Aggressive Quadrotor Flight Using Curiosity-Driven Reinforcement Learning

Qiyu Sun, Jinbao Fang, Wei Xing Zheng, Fellow, IEEE, and Yang Tang, Senior Member, IEEE

Abstract—The ability to perform aggressive movements, which are called aggressive flights, is important for quadrotors during navigation. However, aggressive quadrotor flights are still a great challenge to practical applications. The existing solutions to aggressive flights heavily rely on a predefined trajectory, which is a time-consuming preprocessing step. To avoid such path planning, we propose a curiosity-driven reinforcement learning method for aggressive flight missions and a similarity-based curiosity module is introduced to speed up the training procedure. A branch structure exploration (BSE) strategy is also applied to guarantee the robustness of the policy and to ensure the policy trained in simulations can be performed in real-world experiments directly. The experimental results in simulations demonstrate that our reinforcement learning algorithm performs well in aggressive flight tasks, speeds up the convergence process and improves the robustness of the policy. Besides, our algorithm shows a satisfactory simulated to real transferability and performs well in real-world experiments.

Index Terms—UAVs, Aggressive Flight, Reinforcement Learning

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are well-performing platforms for many tasks, such as exploration [1], rescue assistance [2] and surveillance [3]. In particular, UAVs are suitable for executing tasks such as agilely moving and avoiding obstacles with high linear and angular speed, which are called aggressive flights. The ability of aggressive flight is significant for UAVs, especially in the situations such as entering a damaged building with cluttered obstacles to find trapped persons [2]. There are several representative tasks in aggressive flight problems, such as flying through narrow windows and slalom path scenes with a high speed [4]. In these missions, UAVs are confronted with the challenges of continuously performing precise aggressive movements and stabilizing themselves in aggressive states.

Previous works [4], [5] have demonstrated the possibility of navigating UAVs in aggressive flight missions. However, a time-consuming trajectory planning procedure is commonly required to provide an aggressive trajectory before the navigation of UAVs in these methods. As this pre-required aggressive trajectory planning may take considerable time in some unstructured environments [6], it restricts applications with a tight time constraint such as rescue assistance. Thus, an end-to-end navigation strategy that is able to directly use current state information to generate a control command is of great importance and is considered to be a superior solution in the tasks expecting rapid reactions. One of the promising methods to address this is reinforcement learning, through which UAVs can be navigated without the trajectory planning procedure [7] and which has been widely used in the control and navigation of agents [8]–[11].

Though reinforcement learning methods provide solutions to navigation without trajectory planning, few attempts address aggressive flight problems with reinforcement learning. The reason is that reinforcement learning methods still face some challenges when conducting aggressive flight missions. First, the quadrotor system in the aggressive flight missions is underactuated, which brings challenges to the control of the system. Thus, positive samples are difficult to obtain by naive stochastic sampling, and the sparse reward issue becomes more severe [12], which make the exploration of the control policy difficult and lead to a severely inefficient training procedure for reinforcement learning. Second, reinforcement learning methods commonly suffer from the transferability issue between simulations and real-world experiments [13], [14]. A well-performed algorithm in simulations frequently fails in real-world experiments. As these challenges remain unresolved, few works take advantage of the reinforcement learning techniques to address aggressive flight missions.

In this work, we explore how to conduct aggressive flight missions, including flying through narrow windows and slalom
path scenes, through reinforcement learning using the optimized policy directly instead of tracking the predefined trajectory. The architecture of our algorithm is shown in Figure 1. To obtain more positive samples for training, a curiosity-driven reinforcement learning method is proposed. We use a curiosity module based on the similarity of the states (i.e., the position and velocity of the quadrotor) for aggressive flight missions to improve the efficiency of policy exploration, overcome the sparse reward issue, and accelerate convergence. To enhance the adaptability of our method, we set obstacles with different positions and attitudes during training, which helps the quadrotor adapt to environments with different obstacles. To ensure that our algorithm can transfer from simulations to real-world experiments, we add the branch structure exploration (BSE) strategy (also called vine) [15] to our reinforcement learning algorithm. Through the BSE strategy, we can enrich the diversity of the training samples, which makes the simulation process more similar to the real experiments and enhances the transferability of our method.

In our experiment, we assume that the states of the quadrotor and the obstacles in environment can be detected, and then the quadrotor can make decision according to these states. The experiments contain two phases: the training phase conducted in simulation and the task execution phase executed in simulation or real-world. In training phase, the quadrotor conducts flight missions in a variety of scenarios with diverse obstacles, which positions and attitudes are different, to learn navigation strategies with our proposed method. In task execution phase, the quadrotor executes aggressive flight tasks with the policy trained during the training phase directly, even though the position and attitude of obstacles may be different from those in training phase. Though our method requires an additional training procedure when compared to traditional methods [4], [5] before conducting aggressive missions, the training procedure is conducted for just one time to enable the system the ability of navigation policy generation. Once the network is well-trained, the quadrotor can execute different aggressive flight missions without extra training. We test our method in both simulations and real scenarios. The experimental results show that our method can conduct aggressive flight missions with satisfactory performance in both simulations and real-world experiments, and some ablation experiments demonstrate the effectiveness of our proposed method.

