AUTONOMOUS DRONE FLIGHT THROUGH HOOPS, REINFORCEMENT LEARNING

Rolando Esquivel-Sancho (s230025)

DTU Autonomous Systems

ABSTRACT

This paper explores the application of reinforcement learning (RL) in autonomous drone flight through hoops. Utilizing the Proximal Policy Optimization (PPO) algorithm in the NVIDIA Isaac Gym simulator, our study focuses on training a quadrotor to navigate through a target hoop. The methodology includes detailed task design, reward function formulation, and policy training.

The trajectory analysis presents smooth and controlled paths generated by the RL agent, emphasizing the effectiveness of the learned policies. The paper concludes with insights into the potential of RL in autonomous drone control, acknowledging challenges and proposing avenues for future research.

This work contributes to the evolving landscape of RL in robotics, showing some practical implications and advancements achievable in autonomous drone flight through the integration of learning-based methodologies.

1. INTRODUCTION

In recent years, there has been a notable surge in the exploration and application of reinforcement learning (RL) across various domains. This trend is particularly evident in the field of robotics, with significant advancements observed in the development of aerial robots. Within this context, flight control stands out as a significant area of interest, with specific emphasis on improving the control to perform extreme maneuvers, pushing these robots to their physical limits [1]. This intends to improve the performance in tasks involving the visitation of multiple waypoints, such as delivery, inspection, and drone racing, in which minimizing the time required is often a critical objective [2].

The control of these systems has been explored for many years and several approaches have been implemented based on traditional and also in sophisticated control architectures. While traditional methods have contributed significantly, recent works have unveiled the potential and benefits of integrating learning-based methods.

The code can be found in this repo

One of the most common learning-based methods used in this type of continuous task is the Proximal Policy Optimization (PPO) designed to train policies for reinforcement learning tasks. Introduced in [3], PPO operates on the principle of iteratively updating policies to maximize expected cumulative rewards while ensuring stable learning by following the same principle of Trust Region Policy Optimization (TRPO) method.

Integrating RL techniques in aerial robotics provides a dynamic method for optimizing flight control. The use of RL algorithms improves the accuracy and efficiency of flight robots, allowing them to better adapt to complex environments.

Despite these advances, it is important to acknowledge the challenges associated with integrating RL into flight robotics, where large amounts of data and computing resources are required to enhance the capabilities of aerial robotic systems.

The development of learning-based methods for aerial robots is an intensive data-driven process that requires highly parallelized simulations [4]. To address this, we leverage the capabilities of NVIDIA Isaac Gym to create the environments. This facilitates parallel simulation for thousands of multirotor robots, unlocking the full potential of learning-based methods in complex and challenging environments.

The subsequent sections will delve into the details of our methodology, covering aspects such as Quadrotor and Hoop Design, Reinforcement learning task, Reward function, and others. We used tools like NVIDIA Isaac Gym, and conducted highly parallelized simulations, paving the way for training scenarios involving thousands of multirotor robots.

2. RELATED WORK

Some interesting approaches have been developed in recent years to address the challenges and enhance the capabilities of flight robots, especially in reinforcement learning (RL) applications

Among these, the project presented at [5] is one of the most relevant, where researchers successfully designed a quadrotor to compete in Autonomous Drone Raicing, this drone is capable of surpassing both human pilots and advanced controllers. This achievement highlights the potential

of RL-based methods in achieving high-performance outcomes in dynamic and competitive environments.

Another contribution is the work in [6], where the authors introduce adaptive learning algorithms to control a flying robot. The proposed system dynamically adjusts the controller parameters based on the robot's performance and environmental conditions, resulting in improved flexibility and robustness

Furthermore, the advances in swarm robotics are presented in [7], the authors demonstrate the potential and state-of-the-art reinforcement learning-based control systems for swarm robotics.

In addition, the work [8] focuses on trajectory optimization for flying robots using deep reinforcement learning. By leveraging RL techniques, they achieve precise trajectory planning and control, essential for applications such as inspection and surveillance.

