IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

Robust Adaptive Ensemble Adversary Reinforcement Learning

Peng Zhai, Taixian Hou, Xiaopeng Ji, Zhiyan Dong, Lihua Zhang

***Abstract*—Reinforcement learning needs to learn policies through trial and error. The unstable policies in the early stage of training make it expensive (and time-consuming) to train directly in the real environment, which may cause disastrous consequences. The popular solution is to use the simulator to train the policy and deploy it in a real environment. However, the modeling error and external disturbance between the simulation and the real environment may fail the physical deployment, resulting in the sim2real transfer problem. In this letter, we propose a novel robust adversarial reinforcement learning framework, which uses the ensemble training of multi-adversarial agents that can adaptively adjust adversaries' strength to enhance RL policy's robustness. More specifically, we take the accumulative reward as feedback and construct a PID controller to adjust the adversary's output magnitude to perform the adversarial training well. Experiments in the simulated and the real environment show that our algorithm improves the generalization ability of the policy for the modeling error and the uncertain disturbance simultaneously, outperforming the next best prior methods across all domains. The algorithm was further proven to be effective in a sim2real transfer task through the load experiment of a real racing drone, and the tracking performance is better than the PID-based flight controller.**

***Index Terms*—Reinforcement Learning; Machine Learning for Robot Control; Robust/Adaptive Control**

I. Introduction

DEEP reinforcement learning (RL) methods have made tremendous progress in many high-dimensional tasks. It provides a general method for robots to acquire new skills and has shown promise in various robotic domains such as navigation [1], control [2] and locomotion [3]. Since the RL algorithm requires a lot of data during training and obtaining data from real-world robots is expensive and laborious, those methods are typically trained in the simulator before being deployed in the real world. However, the modeling error and various disturbances in the real environment (sim2real gap) make the policy trained in simulation challenging to generalize to the real world. This effect will be more significant in a low-level control policy, resulting in disastrous consequences [4]. Therefore, a method that can improve policy's robustness is required to bridge the gap between the simulation and the real environment.

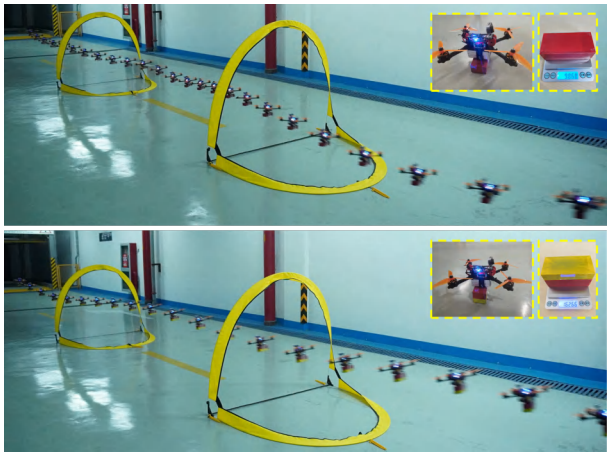

Fig. 1. RAEARL is able to efficiently perform Sim2Real tasks of the quadcopter carrying different payloads flying over the racing gate, which are difficult to control precisely [7]. We demonstrate that the RAEARL controller can accurately track the desired angular velocity in real environments with a smaller tracking error than the PID controller.

In robotic areas, domain randomization is an effective method of learning robustness, asking a designer to define a distribution over environments to which the agent should be robust. The parameters that need generalization (such as friction, mass, damping, etc.) are then randomly assigned during training to ensure the average robustness of the agent to this set of parameters [5]. However, finding the correct distribution to randomize the parameters is complex and requires expertise. If the distribution is too large, the task becomes more difficult to learn the robust policy; if the distribution is too small, the policy will not generalize effectively [6]. Furthermore, as we demonstrate in Section IV, domain randomization is not always effective in tasks sensitive to environmental parameters and even degrades the performance of RL policies.

Another easier method to automate is to model environmental differences as adversarial disturbances. By describing the sim2real problem as a zero-sum game, the normal agent

Manuscript received: August, 20, 2022; Accepted October, 17, 2022.

This paper was recommended for publication by Editor Asfour Tamim and Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by China Postdoctoral Science Foundation under Grant No. BX20220071, in part by Shanghai Municipality Science and Technology Major Project 2021SHZDZX0103 and in part by National Key R&D Program of China under Grant No. 2021ZD0113502. (Corresponding author: Zhiyan Dong and Lihua Zhang.)