The main contributions of this study are as follows:

1) We propose a curiosity-driven reinforcement learning algorithm for aggressive flight missions. Specifically, a similarity-based curiosity module is proposed to obtain more positive samples and overcome the sparse reward problem in reinforcement learning, thus accelerating the convergence speed of the training process.

2) The BSE strategy [15] is applied to guarantee the robustness of the policy, so that the policy trained in simulations can be performed in the real-world directly. Besides, our method shows the adaptability when the obstacles in a specific aggressive flight mission are different in test and training process.

3) Real-world experiments are conducted in both slalom path and narrow window scenes use the policy learned from simulation directly, which demonstrates the simulated to the real (sim2real) transferability of our method is satisfactory. The maximum linear and angular velocity reach 5.65 m/s and 252 deg/s, respectively.

II. RELATED WORK

Traditional quadrotor navigation and aggressive flight methods generate a feasible trajectory at first and the quadrotor tracks the trajectory with a certain controller [4], [6]. In contrast, reinforcement learning quadrotor navigation methods obtain the control strategy by optimizing the policy during the training process [7], [16].

Traditional aggressive flight. Previous works have provided solutions to aggressive flight missions [4], [6]. These works focus on the challenges in aggressive flight, including the aggressive trajectory planning [17], robust controller design [18], and high-rate estimation of aggressive states [19]. In [4], the visual-inertial odometry is presented for perception, and an aggressive feasible trajectory planning method is introduced to obtain the sequence of aggressive movements. A lightweight micro aerial vehicle is navigated by this method to fly through some challenging scenes, such as narrow windows or slalom path scenes. The path planning technique with aggressive movements is further discussed in [6], in which the authors propose a motion-primitive-based graph searching strategy to generate aggressive trajectories in cluttered environments. When the feasibility constraint is presented with an episode quadrotor model, the trajectory can be generated more precisely. In [5], a single onboard camera and an inertial measurement unit (IMU) are used to control a quadrotor and flight through narrow gaps. With the independent yaw-angle planning, the quadrotor is able to continuously face the target while tracking the trajectory. All the methods mentioned above require a path-planning procedure to generate a desired trajectory, and trajectory planning is very time-consuming, especially for aggressive flight missions.

Some recent methods [20]–[22] focus on the faster quadrotor navigation and show significant improvements. In [20], a fast and safe trajectory planner is proposed. The planner can generate trajectories in several seconds and gets about 50% improvement when compared with other state-of-the-art planning methods [23], [24]. In [21], the event-based cameras are used to navigate quadrotors in complex dynamic environments. The method shows the possibility to accomplish the dynamic obstacle avoidance in few milliseconds and the quadrotors are able to avoid multiple obstacles at relative speeds up to 10m/s. Though [21] accomplishes dynamic obstacle avoidance quickly, it utilizes an event-based camera as the sensor and thus is insensitive to static obstacles, and [20] still needs more than 10 seconds for trajectory planning. A receding horizon planning architecture is proposed for reactive obstacle avoidance in [22] and the work is able to perform efficient trajectory planning for high linear speed flight, while

The video of real-world experiments can be found at https://youtu.be/s76iqKB3fNw. The codes of our works can be found at https://github.com/Randy-F/reinforcement-aggressive-flight.\footnote{The video of real-world experiments can be found at https://youtu.be/s76iqKB3fNw. The codes of our works can be found at https://github.com/Randy-F/reinforcement-aggressive-flight.}
our method concentrates on aggressive flight, in which the agents fly with both high linear and high angular speed. Compared with these works, our method attempts to avoid the trajectory planning procedure via reinforcement learning methods. By utilizing the end-to-end decision making process, our method saves the time of planning procedure with the real-time navigation policy.

Reinforcement-learning navigation for quadrotors. Reinforcement-learning methods have attracted increasing attention in numerous tasks such as flight control [25], collision avoidance [26] and acrobatics maneuvers [27]. Based on the existence of a system modeling process, these methods can be divided into two categories: model-based [26], [28], [29] and model-free reinforcement learning methods [30]. Model-based methods model the system dynamics first and then evaluate the policy with these models. In [29], the system dynamics model is built by neural networks and is used to predict future states from past state-action pairs, in which the low-level control is performed using a model predictive control strategy. In [26], the quadrotor is able to avoid obstacles with an uncertainty-aware prediction model after a few training iterations. Model-free methods are also widely used, and they evaluate the performance of a policy without a system modeling process. In [30], a deterministic policy optimization method is presented to design a learning-based controller, which can stabilize the quadrotor from a high initial speed and large initial rotation angle.