3. METHODOLOGY

In this section, we present the methodology employed for designing the task within the NVIDIA Isaac Gym simulator. The task involves a quadrotor navigating through a designated space to reach a dynamic target and pass through a hoop, establishing a challenging reinforcement learning (RL) environment.

3.1. Quadrotor and Hoop Design

We developed a new task for the NVIDIA Isaac Gym simulator, aligned with the structure presented in [9]. This task is created based on the VecTask class to keep the structure and ensuring consistency with the standard method available in the repository Link This approach facilitates the parallelization process and the use of Isaac Gym's APIs as long as the package configuration.

The environment space is represented by a box with dimensions of 5x5x2.5 meters. Within this environment, there are two actors: a quadrotor and a target. The quadrotor is initialized with a random position in the x and y directions, in a range of ± 1.5 meters from the center of the box. The z position can take values between 0.5 and 2.5 meters.

The target consists of two elements: the target point (serving as the goal for the drone) and the hoop position, located 0.3 meters ahead of the target point. This positioning ensures that the drone passes completely through the hoop. The target can be configured with a fixed position or a random position. In both cases, the hoop is set parallel to the xz plane.

To model the Quadrotor, we utilized a URDF file, which was made based on the model presented in [4], with adjustments in the physical configuration. Figure 1 provides a visualization of the model, where the red color corresponds to the front left propeller, green to the front right propeller, and blue to the back propellers. The dimensions of the model are described in Table 1

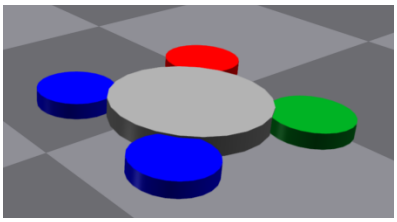


Fig. 1: Drone model, from URDF file.

Table 1: Drone model dimensions.

Element	Radio (m)	Length (m)
Base	0.1	0.025
Propeller	0.05	0.02
Collision mesh	0.2	0.04

For the hoop, we create a URDF file that models a square frame structure, with dimensions of 0.7x0.7 meters.

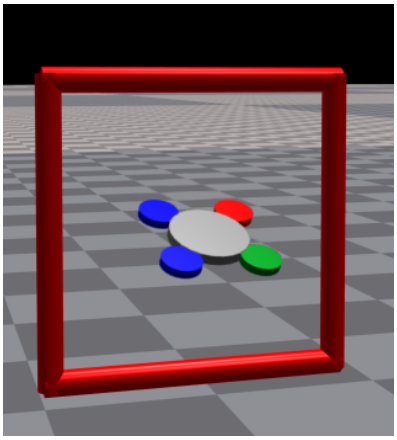


Fig. 2: Hoop and quadrotor model.

To handle the collision we used the collision method provided by Isaac Gym and created the collision mesh for each actor, for the drone this mesh is modeled as a cylinder with a radius of 0.2m and a height of 0.04m, as detailed in Table 1. This cylindrical structure encapsulates the drone, providing a representation of collision events with a single-body structure. On the other hand, the collision mesh for the hoop aligns with the physical structure of the hoop. Figure 3 visually depicts the collision meshes of both actors the drone and the hoop.

3.2. Reinforcement learning task

The task is modeled using the definition of a Markov Decision Process (MDP) as described by the tuple $(\mathcal{S}, \mathcal{A}, p, \gamma)$, follow-

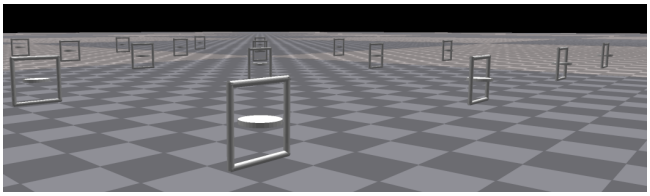


Fig. 3: Collision mesh, drone and hoop

ing the framework introduced by Sutton and Barto in [10]. Here, $\mathcal{S}$ represents the set of all possible states, $\mathcal{A}$ denotes the set of all possible actions, $p(r, s'|s, a)$ is the joint probability of a reward r and the next state s', given the current state s and action a, and γ is a discount factor that balances the trade-off between later and earlier rewards.