Peng Zhai, Taixian Hou are with the Academy for Engineering and Technology, Fudan University, Shanghai 200433, China pzhai@fudan.edu.cn; txhou21@m.fudan.edu.cn

Xiaopeng Ji is with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China xp.ji@cad.zju.edu.cn

Zhiyan Dong is with the Academy for Engineering and Technology, Fudan University, Shanghai 200433, China, and the Ji Hua Laboratory, Foshan 200433, China. dongzhiyan@fudan.edu.cn

Lihua Zhang is with the Academy for Engineering and Technology, Fudan University, Shanghai 200433, China, the Institute of Meta-Medical, Fudan University, Shanghai 200433, China, and the Jilin Provincial Key Laboratory of Intelligence Science and Engineering, Changchun, China lihuazhang@fudan.edu.cn

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

learns a robust policy in an environment with an adversary that disturbs the transition dynamics [8]. The normal and adversarial agents are iteratively optimized to solve a global Nash equilibrium, which provides the worst-case performance bound under a specific disturbance set. Similar to the method of domain randomization, adversarial training requires careful adjustment of the adversary's strength. If the strength is too large, the normal agent's policy learning becomes more difficult; if the strength is too small, the normal agent cannot obtain a robust policy [9].

In this letter, we propose a new algorithm motivated by multi-adversaries with adaptive adversarial strength control, namely the Robust Adaptive Ensemble Adversarial Reinforcement Learning (RAEARL). This algorithm alleviates the overfitting of the normal agent to a specific disturbance while making a trade-off between robustness and training complexity. The contributions of this work are: (1) a new multi-adversary adversarial RL framework introduces the mirror agent to decouple the normal and adversarial agents' training, making it more suitable for distributed computing and enhancing the stability of multi-adversarial training, (2) a PID-controller-based adversarial strength adaptive control strategy adjusts the training complexity by controlling the adversary's output magnitude, (3) experimental evaluation against three state-of-the-art robust RL baselines on four different MuJoCo environments and the sim2real transfer experiment was carried out on the real racing drone. The results show that RAEARL achieves optimal parameter adaptability and anti-disturbance ability in all simulated environments. In addition, the flight experiments on the racing drone show that the RAEARL algorithm obtains a powerful controller which can carry a large-mass load for flexible flight. Its tracking performance is better than the PID-based flight controller of the open source firmware.

II. Related work

Our method is related to the literature on the robust adversarial RL method, ensemble method, and dynamical systems view of optimization.

Robust adversarial reinforcement learning builds upon robust control theory. Its basic idea is to model various system uncertainties as disturbances and find the optimal controller under the worst disturb [10]. Morimoto et al. modeled disturbances as adversarial agents and found the policy robust to disturbance by solving the minimax value function of the normal and adversarial agents [11]. It was the first study to introduce an adversarial method into RL theory. Pinto et al. introduced the idea of minimax into the deep RL and proposed Robust Adversarial Reinforcement Learning (RARL) with considerable empirical success [8]. Tessler et al. proposed Noise Robust Markov Decision Process (NR-MDP), which makes the adversary's disturbance directly superimposed on the normal agent's action [12]. However, as mentioned earlier, setting an appropriate adversarial strength is difficult. A strong adversary will destabilize the system during training, resulting in learning failure, while a weak adversary cannot effectively model the uncertainty of the environment.

The ensemble method brings diversified information through multiple independent components, which is a powerful approach to alleviate the overfitting of the model to a specific domain [13]. Ensemble methods in RL areas improve sample efficiency or performance by constructing multiple RL algorithm components, such as Q-value function [14], policy [15], environment [16], etc. More related to the RAEARL are ensemble techniques in adversarial training. Quan et al. proposed a multi-generator ensemble training approach to address the mode-collapse problem in generative adversarial networks (GAN) [17]. Behnam et al. improved the stability and consistency of the generator by ensemble the discriminator [18]. Hiroaki extended the RARL Algorithm by training multiple adversarial agents for the normal agent [19]. However, Hiroaki pointed out that the ensemble of adversaries made the task much harder for the policy to learn. A strong set of adversaries can permanently destroy the system's stability during learning and lead to task failure.

Several recent works have proposed using classical control theory to analyze the optimization process, including those often applied to deep learning [20]. Hu et al. reinterpreted the first-order gradient optimization as a dynamical system. They regarded the gradient $\nabla_x f$ of the objective as the plant, in which the controller's goal is to drive the plant to zero to obtain the optimal parameters [21]. An et al. reinterpreted SGD as P-control and the momentum method as PI-control. They construct a PID controller to improve the optimization of deep convolutional networks by introducing gradient-change-based derivative terms [22]. Adam et al. interpret the update of Lagrangian multipliers as integral control and introduce proportional and differential terms to stabilize the dynamics of the learning process. In this letter, we consider the training process of adversarial RL as a dynamic system and the accumulative reward of the normal agent as the plant.

