Think Before Acting: The Necessity of Endowing Robot Terminals With the Ability to Fine-Tune Reinforcement Learning Policies

Dawei Feng[1,3], Xudong Gong[1,3], Xunhui Zhang[1,3], Jun Fu[2,*]
[1] College of Computer, National University of Defense Technology, Changsha, Hunan, China
[2] College of Information and Communication, National University of Defense Technology, Wuhan, Hubei, China
[3] State Key Laboratory of Complex & Critical Software Environment, Changsha, Hunan, China
davyfeng.c@qq.com, {gongxudong09, zhangxunhui, fujun}@nudt.edu.cn

***Abstract*—Goal-Conditioned Reinforcement Learning (GCRL) has gained widespread application in robotics. A typical application of GCRL is to pre-train policies in a development environment and then deploy them to robots. In this approach of application, we find that the policies trained by GCRL exhibit discontinuity in goal space, indicating that we cannot effectively estimate the policy's performance in the actual production environment based on its evaluation results in the development environment. To ensure that the robot terminals can complete tasks effectively, we propose the Think Before Acting (TBA) framework, which evaluates and fine-tunes policies on the robot terminal. Within the TBA framework, for a goal to be executed, the policy's performance is first evaluated. If the performance does not meet the requirements, the policy is fine-tuned based on this goal. We conduct experiments on velocity vector control of fixed-wing UAVs to validate the effectiveness of TBA. The results show that for a well-pre-trained policy, less than 10^5 samples and less than 2 minutes of fine-tuning time are required to achieve satisfactory performance on the target goal.**

***Index Terms*—Robotics, Pervasive and Ubiquitous Computing, Reinforcement Learning, Pre-training Policies, Fine-tuning Policies**

I. Introduction

The development of robotics hardware allows engineers to deploy increasingly complex control policies on robot terminals. These policies can process various modalities of data from sensors and directly output signals such as voltage and current to control actuators. Reinforcement learning (RL) [1], a method that learns from interactions with the environment through trial and error without relying on extensive expert design, has been widely used to learn control policies. Since common robotic tasks, such as manipulator control [2] and UAV navigation [3], often involve multi-goal tasks [2], researchers have extended RL to Goal-Conditioned RL (GCRL) [4]. GCRL enables knowledge transfer between different goals, thereby assisting policies in improving learning efficiency and generalizing to unseen goals [5].

The optimization objective of GCRL is to maximize the policy's ability to obtain cumulative rewards on a desired goal distribution [5]. However, this optimization objective does not guarantee that the policy's ability exhibits a continuous variation across the goal distribution. For instance, a policy might be able to achieve goals g_a and g_b, but fail to achieve the goal $\frac{(g_a+g_b)}{2}$. We refer to this issue as **the discontinuity of policy in goal space**. Because (1) nearly all real-world robotics tasks are defined in continuous state spaces, and (2) we can only test the policy's performance on a finite set of goals, the discontinuity of policy in goal space implies that we cannot effectively estimate the policy's performance in the actual production environment based on its evaluation results in the development environment.

To address the challenges posed by the discontinuity of policy in goal space, we propose the Think Before Acting (TBA) framework. In TBA, the robot terminal is equipped with a simulator to support the robot in evaluating and fine-tuning the policy on a specific goal before executing it. This "think before acting" approach ensures that the action results in a better outcome. We evaluate TBA on the velocity vector control task of fixed-wing UAVs [6]. The velocity vector control task of fixed-wing UAVs represents a typical multi-goal and long-horizon problem [7], making it well-suited for validating the performance of GCRL algorithms. The experimental results demonstrate that a well-pre-trained policy can achieve satisfactory performance with only a small amount of training interactions on the robot terminal. Our contributions are summarized as follows:

- We demonstrate that policies trained via GCRL frequently suffer from the problem of discontinuity in goal space by evaluating policies trained from multiple GCRL approaches.
- We propose the Think Before Acting framework for evaluating and fine-tuning RL policies on robot terminals to ensure that robots achieve satisfactory performance on specific goals.
- We evaluate the effectiveness of TBA on the velocity vector control of fixed-wing UAVs and conduct extensive ablation studies to identify the best practices for

* Jun Fu is the corresponding author.

implementing TBA.