As mentioned before, $\mathcal{S}$ denotes the set of all possible states, each representing the current situation or condition of the environment observed by the agent. For this observation, we used an instance of type *Box*, describing an n-dimensional continuous space. This space is bounded, allowing us to define upper and lower limits that specify the valid values our observations can take, as detailed in [11].

The observation space consists of 13 values, wherein the first three ($[0:3]$) represent the quadrotor's position relative to the target, the next four ($[3:7]$) denote the drone's orientation with respect to the target using quaternions, the subsequent three ($[7:10]$) correspond to linear velocities, and the final three ($[10:13]$) the angular velocities.

Concerning the action space $\mathcal{A}$, we utilize four values that correspond to the thrust of each propeller. These values are bounded between -1 and 1 for normalization, ensuring stability and compatibility with the utilized framework.

The definition of the observation and action spaces is as follows:

```
{'observation_space': Box(-inf, inf, (13,),
float32), 'action_space': Box(-1.0, 1.0,
(4,),float32)}
```

3.3. Reward function

The reward function is designed to guide the quadrotor to reach the target while simultaneously passing through the hoop. The function is defined as follows:

$$r = r_p + r_p * (r_u + r_s) + r_c + r_t \tag{1}$$

where r_p represents the position reward and is defined as:

$$r_p = \frac{2.0}{1.0 + d_t^2} \tag{2}$$

Here d_t denotes the Euclidean distance to the target.

The uprightness reward, r_u, and spinning reward, r_s, are intended to maintain the drone upright and prevent spinning when it is close to the target. They are defined as:

$$r_u = \frac{1.0}{1.0 + u^2}, r_s = \frac{1.0}{1.0 + s^2} \tag{3}$$

where u represents the uprightness factor, and s represents the spinning factor.

The collision reward r_c is used to penalize the agent when a collision with the hoop is observed, it is given by

$$r_c = \begin{cases} -20.0 & \text{if collision with gate} \\ 0.0 & \text{otherwise} \end{cases} \tag{4}$$

And, r_t is the target reward, this reward is given when the drone reaches the target, and it is described as follows

$$r_t = \begin{cases} 10000.0 & \text{if } (d_{tp} < 0.05 \text{ and } d_{tn} < 0.4) \\ 0.0 & \text{otherwise} \end{cases} \tag{5}$$

Where d_{tp} is the distance between the drone and the target plane, and d_{tn} is the distance to the normal axis of the target.

3.4. Reset function

Aligned with the reward function, the reset function is intended to reset the environment when the drone reaches an unhealthy condition or when it reaches the target position. This function is given by:

$$\text{reset} = \begin{cases} 1 & \text{if } d_t > 8.0 \text{ or } z_{\text{drone}} < 0.5 \text{ or } u < 0 \\ 1 & \text{if } d_{tp} < 0.0 \text{ and } d_{tn} \geq 0.4 \\ 1 & \text{if } r_t > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

As unhealthy conditions, we consider the following criteria: the drone distance to the target, the drone's z position, a negative uprightness factor, and if the drone crosses the target plane without passing through the hoop.

3.5. Policy training

As mentioned in subsection 3.2, the observation space consists of 13 values represented by the vector o_i described in Equation 7. This vector provides information about the drone in the i-th environment.

$$\begin{aligned} o_i = [&x, y, z, q_1, q_2, q_3, q_4, \\ &v_{lx}, v_{ly}, v_{lz}, v_{ax}, v_{ay}, v_{az}] \end{aligned} \tag{7}$$

Using Isaac Gym Simulator allows us to perform policy rollouts on 4096 environments in parallel, resulting in a significant speed-up of the data collection process. This enhances the diversity of interactions with the environment.