Although reinforcement learning methods have shown great performance in navigation, their sparse reward and poor sim2real transferability remain crucial problems. Sparse reward is a challenging issue in reinforcement learning, and it severely slows down the convergence speed in training process. The curiosity method [31], [32] is proposed as an effective solution to the sparse reward problem. It encourages the agent to explore the state space more efficiently by providing an additional curiosity reward signal, which is based on the difference between the former and current observations of the agent. In [31], a neural network is used to predict the incoming state, and the curiosity reward is generated from the state prediction error. In [32], the reachability of episodic states is used as a curiosity signal, and this reachability is estimated by a network approximator. The sim2real transferability is also an important issue in the reinforcement learning. In [14], the domain adaptation technique is used to improve the transferability of robotic grasping systems. In [13], a quadrotor is navigated in an indoor environment with monocular images, and the experiences in the simulations are used to train a generalizable perception module. Our work relieves the sparse reward issue by a curiosity-driven module and improves the sim2real transferability by the BSE strategy.

III. OUR PROPOSED METHOD FOR AGGRESSIVE FLIGHT

A. Problem Formulation

We describe the aggressive flight missions as a Markov decision process (MDP) [33] and solve it as a standard reinforcement learning problem. There are three important elements in an MDP: state, action, and reward. The quadrotor starts from a certain initial state $s_0$ (such as initial position $X_0$ and velocity $V_0$). By choosing the action $a_t$ (such as attitude $R$ and thrust $f$) by the current policy at time $t$, the states of the quadrotor change from $s_t$ to $s_{t+1}$. The probability distribution of transition $P_{s_t, a_t}^{s_{t+1}}$ is determined by the current state $s_t$ and action $a_t$. The reward $r_t$ is generated by analyzing the quality of the performance caused by the former state $s_t$ and action $a_t$. Assuming that the finite-step process ends in step $T$, a state, action and reward sequence $\{s_t, a_t, r_t \mid 0 \leq t \leq T\}$ is presented. This sequence is used to evaluate the state and action value $Q(s, a)$ and the value updates the action policy $\pi(s)$ by the policy gradient strategy.

We choose the position $X$, attitude $R$ and linear velocity $V$ of the quadrotor along with the parameters of obstacles as the state features of aggressive flight. These features guarantee that the agent obtain sufficient information from the quadrotor and environment. We use the rotation matrix to express the attitude to avoid the discontinuous feature in the Q-value approximation [12], which commonly exists when the Euler angle or quaternion is used. The policy network generates the attitude-thrust command, and this command signal is adequate for the requirements of control precision in the aggressive flight mission. Moreover, the attitude-thrust command is a general control command for UAVs with the difference in inner parameters (i.e., dynamic model of the motor), so the policy trained in a simulated environment can obtain a robust performance on a real quadrotor.

To provide guidance for the quadrotor, an extrinsic reward is generated by analyzing the performance in the mission. We divide the extrinsic reward into two parts in the aggressive flight missions: the goal reward providing an encouraging reward and obstacle collision reward providing a punishing reward. The goal reward $r_{\text{goal}}$ considers the quality of target movement using the following heuristic function:

$$ r_{\text{goal}} = \lambda_x |X - X_{\text{goal}}| + \lambda_r |R - R_{\text{goal}}| + \lambda_v |V| + \lambda_\omega |\omega|, $$

where $X$, $R$, $V$, and $\omega$ denote the state vectors of position, attitude, linear velocity and angular velocity, $\lambda_x$, $\lambda_r$, $\lambda_v$, and $\lambda_\omega$ denote the weights of the corresponding terms, the subscript “goal” denotes the reference state of the goal, and the symbol $| \cdot |$ denotes the Euclidean norm.

![Fig. 2: Left: ellipsoid model of quadrotor. Right: illustration of the collision reward generated by the quadrotor model and obstacle.](image)
The obstacle collision reward is generated by the geometric relationship between the quadrotor and obstacles. We model the quadrotor as an ellipsoid, and this ellipsoid model allows the quadrotor to pass obstacles such as a narrow window with an aggressive attitude [6]. With the known position \( \mathbf{X} \), attitude \( \mathbf{R} \), ellipsoid radius \( l \), and height \( h \), the ellipsoid \( \epsilon \) is considered as a point set:

\[
\epsilon(\mathbf{X}, \mathbf{R}) := \{ p = R\Sigma R^T d + \mathbf{X} \mid \|d\| \leq 1 \},
\]

where

\[
\Sigma = \begin{bmatrix} l & 0 & 0 \\ 0 & l & 0 \\ 0 & 0 & h \end{bmatrix}.
\]

The diagonal matrix \( \Sigma \) denotes the quadrotor body configuration, and the vector \( d \) assists the description of the point \( p \) that meets the conditions. The ellipsoid model is shown in Figure 2.

The obstacle collision reward is:

\[
r_{\text{obstacle}} = \lambda_{\text{obstacle}} \frac{\text{card}(\epsilon \cap \mathcal{O})}{\text{card}(\epsilon)}. \tag{4}
\]

This reward is generated by calculating the proportion of the intersection area between ellipsoid \( \epsilon \) and obstacle \( \mathcal{O} \). \( \lambda_{\text{obstacle}} \) denotes the weight of the obstacle collision reward, the symbol \( \text{card}(\cdot) \) denotes the number of elements and the set \( \epsilon \) and \( \mathcal{O} \) are sampled using a point cloud before calculation. The total extrinsic reward is represented as:

\[
r_{\text{extrinsic}} = r_{\text{goal}} + r_{\text{obstacle}}. \tag{5}
\]