II. Related Work and Background

Goal-Conditioned Reinforcement Learning. Let $\Delta(\mathcal{X})$ denote the probability distribution over a set $\mathcal{X}$. Goal-conditioned RL is described by goal-augmented MDP [4], [8] $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \mathcal{G}, p_g, \phi \rangle$, where $\mathcal{S}, \mathcal{A}, \gamma$ are the state space, action space, and discount factor, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, $r = \{r_g | r_g : \mathcal{S} \to \mathbb{R}, g \in G\}$ is the goal-conditioned reward functions, $\mathcal{G}$ is the space of goals, p_g is the desired goal distribution, and $\phi : \mathcal{S} \to \mathcal{G}$ is a tractable mapping function that maps the state to a specific goal. For a fixed goal g, solving it means finding an optimal policy from a standard MDP $M_g = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r_g, \gamma \rangle$. There are three kinds of goals in goal-conditioned RL: desired goal is the requirements of the task, and the desired goal distribution is denoted as p_{dg}; achieved goal is the corresponding goal achieved by the current timestamp and state, and the achieved goal distribution is denoted as p_{ag}; behavioral goal is the target goal for sampling trajectory in the current episode [4]. The policy parameterized by θ is modeled as $\pi_\theta : \mathcal{S} \times \mathcal{G} \to \Delta(\mathcal{A})$ based on the idea of universal value function approximators (UVFA) [5]. The objective of goal-conditioned RL is to maximize the expectation $\mathbb{E}_{g \sim p_{dg}, \pi}[\sum_{t=0}^{\infty} \gamma^t r_g(s_t)]$.

In the learning process of GCRL, selecting goals of appropriate difficulty for the current learning policy is crucial [8]. This involves two sub-problems [9]: the first is to estimate the goal-achieving ability p_{ag}^{π} of the current policy; the second is to select goals of appropriate difficulty based on p_{ag}^{π} for training. For the first problem, off-policy evaluation (OPE) [10], [11] can be used for efficient estimation. One of the most practical methods is to periodically evaluate the training policy during training, each time evaluating N trajectories, attaching a weight w to each result, and decaying historical w with a factor κ. Then, a Gaussian Mixture Model (GMM) [12] can be used to fit p_{ag}^{π} based on these weighted evaluation results. For the second problem, there are three methods to sample goals from p_{ag}^{π}: (1) RIG: directly sampling goals from p_{ag}^{π} [13], (2) DISCERN: using uniform distribution to sample goals from the support set of p_{ag}^{π} [14], and (3) MEGA: using inverse probability weighting [15] to sample goals p_{ag}^{π} [8].

Pre-train and fine-tune RL policy. When demonstrations consist of sequences of states and actions (without rewards), Imitation Learning (IL) [16] is commonly employed to pre-train policies. IL is a sample-efficient method for learning policies by imitating demonstrators [17]. Among IL methods, Behavioral Cloning (BC) [18] is widely utilized for pre-training policies due to its simplicity and effectiveness [3], [19], [20]. For a set of demonstrations $\mathcal{D}_E = (\tau_1, \ldots, \tau_N)$, where $\tau_i = \{(s_1, a_1), \ldots, (s_{T_i}, a_{T_i})\}$ is a sequence of length T_i of state-action pairs sampled by the demonstrator $\pi_E(\cdot|s_t)$ through environment dynamics $\mathcal{T}(\cdot|s_t, a_t)$, BC learns the policy by optimizing the following objective [21]:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(s,a) \sim \mathcal{D}_E}[\log \pi_\theta(a|s)]. \tag{1}$$

On-policy RL [22] is commonly used to fine-tune RL policies. During the fine-tuning process, it is typical to incorporate a regularization penalty between the training policy and the pre-trained policy into the original RL optimization objective to prevent the training policy from deviating too much from the pre-trained policy [3], [19], [20]. The Kullback–Leibler (KL) divergence [23] is most frequently used as the regularization penalty [24],

$$J(\pi) = \mathbb{E}\big[\sum_{t=0}^{H-1} \gamma^t \big(r(s_t, a_t) - \lambda log(\frac{\pi(a_t|s_t)}{\pi^0(a_t|s_t)})\big)\big]. \tag{2}$$

The KL regularization prevents the policy from forgetting the skills acquired in the pre-training stage during the fine-tuning process [24]. Additionally, it stabilizes the fine-tuning process [3].

Iterative Regularized Policy Optimization (IRPO) [3] is a recently proposed on-policy GCRL algorithm. During training, IRPO alternates between (1) Training policies based on demonstrations using IL. (2) Continuously training policies using on-policy RL methods. (3) Using the trained policies to boost the quality of demonstrations. In terms of performance, IRPO can improve both the policy's performance and the quality of the demonstrations with the progression of training iteration. This study utilizes IRPO for pre-training policies during the pre-training stage. Although IRPO enhances the policy's goal achievement rate, it does not guarantee the trained policy a continuous and stable performance in the goal space. This is the motivation behind this research, aiming to ensure that the policy can achieve good performance on specific goals after being deployed on robot terminals. To achieve this, a simulator is deployed on the robot terminals, allowing them to evaluate and fine-tune the policy based on specific goals.

III. Methodology

In this section, we first introduce a method for quantitatively analyzing the discontinuity of policy in goal space, then the TBA framework.