III. Methodology

Our goal is to bring diverse disturbance information through adversary ensemble and ensure the stability of the training process to obtain a robust policy. Fig. 2 shows the overall framework of the proposed approach. We improve the stability during multi-adversarial training from two aspects. First, we suggest a novel ensemble mirror adversarial RL framework, which decouples the training process of the normal and the adversarial agent to ensure the continuity and stability of the adversarial agent training process. Then, considering the adversarial training process as a dynamic system, we designed a PID controller to adaptively control the adversarial strength during training to ensure the system's stability further. Finally, we discuss the implementation details of the algorithm.

A. Ensemble Adversarial Reinforcement Learning Framework

In this section, we propose the concept of mirror agent, in which the mirror agents have the same network structure and initialized parameters as their counterparts, but the parameter updates lag one step behind. In each rollout, an adversary's mirror agent is randomly selected for adversarial training with the normal agent. The environment of each adversary is

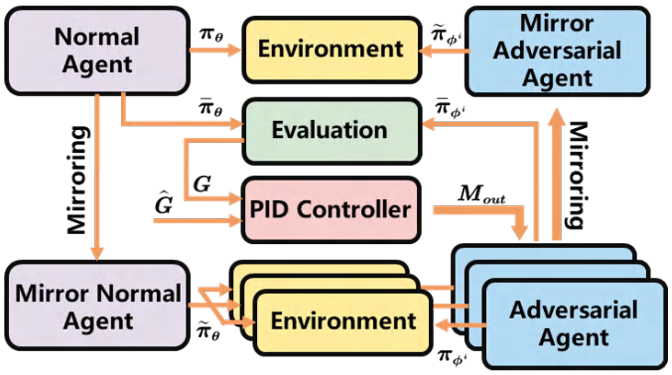

Fig. 2. The proposed Robust Adaptive Ensemble Adversarial Reinforcement Learning framework.

independent of each other and trained with the mirror of the normal agent. Denoting π_θ as the policy of the normal agent, π_{ϕ^i} as the policy of the i-th adversary, and $i \sim U(1, H)$ as the discrete uniform distribution from 1 to H, the normal agent's objective is:

$$\max_{\theta} \mathbb{E}_{i\sim U(1,H)} \left[\mathbb{E}_{a_t\sim\pi_\theta, \tilde{\omega}_t^i \sim \tilde{\pi}_{\phi^i}} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t, \tilde{\omega}_t^i) \right] \right] \quad (1)$$

The i-th adversarial agent's objective is:

$$\min_{\phi^i} \mathbb{E}_{\tilde{a}_t\sim\tilde{\pi}_\theta, \omega_t^i \sim \pi_{\phi^i}} \left[\sum_{t=0}^{T} \gamma^t r(s_t, \tilde{a}_t, \omega_t^i) \right]. \quad (2)$$

Where T is the horizon length of each episode, γ is the discount factor, a_t is the normal agent's action, ω_t^i is the adversarial agent's action, r is the reward, $\tilde{a}_t$ and $\tilde{\pi}_\theta$ is the action and policy of the mirror normal agent. $\tilde{\omega}_t^i$ and $\tilde{\pi}_{\phi^i}$ are the action and policy of the i-th mirror adversarial agent. The details of the training process are shown in Fig. 2 and Algorithm 1. In this framework, the normal and adversarial agents do not directly train with each other but indirectly through the mirror agents. Since mirror agents are updated one step behind, it is somewhat weaker than the other side during the adversarial training. In the normal agent's training stage, the mirror adversarial agent's policy update lags behind the real adversarial agent, which puts the normal agent in the dominant position to make the adversarial training more stable, and vice versa. Furthermore, the training process of each adversarial agent is continuous and independent, which brings richer adversarial information to the normal agent.

It should be noted that although the ensemble adversarial RL framework generates an additional environment for each adversary and increases the amount of computation, this framework is naturally suitable for distributed parallel computing and will not increase the algorithm execution time. On the other hand, the mirror agent can be directly implemented using the old policy for calculating the policy update distance in the on-policy methods such as Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO) without introducing additional computational complexity.

B. Control View of Adversarial Reinforcement Learning

In the adversarial RL method, the adversary influences the normal agent's accumulated reward by disturbing the environment. Among them, the adversary's policy performance and the magnitude of its output are the critical factors affecting the accumulated reward [12]. Under the same adversary's policy, the greater the disturbance magnitude, the lower the normal agent's accumulated reward, the harder the system is to train, and vice versa. Therefore, in this letter, we interpreted the normal agent's accumulated reward as the control objective and the adversarial agent as the actuator, controlling the accumulated reward by changing the adversarial agent's output magnitude and then regulating the adversarial strength of the system.