### B. Similarity-based Curiosity Module

We present a curiosity module to address the sparse reward problem and improve sampling efficiency. In an aggressive flight mission, the quadrotor is required to perform movements in extreme conditions. Therefore, a successful reward is very difficult to obtain by simply using a random sampling strategy, which makes the sparse reward problem appear. To address this problem and encourage the agent to experience novel states, we propose a similarity-based curiosity module for the aggressive flight mission. Unlike using reachability checking or state prediction error, our curiosity module judges the similarity of the time-state curves between the current and former episodes. This method describes the similarity relationship between different episodes more precisely and makes full use of the information in the entire training process. Considering a low similarity as a high curiosity reward, the similarity of the states between different episodes provides a guideline to lead the quadrotor to explore new states. However, a simple comparison of similarities without a proper time alignment is misleading. For example, in Figure 3, two curves of the states during different training episodes are shown to demonstrate the effect of the time alignment. Note that in Figure 3(a), these two episodes have quite similar positional trajectories; however, the measurements is with highly dissimilar when we directly compare them in the time-state curves in Figure 3(b).

Thus, we perform time alignment operation before the comparison to make the similarity measurement between these curves more accurate. Assume that the quadrotor performs an aggressive flight several times and \( S = \{s_i \mid i = 1, 2, \ldots, n\} \), \( S' = \{s'_j \mid j = 1, 2, \ldots, m\} \) are two of the state sequences in these episodes. To perform a proper time alignment, we use the dynamic time warping method [34] for the curve similarity measurement with matrix \( \mathbf{A}_{n \times m} \). Its element \( a(i,j) \) is the distances between each state \( s_i \) and \( s'_j \) in episode \( S \) and \( S' \). Then, the minimized distance with time alignment can be obtained by finding a set of consecutive elements in the matrix and minimizing the sum of the all the elements in the set. Specifically, in matrix \( \mathbf{A}_{n \times m} \), a set of elements starting from the top left corner and ending in the bottom right corner denotes a possible choice for time alignment, and the element \( a(i,j) \) in the chosen set denotes that the states \( (s_m, s'_j) \) are aligned. We can obtain the warping set by iteratively calculating the aligned distance:

\[
D(s_i, s'_j) = d(s_i, s'_j) + \min(D'_{i,j}), \tag{6}
\]

where

\[
D'_{i,j} = \{D(s_{i-1}, s'_{j-1}), D(s_{i-1}, s'_j), D(s_i, s'_{j-1})\}, \tag{7}
\]

and \( D(s_i, s'_j) \) denotes the accumulated distance from state \( (s_1, s'_1) \) to \( (s_i, s'_j) \) along the warping set. With this expression, we can get the distance between \( S \) and \( S' \) using dynamic time warping (DTW):

\[
D_{\text{dtw}}(S, S') = D(s_n, s'_m). \tag{8}
\]

The curiosity reward is generated based on this aligned similarity measurement. It is calculated by the minimum state distance when the current episode is compared with each of the former episodes. The final curiosity reward is

\[
r_{\text{curiosity}} = 1 - \exp(-\min_i \{\sum_{n=1}^{N} D_{\text{dtw}}(S_n, S'_n)\}), \tag{9}
\]

where \( S_n \) denotes the \( n \)th state in the current episode, \( S'_n \) denotes the \( n \)th state in the \( i \)th former episode, and we use position, attitude and velocity for similarity measurement. We use the negative exponential function to restrict the value of the curiosity reward to a reasonable range. The similarity-based curiosity module encourages state space exploration and reduces the time cost of the training time with high-quality samples. We generate the total reward function

\[
r = r_{\text{extrinsic}} + \lambda_r r_{\text{curiosity}}, \tag{10}
\]

where \( \lambda_r \) is the weight of \( r_{\text{curiosity}} \).

### C. Exploration Strategy

We use the BSE strategy to improve the robustness of the policy [15]. During the process of convergence in the training procedure, the agent tends to choose the fixed action sequence, and the trajectories of the quadrotor in each episode will be similar to each other. In simulations, this type of strategy is able to work and ensure a satisfactory result. However, when it comes to real-world environments, the performance of this kind of strategy is far from satisfactory. Because there exist different kinds of ubiquitous and unavoidable disturbances (e.g., actuator bias or uncertain airflow) exist in reality and
these disturbances are difficult to consider in simulations, the quadrotor will experience new states and act inappropriately due to these disturbances when real-world experiments are conducted.

