Optimising Drone Navigation with Reinforcement Learning

Nikolay Kokinov, Shayan Shafquat, and Dhruv Soni

With the increase in popularity of drones, there is a necessity for developing an efficient and stable autonomous navigation system. We propose two heuristic approaches and a Q-learning model for autonomous navigation in a two-dimensional environment. Through state-action space discretisation and utilisation of multiple reward functions to train and alter the behaviour of the drone, we enhance the agent's performance through the reinforcement learning framework. H1 proves to be more suitable for quick movements toward the objective, while H2 results in more stable motion by application of damping constants. Learning from experience, the Q-learning model provides significant improvements to the heuristic approaches in terms of expected return. We discuss the potential implications of the three approaches in 2D autonomous drone navigation.

I. INTRODUCTION

In recent years, the popularity of drones has soared across various applications [1]. Researchers have since set their sights on creating an autonomous navigation system surpassing the capabilities of expert pilots [2]. This pivotal challenge holds significant real-world implications, particularly in the area of goods transportation [1, 3]. In this paper, we present three models for developing a two-dimensional drone flight controller, enhancing performance through the application of Reinforcement Learning (RL).

RL manifests when an agent becomes increasingly competent at selecting actions that maximise their cumulative rewards (G) [4]. This form of learning is particularly suitable for autonomous navigation, facilitating efficient behavioural improvements despite unpredictable environmental conditions that drones often encounter [5]. RL achieves this by establishing the uncertainty of the interaction between agent and environment as a sequence of states, actions, and rewards, providing a stable learning framework [4]. Hence, we formulate the drone's performance as a RL problem.

Our methodology involves implementing two heuristic approaches (H1 and H2). These assess whether the drone should move horizontally and vertically to reach the target location, guiding it based on specific parameters. Heuristics prove invaluable in real-world emergency situations involving autonomous drones, as they can rapidly yield practical and attainable solutions [6, 7]. Additionally, we employ a Q-learning algorithm, prompting the agent to visit and update values associated with different state-action pairs, eventually approximating the optimal Q function. This iterative process enables the algorithm to identify the optimal policy, constructed by the sequence of actions with the highest rewards [2, 8].

We aim to demonstrate the efficacy of these approaches in a simulated 2D drone environment by showcasing the drone's adept navigation towards a series of targets. We compare the learning progress and performance of different drone flight controllers, presenting visualisations of learning curves to observe the algorithms' convergence.

II. METHODS

Environment: We use a 2D environment, with the drone's initial position at its centre. The goal is to reach the marked objectives and there are no obstacles. As the environment is 2D, the drone has two thrust values (T_1 and T_2), depending on which it can move vertically and horizontally. If $T_1 + T_2 > mg$ (the sum of the thrusts is larger than the force of gravity), the drone will move upwards along the y-axis. If $T_1 + T_2 < mg$ it will move downwards, and if $T_1 + T_2 = mg$ the drone remains still. For our simulation, we use $mg = 1N$. The thrusts' initial values are $T_1 = T_2 = 0.5N$, resulting in an equilibrium state. The thrusts power ranges from 0N to 1N. In the case of horizontal motion, one thrust has to be larger than the other. This would cause the drone to rotate and create an angle θ with the y-axis, the pitch of the drone. Positive θ leads to clockwise rotation, negative θ leads to counterclockwise rotation.

Discretisation of State Space: The continuous nature of the drone's environment necessitates discretisation of the state space. For H1 and H2, the states can be represented through discretisation of the parameters involved in altitude and pitch control, which then defines a finite set of state spaces. We achieve this through creating a range of parameter values through equal intervals. In the Q-learning model the discretisation process involves partitioning the continuous state variables of distance, velocity, pitch, and pitch velocity, into distinct bins or intervals. We create a total of five bins: distance bins (capturing the distance between agent and target), velocity bins (representing the drone's vertical velocity), pitch velocity bins (reflecting pitch angle's rate of change), directional bins (indicating agent's direction of movement along x and y axes), and pitch bins (distinguishing positive and negative pitch angles). These discretised states can be represented as indices corresponding to different bins, which serve as categorical representations. This breaks down the continuous information into discrete states, transforming the environment into a finite set of representative states, facilitating learning.