To process this data and to enable the drone to navigate effectively and solve the task, we use the Proximal Policy Optimization (PPO) algorithm [3]. Which is well-suited for optimizing systems in continuous control environments, making

it a strong choice for training drones and other automated operators. Our implementation utilizes the PPO algorithm from the rl-games library [12], this library is provided as part of the Issac Gym simulator.

The network architecture is configured as presented in Figure 4. The structure of the network is designed to capture the complexity of the task, ensuring that the learned policies are both effective and efficient in navigating through the defined environments.

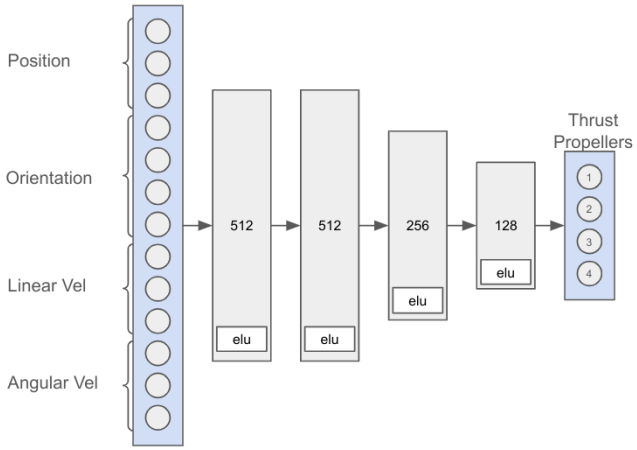


Fig. 4: Network architecture

4. RESULTS

In this section, we present the results obtained from our training experiments using the Proximal Policy Optimization (PPO) algorithm. The outcomes provide insights into the learning progress, performance metrics, and the overall effectiveness of our proposed methodology.

4.1. Training

The training phase is a critical aspect of our study, where the reinforcement learning agent learns to navigate and accomplish the defined task. We evaluate the training progress through key metrics, including the training reward and episode length.

Figure 5 shows the training reward during the training iterations. The plot shows a clear increasing trend in the training reward, indicating that the agent is learning and accumulating more rewards as the training progresses. Eventually, the reward converges to a stable value, suggesting that the agent has found a maximum of the policy function. Which means that the agent has learned a policy.

In Figure 6, we observe the length of episodes during the training process. Initially, the episode length increases, reflecting the exploration phase where the policy contains

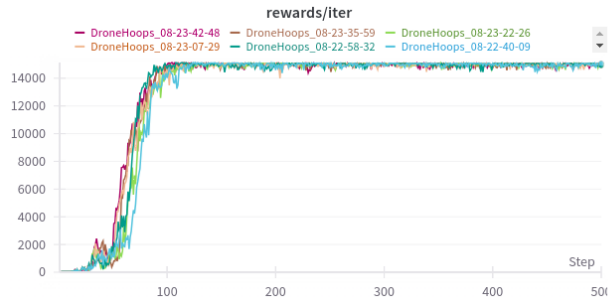


Fig. 5: Training reward

randomness, and the agent explores different actions. However, as the agent learns and the reward increases, the episode length starts to decrease. This reduction in episode length signifies that the agent becomes more efficient in navigating the environment, completing episodes more quickly as it refines its policy.

