Reinforcement Learning for Agile Quadrotor Control with CTBR in Isaac Gym

s232896 Yiming Zhang, s232179 Huan Hu
Supervisor: Silvia Tolu, Luca Zanatta, Sebastiano Mengozzi

***Abstract*—Reinforcement Learning (RL) has gained prominence in various domains, particularly in robotics, for its ability to handle the complex dynamics of robot control tasks without requiring explicit modeling of these dynamics. RL facilitates the development of control policies through a process of exploration and exploitation, enabling robots to refine their strategies and achieve optimal behaviors. This adaptability is especially beneficial for quadrotors, which are nonlinear dynamical systems that challenge traditional control methods. RL-based control policies offer a solution by enabling direct mapping from high-dimensional sensory inputs to actions. Furthermore, training these policies in simulation is both efficient and practical, circumventing the sample inefficiency issues encountered when training on real hardware. In this project, we make two contributions:(i) We have created a dataset with complex trajectories, significantly surpassing the quantity of previous works. Additionally, we developed a method to assess the complexity of these trajectories and effectively categorized the complexity levels of agile trajectories. (ii) We managed to train the quadrotor to perform difficult maneuvers and can follow complex flight trajectories in a simulation environment.**

SUPPLEMENTARY MATERIAL

Project repository: https://github.com/zerojuhao/isaacgym.git
For demo videos, please visit: https://youtu.be/mamfwyh-Cug?si=QimOkQi0XmyVdKpa

I. Introduction

Reinforcement Learning (RL) has become a prevalent method across numerous fields [1], especially in robotics. Traditional control techniques frequently struggle with the complex dynamics involved in robot control tasks. RL addresses this issue by bypassing the requirement to explicitly model the robot's dynamics. Instead, it enables robots to develop their control policies through a balance of exploration and exploitation during training [2]. This means that RL allows robots to learn optimal behaviors by trying out various actions and refining their strategies based on the outcomes [3], leading to more effective and adaptable control solutions.

Quadrotors, characterized by their highly nonlinear dynamical systems, present significant challenges in pushing their physical limits, even with carefully tuned controllers. Previous research has proposed various learned control policies using different control input modalities for the underlying platform. Some approaches directly specify motor commands [4], [5], [6], while others output desired collective thrust and body rates [7], [8], which are then executed by a low-level controller. Additionally, some methods output velocity commands [9], [10] that are carried out by a control stack, or even a sequence of future waypoints [11]. However, most published approaches do not provide a justification for their choice of control input modality. This lack of explanation hinders performance comparisons among different approaches and, consequently, scientific progress. Due to the high sample complexity of learning-based policies, they are often trained in simulation. Reinforcement Learning (RL)-based control policies enable direct mapping from high-dimensional raw sensory inputs to actions. Training these policies in a simulation environment is both efficient and practical, addressing the inherent sample inefficiency of training on real hardware. Agile quadrotor flight requires fast and accurate control strategies, making it a particularly challenging problem. Recently, numerous learning-based controllers have been proposed for quadrotors. Unlike traditional control methods, [4], [11]–[13] present these learned control policies can directly map sensory information to actions, eliminating the need for explicit state estimation.

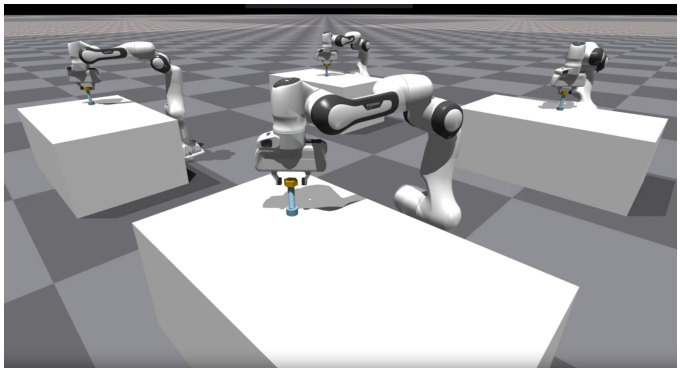

Fig. 1: Isaac Gym environment

The recent advancements in simulation software, such as Nvidia Isaac Gym, have further facilitated the adoption of DRL techniques. These simulators mitigate the traditionally prohibitive data and trial requirements by enabling parallel environment generation and realistic interactions, thus expediting the training convergence of neural networks.