Discretisation of Action Space: Discretisation of the continuous action space is essential for the learning process. It enables the models to effectively navigate

through a range of actions and learn optimal policies. For H1 and H2, the action space is defined as the changes in parameters' value. These variations are evenly spaced within a specific range. In this way, the changes in the parameter values are discretised and define a finite set of actions. For the Q-learning model the action space discretisation involves selecting discrete actions that the drone can take at each time step through utilisation of H1 and H2. The decision to incorporate both heuristics stems from their complementary strengths. The first approach is simpler and results in rapid motion towards the target, while the second introduces constants that ensure movement stability. The parameter values of the two heuristics that lead to optimal performance are sampled, which defines a finite set of discrete actions.

Reward Function and Logic: The reward function has to be designed in a way that encourages learning of policies that reach the target in a time and resource-efficient manner. It is defined as follows:

$$R = \alpha_1(\delta) - \alpha_2(\sqrt{(dx)^2 + (dy)^2}) - \alpha_3(T_1 + T_2) - \alpha_4, \quad (1)$$

where δ is a binary variable that denotes whether the drone has reached the target, dx and dy denote the difference between the current and target position of the drone along the x and y axes, and α_1, α_2, α_3, α_4 are weight factors. This function gives the agent a positive reward for reaching the target and negative rewards for increased distance and thrust usage to encourage efficiency. We include a small negative constant to discourage prolonged episodes. This reward function also provides an accurate method to assess the agent's performance.

Training Regimes: We use a total of four training regimes on the drone controllers. First, we train them on a set of four deterministic targets. Each target is presented one at a time. Once the drone reaches all four targets one epoch of the simulation ends.

Second, we use two different sets of weight values for the reward function presented in Eq. (1). In the first regime (R1) we use $\alpha_1 = 100, \alpha_2 = 10, \alpha_3 = 1, \alpha_4 = 0.1$. In the second regime (R2) we make $\alpha_3 = 0.1$ and $\alpha_4 = 1$, while maintaining α_1 and α_2. In R2 the agent is punished less for thrust usage and more for episode length. These training regimes help define an optimal reward function.

The third regime involves training the Q-learning drone controller on a small state size of 96 and then on a larger state size of 384. A higher level of granularity of the state space can lead to more precise, informed decisions of the agent, but can also result in slower initial learning [2]. On the contrary, a lower state size can lead to faster initial learning, but can limit the ability of the agent to learn more precise and effective policies in later stages [4]. Hence, different levels of granulation of state spaces can significantly affect the agent's performance.

Finally, we train the Q-learning drone controller in low and high action spaces. The small action size condition consists of four defined actions (top 2 from each heuristic), reducing the complexity of the action space and allowing for faster exploration. The larger action size consists of six defined actions (top 3 from each heuristic). This helps the agent learn more expressive and optimal policies to solve the problem, but may hinder its ability to rapidly explore the environment.

Models

Heuristic 1 (H1): To change the drone's position the model varies the thrust values T_1 and T_2 as:

$$T_1 = f_{Tmin,Tmax}0.5 + dyk_y + \delta_\theta \quad (2)$$

$$T_2 = f_{Tmin,Tmax}0.5 + dyk_y - \delta_\theta \quad (3)$$

Here δ_θ denotes the required change in the drone's pitch to direct the agent towards the objective. It is calculated as $\delta_\theta = \theta' - \theta$, where θ' is the target pitch and can be derived through $\theta' = dxk_x$. The proportional constants k_y and k_x facilitate vertical and horizontal motion based on the distance between the target and the agent. This heuristic also incorporates a RL Q-learning tuning mechanism, which dynamically adjusts the parameters through an iterative process.

Heuristic 2 (H2): The drone's movement again results from variations in the thrust values. This approach introduces damping coefficients to allow for stable adjustments in altitude (b) and pitch (b_θ). Altitude adjustment can be summarised as $T = \frac{1}{2}mg + \varepsilon$, where:

$$\varepsilon = f_{\varepsilon_{min},\varepsilon_{max}}(k(dy) - b\dot{y}). \quad (4)$$

Hence, if $\varepsilon > 0$ the drone moves upwards the y-axis and downwards if $\varepsilon < 0$. Lateral adjustments are achieved through introducing a variable γ that affects the thrusts simultaneously: $T_1 = T_{eq} + \gamma$, $T_2 = T_{eq} - \gamma$, where:

$$\gamma = f_{\gamma_{min},\gamma_{max}}(k_\theta(\theta' - \theta) - b_\theta\dot{\theta}). \quad (5)$$

Therefore, lateral adjustments are achieved in respect to the pitch, which influences the drone's thrusts, rotating and moving it along the x-axis. The proportional and damping constants applied to the vertical and horizontal motion serve to prevent overshooting of the target or rapid movements leading to instability. This heuristic also incorporates a Q-learning-based tuning mechanism.

