Drone Flight Controller, Project 9 - MLiS Part I

Jarrad Foley[1,*] and Stephen McSweeney[1,†]

[1] *School of Physics and Astronomy, University of Nottingham, Nottingham, NG7 2RD, UK*

(Dated: January 19, 2023)

This paper outlines the process and intricacies that were explored with creating a drone that is controlled autonomously through reinforcement methods. The method implemented is a state-action value algorithm with a soft policy to ensure all states have a non-zero probability of being taken. This is kept to the bounds of a defined space and set of targets and a specified amount of timesteps. [RESULT] [CONCLUSION]

I. INTRODUCTION

Drones in the current age are becoming more and more popular, encouraging the public to build their own versions of these drones from their homes. Usually drones are programmed with a flight controller to define its movements for a given task. While there are many ways to control a drone such as user controlled, this project focuses on implementing autonomous flight in which the drone will fly within a defined space to hit a number of targets. The drone is controlled using two motors which define the thrust independently allowing for navigation. The task is to write a flight controller which uses reinforcement learning to have the drone traverse the space to hit as many targets as it can within a specified amount of time.

II. IMPLEMENTATION

For the implementation for this project we used a Soft policy Monti Carlo algorithm implementing a state-action value function ensuring that all actions can be taken with a non-zero probability.

A. States

The states for this model uses the pitch of the drone, distance from the target both x and y, and velocity. All these values are scaled to be integers within certain bounds and rounded down to fit within a grid which is used for action selection. The state space is 4 dimentional of size 20 x 40 x 30 x 10.

B. Actions

There are 6 actions used within this program to define what movement the drone does.

$$Ta = \frac{1}{10}\max(-1, \min(10\Delta y, 1))$$

The first action is used to counteract the drones current pitch if it is facing the wrong direction from the target.

$$\begin{cases} T_1 = \frac{1}{2} \\ T_2 = \frac{1}{2} & \sin(\Delta x) = \sin(\Theta)) \\ \\ T_1 = \max(1, \min(\frac{\sin(\Delta x)}{2} + 1), 0)) & \sin(\Delta x) \neq \sin(\Theta)) \\ T_2 = \max(1, \min(-(\frac{\sin(\Delta x)}{2} + 1)), 0)) \end{cases}$$

The second action adjusts the thrusts based on where the target is in the space and the angle of change that the drone will need to adjust to be directed at the target multiplied by 2. If the target is to the left of the drone it will have a lower left thrust than right thrust and vice versa.

$$T_1 = \max(1, \min(\tfrac{2}{5} + Ta + 2\Delta pitch, 0))$$
$$T_2 = \max(1, \min(\tfrac{2}{5} + Ta - 2\Delta pitch, 0))$$

The third action gets the thrust based on the velocity of the drone's pitch to adjust which thrust it needs more power to counteract the current rotation when it's not rotating towards the target. This action is unstable compared to another action which does the same thing just at a slower rate.

$$T_1 = \tfrac{1}{2} - \tfrac{2}{5}\sin(pitchvelocity)$$
$$T_2 = \tfrac{1}{2} + \tfrac{2}{5}\sin(pitchvelocity)$$

The fourth action changes the thrust depending on the current pitch velocity of the drone. If the pitch velocity is in the direction of the target then the thrusts will more equal to slow down how much it spins, however if the pitch velocity is going in the opposite direction of the target then it strongly sets the thrusts to alter the pitch velocity to be towards the target, spinning it into the correct direction.

$$a = \|TargetDistance\|$$

The fifth action adjusts the thrusts based on where the target is in the space and the angle of change that the drone will need to adjust to be directed at the target. If the target is to the left of the drone it will have a lower left thrust than right thrust and vice versa.

$$T_1 = \max(1, \min(\tfrac{1}{2} + Ta + \Delta pitch, 0))$$
$$T_2 = \max(1, \min(\tfrac{1}{2} + Ta - \Delta pitch, 0))$$