Isaac Gym offers a high performance learning platform to train policies for wide variety of robotics tasks directly on GPU and is characterized with blazing fast training times for complex robotics tasks [14]. Also, it supports simultaneously RL training with multiple environments as which is shown in Figure 1. We implemented PPO algorithm through skrl, a reinforcement learning library for Isaac environments.

Significant strides in Artificial Intelligence (AI) paradigms and sophisticated optimization algorithms are also contributing

to this technological revolution. These advancements, driven by extensive datasets, realistic physical simulations, and enhanced hardware accelerators for massively parallel computation, are pushing the boundaries of AI capabilities. As a result, we are on the verge of achieving autonomous and intelligent robotic systems poised to transform daily human activities. These robots will eventually replace humans in performing hazardous and monotonous tasks, thereby expanding human potential and reshaping societal structures—from education and work to our broader existential perspectives.

Despite these advancements, a critical gap remains: the development of reliable algorithms that enable robots to navigate and comprehend a human-centric world as intuitively as humans do. Quadrotors, a popular subject within the robotics community, present multiple engineering challenges due to their mechanical architecture. These challenges include performing high-speed maneuvers in complex environments, which test the limits of flight autonomy, given the constraints of small batteries and low energy density. The inherent instability and highly nonlinear dynamics of quadcopters necessitate advanced control mechanisms, all while operating within the limited computational power of embedded systems.

The overarching aim of this project is to establish a framework for robotic learning that mirrors the adaptive and responsive behaviors observed in animals. Specifically, this work focuses on leveraging DRL techniques to enable a quadrotor to perform agile flights by navigating through a fast-moving gate to reach a target position without collisions, akin to human piloting. This involves training a neural network to map state observations directly into motor control commands, bypassing explicit perception, trajectory planning, and model-based control stages.

This not only contributes to the field of robotics autonomy but also paves the way for future research in zero-shot transfer from simulation to reality. By learning in a high-fidelity, randomized setup, robots can discover a wide range of behaviors and policies, leading to more robust and generalizable frameworks for rapid development and prototyping of high-performance robotic systems.

In the following report primarily includes our four main works: we managed to create a dataset, which outperform the related work in quantity.

Moreover, we tried many ways to assess the complexity of these trajectories and in the end, we find a suitable method to achieve that and also it is enable to effectively categorize the complexity levels of agile trajectories.

Additionally, We are able to train the quadrotor to perform difficult maneuvers and can follow complex flight trajectories in a simulation environment.

Finally, we conduct a comparative analysis based on the baseline to evaluate the performance and draw conclusions from the analyzed data.

II. RELATED WORK

Compared to specifying linear velocity commands, controlling collective thrust and body rates has been demonstrated to enable significantly more aggressive maneuvers. In [8], a racing policy is designed to directly map image observations to collective thrust and body rate commands. While this policy performs well in simulation, navigating challenging race tracks, it has not been deployed on a real platform. The authors in [15] propose a hybrid approach that combines a classical controller with a learned residual command to compensate for aerodynamic disturbances like ground effect during near-hover flight. In another study [7], privileged learning is used to imitate a model predictive controller (MPC) for executing acrobatic maneuvers. This method successfully demonstrated acrobatic flight on a real platform, but it was limited to single maneuvers and required a separate policy for each trajectory.

[16] led by Elia Kaufmann finished a benchmark comparison of learned control policies for agile quadrotor flight. They compared policies that specify individual rotor thrusts, collective thrust and bodyrates, and linear velocity commands. While all tested policy types were able to learn a universal flight controller, they differed strongly in terms of peak performance and robustness against dynamics mismatch. And they finally identified that policies producing collective thrust and bodyrates present strong resilience against dynamics mismatch and transfer well between domains while retaining high agility.