Q-Learning Model: The Q-learning model we used is an off-policy algorithm that is derived from Watkins' Q-learning [9]. It is summarised by the equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_a Q(s',a') - Q(s,a)], \quad (6)$$

where $Q(s,a)$ represents the expected reward for taking an action a in state s, s' and a' refer to the next state and action, respectively, and R denotes the obtained reward. In this algorithm, the agent executes an action, which results in a new state and reward acquisition. The Q-values are updated based on Eq. (6). This process repeats until no Q-values are updated in the last iteration.

The key hyperparameters are α (learning rate), γ (discount factor), and ϵ (exploration-exploitation trade-off). Essentially, α controls the rate of update of the Q-values.

Approach	Regime	G / step	Avg. Steps	Avg. thrusts	Avg. distance
Before tuning H1	R1	-6.21	508.2	0.74	0.47
	R2	-5.77	508.2	0.74	0.47
After tuning H1	R1	**-5.557**	682.6	**0.568**	**0.439**
	R2	**-5.299**	710.4	0.77	**0.42**
Before tuning H2	R1	-6.27	416	0.70	0.48
	R2	-5.90	**416**	0.70	0.48
After tuning H2	R1	-6.081	**378.2**	**0.624**	0.478
	R2	-5.644	655.8	0.668	0.46

TABLE I. Summary performance result after tuning of the heuristic parameters using RL based state-action pairs. Bolded text indicates best performance.

It is important to find a balanced value that ensures stable, precise convergence during the training process [4]. The discount factor governs the trade-off between prioritising future or immediate rewards [2, 5]. Representing the exploration rate, a high initial ϵ value in combination with an $\epsilon-$decay factor allows initial exploration of the state-action space with gradual shift to exploitation policy as learning progresses [4]. The drone controller's performance is evaluated through calculating cumulative rewards of multiple simulation episodes, in respect to the number of steps the agent takes in these simulations. Rewards are averaged over all evaluation episodes.

III. RESULTS

Comparison of the two heuristic approaches are summarised in Table I. They are evaluated based on the cumulative rewards they acquire per step. Noticeably, both H1 and H2 perform better after parameter tuning, as the cumulative rewards (G) for both approaches are improved. H1 performs better in G acquisition and average distance to target, after both heuristics' parameters are tuned. On the contrary, H2 can achieve significantly lower number of steps needed to reach the target. Hence, H1 is more suitable for quicker movement towards the target, while H2 provides improved motion stability, leading to a lower number of steps to get to the objective. Finally, training the heuristic controllers with R1 leads to lower average thrust values and step numbers, while training with R2 results in lower average distance.

These results are consistent with those presented in Table II, displaying the hyperparameter values which lead to optimal performance of the Q-learning model for all training regimes. When utilising the R1 training regime, this drone controller has, on average, lower thrust values and steps needed to reach the objective. Contrarily, training the controller with R2 leads to shorter average distance. Hence, the two reward training regimes impact the heuristic and Q-learning models' performance in different manners, which can be attributed to R1 penalising thrust usage more, resulting in higher thrust usage efficiency and reduction in the number of steps. Additionally, R2 uses higher time penalty, resulting in lower average distances. The higher step numbers when using R2 can be attributed to a loss of motion stability as the agent reduces the distance to the target more rapidly.