A. The Discontinuity of Policy in Goal Space

Let $|\mathcal{G}|$ denotes the dimension of the goal space $\mathcal{G}$, $\Delta \boldsymbol{g} = (\Delta g_1, \Delta g_2, \cdots, \Delta g_{|\mathcal{G}|})$ the perturbation. For a specific goal $\boldsymbol{g} = (g_1, g_2, \cdots, g_{|\mathcal{G}|})$, define the perturbed goal set $p_{\Delta \boldsymbol{g}}(\boldsymbol{g}) = (\boldsymbol{g}'_1, \boldsymbol{g}'_2, \cdots, \boldsymbol{g}'_{2|\mathcal{G}|})$, where $\boldsymbol{g}'_{2i-1} = (g_1, g_2, \cdots, g_i + \Delta g_i, \cdots, g_{|\mathcal{G}|})$ and $\boldsymbol{g}'_{2i} = (g_1, g_2, \cdots, g_i - \Delta g_i, \cdots, g_{|\mathcal{G}|}), i \in [1, 2, \cdots, |\mathcal{G}|]$.

We use $C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g}))$ to represent the number of goals in $p_{\Delta \boldsymbol{g}}(\boldsymbol{g})$ that the policy π can achieve when it fails to achieve $\boldsymbol{g}$:

$$C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g})) = \begin{cases} \sum_{i=1}^{2|\mathcal{G}|} f(\boldsymbol{g}'_i), & \text{if } f(\boldsymbol{g}) = 0 \\ -1, & \text{else} \end{cases}, \tag{3}$$

where $f(\boldsymbol{g})$ represents whether the policy π can achieve the goal $\boldsymbol{g}$, with $f(\boldsymbol{g}) = 1$ indicating that it can and $f(\boldsymbol{g}) = 0$

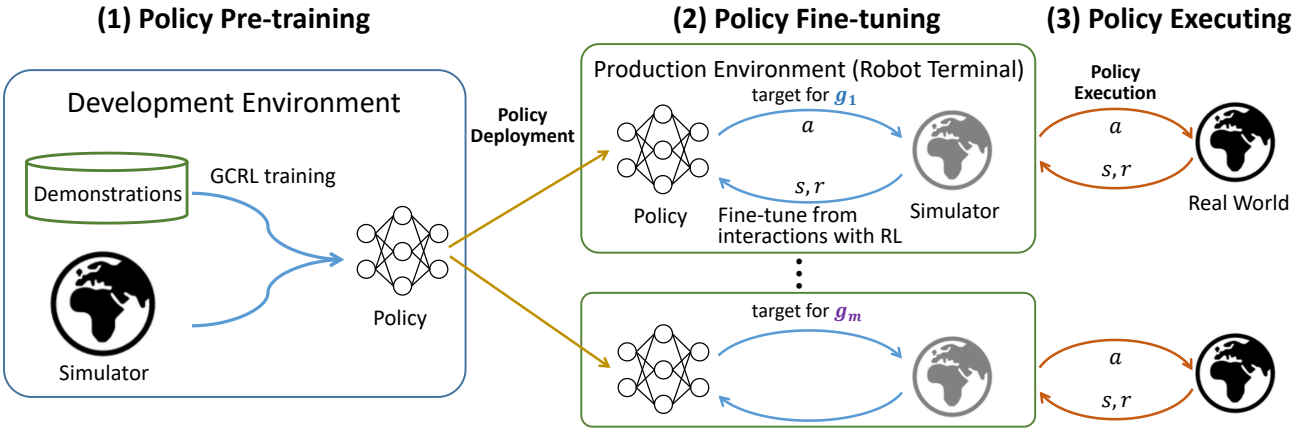

Fig. 1. The TBA framework.

indicating that it cannot. Based on this, we define the discontinuity as follows:

$$s_k(\Delta \boldsymbol{g}) = \sum_{\boldsymbol{g} \in \mathcal{G}_{TEST}} \frac{\mathbb{I}(C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g})) = k)}{|\mathcal{G}_{TEST}|}, \tag{4}$$

where $\mathbb{I}(\cdot)$ is an indicator function; if $C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g})) = k$, then $\mathbb{I}(C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g})) = k) = 1$, otherwise, $\mathbb{I}(C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g})) = k) = 0$, and $\mathcal{G}_{TEST}$ is the set of test goals. We refer to $s_k(\Delta \boldsymbol{g})$ as the k-degree discontinuity of policy π with respect to the perturbation $\Delta \boldsymbol{g}$. This term represents the proportion of test goals $\boldsymbol{g}$ that the policy π cannot achieve, but it can exactly achieve k perturbation goals of $\boldsymbol{g}$ defined on $\Delta \boldsymbol{g}$. A high value of s_k indicates that the policy π is capable of achieving k neighboring goals around a specific goal $\boldsymbol{g}$, but it fails to achieve $\boldsymbol{g}$ itself. This highlights the discontinuity of policy π, and the larger the k in s_k, the more it reflects the discontinuity of π. We provide a detailed analysis of the discontinuity in goal space for policies trained by various GCRL methods in Section IV-B.