However, they use only 8 trajectories for testing, design the reward using timestep, and did not quantitatively grade the complexity of the trajectories. In this project, We created more trajectories as a dataset and proposed a new method for the design of reward and evaluating trajectory complexity.

III. BACKGROUND

A. Reinforcement Learning

Reinforcement Learning is a subset of machine learning where the RL agent learns to make decisions by interacting with an environment, as illustrated in Figure 4.

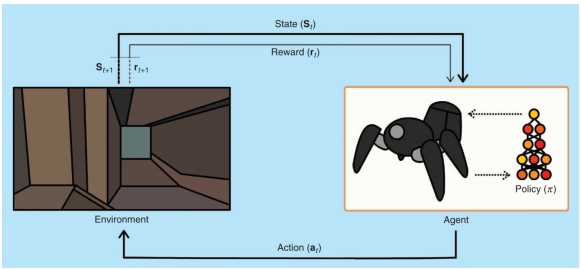

Fig. 2: How RL agents learn by interacting with environment [1]

The agent leverages exploration and exploitation to gather observations that enhance its policy through environmental interactions. By trial and error, the RL agent can often outperform humans in completing complex tasks. RL finds applications in various domains, including robotics, gaming, and autonomous driving.

Reinforcement learning algorithms are primarily divided into two categories: value-based algorithms, such as deep Q-network, and policy search algorithms, like Policy Gradient.

This project employs Proximal Policy Optimization (PPO), an advanced RL algorithm for continuous action spaces. PPO limits significant policy updates to prevent large deviations

from the existing policy. Utilizing a replay buffer, the training efficiency of PPO is enhanced since the stored data can be reused for multiple training steps.

B. Proximal Policy Optimization

Proximal Policy Optimization (PPO) is an efficient and stable reinforcement learning algorithm widely used for training neural network policies. It balances simplicity and performance by limiting policy updates to prevent drastic changes, ensuring stable and efficient learning.

Clipped Objective Function: PPO uses a clipped surrogate objective to restrict policy updates, enhancing stability:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{1}$$

where $r_t(\theta)$ is the probability ratio of new to old policies, and $\hat{A}_t$ is the advantage estimate, typically around 0.1-0.2. The advantage $\hat{A}_t$ compares action a_t to the average. By optimizing this clipped objective, PPO bances exploiting knowledge and exploring new actions to learn stably. In this project, PPO is used to train the quadrotor's neural network policy, allowing it to perform agile maneuvers efficiently and stably, adapting to complex flight trajectories.

Algorithm: The Proximal Policy Optimization (PPO) algorithm, which employs fixed-length trajectory segments, is described as follows. In each iteration, N parallel actors gather T timesteps of data. The surrogate loss is then formulated based on these NT timesteps of data and optimized using minibatch Stochastic Gradient Descent (SGD) or, more commonly for improved performance, the Adam optimizer, over K epochs. Each step of the PPO algorithm is shown in the Figure 3 below.

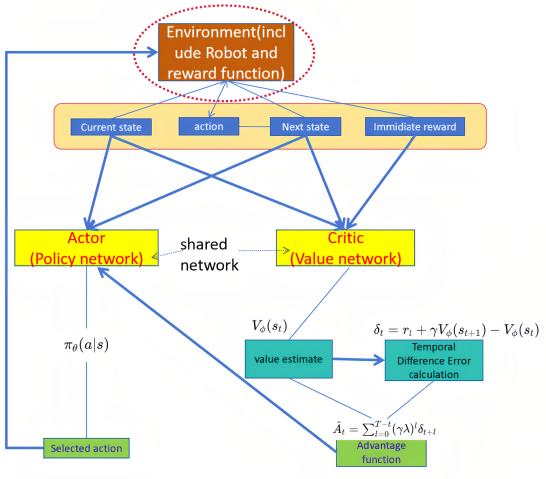

Fig. 3: Proximal Policy Optimization algorithm diagram

C. Collective Thrust & Bodyrates Control

CTBR controller affects the drone's action by controlling its collective thrust and body rate. Unlike high-level command generation, specifying collective thrust and body rates does not require full state estimation of the platform [17]; instead, it only needs inertial measurements to perform feedback control on the body rates. This information is readily available at high frequencies in modern flight controllers, [18] making collective thrust and body rates the preferred control input modality for professional human pilots. The algorithm diagram is shown below in figure 4. Also, in the simulation environment, the delay part inside the control loop can be omitted.

