UNIVERSITY OF WARWICK

UAVSI

UNMANNED AERIAL VEHICLE SWARM INTELLIGENCE

Final Report

Antonio Brito

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 $9.81\frac{m}{s^2}$. For the purposes of the simulation, air resistance is ignored. The agent is modelled as a quadcopter, with four propellers. These can be independently controlled to affect the movement of the agent.

Figure 1: Propeller Diagram

For this investigation, a quadcopter was chosen as the type of agent, rather than a fixed-wing aircraft, due to increased maneouverability **etc etc etc**

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 2: Control Operations and Thrust Levels

It is noted that yaw is omitted from Fig. **??**. 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 3: 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. **??**.

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

Figure 4: Thrust Constants for a drone of mass m kg

2 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.

insert feedback loop diagram here

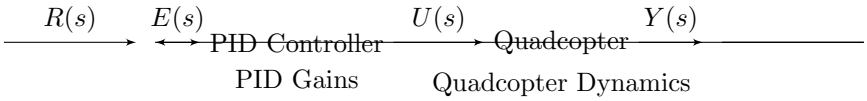

Figure 5: Feedback Control System with PID Controller for Quadcopter

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.

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.

3 Introducing Autonomy: Boids