In the evaluation part of each epoch, the deterministic policies of the normal and the adversarial agent interact with the environment ($\bar{\pi}_\theta$ and $\bar{\pi}_{\phi^i}$ in Fig. 2) and feedback the normal agent's accumulated reward to the PID controller. The PID controller calculates the adversarial agent's output magnitude according to the difference between the expected accumulated reward and the actually accumulated reward. The control rate can be expressed as:

$$M_{PID} = K_P e_G + K_I \sum e_G + K_D \dot{e}_G. \quad (3)$$

Where M_{PID} is the output of the PID controller; K_p, K_i, K_d represent the proportional, integral and differential constants of the PID controller, respectively; e_G represents the normalized difference between actual accumulated reward G and the expected accumulated reward $\hat{G}$: $e_G = (G - \hat{G})/\hat{G}$. The PID controller makes the actual accumulated reward approach the expected by adjusting the adversary's output magnitude. Therefore, the expected accumulated reward can implicitly determine adversarial strength, which needs to be reasonably designed. In this letter, we define the expected accumulated reward as:

$$\hat{G} = \xi \sum_{t=0}^{T} \hat{r}_t. \quad (4)$$

Where $\hat{r}_t$ is the theoretical maximum value of the reward function, which can easily calculate in tasks with known reward functions. For example, in the inverted pendulum task, the reward function is usually set to the angle θ between the pendulum and the body. To keep the pendulum upright, the theoretical maximum value of the reward function is $\hat{r}_t = \pi/2$. $\xi \in (0, 1)$ is a proportional coefficient, which determines the adversarial strength. A large ξ makes the expected accumulated reward closer to the upper limit of the theoretical accumulated reward, indicating that the adversarial agent cannot disturb the environment too much. Conversely, low ξ will further increase the adversary's output magnitude to prevent the normal agent from obtaining high rewards.

It should be noted that in the initial stage of training, the normal agent did not learn an effective policy, and the actually accumulated reward is much smaller than expected. The PID controller will rapidly reduce the adversary's output magnitude to 0, making the adversarial agent invalid and unable to obtain

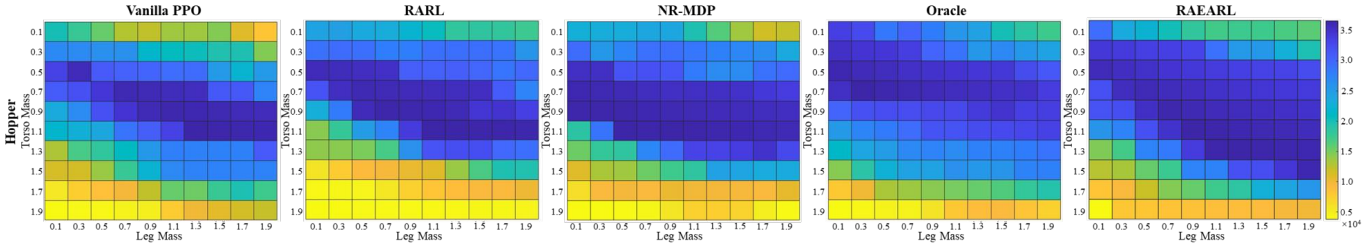

Fig. 3. Average accumulative reward across seven seeds on each test set of the Hopper environment. The x- and y-axes represent the mass changes of different parts of the robot, respectively.

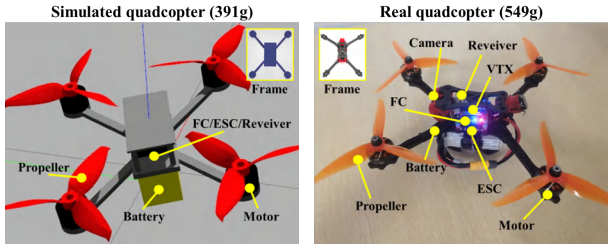

Fig. 4. Comparison of the quadcopter in the simulation and real environment.

adversarial samples; in the later stage of training, with the improvement of the performance of the normal agent's policy, the actually accumulated reward will approach the upper limit of the theoretical accumulated reward. The PID controller may continue to increase the adversary's output magnitude, and excessive disturbance may prevent the normal agent's policy from converging. To solve this problem, we set a saturation region for the PID controller's output:

$$M_{out} = min(max(M_{PID}, M_{min}), M_{max}). \tag{5}$$

Where M_{out} is the adversary's output magnitude set by the controller, M_{min} and M_{max} are the lower and upper limits of the output magnitude, respectively. M_{min} ensures the minimum adversarial disturbance at the initial stage of training and improves the normal agent's exploration efficiency. M_{max} ensures that the adversarial agent will not be too strong to affect the stability of training. M_{PID} maintains a certain intensity of confrontation in the middle stage of training.