Therefore, to extend the exploration area and improve the quality of the action in a wider state space, we perform an exploration strategy in the branch structure. The BSE is illustrated in Figure 4. We start the experiments with the current policy \( \pi_\phi \), obtaining the initial samples \( s_0, s_1, \ldots, s_{T_{\text{init}}} \). Then, a branch set of the reachable states are generated following the initial samples. For the states in this branch set, we perform an action with additional random noise to generate a branch trajectory. The following trajectory sampled with the current policy starts from the end of the branch trajectory. Compared with the simple single path exploration, the BSE method guarantees a broader distribution of sampling area. Thus, the robustness of the control policy is improved as well.

\[ G_t(s_t, a_t) = v_t + \gamma \min_{i=1,2} Q_{\theta_i'}(s_{t+1}, \pi_{\phi'}(s_{t+1})), \]

where \( G_t(s_t, a_t) \) is the target estimation of state \( s_t \) and action \( a_t \), and \( v_t \) is the preliminary approximation of the value in the \( t^{th} \) step using Monte-Carlo samples [12]. \( \gamma \) is the discount.
factor [12] in the MDP and $\theta'_1$, $\theta'_2$, and $\phi'$ are the weights of the target actor-critic networks. The policy network is updated using the DDPG [36]:

$$
\nabla_{\phi} L(\phi) = \frac{1}{N} \sum_{k=0}^{N} \nabla_{\phi} Q_{\theta_1}(s, a)_{s=\pi_\phi(s)} \nabla_{\phi} \pi_{\phi'}(s),
$$

where $\phi$ is the weights of the actor network, $N$ is the mini-batch sample size and $k$ is the sample index. Finally, we update the target network by

$$
\phi' \leftarrow \rho \phi + (1 - \rho) \phi',
$$

$$
\theta'_1 \leftarrow \rho \theta_1 + (1 - \rho) \theta'_1,
$$

with the same frequency as the policy update. To stabilize the learning process, we update the policy and target networks when the critic networks are updated more than once. This delayed policy update strategy guarantees a small estimation bias in the policy update process. Our curiosity-driven deterministic policy gradient algorithm is summarized in Algorithm 1.

IV. EXPERIMENTAL RESULTS

A. Simulation Setup

Unreal Engine 4\(^2\) is used as the physical engine to simulate the dynamic model of quadrotor and training environment. The simulation platform is equipped with a 2.20GHz i7-8750H CPU, NVIDIA GTX 1070 GPU, and 16GB memory space. The training process is implemented in an Airsim\(^3\) aerial vehicle simulator. We implement our proposed reinforcement learning algorithm in two aggressive flight missions, flying through the slalom path and a narrow window. The slalom path scene contains two columnar obstacles, and the obstacles are in front of the initial position of the agent. The goal of the task is to pass through both obstacles in opposite horizontal directions. For the narrow window scene, the goal of the task is to pass through the narrow window gap with a certain attitude. We terminate a training episode when the target is accomplished or the quadrotor is outside the flight region. The curiosity reward is generated when an episode is terminated, the extrinsic reward is generated when a collision is detected or the target is reached, and we set $\lambda_c = 4$.

We conduct the training procedure in both slalom path and narrow window scenes, and we compare our methods with DDPG [36] and the Soft Actor-Critic (SAC) [39]. To prove the effectiveness of our curiosity module on the convergence speed, ablation experiments are conducted as well. Following previous works [36], [39], we adopted the average reward as the evaluation criterion for the performance of the reinforcement learning algorithms. From the average reward learning curves shown in Figure 6, it is obvious that our method achieves the best performance when compared to DDPG [36] and SAC [39]. The method without the curiosity module takes more time for training and obtains lower rewards, showing that our curiosity module improves the exploration efficiency and performance. In particular, in the narrow window mission, our method obtains a satisfactory reward in about 20 minutes of

\[^2\]https://www.unrealengine.com/

\[^3\]https://github.com/Microsoft/AirSim

| TABLE I: Statistics in narrow window scene with different setup of noises. |
|-------------------------------|-----------------|-----------------|-----------------|
| STD noise | method         | Position error (m) | Average reward | Successful rate |
| 0.0°   | Our method     | 0.0116           | 2.9154           | 99.2%           |
|        | SPE method     | 0.0158           | 2.9045           | 98.6%           |
| 1.5°   | Our method     | 0.0138           | 2.9067           | 96.8%           |
|        | SPE method     | 0.0219           | 2.8973           | 79.5%           |
| 3.0°   | Our method     | 0.0164           | 2.8973           | 91.2%           |
|        | SPE method     | 0.0424           | 2.7854           | 50.6%           |

| TABLE II: Statistics in slalom path scene with different setup of noises. |
|-------------------------------|-----------------|-----------------|-----------------|
| STD noise | method         | Position error (m) | Average reward | Successful rate |
| 0.0°   | Our method     | 0.0149           | 24.179           | 99.1%           |
|        | SPE method     | 0.0163           | 24.096           | 98.8%           |
| 1.5°   | Our method     | 0.0158           | 23.985           | 96.5%           |
|        | SPE method     | 0.0192           | 23.885           | 86.3%           |
| 3.0°   | Our method     | 0.0185           | 23.829           | 93.6%           |
|        | SPE method     | 0.0254           | 23.429           | 61.2%           |
training and the average reward learning curve converges in about 30 minutes. In the slalom path mission, our method reaches convergence in 45 minutes, while other methods are not able to achieve the same performance with even in 100 minutes. Facing multiple sequential targets in a slalom path scene, our method performs much better than the others in terms of exploration efficiency.