B. The TBA Framework

Algorithm 1 The algorithm of TBA running on robot terminals

Input: pre-trained policy π, the target goal: $\boldsymbol{g}$
Output: π

1: evaluate π on $\boldsymbol{g}$ with observation noise to get the performance on $\boldsymbol{g}$: $J[\pi(\cdot|s, \boldsymbol{g})]$
2: **if** $J[\pi(\cdot|s, \boldsymbol{g})]$ does not meet the performance requirement **then**
3: fine-tune π by interacting with the simulator with GCRL on $\boldsymbol{g}$
4: **end if**
5: execute π in real-world environment

The structure of the TBA framework is depicted in Figure 1. The TBA framework consists of two parts: the pre-training algorithm running in the development environment and the fine-tuning algorithm running in the production environment (robot terminal).

During the development stage, the primary goal is to train a goal-conditioned policy that is as general as possible, capable of achieving a wide range of various goals. To achieve this, we define $p_{dg} = \mathcal{U}(\mathcal{G})$, where $\mathcal{U}(X)$ denotes the uniform distribution over the set X. This ensures that the policy is trained on a diverse set of goals. Subsequently, GCRL algorithms such as off-policy algorithms like HER [25] and MEGA [8], as well as demonstration-based algorithms like IRPO [3], are employed to train the policy. Once training is complete, the policy is deployed to the robot terminals.

In the production stage, the main purpose is to ensure that the robot can achieve a specific goal $\boldsymbol{g}$. Due to the discontinuity of the policy in goal space, it is necessary to evaluate the policy's performance on $\boldsymbol{g}$ in the real-world environment before actually executing π. Since in practice, the robot terminal can only evaluate goal $\boldsymbol{g}$ within the simulator, to approximate the effects of evaluating in the real world, we refer to commonly used methods in sim-to-real research and add random noise to the observations during evaluation [26]. If the performance does not meet the requirements, fine-tuning of π is required. The algorithm employed during this stage is depicted in Algorithm 1. First, the robot evaluates π on $\boldsymbol{g}$ to determine if it can meet the performance requirements (line 2). If cannot, the robot fine-tunes π by interacting with the simulator on $\boldsymbol{g}$ using GCRL algorithms (line 3). When completing the fine-tuning, the robot executes π in the real-world environment (line 5).

In Section IV-C, we demonstrate through experiments that a policy that has been well-pre-trained can achieve ideal performance with only a small number of interactions, showcasing the feasibility of deploying the TBA framework on actual robotic systems.

IV. EXPERIMENTS

A. Settings

We employ the velocity vector control of fixed-wing UAVs [6] as the experimental scenario. The velocity vector control of fixed-wing UAVs exhibits the following characteristics: (1) Multi-goal problem: The fixed-wing UAV is required to achieve any velocity vector in three-dimensional space, which is a typical multi-goal problem. (2) Long control sequences: For well-trained policies, the average number of steps required to achieve a goal is over 100, and more challenging goals can demand upwards of 300 steps. The long horizon of this task makes it more challenging than common multi-goal tasks, such as robot arm control [2], and articulated-body control [27]. Thus the velocity vector control of fixed-wing UAVs represents a difficult multi-goal problem suitable for validating GCRL algorithms. The specific settings and hyper-parameters for velocity vector control are referenced from [6].

To illustrate the ubiquity of discontinuity in goal space among policies trained by GCRL algorithms, we employ a variety of GCRL methods: (1) IRPO [3]: IRPO is an on-policy GCRL algorithm that alternately trains policies and boosts the quality of demonstrations. IRPO is capable of obtaining policies of increasingly higher quality as iterations progress. (2) Self-curriculum GCRL methods: When employing OPE methods to estimate p_{ag}, different values of N and κ yield different estimates of p_{ag}. Consequently, policies obtained from GCRL using these different estimates of p_{ag} exhibit varying performance. We evaluate policies obtained under different conditions using the aforementioned algorithms and analyze their discontinuity in goal space, which is detailed in Section IV-B.

In the implementation of TBA, during the pre-training stage in the development environment, we utilize IRPO to train policies. IRPO initially employs BC to train policies offline on the demonstrations, followed by Proximal Policy Optimization (PPO) [28] with an extra KL regularization loss (Eq. 2) in the objective [24] for online policy training through interactions with the environment. During the PPO training stage, we utilize MEGA to sample goals of appropriate difficulty for training purposes. We conduct four training iterations with hyper-parameters listed in [3]. In the fine-tuning stage of the production environment, we employ PPO for policy fine-tuning, maintaining the same hyper-parameters as those used in the pre-training stage.

In the experiments, we employ the discrete goal set defined in Table 5(b) of [3] as the test goal set. For the goal space defined in the environment [6], we define perturbations only in the flight path elevator angle μ and flight path azimuth angle χ, with $\Delta_\mu = 5, \Delta_\chi = 5$, respectively.

The configuration of the robot terminals used in the experiments is as follows: Intel Core i9-10980XE CPU, 8GB memory, no GPU, Ubuntu 20.04 operating system. The Stable Baselines3 framework [29] is utilized for the training of PPO and IRPO and the Imitation framework [30] is employed for the training of BC. All the algorithms' hyper-parameters are referenced from [3].