C. Robust Adaptive Ensemble Adversarial Reinforcement Learning

We summarize RAEARL in Algorithm 1. Where H is the total number of adversarial agents; N_{iter} is the total number of the rollout; N_{Sample} is the length of each rollout trajectory; $\tilde{\pi}$ represents the mirroring policy of the corresponding agent. The RAEARL algorithm uses the following alternating procedure to optimize each agent. In each rollout, the normal agent training with the randomly selected i-th mirror adversarial agent while each adversary is trained with the mirror normal agent, and the PPO algorithm is used to update the policy parameters of all agents. The adversarial strength is then evaluated, and the PID controller is used to adjust the i-th adversary's output magnitude. This sequence is repeated until convergence.

Algorithm 1 RAEARL

Input PID parameters K_P, K_I, K_D, adversarial strength coefficient ξ, adversary's output range M_{min} and M_{max}
Initialize Environment $\mathcal{E}_\theta, \mathcal{E}_{\phi^1}, \ldots, \mathcal{E}_{\phi^H}$, rollout buffer $D_\theta, D_{\phi^1}, \ldots, D_{\phi^H}$
Initialize Normal agent policy $\pi_\theta(a\,|s)$, adversarial policies $\pi_{\phi^1}(\omega_1\,|s), \ldots, \pi_{\phi^H}(\omega_H\,|s)$ and corresponding parameters $\theta, \phi^1, \ldots, \phi^H$

for k in N_{iter} **do**
  Sample adversary $i \sim U(1, H)$, get mirror adversarial agent $\tilde{\pi}_{\phi^i}$
  for t in N_{Sample} **do**
    Run policies $a_t \sim \pi_\theta$, $\tilde{\omega}_t \sim \tilde{\pi}_{\phi^i}$ in $\mathcal{E}_\theta$
    Store transitions (s_t, a_t, r_t, s_{t+1}) in D_θ
  end for
  Get mirror normal agent $\tilde{\pi}_\theta(\tilde{a}\,|s)$
  for h in H **do**
    for t in N_{Sample} **do**
      Run policies $\tilde{a}_t \sim \tilde{\pi}_\theta$, $\omega_t \sim \pi_{\phi^h}$ in $\mathcal{E}_{\phi^h}$
      Store transitions $(s_t, \omega_t, r_t, s_{t+1})$ in D_{ϕ^h}
    end for
  end for
  Update $\theta, \phi^1, \ldots, \phi^H$ using PPO and rollout buffers $D_\theta, D_{\phi^1}, \ldots, D_{\phi^H}$
  Evaluate policy π_θ and π_{ϕ^i}, calculate accumulated reward G
  Calculate the PID controller's output using (3)
  Reset the i-th adversary's output magnitude using (5)
end for

IV. Experiments and Results

Our experiments aim to investigate the following: (1) Does our method improve generalization to model mismatch and external disturbances? (2) How does the PID controller work in adversarial training? (3) Can our method transfer efficiently in the sim2real task? To answer these questions, we designed five groups of experiments. The first three groups of experiments were performed in the MuJoCo environment. The first and second groups of experiments analyze the robustness of the proposed algorithm under environmental parameter changes and external disturbances. The third group of experiments ablated the PID controller and examined how it adjusted the adversary's magnitude during training. The fourth group of experiments was conducted in the GymFc, an RL environment

for developing attitude flight controllers for the unmanned aerial vehicle (UAV). We test the performance of different algorithms under actuator disturbances. Finally, we will directly deploy the trained policy in the GymFc environment to a real racing drone to demonstrate the potential of our method in sim2real tasks.

A. Experimental Setting

In the MuJoCo environment, we select four tasks from a modified adversarial Gym [8]: InvertedPendulumadv1, InvertedDoublePendulumAdv-v1, HopperAdv-v1, and HalfCheetahAdv-v1 to test our algorithm. All environments were trained with three adversaries. We used the PPO algorithm implemented by openAI baselines [23] as the policy optimizer for all the algorithms to compare the baseline algorithms to ours fairly. Specifically, we compared Vanilla PPO, RARL, NR-MDP, and domain randomization. We called the domain randomization method the Oracle method because it is trained directly on the test domain. In addition, we performed ablation experiments on the PID controller and mirror agents components of the RAEARL algorithm. In the N-mirror algorithm, the adversarial training is performed directly without the mirror agents. In the Annealing algorithm, the output magnitude of all adversarial agents increases with the timestep.

TABLE I
SUCCESS RATES (%) AND STANDARD DEVIATIONS OF DIFFERENT ALGORITHMS WITH 100 MASS COMBINATIONS ARE COMPARED. WHERE THE PENDULUM AND DOUBLE DENOTE THE INVERTEDPENDULUM AND INVERTEDDOUBLEPENDULUM TASKS, RESPECTIVELY. THE BEST RESULTS IN EACH CATEGORY ARE IN BOLD.