Since reinforcement learning methods are poor in sim2real transferability, we use the BSE strategy [15] to improve the robustness of our system and to improve the sim2real transferability. The disturbances in real-world experiments could come from the voltage instability of batteries, nonstandard dynamics of rotors, uneven mass distribution, etc., which lead to the actions (attitude commands in our experiments) in real experiments deviating from the simulated cases. Therefore, we add action noises to simulate the uncertainty in real-world experiments. The action noises are denoted by the attitude command error and are applied to each attitude axis independently. In the experiments, we set the standard deviation (STD) noises at 0.0°, 1.5°, and 3.0°.

The quantitative experimental results are shown in Tables I and II. We also conduct an ablation experiment to verify the effectiveness of the BSE strategy by replacing it with the single path exploration method (SPE method) [15] and test both of the methods with additional action noises. When the STD noise is 1.5°, our method maintains the accuracy while the error of the SPE method increases significantly. When the STD of noise is 3°, the SPE method tends to fail while our method has a much higher success rate owing to its wider sample border. Each experiment was performed 1000 times. The trajectories in the simulation experiments are demonstrated in Figure 7 and Figure 8. The trajectory in the slalom path scene is shown in Figure 7, in which we choose an episode to show the performance of the policy. The trajectories in different narrow window scenes are plotted in Figure 8, in which we perform the experiments on diverse narrow windows using the same pre-trained policy. The narrow window scenes are different in the various rotation angles and the distances to the center of the scenes. Specifically, the angles of these windows are 0.3, 0.4 and 0.6 rad, and the distances are -0.18, 0.0 and 0.3 m, respectively. The figures demonstrate the generalization of our method to navigate the quadrotor with variable obstacles parameters.

To further verify the superiority of our method, we run a Monte-Carlo simulation for a collection of scenes containing random unstructured obstacles. In the generated scenes, there exists at least one path that is in accord with quadrotor’s dynamics constraints for the agent to traverse across. We execute navigation in the generated scenes using our method and an example of the experimental result is shown in Figure 9. The result demonstrates that our method not only performs well in classical aggressive flight missions like specific narrow window and slalom path scenarios, but also works well in other unstructured environments. More experiments can be found in Appendix A.

**Algorithm 1** Curiosity-Driven Deterministic Policy Gradient

```plaintext
Initialize critic-actor networks and target networks
Initialize the replay buffer B and delayed parameter d
Observe s₀
for t = 0, 1, 2, ..., T do
    Sample action aₜ using current policy
    if episode is terminated then
        Calculate curiosity reward r_{curiosity}
    end if
    Observe reward rₜ and new state s_{t+1}
    Store transition (sₜ, aₜ, rₜ, s_{t+1}) in B
    Sample N transitions from B
    Update critics using gradient descent step
    if t mod d == 0 then
        Update actor network according to Eq. 12
        Update target network according to Eq. 13
    end if
end for
```

**B. Experiments on a Real Quadrotor**

In this section, we perform aggressive flight missions in real-world environments using the policy learned from sim-
Fig. 9: The trajectory in an unstructured environment.

Fig. 10: System diagram of our learning-based aggressive flight experiment. $X, R$ are position and orientation from Vicon, and $\hat{X}, \hat{R}, \hat{V}$ are filtered position, orientation and velocity.

TABLE III: Parameters of the quadrotor in our experiments.

| Parameters               | Value (units) |
|--------------------------|---------------|
| Weight                   | 0.547 (kg)    |
| Size                     | $0.44 \times 0.44 \times 0.12$ (m) |
| Rotor diameter           | 20.32 (cm)    |
| Maximum speed            | 15 (m/s)      |
| Maximum thrust           | 20 (N)        |
| Power (each motors)      | 80 (W)        |
| Inertia $I_{xx}, I_{yy}, I_{zz}$ | $0.033, 0.033, 0.058$ (kg·m²) |

Fig. 11: A snapshot of the quadrotor during our aggressive flight experiment.

Fig. 12: The trajectories in our aggressive flight experiments.

(a) The experimental scenes and results.

(b) Comparisons between experimental and simulated trajectories.

We perform aggressive flights in real-world experiments in both slalom path and narrow window scenes. For the slalom path scene, we used two columnar obstacles placed 1.5m away from each other. For the narrow window scene, we build a narrow window gap with a 30° rotation angle and Figure 11 is a snapshot of the experiment. A 3D illustration of experimental trajectories is shown in Figure 12(a), and the comparisons between the experimental and simulated trajectories are also shown in Figure 12(b). Because there are differences between the simulations and real-world experiments, the trajectories of the quadrotor in the real-world experiments inevitably deviate from their trajectories in the simulations, as shown in Figure 12(b). Despite this, the policy trained with our exploration strategy has a satisfactory transferability and can give a relatively satisfactory action selection in the new states.

We conduct ablation and comparison experiments to demonstrate the performance and transferability of our method. In Table IV, we calculate the average position and rotation differences between our trajectories and the goal trajectories ($\text{Goal Pos. Error}$ and $\text{Goal Ang. Error}$) to evaluate the performance.