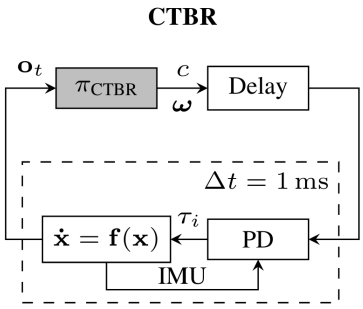

Fig. 4: CTBR controller strategy

D. Isaac Gym

Isaac Gym is an advanced simulation platform developed by NVIDIA specifically designed for training and developing reinforcement learning (RL) models in robotics and other dynamic systems [14]. It leverages the power of GPU acceleration to provide high-fidelity physics simulations at unprecedented speeds, enabling efficient training of complex control policies.

In our project, Isaac Gym is used to simulate the flight dynamics of a quadrotor. By accurately modeling the physics of quadrotor flight, we can train reinforcement learning policies to perform agile maneuvers. The high performance and scalability [19] of Isaac Gym allow us to generate a large dataset of flight trajectories and optimize the control policies efficiently. This simulation capability is critical for ensuring that the trained policies are robust and capable of transferring to real-world quadrotor platforms [20].

IV. METHODOLOGY

A. Complexity of trajectory

In our previous attempts to evaluate the complexity of trajectories, we explored algorithms based on the volume of three-dimensional space occupied by the trajectory, as well as the accumulation of the trajectory's straightness and density entropy. However, these algorithms presented certain issues and limitations during testing. In the end, we managed to find a suitable function and we adopted a curvature-based algorithm. The evaluation value we obtained is related to the average curvature, effectively avoiding the problem where the number of sampling points significantly influences the final result, which was a major drawback of the accumulation algorithms.

We calculate the complexity using the following formula:

$$C = \frac{\sum_{i=0}^{n-1} l_i}{\log(\sum_{i=0}^{n-2} \alpha_i)} \tag{2}$$

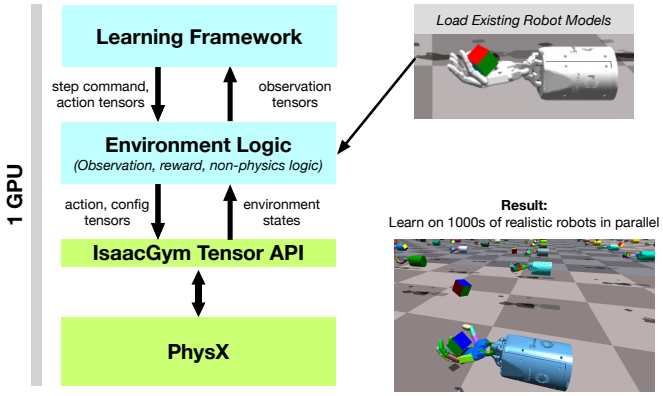

Fig. 5: Isaac Gym Structure

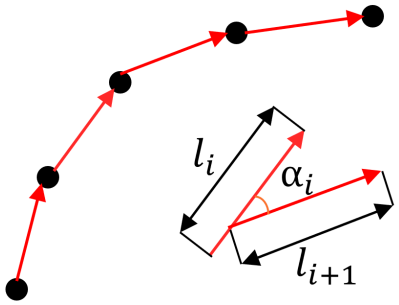

Fig. 6: Evaluation of the complexity

In a trajectory, as in Figure 6, two neighboring points form a vector, and two vectors form an angle. In the formula above, l_i is the length of the vector and α_i is the angle. This formula reflects the complexity of the trajectory by calculating the average rate of change of angle with distance.