Algorithm	Pendulum	Double	HalfCheetah	Hopper
PPO	59.2±0.4	36.3±1.19	66.7±2.24	14.6±0.49
RARL	72.8±0.75	40.6±1.62	82.5±1.96	20.4±0.66
NR-MDP	71.2±0.4	32.5±1.67	84.6±1.91	28.2±0.98
Oracle	**78.0±0.2**	33.3±1.49	84.3±1.27	22.0±0.45
Annealing	70.4±0.48	34.2±1.49	77.6±1.36	24.3±0.8
N-mirror	71.5±0.32	34.7±1.84	83.2±1.16	30.2±1.01
RAEARL	**78.0±0.43**	**50.2±1.92**	**88.4±1.43**	**41.5±0.85**

In the GymFc environment, an First Person View (FPV) racing quadcopter named NF1 is used as a model and training with two adversaries. We compare the simulation and the real environment in Fig. 4, the quadcopter in the real environment has a higher mass and asymmetrical frame, which makes it have a more significant sim2real gap. Further environment's setting, pseudocode, and hyperparameter ablation study can be found in the supplement.

B. Robustness Under the Modeling Error

In this experiment, we evaluated the generalization of all algorithms to the modeling errors. Specifically, the policy is trained with fixed robot mass in the training domain. In the test domain, we changed the masses of the two body parts of the robot. The range of mass change is [0.1, 2.1) with an interval of 0.2, and the two body parts are combined into 100 groups of different mass combinations. Seven random seeds initialize the trained policies, and 700 episodes are tested for each mass group. The threshold of timesteps was set for each task. If the timesteps of the agent are larger than the threshold, the task is considered successful; otherwise, the task fails (see supplementary material for details). Table I shows that the RAEARL algorithm outperforms all baselines in most environments. In the InvertedPendulum, the RAEARL and Oracle algorithms have comparable adaptability, while in the InvertedDoublePendulum and Hopper environments, the RAEARL algorithm performs significantly better than other baselines.

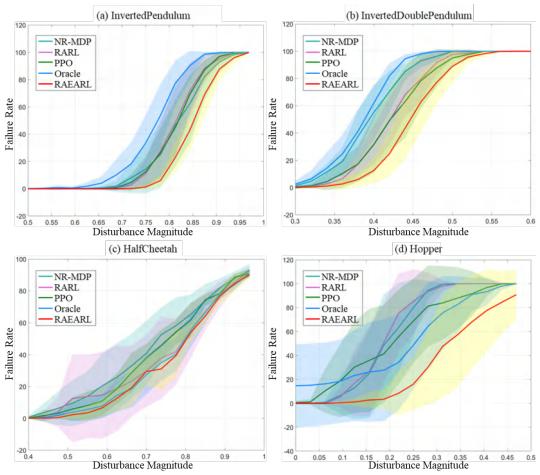

Fig. 5. Success rates of different algorithms with actuator disturbance. The trained policies are initialized by 7 random seeds, and 700 episodes are tested for each magnitude.

The results of ablation experiments show that the use of mirror agents can improve the robustness of the policy, and the simple annealing scheduler algorithm does not effectively improve the robustness of the policy. The possible reason is that the annealing scheduling algorithm cannot dynamically adjust the output magnitude according to the adversarial agents' policy performance, which leads to inappropriate adversarial strength.

Fig. 3 shows the accumulated reward heatmap for all algorithms (except the ablation algorithms) in the Hopper environment. The heatmaps of accumulated rewards and failure rates for other environments can be found in the supplementary material. We observe that the RAEARL algorithm reduces the failure rate of the agent even in environments with more extreme mass changes (the borders of the heatmap). It is worth noting that the generalization of the Oracle algorithm in the Hopper environment has deviations, which have a lower failure rate in lighter-mass environments and a higher failure rate in other environmental parameters. This suggests that the domain randomization method does not consistently achieve average robustness to the set of environmental distributions.

C. Robustness Under the External Disturbance

In this experiment, we test the adaptability of all algorithms to actuator disturbance. The random disturbance is superimposed on the actions of the normal agent by weighting. The experimental results are shown in Fig. 5, and the RAEARL

