WHY WHY WHY?

1 Quadcopter Model

For the physics model, the coordinate space is referred to using *local tangent plane coordinates*, namely that z is east, x is north and y is up. The principal axes of movement are *pitch*, along the transverse (z) axis, *roll*, along the longitudinal (x) axis, and *yaw*, along the vertical (y) axis.

A UAV can be modelled as a rigid body with mass m, on Earth. As such, gravity acts on the agent at $9.81\frac{m}{s^2}$. The drag coefficient is estimated at 0.975 considering the mass of the body [1].

The dimensions and chracteristics of the agent have been determined using both realistic averages [2] and estimations. These can be seen in Fig. 1.

Parameter	Value
Mass, m	10 kg
Length, l	0.7 m
Width, w	0.7 m
Height, h	0.25 m
Propeller Area, A	0.16 m^2
Distance To Propeller Centre, d	0.5 m
Drag Coefficient, C_d	0.975

Figure 1: Dimensions of the Agent

The agent is modelled as a quadcopter, with four propellers. These can be independently controlled to affect the movement of the agent.

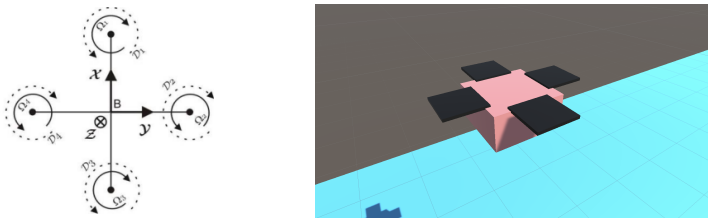

Figure 2: Propeller Diagram & Simulated Agent

For this investigation, a quadcopter was chosen as the type of agent, rather than a fixed-wing aircraft, due to increased maneouverability, simplicity when considering autonomous flight and simplicity in takeoff and landing procedures [3].

There are four elements of control for an agent:

- Roll angle (in radians), ψ
- Pitch angle (in radians), θ
- Yaw rate (in $\frac{rad}{s}$), ϕ
- Vertical thrust (in Newtons), T

For simplicity, these can be solely controlled by combinations of propeller thrust levels.

A base thrust level, T_b, can be defined, which is the minimum thrust level required to maintain a stable hover. This is the thrust level required to counteract gravity, so $T_b = mg$. For each control

operation, T_b can be augmented by a thrust level, T_{add}, such that $|T_{add}| < T_b$, which is the additional thrust required to perform the operation.

As such, the control parameters can be operated by the following eight combinations of thrust levels for each propeller, with a high thrust level $T_{add} > 0$ and a low thrust level $T_{add} < 0$:

Operation	High Thrust	Low Thrust
Positive Roll, ψ_+	D_4	D_2
Negative Roll, ψ_-	D_2	D_4
Positive Pitch, θ_+	D_3	D_1
Negative Pitch, θ_-	D_1	D_3
Positive Thrust, T_+	D_1, D_2, D_3, D_4	none
Negative Thrust, T_-	none	D_1, D_2, D_3, D_4

Figure 3: Control Operations and Thrust Levels

It is noted that yaw is omitted from Fig. 3. For the purposes of the simulation, the propellers do not rotate. This lends itself to needing a workaround for simulating yaw. In *Unity*'s physics engine, the rotation of opposite propellers can be simulated by applying forces along the x and z axes. As such, yaw is simulated according to the table below:

Operation	Propeller	Direction of Force (respectively)
Positive Yaw, ϕ_+	D_1, D_2	Z_-, X_-
Positive Yaw, ϕ_+	D_3, D_4	Z_+, X_+
Negative Yaw, ϕ_-	D_1, D_2	Z_+, X_+
Negative Yaw, ϕ_-	D_3, D_4	Z_-, X_+

Figure 4: Yaw Directions and Respective Forces

In real world physics, a couple of propellers could not provide thrust in both extremes of the same axes, as they could not switch rotation direction. Likewise, forces in adjacent propellers would not cause rotation in the same direction. However, for the purposes of simplicity within the simulation, this is ignored.

It is then possible to set some thrust constants for each control operation. An example assigment for the pitch and roll operations is seen below in Fig. 5.

Thrust Level	Thrust (N)
High	$\frac{2m \cdot g}{4}$
Normal	$\frac{m \cdot g}{4}$
Low	$\frac{0.5m \cdot g}{4}$

Figure 5: Thrust Constants for a drone of mass m kg

2 Stabilisation: Proportional-Integral-Derivative Controller

An issue arises with this implementation; the simulation becomes unstable as thrust cannot be provided in an accurate enough manner to counteract excessive rotation, notably in the roll and pitch axes.