B. The Discontinuity of Policy in Goal Space

We refer to the policies trained by [3] and evaluate these policies using the method detailed in Section III-A, with the results presented in Fig. 2. These evaluated policies originate from different iterations and different KL regularization strengths within the IRPO framework. As [3] demonstrates, IRPO iteratively enhances the success rate of the policy. In the earlier training iterations, the policy is capable of achieving some simpler goals, and a relatively small strength of KL regularization is found to be beneficial for the policy to achieve a greater number of goals. However, as training progresses to later iterations, the policy is able to achieve a broader spectrum of goals, including some more challenging ones. In these later iterations, a relatively large strength of KL regularization is observed to be advantageous for the policy to achieve a greater number of goals. From Fig. 2, it is evident that (1) $\sum_{k=1}^{4} s_k$ for almost all policies exceeds 0.1, indicating that unless we precisely evaluate the policy on $\boldsymbol{g}$, we can only be reasonably confident that our evaluation of π's performance on $\boldsymbol{g}$ is accurate to within 90%. The higher the goal achievement rate of a policy, the less continuous its goal achievement ability in the goal space. This suggests that policies with better goal achievement performance should be approached with caution when evaluating their performance.

We train policies with self-curriculum methods detailed in Section II and evaluate these policies using the method detailed in Section III-A, with the results presented in Fig. 3. Utilizing larger values of N and smaller values of κ slightly improves the policy's performance, but overall, policies trained with different combinations of N and κ do not exhibit significant performance differences. When analyzing the policy's discontinuity in goal space, the differences between policies trained with various N and κ values are minimal, yet all policies have a $\sum_{k=1}^{4} s_k$ value exceeding 0.2, suggesting that we can only be reasonably confident that our evaluation of π's performance on $\boldsymbol{g}$ is accurate to within 80%.

In summary, although different GCRL algorithms and hyper-parameters can train policies with varying performance, these policies generally exhibit the problem of discontinuity in goal space. This is the motivation behind the design of the TBA framework, which evaluates the policy π on goal $\boldsymbol{g}$ before the robot executes $\boldsymbol{g}$ and determines whether to fine-tune π on $\boldsymbol{g}$ based on the evaluation results.

C. The Performance of TBA Framework

To evaluate the performance of the TBA framework, we evaluate the policies obtained from the fourth training iteration of IRPO. We randomly select goals that the policies could not achieve and analyze their discontinuity in goal space. Subsequently, we fine-tune the policies on the robot terminals based on the corresponding goals, with the results presented in Table I. It can be observed that:

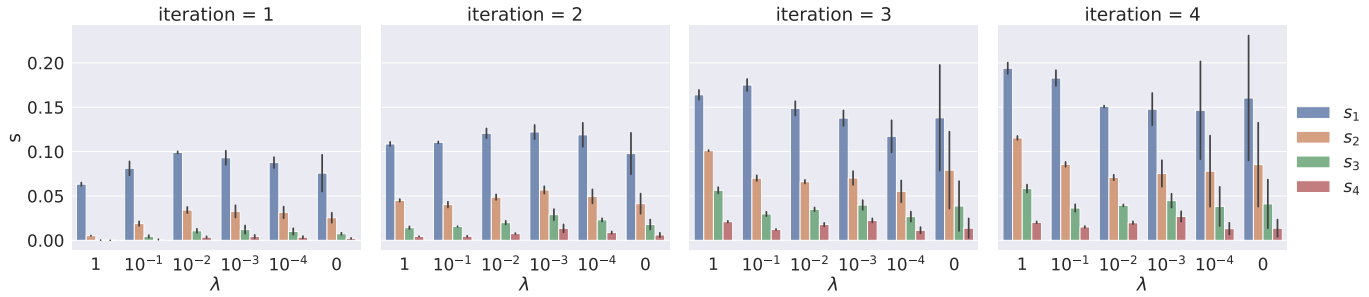

Fig. 2. s_1, s_2, s_3, and s_4 of policies from different iterations and different KL regularization strengths of IRPO. Results are derived from experiments across 5 random seeds.

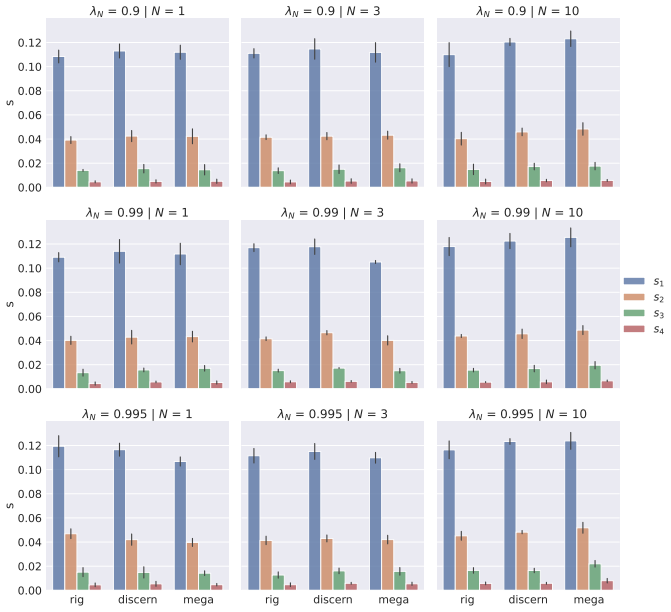

Fig. 3. s_1, s_2, s_3, and s_4 of policies from different N and κ of self-curriculum methods of GCRL. Results are derived from experiments across 5 random seeds.