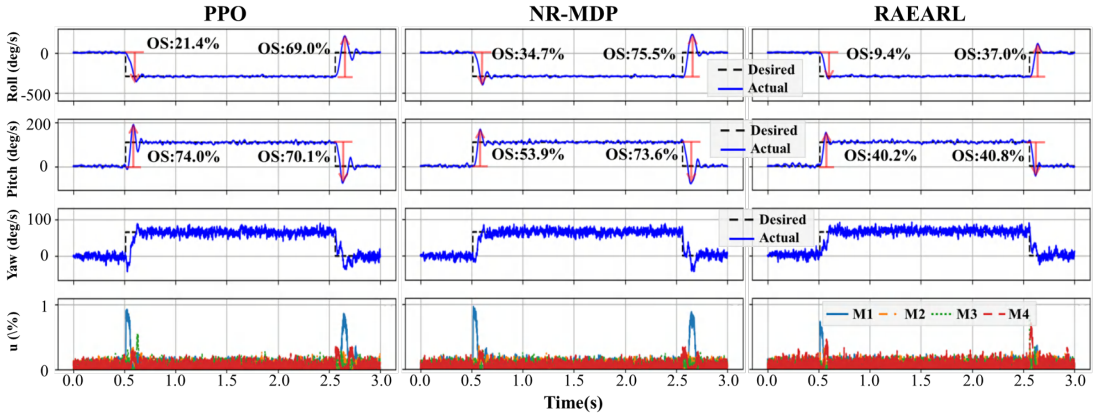

Fig. 6. Step responses and control signals of the three algorithms in the GymFc training environment. OS is short for overshoot, whereas blue line represents the actual angular velocity, and dashed black line represents the desired angular velocity.

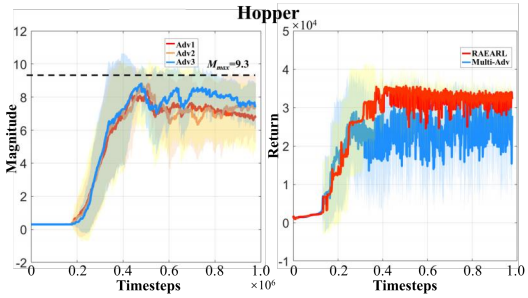

Fig. 7. The curves of adversary's output magnitude and accumulated reward in the Hopper environment.

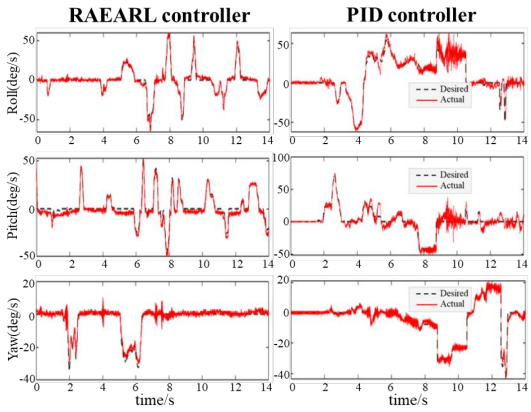

Fig. 8. Flight data of RAEARL and PID-based flight controller. Black curve shows the remote-control input, whereas the red curve shows the angular velocity measured by the IMU. The flight controller sampling frequency is 0.01s, whereas the sampling duration is 14s.

algorithm achieves the best anti-disturbance ability in all environments. Furthermore, although the Oracle algorithm is competitive in several environments with varying environmental parameters, its adaptability to other types of uncertainty is weakened due to the over-fitting environmental distribution. In the Hopper environment, as described in Section IV-B, the policies obtained by the Oracle algorithm adapt to lighter-mass environments, which tend to be sensitive to disturbance and even fail under small disturbances.

D. Influence of PID Controller

In this experiment, we recorded the adversary's output magnitude of the RAEARL algorithm and the normal agent's accumulated reward during training. Fig. 7 shows the relevant curves in the Hopper environment, curves for other environments can be found in the supplementary material. M_{max} is the upper limits of the output magnitude, represented by a black curve; Multi-Adv is a comparison algorithm that adopts the same adversarial framework as the RAEARL, but the adversarial agent output magnitude is set to a fixed M_{max}. It can be seen that in the RAEARL algorithm, each adversary has a different policy. The PID controller can dynamically adjust the output magnitude according to the adversary's policy. Compared with Multi-Adv, RAEARL obtains a higher accumulated reward and has lower volatility and variance. This demonstrates that the PID controller can improve the stability of adversarial training. From the perspective of curriculum learning, the PID controller can also be regarded as a curriculum generator, which dynamically adjusts the difficulty of curriculum training (the adversarial agent output magnitude) according to the policy performance of the normal agent.

E. Sim-to-Real Task

1) Simulation Environment: Fei et al. reported that the disturbance to the motor signal might significantly degrade the flight controller's performance and cause a severe failure [7]. In the test domain of the GymFc environment, we superimpose 10% random noise on the normal agent's action, which makes the disturbance act directly on the motors of the quadcopter. We compared the step responses of PPO, NR-MDP, and RAEARL; the results are presented in Fig 6. It can be observed that the RAEARL algorithm has the smallest overshoot and the fastest rise time, making it more suitable for deployment in real systems. In addition, we noticed that all three algorithms have oscillations in the yaw axis. The possible reason is that disturbances add a moment of inertia noise to each motor, which increases the complexity of controlling the yaw axis.