B. Assigning target points

In the initial state, we assign 3 future target points for the drone. And here we define: when the distance between the drone and the 1st target point is less than 5 cm, it represents that the drone is very close to the target point. At this point, 3 new target points can be updated. The information of these points will be used to compute rewards and inputs to the neural network.

C. Reward design

Reward consists of three categories: distance, access, and velocity.

$R_{distance}$: There are three rewards regarding distance, and

$$R_{distance} = R_1 + R_2 + R_3 \tag{3}$$

1) R_1: We use the formula below to calculate the first part of reward regarding distance, that is R_1. Here $distance_i$ is the distance between the drone's current position and its 3 future target points.

$$R_1 = \frac{1}{1 + distance_1^2} + \frac{1}{1 + distance_2^2} + \frac{1}{1 + distance_3^2} \tag{4}$$

2) R_2: We record the current position of the drone and the position of the last step. Only if the Euclidean distance between the current position and the first future target point decreases compared to the previous step, the reward is given, which is the same as R_1. In all other cases, the reward is 0.
3) R_3: Similar to the last oneonly if the distance between the current position and the first future target point decreases in x-y-z 3 directions compared to the previous step, the reward is given, which is the same as R_3. In all other cases, the reward is 0.

R_{access}: When the distance between the drone and the first future target point is less than 5 centimeters, this reward can be earned, which is a relatively large number. Here we set

$$R_{access} = 10 \tag{5}$$

$R_{velocity}$: The reward regarding velocity is the value of velocity, which means that the greater the velocity, the greater the reward.

$$R_{velocity} = velocity \tag{6}$$

To summarize, the total reward is:

$$Total\ Reward = R_{distance} + R_{access} + R_{distance} * R_{velocity} \tag{7}$$

D. CTBR control and air resistance

In this project, CTRB control takes care of:

- tracking the body rates through a $Proportional + Feedback$ linearization controller.
- saturating inputs to the quadrotor

The controller controls the motion of the drone by processing the input from NN and outputting torque in all three directions x-y-z and a collective thrust in the z-direction only.

Here, we use the formula below to simulate air resistance:

$$f = -k * sign(velocity) * velocity^2 \tag{8}$$

where k is the coefficient of friction, $sign$ is used to get the sign of velocity. Here we set $k = 0.001$.

E. System overview

The structure is shown in figure 7. The inputs to the neural network are 19 elements obtained using API from Isaac Gym, which are the difference between the position of the UAV and three future points in the xyz direction(3 × 3), quaternion(4), linear velocity(3), and angular velocity(3). The neural network then outputs torque(3) and collective thrust(1) to the CTBR controller. The controller further processes these inputs and finally continue to output torque and thrust to be applied to the UAV in the environment.

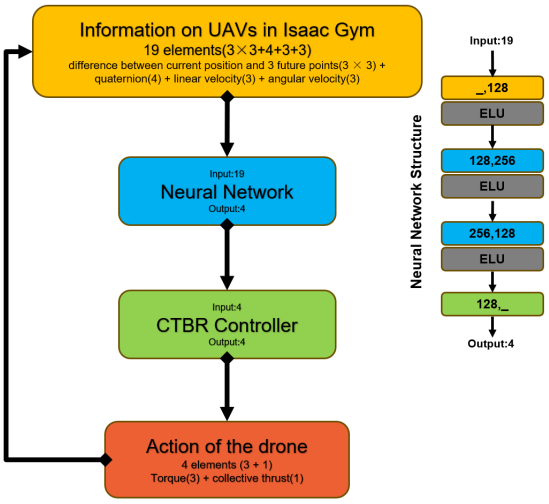

Fig. 7: System structure

V. EXPERIMENTS AND RESULTS

We created simulation environment in Isaac Gym first. Then we finished the design of reward function and the build of dataset. We successfully constructed several three-dimensional curve shapes using parametric equations to represent trajectories, supplemented by common flight paths. In the end, we used Python to plot these trajectories in space, generating a dataset with 36 different trajectories. These trajectories exhibit varying complexities and present different shapes.