TABLE I
CASE STUDIES.

Target $\boldsymbol{g}$	$C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g}))$	Pre-train	Fine-tune			
		Return	Train steps	Time (min)	Success?	Return
(130,80,165)	0	-152.97	$< 2 * 10^5$	< 2	True	-43.81
(180,55,160)	0	-78.67	$< 2 * 10^5$	< 2	True	-42.52
(290,65,95)	0	-114.48	$4 * 10^6$	65	False	/
(300,50,55)	0	-104.49	$< 2 * 10^5$	< 2	True	-74.59
(120,-75,50)	0	-197.61	$4 * 10^6$	65	False	/
(290,-85,-145)	1	-182.73	$< 1 * 10^5$	< 1	True	-46.26
(100,-35,-45)	1	-104.25	$4 * 10^6$	65	False	/
(120,80,95)	1	-171.91	$< 1 * 10^5$	< 1	True	-39.48
(110,-20,55)	1	-61.16	$< 2 * 10^5$	< 2	True	-38.22
(120,-65,-140)	1	-110.71	$< 1 * 10^5$	< 1	True	-48.28
(290,50,0)	2	-198.71	$< 2 * 10^5$	< 2	True	-60.10
(280,55,-40)	2	-99.95	$< 1 * 10^5$	< 1	True	-61.46
(120,80,105)	2	-174.39	$< 1 * 10^5$	< 1	True	-40.55
(250,70,60)	2	-118.06	$< 1 * 10^5$	< 1	True	-59.91
(290,55,-45)	2	-103.20	$< 1 * 10^5$	< 1	True	-75.52
(300,-85,-150)	3	-192.86	$< 2 * 10^5$	< 2	True	-47.46
(110,0,165)	3	-67.18	$< 1 * 10^5$	< 1	True	-67.05
(290,60,-90)	3	-153.24	$< 2 * 10^5$	< 2	True	-89.89
(300,-80,-15)	3	-187.25	$< 1 * 10^5$	< 1	True	-44.36
(280,60,-75)	3	-156.33	$< 1 * 10^5$	< 1	True	-75.12
(280,25,0)	4	-183.52	$< 1 * 10^5$	< 1	True	-40.41
(290,-10,5)	4	-185.53	$< 1 * 10^5$	< 1	True	-32.22
(130,-65,-130)	4	-98.74	$< 1 * 10^5$	< 1	True	-40.25
(290,20,15)	4	-188.15	$< 2 * 10^5$	< 2	True	-47.54
(280,30,-5)	4	-183.45	$< 1 * 10^5$	< 1	True	-38.63

TABLE II
PERFORMANCE OF POLICIES FINE-TUNED ON THE FIRST-ITERATION POLICY OF IRPO.

Target $\boldsymbol{g}$	$C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g}))$	Pre-train	Fine-tune			
		Return	Train steps	Time (min)	Success?	Return
(290,-85,-145)	1	-182.73	$4 * 10^6$	65	False	/
(120,80,105)	2	-174.39	$4 * 10^6$	65	False	/
(290,60,-90)	3	-153.24	$4 * 10^6$	65	False	/
(130,-65,-130)	4	-98.74	$4 * 10^6$	65	False	/

Firstly, from an overall perspective, fine-tuning almost enables the policies to reach all goals, especially for goals with a larger value of $C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g}))$. This indicates that goals that cannot be achieved due to the policy's discontinuity can, in most cases, be achieved through fine-tuning.

Secondly, a small number of goals remain unachievable even after fine-tuning. These goals typically have a smaller value of $C(\boldsymbol{g}, p_{\Delta \boldsymbol{g}}(\boldsymbol{g}))$, suggesting that the policies from the pre-training stage not only fail to achieve $\boldsymbol{g}$ but also the surrounding goals, indicating that the failure to achieve $\boldsymbol{g}$ is not solely due to the policy's discontinuity but rather that the policy has not learned these goals at all.

Thirdly, for all goals that can be achieved through fine-tuning, the number of sampling steps required is less than $2 * 10^5$, corresponding to a wall clock time of less than 2 minutes. This demonstrates that fine-tuning π on a fixed $\boldsymbol{g}$ requires minimal computational resources and will not overload the robot terminal with computational demands.

D. Ablation Studies

In this section, we analyze: (1) The impact of pre-trained policies on fine-tuning. (2) The overall impact of fine-tuning on the policy's performance. (3) Whether fine-tuning the policy on goals it already achieves can enhance its achievement performance. (4) Strategies for fine-tuning difficult goals.