The final action is used to adjust the drones y value. If the drones y value is lower than the target's y value then the drone will ascend and if the drones y value is above the target's y value then the drone will descend.

$$T_1 = \max(0, \min(\sin(\Delta y), 1))$$
$$T_2 = \max(0, \min(\sin(\Delta y), 1))$$

* ppxjf2@nottingham.ac.uk
† ppxsm5@nottingham.ac.uk

Target	1	2	3	4	5	6	7	8	9	10
x	0.35	-0.35	0.5	-0.35	0.35	-0.15	-0.35	0.35	-0.5	0.35
y	0.3	0.4	-0.4	0	0.4	-0.1	-0.3	-0.4	0.4	0

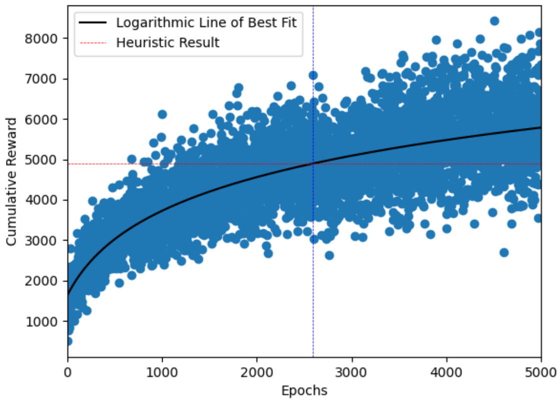

FIG. 1. Shows an example of a figure.

C. Rewards

We use the following reward function:

$$R = \frac{(-3a+0.1b-c+2d^3)}{100}$$

Where:

$$a = \|TargetDistance\|$$
$$b = abs(DroneVelocity)$$
$$c = \lfloor abs(DronePitch) \rfloor$$
$$d = 1 + TargetsHit$$

III. RESULTS

Here, one can display figures, such as in Figure 2.

Each assessment was done with the following target coordinates.

Each simulation was also run for 3000 simulation steps.

In this simulation, the heuristic algorithm hit 5 targets and achieved a value for the sum of rewards as 4894.42, against our chosen reward function. The following graph shows an example of the Soft state Monti-Carlo state machine algorithm's results, under the same conditions.

We modelled the cumulative reward function for any given epoch using the least-sqares optimized log function. 1475.01044 *ln(0.00310817391*CumReward)+ 1642.65857

IV. EVALUATION

The implementation is not deterministic therefore an average of five simulations is used to reduce the error in our least squares exponential model.

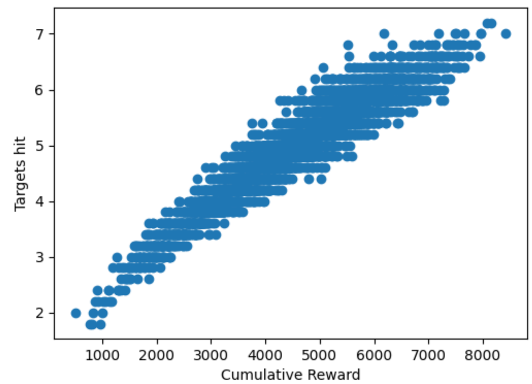

FIG. 2. Shows an example of a figure.

The best fit logarithmic model suggests that we would expect the simulation to overtake the heuristic model with its average simulation after 2596 epochs.

However, the model has a large Mean Squared Error due to only averaging over 5 simulations. The model has a residual standard deviation of 711.29847146, and we would therefore require the cumulative reward to be 7028 for the heuristic model's result to lie outside of 3 standard deviations. The model therefore suggests that after 12395 epochs we would expect any results worse than the heuristic algorithm to be anomalies. Due to our model only being trained to 5000 epochs, however, the model may struggle to be representative of the data at this point.

V. CONCLUSIONS

Write your conclusions here.