TABLE II
AVERAGE ABSOLUTE ERROR OF PITCH, ROLL, AND YAW AXES OF RAEARL AND PID-BASED ATTITUDE CONTROLLERS IN THE REAL WORLD. THE BEST RESULTS IN EACH CATEGORY ARE IN BOLD.

Controller	Pitch (deg/s)	Roll (deg/s)	Yaw (deg/s)
PID-based	3.2556	3.8652	2.9717
RAEARL	**1.9709**	**3.1981**	**1.2091**

2) Real Environment: In the last group of experiments, we directly deployed the RAEARL's policy trained in the GymFc environment on a real racing quadcopter without further tuning. Specifically, we make the quadcopter fly over the racing gates carrying different payload masses. For a small UAV, approximately 10-100 mW is required for 1 g additional take-off weight for hovering [24]. Even a small payload will affect the performance of the aircraft. In our experiments, we make the quadcopter carry 98.68g and 162.66g loads, equivalent to 17.9% and 29.7% of the total weight of the quadcopter. The flight trajectory is shown in Fig. 1. The quadcopter flies smoothly and quickly across the two racing gates while carrying a large mass load. This indicates that the RAEARL algorithm can effectively bridge the sim2real gap and has well generalization for serious parameter changes in real environments.

Finally, we compare the angular velocity error of the RAEARL and the PID-based flight controller of the open source flight controller firmware Betaflight on a real drone. The controller's tracking curve and average absolute error are shown in Fig. 8 and Table II, respectively. Compared with the PID-based flight controller, the RAEARL achieves smaller tracking errors in all three axes and can adapt to real drones that are mismatched with the simulated environment.

V. CONCLUSION

In this letter, we propose RAEARL, a novel multi-adversarial agent reinforcement learning framework. The critical idea of this framework is to adjust the strength of adversarial agents through a PID controller to stabilize the adversarial training process while aggregating the most diverse information to improve the policy's robustness. Furthermore, we decouple the training of the normal and adversarial agents through mirror agents to further improve the stability during training and make the adversarial framework suitable for distributed computing. Experiments on multiple tasks in simulation and sim2real transfer demonstrate the strong generalization ability of our approach.

Some interesting future research directions include exploiting multi-adversaries to attack different parts of the environment, such as attacking the robot body and actuator simultaneously or attacking different components of the RL algorithm, and the investigation of the adaptive adversarial strength controller in the above attacks.

REFERENCES

[1] Kai Zhu and Tao Zhang. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691, 2021.

[2] Ning Wang, Ying Gao, and Xuefeng Zhang. Data-driven performance-prescribed reinforcement learning control of an unmanned surface vehicle. *IEEE Transactions on Neural Networks and Learning Systems*, 32(12):5456–5467, 2021.

[3] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[4] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[5] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[6] Joanne Truong, Sonia Chernova, and Dhruv Batra. Bi-directional domain adaptation for sim2real transfer of embodied navigation agents. *IEEE Robotics and Automation Letters*, 6(2):2634–2641, 2021.

[7] Fan Fei, Zhan Tu, Dongyan Xu, and Xinyan Deng. Learn-to-recover retrofitting uavs with reinforcement learning-assisted flight control under cyber-physical attacks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7358–7364. IEEE, 2020.

[8] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[9] Peng Zhai, Jie Luo, Zhiyan Dong, Lihua Zhang, Shunli Wang, and Dingkang Yang. Robust adversarial reinforcement learning with dissipation inequation constraint. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5431–5439, Jun. 2022.

[10] Shen TieLong. *H_∞ Control theory and application*. Tsinghua University Press, 1996.

[11] Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

[12] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

[13] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[14] Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *International Conference on Learning Representations*, 2020.

[15] Marco A Wiering and Hado Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008.

[16] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[17] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. MGAN: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations*, 2018.

[18] Behnam Neyshabur, Srinadh Bhojanapalli, and Ayan Chakrabarti. Stabilizing GAN training with multiple random projections, 2018.

[19] Hiroaki Shioya, Yusuke Iwasawa, and Yutaka Matsuo. Extending robust adversarial reinforcement learning considering adaptation and diversity. *ICLR Workshop*, 2018.

[20] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

[21] Bin Hu and Laurent Lessard. Control interpretations for first-order optimization methods. In *2017 American Control Conference (ACC)*, pages 3114–3119. IEEE, 2017.

[22] Wangpeng An, Haoqian Wang, Qingyun Sun, Jun Xu, Qionghai Dai, and Lei Zhang. A pid controller approach for stochastic optimization of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8522–8531, 2018.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv1707.06347*, 2017.

[24] Leutenegger et al. Flying robots. In *Springer Handbook of Robotics*, pages 623–670. Springer, 2016.