TABLE III
SUCCESS RATE (%) OF POLICY BEFORE AND AFTER FINE-TUNING. THE PRE-TRAINING RESULTS ARE DERIVED FROM EXPERIMENTS ACROSS 5 RANDOM SEEDS, AND THE FINE-TUNING RESULTS ARE AVERAGED OVER ALL POLICIES TRAINED IN TABLE I.

Success rate before fine-tuning	Success rate after fine-tuning
71.68±2.8	19.16±11.55

TABLE IV
PERFORMANCE OF POLICIES FINE-TUNED ON THEIR ACHIEVABLE GOALS.

Target $\boldsymbol{g}$	Pre-train		Fine-tune		
	Success?	Return	Train steps	Success?	Return
(280,-85,120)	True	-41.40	$4 * 10^6$	True	-41.58
(280,-10,160)	True	-64.38	$4 * 10^6$	True	-64.36
(300,-35,-160)	True	-68.24	$4 * 10^6$	True	-64.57
(300,-30,-150)	True	-67.35	$4 * 10^6$	True	-66.88
(220,60,-110)	True	-43.27	$4 * 10^6$	True	-46.50

Poorly performing pre-trained policies are detrimental to the subsequent fine-tuning of the policy. We utilize the policy obtained from the first training iteration of IRPO as the pre-trained policy, which has a success rate of 38.31%, reaching only 53.4% of the success rate achieved by the fourth-iteration policy. As can be seen from Table I, on these four goals, the policy based on the fourth-iteration policy could be successfully fine-tuned. Conversely, as indicated in Table II, the policy based on the first-iteration policy could not be successfully fine-tuned. This demonstrates the importance of the performance of the pre-trained policy for subsequent fine-tuning.

Fine-tuning a goal-conditioned policy on a specific goal can lead to a decrease in the policy's performance on the desired goal distribution p_{dg}. We evaluate the policy's performance on p_{dg} both before and after fine-tuning, and present the corresponding results with Table III. It can be observed that, before fine-tuning, the policy's success rate reaches 71.68%, whereas, after fine-tuning, the success rate drops to 19.16%. The reason for this decline is that the optimization objective of GCRL is to maximize the policy's cumulative rewards on p_{dg} [4]. However, using fixed goal sampling data does not align with the maximization of the GCRL optimization objective. Consequently, fine-tuning the policy on a specific goal comes at the cost of reducing its ability to achieve other goals.

Do not fine-tune policy on its already achievable goals. Fine-tuning a policy on its achievable goals will not significantly enhance the policy's performance on those goals. We fine-tune the policy on some of the goals that the pre-trained policy could already achieve and evaluate the performance of the fine-tuned policy on those goals. The results are presented in Table IV. It can be seen that there is no significant difference in performance between the fine-tuned and pre-tuned policies on these goals. However, in light of the previous point, fine-tuning the policy on a specific goal comes at the cost of reducing its ability to achieve other goals. Therefore, we suggest that do not fine-tune the policy on goals it can already achieve.

TABLE V
PERFORMANCE OF POLICIES FINE-TUNED ON $\boldsymbol{g}$ AND ITS NEIGHBORING GOALS g'.

Target $\boldsymbol{g}$	$C(\boldsymbol{g}, p_{\triangle \boldsymbol{g}}(\boldsymbol{g}))$	Pre-train	Fine-tune		
		Return	Train steps	Success?	Return
(290,65,95)	0	-114.48	$4 * 10^6$	True	-85.81
(120,-75,50)	0	-197.61	$4 * 10^6$	True	-46.20
(100,-35,-45)	1	-104.25	$4 * 10^6$	True	-53.84

For a specific difficult goal $\boldsymbol{g}$, sampling multiple neighboring goals g' during fine-tuning can help the policy achieve $\boldsymbol{g}$. For goals in Table I that remain unachievable after fine-tuning, we randomly sample neighboring goals g' in their vicinity for training and show the results in Table V. It can be observed that training with neighboring goals g' promotes the policy's ability to achieve these difficult goals. This aligns with the optimization principle of GCRL and suggests that learning multiple goals simultaneously is more beneficial for the policy to grasp the ability to achieve goals [25].

V. DISCUSSIONS

The TBA framework currently represents a basic framework and there are many aspects that can be optimized:

First, if the robot can communicate with the development environment during task execution, then: (1) Fine-tuning can be performed in the development environment, eliminating the need to deploy a simulator on the robot terminals, which can reduce the hardware requirements for the robot terminals. (2) The robot terminals can return the interaction data with the real-world environment to the development environment to assist in pre-training higher-quality policies. This can create a closed loop where the development environment and the robot terminals continuously improve policies through pre-training, deploying, fine-tuning policies and collecting interactions.