Figure 8 shows some common curves in the dataset , including straight lines, circles, ellipses, and spirals etc.(variants of the same shape are not displayed). Uncommon curves are shown in the figure 9(similar variants are also not included).

In this project, we only select common trajectories for evaluating complexity and training.

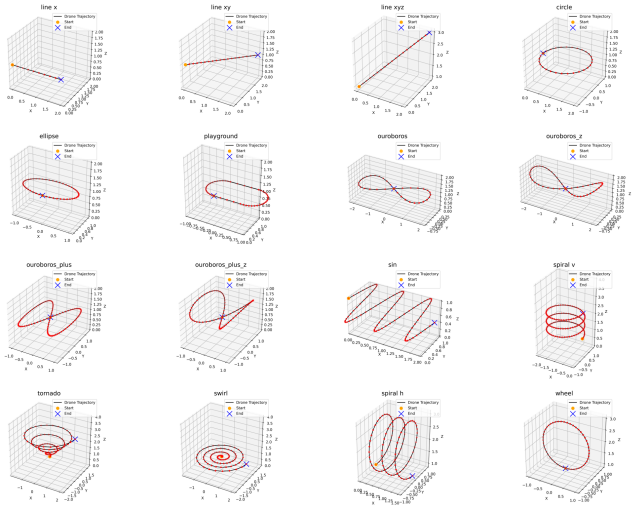

Fig. 8: Common Trajectories in the Dataset

Then we added CTBR controller (provided by Sebastiano Mengozzi) to process the output of neural network. The implementation of RL training part is build upon *skrl*, a specialized RL library designed to solve reforcement learning tasks.

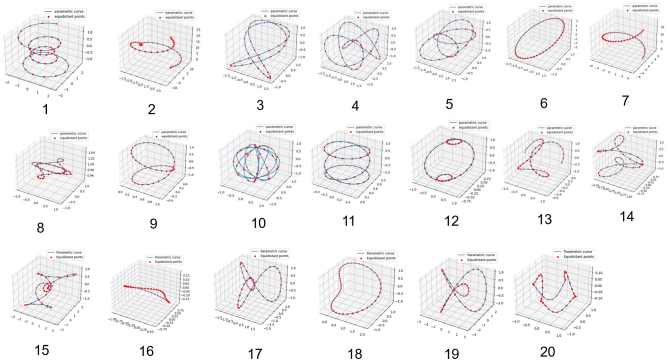

Fig. 9: Examples of Uncommon Trajectories in the Dataset

We successfully completed the evaluation of trajectory complexity and the training of the neural network, and tested the performance on these common trajectories. We created 1000 environments in Isaac Gym and trained the drones using the NVIDIA 4060 Laptop GPU. For most complex trajectories, the neural network was able to converge in less than 160 steps. The results of the evaluation of the trajectory complexity are shown in figure 10. Regarding the training, some hyperparameters are set as shown in Table I.

TABLE I: Training Hyperparameters

Hyperparameter	Value
rollouts	16
γ (discount factor)	0.98
Actor learning rate	3e-4
Critic learning rate	3e-4
Entropy regularization	1e-2
ε (importance ratio clipping)	0.2

It is important to note that **here rollouts must be** ≥ 16 **to avoid error in Isaac Gym**.