As such, the need for a control system becomes apparent. A PID (Proportional-Integral-Derivative) is a feedback control system, which uses the error between the current state and the desired state to calculate the control parameters.

Fig. 6 shows the feedback loop of the PID controller. The error, $e(t)$, is calculated as the difference between the desired state, r, and the current state, y. The error is then fed into the three controllers. The output of each controller is then summed to produce the control variable, $u(t)$, which is then fed into the system. The system then produces the output, y, which is fed back into the error calculation.

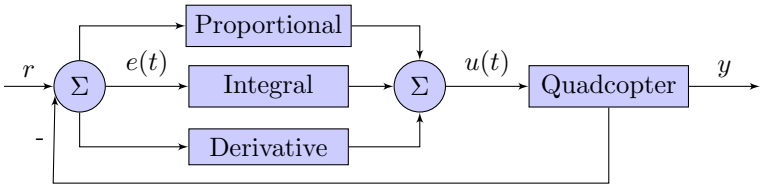

Figure 6: PID Controller

The PID controller is defined by three constant parameters, K_p, K_i and K_d, which are the proportional, integral and derivative gains respectively. The control variable, $u(t)$, is then defined as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

where $e(t)$ is the error at time t. Considering our simulation operates in discrete time intervals (frames), this can be approximated, using the *Laplace transform* as:

$$u(t) = K_p e(t) + K_i \frac{(e_t + e_{t-1})t}{2} + K_d \frac{e_t - e_{t-1}}{t}$$

Hence, a feedback loop of the entire system can be produced, taking into account different PID controllers for each control operation. This is shown in Fig. 7.

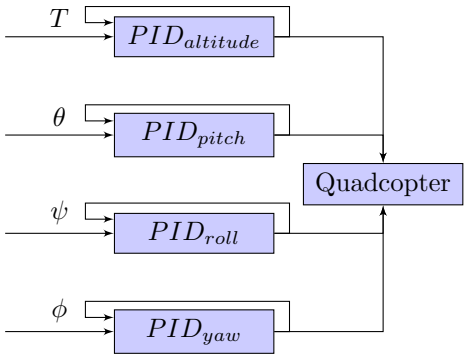

Figure 7: System Model

The constants are then tuned to generate expected stabilisation behaviour. As mentioned previously, each propeller has a base thrust level equal to the force required to hover, $T_b = \frac{mg}{4}$. The PID controller is then used to calculate the thrust level to be added to each propeller, T_{add}, in every frame of the simulation, to perform the desired operation.

Using manual tuning, the values were optimised as shown in Fig. 8

Control Operation	K_p	K_i	K_d
Thrust	6	5	2
Pitch	10	10	2
Roll	10	10	2
Yaw	10	10	2

Figure 8: Tuning Constants

It is noted that the agent is slightly unstable at very small angles. This is negligible.

3 Introducing Autonomy

The initial aim is to achieve position control for an agent, such that when given a current position $[x_0, y_0, z_0]$ and a goal position $[x_1, y_1, z_1]$, the agent will move to the goal position.

There are several methods for this. An agent may yaw to face the goal, then pitch towards it. In the case of this project, time-sensitive position decisions will need to be made when considering swarm dynamics and collision avoidance, so a more efficient method is required. As such, the aim will be to pitch and roll towards the position.

This can be split up into two main goals:

1. Apply a force in the opposite direction when avoiding other agents.
2. Use trigonometry to apply relative ϕ and θ forces, reducing these forces when closer to the goal using a separate PID position controller.

Initially, multiple agents are now introduced into the environment. They are programmed to hover. They should then move away from each other according to rule 1 above. This is the first principle that Reynolds [4] introduced for autonomous agents - *separation*.

NB - note importance of both goals - should they be equal?

References

[1] G. Hattenberger, M. Bronz, and J.-P. Condomines, "Evaluation of drag coefficient for a quadrotor model," *International Journal of Micro Air Vehicles*, vol. 15, p. 4, 2023. [Online]. Available: https://doi.org/10.1177/17568293221148378

[2] M. Figliozzi, "Multicopter drone mass distribution impacts on viability, performance, and sustainability," *Transportation Research Part D: Transport and Environment*, vol. 121, p. 3, 2023.

[3] F. P. Thamm, N. Brieger, K. P. Neitzke, M. Meyer, R. Jansen, and M. Mönninghof, "Songbird - AN Innovative Uas Combining the Advantages of Fixed Wing and Multi Rotor Uas," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL1, p. 345, Aug. 2015.

[4] C. W. Reynolds, *Flocks, Herds, and Schools: A Distributed Behavioral Model.* New York, NY, USA: Association for Computing Machinery, 1998, pp. 273–282. [Online]. Available: https://doi.org/10.1145/280811.281008