the position of the drone in the global coordinate frame, we know $\boldsymbol{P^3_{3D}}$ from the drone, and we predetermined $\boldsymbol{P^1_{13}}$. This should be simple vector addition, however they are technically in different coordinate frames and require a rotation matrix, resulting in:

$$P^1_{1D} = R^1_3 * P^3_{3D} + P^1_{13}$$

However, by aligning all mission pads in the same direction, R_3^1 simplifies to the identity, resulting in:

$$P_{3D}^1 = P_{3D}^3 \rightarrow P_{1D}^1 = P_{3D}^3 + P_{13}^1$$

Now knowing the global coordinate, we can now apply proportional control whenever the drone begins to fly out of the approved airspace.

For example, if the drone crosses a boundary defined along the x or y axis, we can define an error as:

$$\boldsymbol{e} = \begin{bmatrix} x_{boundary} \\ y_{boundary} \\ 0 \end{bmatrix} - \begin{bmatrix} x_D \\ y_D \\ 0 \end{bmatrix}$$

Then, a restoring velocity, $\boldsymbol{v}$ would be based on some gain k_p.

$$\boldsymbol{v} = k_p \cdot \boldsymbol{e}$$

However, this restoring velocity is in the global frame, so let us denote it as $\boldsymbol{v^1}$. In order to put this velocity into the drone's frame, we need to multiply it back by a rotation matrix, based upon the yaw of the drone.

$$\boldsymbol{v^D} = \boldsymbol{R_1^D} \cdot \boldsymbol{v^1}$$

Where:

$$\boldsymbol{R_1^D} = \begin{bmatrix} \cos(yaw) & -\sin(yaw) & 0 \\ \sin(yaw) & \cos(yaw) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This results in the following:

$$\boldsymbol{v^D} = \begin{bmatrix} \cos(yaw) & -\sin(yaw) & 0 \\ \sin(yaw) & \cos(yaw) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot k_p \cdot \left(\begin{bmatrix} x_{boundary} \\ y_{boundary} \\ 0 \end{bmatrix} - \begin{bmatrix} x_D \\ y_D \\ 0 \end{bmatrix} \right)$$