---

4http://www.asctec.de
5https://www.vicon.com/
TABLE IV: Comparison and ablation experiments in real-world aggressive flight missions.

| Scene          | Method                  | Goal Pos. Error (m) | Goal Ang. Error (deg) | Sim2real Pos. Difference (m) | Sim2real Ang. Difference (deg) | Planning Time (s) |
|----------------|-------------------------|---------------------|-----------------------|-----------------------------|-------------------------------|-------------------|
| Narrow Window  | Our Method              | 0.034               | 2.15                  | 0.273                       | 4.78                          | /                 |
|                | TD3 [38] + Curiosity    | 0.055               | 3.34                  | 0.448                       | 5.50                          | /                 |
|                | TD3 [38] + BSE          | 0.116               | 5.92                  | 0.319                       | 4.97                          | /                 |
|                | TD3 [38]                | 0.521               | 7.51                  | 0.484                       | 6.76                          | /                 |
|                | Nonlinear Tracking Ctrl [4] | 0.049               | 2.82                  | –                           | –                             | 41.90             |
| Slalom Path    | Our Method              | 0.044               | 2.06                  | 0.187                       | 6.02                          | /                 |
|                | TD3 [38] + Curiosity    | 0.068               | 2.76                  | 0.371                       | 8.39                          | /                 |
|                | TD3 [38] + BSE          | 0.084               | 4.16                  | 0.210                       | 6.41                          | /                 |
|                | TD3 [38]                | 0.456               | 5.31                  | 0.431                       | 8.81                          | /                 |
|                | Nonlinear Tracking Ctrl [4] | 0.057               | 3.17                  | –                           | –                             | 21.47             |

1 Reinforcement learning based methods do not require trajectory planning procedure and the planning time is presented as “/”.
2 The sim2real transferability is tested between the reinforcement learning based methods, therefore, the simulated error of nonlinear tracking controller is presented as “–”.

The transferability is evaluated by calculating the differences between the simulated and experimental trajectory results (Sim2real Pos. Difference and Sim2real Ang. Difference).

To prove the effectiveness of the modules used in our method, we compare our method with the original TD3 [38], TD3 + Curiosity and TD3 + BSE. The results show that both of the curiosity module and the BSE strategy narrow the differences between the simulated and experimental trajectories, and the performance is improved as well.

To demonstrate the performance of our curiosity-driven reinforcement learning method further, we compare our method with the planning-based aggressive flight method [4]. The method uses the aggressive trajectory planning [6] and a nonlinear controller [40]. The quantitative results are shown in Table IV and the trajectories are plotted in Figure 13. The results in Table IV show that our method obtains a smaller error without trajectory planning, which demonstrate that our method have a better performance and transferability when compared to [6]. In Figure 13, our method is more far from the obstacles in the slalom path mission and more closer to the goal in narrow window mission.

Fig. 13: Comparisons of position and rotation trajectories in different aggressive flight missions.

V. CONCLUSIONS

In this paper, we confront the problem of aggressive flight with a curiosity-driven reinforcement learning method. We introduce a similarity-based curiosity module to overcome the sparse reward problem in reinforcement learning, which can achieve a satisfactory performance with a fast convergence speed. Besides, the BSE strategy improves the robustness of our learning-based controller, which makes the policy trained in simulation can be directly used with a real quadrotor. Our method accelerates the execution of the aggressive movements by reducing the dependence of trajectory planning step, and it shows the sim2real transferability.

In the future, we will try to conduct more aggressive tasks with higher speed and rotation angles through reinforcement learning. Since the higher rotation angle leads to higher traversing speed, the quadrotor needs faster execution rate and more precise control commands. To handle these, we plan to deploy our algorithm onboard with the matrix arithmetic method. Besides, model-based reinforcement learning methods will be considered. The dynamic model of the quadrotor will be used as a state predictor to make full use of the theoretical
information in the learning process and control the quadrotor more precisely. We will also explore the possibilities of visual-based reinforcement learning methods for aggressive flight, in which different cameras can be used, such as the event-based and RGB-D cameras.

REFERENCES

[1] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. J. Grika, F. Raess, M. Suppa, and D. Burschka, “Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue,” IEEE Robotics & Automation Magazine, vol. 19, no. 3, pp. 46–56, 2012.

[2] N. Michael, S. Shen, K. Mohta, Y. Mulaqoankar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, “Collaborative mapping of an earthquake-damaged building via ground and aerial robots,” Journal of Field Robotics, vol. 29, no. 5, pp. 832–841, 2012.

[3] D. Zhu, Y. Du, Y. Lin, H. Li, C. Wang, X. Xu, and M. Q.-H. Meng, “Hawkeye: Open source framework for field surveillance,” in Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6083–6090, 2017.

[4] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, “Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU,” IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 404–411, 2016.

[5] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, “Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision,” in Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5774–5781, 2017.

[6] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, “Search-based motion planning for aggressive flight in SE(3),” IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2439–2446, 2018.

[7] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. “Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search,” in Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 528–535, 2016.

[8] H. Zhang, H. Jiang, Y. Luo, and G. Xiao, “Data-driven optimal consensus control for discrete-time multi-agent systems with unknown dynamics using reinforcement learning method,” IEEE Transactions on Industrial Electronics, vol. 64, no. 5, pp. 4091–4100, 2016.