Second, if the simulator requires substantial resources, two methods can be explored: (1) Model-Based RL [31]: Train a lightweight world model using a model-based method and deploy it on the robot terminals. This approach reduces the demand on the robot terminals' hardware for simulation. (2) Data-Driven Fine-tuning [20]: Store a subset of high-quality data on the robot terminals and use BC to fine-tune policies based on this data. While BC fine-tuning may not achieve the same level of performance as RL fine-tuning, it can still improve the policy's performance to some extent, eliminating the need to deploy a simulator on the robot terminals. The work referenced in [20] demonstrates the effectiveness of BC fine-tuning in this context.

Third, we consider TBA to be a transitional method. A more effective approach would be to train policies during the development stage that exhibit continuous and stable

performance in the goal space. We observe that in single-goal RL, [32], [33] have studied methods for obtaining policies with continuous and stable outputs in the action space. The core idea of these methods is to ensure that the RL policy satisfies the Lipschitz continuity condition [34]. Exploring how to extend these methods to the goal space is a promising research direction.

VI. Conclusion

This paper first demonstrates experimentally that GCRL policies generally exhibit the problem of discontinuity in goal space, which prevents us from effectively estimating the policy's performance in the actual production environment based solely on its evaluation results in the development environment. To ensure satisfactory performance of the policy in the production environment, we propose the TBA framework. Furthermore, we validate the effectiveness of TBA on the velocity vector control of fixed-wing UAVs. Additionally, we conduct extensive ablation studies to identify the best practices for employing TBA.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2021ZD0112904).

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv preprint arXiv:1802.09464*, 2018.

[3] X. Gong, D. Feng, K. Xu, Y. Zhai, C. Yao, W. Wang, B. Ding, and H. Wang, "Iterative regularized policy optimization with imperfect demonstrations," in *International Conference on Machine Learning*. PMLR, 2024.

[4] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," in *International Joint Conference on Artificial Intelligence*, 2022, pp. 5502–5511.

[5] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International conference on machine learning*. PMLR, 2015, pp. 1312–1320.

[6] X. Gong, D. Feng, K. Xu, W. Wang, Z. Sun, X. Zhou, B. Ding, and H. Wang, "Flycraft: An efficient goal-conditioned reinforcement learning environment for fixed-wing uav velocity vector control," https://github.com/GongXudong/fly-craft, 2024.

[7] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1025–1037.

[8] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba, "Maximum entropy gain exploration for long horizon multi-goal reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7750–7761.

[9] G. Xudong, F. Dawei, X. Kele, D. Bo, and W. Huaimin, "Goal-conditioned on-policy reinforcement learning," *Advances in neural information processing systems*, 2024.

[10] P. Thomas, G. Theocharous, and M. Ghavamzadeh, "High-confidence off-policy evaluation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[11] M. Uehara, C. Shi, and N. Kallus, "A review of off-policy evaluation in reinforcement learning," *arXiv preprint arXiv:2212.06355*, 2022.

[12] D. A. Reynolds *et al.*, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[13] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," *Advances in neural information processing systems*, vol. 31, 2018.

[14] D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih, "Unsupervised control through non-parametric discriminative rewards," in *International Conference on Learning Representations*, 2018.

[15] A. D. Jacq, M. Orsini, G. Dulac-Arnold, O. Pietquin, M. Geist, and O. Bachem, "On the importance of data collection for training general goal-reaching policies," in *Sixteenth European Workshop on Reinforcement Learning*, 2023.

[16] T. Osa, J. Pajarinen, G. Neumann, J. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[17] S. Belkhale, Y. Cui, and D. Sadigh, "Data quality in imitation learning," *Advances in neural information processing systems*, 2023.

[18] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[19] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[20] B. Baker, I. Akkaya, P. Zhokov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune, "Video pretraining (vpt): Learning to act by watching unlabeled online videos," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 639–24 654, 2022.

[21] F. Sasaki and R. Yamashina, "Behavioral cloning from noisy demonstrations," in *International Conference on Learning Representations*, 2020.

[22] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "A closer look at deep policy gradients," in *International Conference on Learning Representations*, 2020.

[23] V. K. Rohatgi and A. M. E. Saleh, *An introduction to probability and statistics*. John Wiley & Sons, 2015.

[24] N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist, "Leverage the average: an analysis of kl regularization in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 163–12 174, 2020.

[25] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[26] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen, "Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization," in *2019 international conference on unmanned aircraft systems (ICUAS)*. IEEE, 2019, pp. 523–533.

[27] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[29] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[30] A. Gleave, M. Taufeeque, J. Rocamonde, E. Jenner, S. H. Wang, S. Toyer, M. Ernestus, N. Belrose, S. Emmons, and S. Russell, "imitation: Clean imitation learning implementations," arXiv:2211.11972v1 [cs.LG], 2022. [Online]. Available: https://arxiv.org/abs/2211.11972

[31] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[32] S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko, "Regularizing action policies for smooth control with reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1810–1816.

[33] Q. Shen, Y. Li, H. Jiang, Z. Wang, and T. Zhao, "Deep reinforcement learning with robust and smooth policy," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8707–8718.

[34] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," *Advances in Neural Information Processing Systems*, vol. 31, 2018.