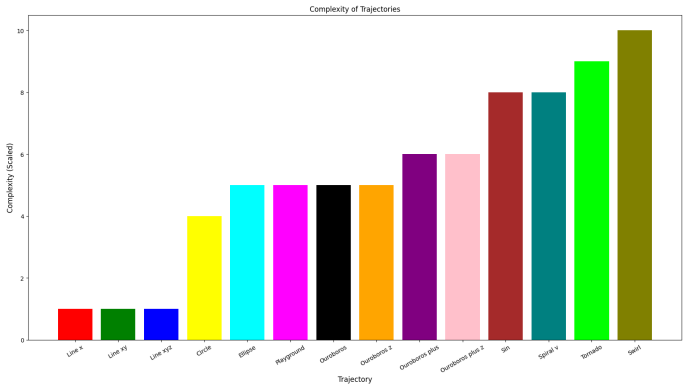

Fig. 10: Evaluation of complexity

Regarding the calculation of positional tracking error, we connect the line between the previous target of the drone and its 1st future target. The perpendicular distance between the drone and this line is the error data we need. Table II shows the mean, standard deviation, maximum and minimum values of the error.

TABLE II: Positional tracking error in centimeter

Trajectory	Error	Max	Min
Line x	1.74 ±0.87	3.62	0.02
Line xy	1.74 ±0.91	3.71	0.02
Line xyz	1.75 ±0.92	3.72	0.03
Circle	1.08 ±0.69	3.53	0.03
Ellipse	1.12 ±0.89	3.65	0.05
Playground	1.09 ±0.75	3.54	0.04
Ouroboros	1.34 ±0.92	3.82	0.05
Ouroboros z	1.35 ±0.91	3.84	0.05
Ouroboros plus	1.37 ±0.95	3.75	0.06
Ouroboros plus z	1.41 ±0.82	3.85	0.06
Sin	1.86 ±0.98	4.51	0.23
Spiral v	1.65 ±0.71	3.94	0.09
Tornado	1.70 ±0.98	5.21	0.03
Swirl	1.52 ±0.95	3.82	0.06

In these trajectories, the drone can achieve a maximum speed of 6m/s with a low error. A series of videos illustrating the test on these trajectories is available on **YouTube**.

VI. CONCLUDSION

In this project, the trained model with the CTBR controller demonstrated the ability to follow complex curves, indicating that the drone can perform agile and flexible maneuvers. Our experimental results show that, although the trained drone exhibited larger errors in tasks such as straight-line dives and cruising, it maintained relatively low errors on complex curves. As the trajectory complexity increased, the error values showed only a minor increase compared to simpler curves, such as circles. It was only when the trajectory complexity reached very high levels that the error increased significantly. Therefore, the system exhibits high stability and good adaptability to different trajectory tasks.

VII. FUTURE WORK

Due to time constraints, we have only conducted training and testing in simulation. The next step involves deploying the trained model on a physical drone and testing its performance in a specially designed testing environment, where we will record the results. Additionally, we aim to explore random multi-trajectory training to enhance the model's performance across various complex trajectories.

REFERENCES

[1] Y. Li, "Deep reinforcement learning: An overview," *CoRR*, vol. abs/1701.07274, 2017. [Online]. Available: http://arxiv.org/abs/1701.07274

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[3] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction."

[4] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 528–535, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:5539001

[5] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, p. 2096–2103, Oct. 2017. [Online]. Available: http://dx.doi.org/10.1109/LRA.2017.2720851

[6] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," 2019.

[7] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," 2020.

[8] M. Müller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, "Teaching uavs to race: End-to-end regression of agile controls in simulation," 2018.

[9] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[10] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[11] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, Oct. 2021. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abg5810

[12] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," in *Robotics: Science and Systems XVI*, ser. RSS2020. Robotics: Science and Systems Foundation, Jul. 2020. [Online]. Available: http://dx.doi.org/10.15607/rss.2020.xvi.040

[13] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, pp. 39:1–39:40, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:7242892

[14] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.

[15] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2019.8794351

[16] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," 2022.

[17] S. Mengozzi, "Learning agile flight using massively parallel deep reinforcement learning," Ph.D. dissertation, Alma Mater Studiorum-University of Bologna, 2022.

[18] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[19] T. Kim, M. Jang, and J. Kim, "A survey on simulation environments for reinforcement learning," *2021 18th International Conference on Ubiquitous Robots (UR)*, pp. 63–67, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236521645

[20] D. Makoviichuk and V. Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," *Denys88/rl_games*, 2022.