[9] Z. Li, J. Liu, Z. Huang, Y. Peng, H. Pu, and L. Ding, “Adaptive impedance control of human–robot cooperation using reinforcement learning,” IEEE Transactions on Industrial Electronics, vol. 64, no. 10, pp. 8013–8022, 2017.

[10] Y. Yuan, Z. Li, T. Zhao, and D. Gan, “Dmp-based motion generation for a walking exoskeleton robot using reinforcement learning,” IEEE Transactions on Industrial Electronics, vol. 67, no. 5, pp. 3830–3839, 2020.

[11] B. Luo, H.-N. Wu, and T. Huang, “Optimal output regulation for model-free quadrotor helicopter with msetp q-learning,” IEEE Transactions on Industrial Electronics, vol. 65, no. 6, pp. 4953–4961, 2017.

[12] R. S. Sutton, A. G. Barto et al., Introduction to reinforcement learning, Cambridge, MA: MIT Press, 1998.

[13] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, “Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight,” in Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), pp. 6008–6014, 2019.

[14] K. Bousmalis, C. Brunner, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downes, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, “Using simulation and domain adaptation to improve efficiency of deep robotic grasping,” in Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 4243–4250, 2018.

[15] J. Xu, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in Proceedings of the 2015 International Conference on Machine Learning (ICML), pp. 1889–1905, 2015.

[16] S. Lupashin, A. Schöllig, M. Sherbach, and R. D’Andrea, “A simple learning strategy for high-speed quadcopter multi-flips,” in Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 1612–1618, 2010.

[17] B. Xian, S. Wang, and S. Yang, “An online trajectory planning approach for a quadrotor UAV with a slung payload,” IEEE Transactions on Industrial Electronics, vol. 67, no. 8, pp. 6669–6678, 2020.
VI. APPENDIX

A. Experiments in unstructured environments

We add some experiments in unstructured environments by running Monte-Carlo simulations for a collection of scenes containing random unstructured obstacles. In the generated scenes, there exists at least one path that is in accord with quadrotor’s dynamics constraints for the agent to traverse across. The result demonstrates that our method not only performs well in classical aggressive flight missions like specific narrow window and slalom path scenarios, but also works well in other unstructured environments, as shown in Figure 14. To show the 3d geometry of the unstructured environments in a clearer way, we have uploaded the experimental video for the navigation in an unstructured environment and you can watch the video at https://youtu.be/5ZI9qyZNwjo.

![Scene 1](image1)
![Scene 2](image2)
![Scene 3](image3)

Fig. 14: The trajectories in an unstructured environment.

B. Scene with two narrow windows

The simulation experiment with two narrow windows is also conducted to prove that our method works well in more complex environment, as shown in Figure 15.

![Scene 2](image4)
![Scene 3](image5)

Fig. 15: Scene with two narrow windows.

C. Visual comparison of our work and [4]

We plot the trajectories of different methods in Table IV for a clearer visual comparison. As shown in Figure 16, our method is more far from the obstacles in the slalom path mission and has a smoother curve that is in accord with the dynamics constraints of the quadrotor. We can also find that our method is more closer to the goal in narrow window mission in Figure 13, compared to traditional tracking controller [4].

![Our Method vs Tracking Controller](image6)

Fig. 16: Visual comparison of different methods.
TABLE V: The comparison of our method and state-of-the-art methods.

| Method                  | Trajectory Planning | Pre-training | Applications           |
|-------------------------|---------------------|--------------|-------------------------|
|                         | Time                | Frequency    | Time                    | Frequency       |                      |
| Loianno et al. [4]      | Dozens of seconds   | Before every flight | Not necessary | Aggressive flights |
| Liu et al. [6]          | Dozens of seconds   | Before every flight | Not necessary | Aggressive flights |
| Falanga et al. [21]     | Real-time           | Not necessary | Obstacle avoidance      |                  |
| Ryll et al. [22]        | Real-time           | Not necessary | High speed flights      |                  |
| Ours                    | Not necessary       | 1 hour       | Once                    | Aggressive flights |

D. Comparison of different methods

We add a table to compare our method with some other state-of-the-art methods in terms of various aspects.

From Table V, we can find that our method can execute real-time aggressive flight missions without trajectory planning after the one-time offline pre-training, while traditional methods [4], [6] need dozens of seconds for trajectory planning before every flight. Though reference [21] can conduct trajectory planning in real-time and [22] is able to perform efficient trajectory planning for high linear speed flight, our method concentrates on aggressive flight, in which the agents fly with both high linear and angular speed.

E. Generalization of our method

We perform the experiments on diverse narrow windows using the same pre-trained policy to demonstrate the generalization of our method in Figure 8. The experiments shown in Figure 8 are the successful cases, and we continue to change the setup of the narrow window to explore when the negative results appears. We find that when the rotation angle and the distance are set nearly to 1.2 rad and -0.5 m, the unsuccessful cases begin to appear, as shown with gray line in Figure 17. Based on the experimental results, we find that when marked changes appear in environment, our method may fail to fulfill the task. How to further improve the generalization of our method is one of our further research directions.

Fig. 17: Position and rotation trajectories in narrow window scene.