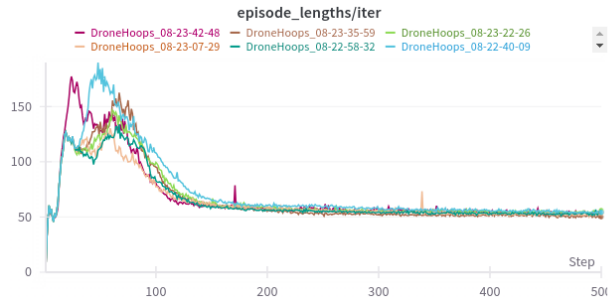


Fig. 6: Episode length

4.2. Inference

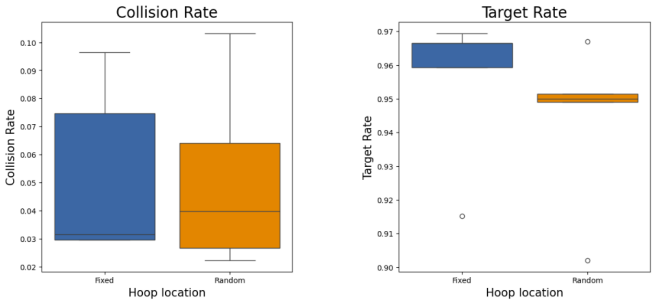
To evaluate the performance of the trained task, we used the trained agent in the test configuration. This inference mode provided valuable data, including actions taken by the agent, obtained rewards, collision occurrences, successful target reach events, and the number of iterations.

We conducted two short experiments using different configurations of the hoop, as described in subsection 3.1. One experiment with the static hoop configuration, while the other set the hoop in a random position. A total of 10 agents were trained for each experiment, with 1000 episodes tested per agent.

Some of these results are shown in Figure7. Where the collision rate figure (Figure 7a) provides insights into the frequency of collisions during the inference phase. Analyzing this rate is crucial for assessing the agent's ability to navigate through the hoop without collisions. For both experiments, the resulting rate was approximately 5%, indicating that in around 5% of the episodes, the drone experienced a collision.

On the other hand, the target reach rate figure (Figure 7b) offers information on the agent's success in reaching the predefined targets. A high target reach rate, such as the obtained 95%, indicates the effectiveness of the learned policy in accomplishing the desired task.

Based on this data and the number of samples, we cannot determine that one configuration works better than the other.



(a) Collision rate. (b) Target rate.

Fig. 7: Inference results of collision and target rates

Additionally, we can assess the behavior of the agent by observing the trajectory path followed by the quadrotor. Figure 8 displays the trajectory plot of multiple episodes. From this plot, we can observe how the agents generate a smooth trajectory towards the target. This observation is further detailed in Figure 9, this top-view of the trajectory exposes the path in x and y direction. Typically, these directions demand more control complexity in traditional control approaches.

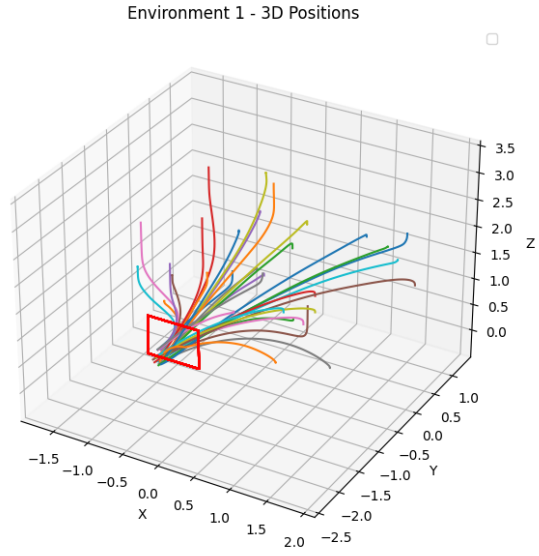


Fig. 8: Drone trajectory paths

Figure 10 shows the actions applied to each propeller in five episodes. It is evident that propellers 1 and 3, positioned