Regime	State size	Action size	α	γ	$\epsilon-$decay	Cum. Rewards / step	Avg. Steps	Avg. Thrust	Avg. Distance
R1	96	4	0.10	0.85	0.990	-5.277	1110.33	0.574	0.416
		6	0.05	0.95	0.990	-5.180	**1110.33**	0.572	0.407
	384	4	0.05	0.85	0.990	**-5.169**	1200.67	**0.565**	**0.407**
		6	0.10	0.85	0.990	-5.298	1137.33	0.566	0.418
R2	96	4	0.10	0.85	0.990	-5.106	1177.00	0.637	0.407
		6	0.05	0.95	0.990	-5.081	1188.67	0.656	0.404
	384	4	0.05	0.85	0.990	-5.022	**1170.33**	**0.598**	0.400
		6	0.10	0.85	0.995	**-4.987**	1181.33	0.638	**0.396**

TABLE II. Hyperparameters for Q-learning model for reward function, low/high state sizes and low/high action sizes training regimes. Bolded text indicates best performance.

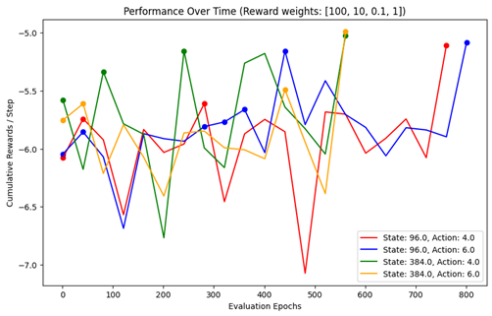

FIG. 1. Q-learning model learning curves across evaluation epochs, tracking enhancements until reaching peak performance. Four state-action pairs are depicted, focusing on the R2 reward regime.

The learning curve of the Q-learning model with respect to evaluation epochs is presented in Fig. 1. A general increasing trend in terms of G in each of the state-action pairs can be observed, with a reduction in initial fluctuations as learning progresses. Furthermore, pairs with higher action sizes eventually obtain a higher G per step than pairs with the same space sizes, but lower action spaces. Results also demonstrate the benefits of high state space sizes, as these pairs perform better in later stages of learning, converging significantly faster when compared to low state-action pairs. One explanation is that granularity of space sizes allows for more expressive policies that eventually lead to faster convergence to an optimal policy. The model performs better in cumulative rewards acquisition and average distance to target when compared with H1 and H2.

IV. CONCLUSIONS

Overall, we have explored the development of a 2D drone flight controller utilising RL. It encompassed parameter tuning of two heuristic approaches (H1 and H2) and a Q-learning model. Models were evaluated and trained on multiple reward regimes. H1 proved more useful for rapid movements towards the objective, while

H2 allowed for higher motion stability. The Q-learning model performed significantly better than the two heuristic approaches in terms of expected return, demonstrating its learning capabilities of optimal trajectories. A next step may involve training the drone on randomly selected target sets, enabling it to demonstrate proficiency in navigating through uncertain environments. Implementations of other models, like Deep Q-Networks (e.g. [2, 8]), may also be necessary to enable efficient utilisation of autonomous drone navigation systems.

[1] D. Floreano and R. J. Wood, Science, technology and the future of small autonomous drones, nature **521**, 460 (2015).

[2] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, Autonomous drone racing with deep reinforcement learning (2021), arXiv:2103.08624 [cs.RO].

[3] D. D. Nguyen, J. Rohacs, and D. Rohacs, Autonomous flight trajectory control system for drones in smart city traffic management, ISPRS International Journal of Geo-Information **10**, 338 (2021).

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction* (MIT press, 2020).

[5] D. F.-S. L. V. N. Huy X. Pham, Hung M. La, Autonomous uav navigation using reinforcement learning (2018), arXiv:1801.05086 [cs.RO].

[6] J. d. Silva Arantes, M. d. Silva Arantes, C. F. Motta Toledo, O. T. Junior, and B. C. Williams, Heuristic and genetic algorithm approaches for uav path planning under critical situation, International Journal on Artificial Intelligence Tools **26**, 1760008 (2017).

[7] R. Alyassi, M. Khonji, A. Karapetyan, S. C.-K. Chau, K. Elbassioni, and C.-M. Tseng, Autonomous recharging and flight mission planning for battery-operated autonomous drones, IEEE Transactions on Automation Science and Engineering **20**, 1034 (2022).

[8] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, and G. Casalino, Drone deep reinforcement learning: A review, Electronics **10**, 999 (2021).

[9] C. J. C. H. Watkins, Learning from delayed rewards, (1989).