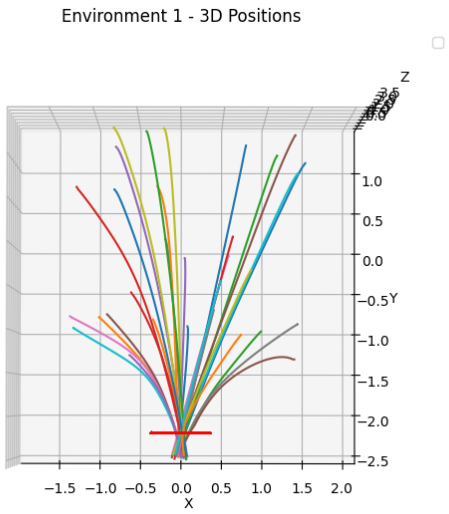


Fig. 9: Drone trajectory paths (top-view)

farthest from the hoop, exerted maximum thrust at the beginning of the episode to orient the drone in the direction of the hoop. Subsequently, the remaining propellers worked to compensate and stabilize the quadrotor while flying toward the hoop.

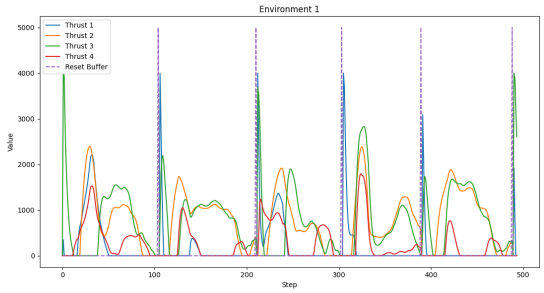


Fig. 10: Thrust applied to each propeller in 5 episodes, with dotted lines indicating the end of each episode.

5. CONCLUSIONS

In this work, we have explored the application of reinforcement learning (RL) in the context of autonomous drone flight through hoops. The use of RL algorithms, particularly Proximal Policy Optimization (PPO), has shown promising results in enhancing the control and maneuverability of aerial robots. Our methodology involved designing a task in the NVIDIA Isaac Gym simulator, incorporating a quadrotor and hoop, and training the agent using the PPO algorithm.

Moreover, the target reach rate was high, reaching approximately 95%, indicating the success of the learned policies in

accomplishing the specified task. Trajectory plots revealed smooth paths generated by the agents, showcasing their ability to navigate in a controlled and precise manner.

Our work contributes to the growing field of RL in robotics, providing insights into the effectiveness of learning-based methods for autonomous drone control. Despite the challenges associated with integrating RL into flight robotics, our results demonstrate the potential benefits, especially when leveraging parallelized simulations and advanced tools like NVIDIA Isaac Gym.

As future work, further optimization of the reward function, exploration of different RL algorithms, and experimentation with more complex scenarios could be pursued to enhance the capabilities of autonomous drone flight through hoops. Additionally, real-world testing and validation of the trained policies could provide valuable insights into the generalization of the learned behaviors.

6. REFERENCES

[1] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza, "Autonomous drone racing with deep reinforcement learning," 2021.

[2] Philipp Foehn, Angel Romero, and Davide Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, July 2021.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," 2017.

[4] Mihir Kulkarni, Theodor J. L. Forgaard, and Kostas Alexis, "Aerial gym – isaac gym simulator for aerial robots," 2023.

[5] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[6] Ali Barzegar and Deokjin Lee, "Deep reinforcement learning-based adaptive controller for trajectory tracking and altitude control of an aerial robot," *Applied Sciences*, vol. 12, pp. 4764, 05 2022.

[7] Marc-Andrė Blais and Moulay A. Akhloufi, "Reinforcement learning for swarm robotics: An overview of applications, algorithms and simulators," *Cognitive Robotics*, vol. 3, pp. 226–256, 2023.

[8] Harald Bayerlein, Paul De Kerret, and David Gesbert, "Trajectory optimization for autonomous flying base station via reinforcement learning," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.

[9] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.

[10] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, second edition, 2018.

[11] Gymnasium Documentation, "Basic usage - gymnasium documentation," 2023, Accessed: 01-07-2024.

[12] Denys Makoviichuk and Viktor Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," `https://github.com/Denys88/rl_games`, May 2021.