2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# Cooperative-Competitive Team Play of Real-World Craft Robots

Rui Zhao[1*], Xihui Li[1,2*], Yizheng Zhang[1*], Yuzhen Liu[1*],
Zhong Zhang[1], Yufeng Zhang[1], Cheng Zhou[1], Zhengyou Zhang[1], Lei Han[1]

*Abstract*— **Multi-agent deep Reinforcement Learning (RL) has made significant progress in developing intelligent game-playing agents in recent years. However, the efficient training of collective robots using multi-agent RL and the transfer of learned policies to real-world applications remain open research questions. In this work, we first develop a comprehensive robotic system, including simulation, distributed learning framework, and physical robot components. We then propose and evaluate reinforcement learning techniques designed for efficient training of cooperative and competitive policies on this platform. To address the challenges of multi-agent sim-to-real transfer, we introduce Out of Distribution State Initialization (OODSI) to mitigate the impact of the sim-to-real gap. In the experiments, OODSI improves the Sim2Real performance by 20%. We demonstrate the effectiveness of our approach through experiments with a multi-robot car competitive game and a cooperative task in real-world settings.**

## I. INTRODUCTION

Deep Reinforcement Learning (RL) has demonstrated superhuman performance in multi-player games, such as Dota [1], StarCraft [2], [3], and Hide and Seek [4]. In robotics, RL has been applied to different kinds of robots. For single-robot systems with high-dimensional nonlinear dynamics, RL has advanced locomotion behaviors of real-world quadrupedal robots [5]–[15] and bipedal robots [16]–[19], empowered these robots to learn agile motions. For multi-robot scenarios, multi-agent RL has shown progress in learning cooperative robotic systems, including mobile robots [20]–[27], drones [28], [29], and robotic arms and hands [30], [31]. Multi-agent cooperation can greatly improve the efficiency of transportation [32], warehouse management [33], and construction [34], which further potentially promotes social production and development. Despite the agility behaviors learned by robots with deep RL, the ability to cooperate with others, i.e., collective intelligence, is of great importance for multi-agent robotic systems.

The cooperative effort of a group allows for greater achievements than any individual could achieve. In the natural world, ants evolve collective intelligence through millions of years. To maximize the interest of the species, ants divide labors and assign sub-tasks for different roles. Through teamwork, ants achieve great results, such as building complex underground nets [35].

Enabling robots to emergent similar swarm intelligence has been a long-standing challenge. As the complexity and the number of robots increases, manually building a multi-agent control system [33] becomes impractical. Unlike conventional methods that depend on manual design or in-depth knowledge of system dynamics, RL can potentially deliver more robust policies since it needs minimal domain expertise, only a reward signal. Recent advances in deep RL, such as self-play [36], could potentially transcend multi-agent intelligence from games into robotics.

In contrast to RL in game AI, training cooperative policies for robots poses different challenges. Video games are natural environments in which to train AI systems. There is no difference between the training environments and the testing environments. In robotics, collecting real-world data from physical robots is extremely expensive and time-consuming. To mitigate this problem, we need to build representative and efficient simulations to generate data in scale for training data-driven approaches. How to balance the complexity and the fidelity of the simulation is one of the challenges. Practical design choices are necessary for an efficient learning-based robotic system. It is impossible to build a simulated environment that is exactly the same as the real world. The Sim-to-Real (Sim2Real) gap greatly affects the performance of multi-agent reinforcement learning for robotics, which is a major challenge.

Previous works on cooperative-competitive multi-agent RL via self-play are mainly in the virtual world, such as StarCraft [37], MineCraft [38], and simulated football [39]. Examples of using RL for multi-robot cooperation and multi-team competition in the real world remain elusive. This is partly attributed to the increased complexity of interactions among agents and teams. This complexity challenges the reliance of modern RL algorithms on large amounts of real-world data. Recent works have bypassed real-world data collection by training policies in simulations via domain randomization [40]–[45]. However, domain randomization has scalability issues. As the task becomes more complex, its simulated environment must utilize more randomizations. For example, a single-agent task may only need to randomize properties associated with the agent, but if multiple agents are introduced, the relative properties among agents need to be randomized as well. Parallel to the scalability challenges and the hyperparameter tunning of the randomizations, if too many randomizations are used, the learned policy might be too conservative [46]. With these conservative policies, the agents might be unable to properly solve the original task. Compared to single-robot systems, multi-robot systems pose more challenges in efficient learning and sim-to-real transfer, as the dimensionality and complexity increase. More practical and efficient sim-to-real techniques are needed for

*Equal contribution. [1]The authors are with Tencent Robotics X Laboratory, Shenzhen, Guangdong, China. [2]The author is with Tsinghua University, Shenzhen, Guangdong, China. Corresponding to rui.zhao.ml@gmail.com, leihan.cs@gmail.com

the multi-agent settings.

We propose to utilize guided RL [47] to increase learning efficiency and develop a novel sim-to-real method named Out of Distribution State Initialization (OODSI) to mitigate the discrepancy of multi-agent dynamics between simulation and the real world. To test the cooperative-competitive policy and its resilience against the sim-to-real gap, we build a multi-team game environment with craft robots in the real world. The craft-robots are essentially mobile robots with the ability to move and build constructions with different objects, such as blocks and slopes. The robots can move and unfold the slopes to build different constructions and change the layout of the landscape. We also build the corresponding simulations, the first simulation is based on pyBullet, which has discrete observation and action spaces and is fast to run rollouts for training. The second simulation is based on Gazebo, which has continuous signals and is relatively slow but more realistic to the physical robot scenarios. We use the pyBullet-based simulation for training. We use the Gazebo-based simulation for testing before deploying the policies on the real robots.

In this robotic platform, we design two games. The first one is a two-team competition game. In this competition, there are limited amount of resources, including the block and slopes. The goal for each team is to build a second-floor structure. The resources is only enough for one team to finish the building. Therefore the two teams need to compete with each other to secure objects and build fast. During learning, we observe the evolution of attack and defense strategies. The second game is a simple word-building task, which is used for more comprehensive ablation tests and in-depth analysis of the algorithms.

In this paper, we build a complete robotic AI system for multi-agent mobile robots, including the simulations, learning framework, and physical robots. We show that design choices are important to make the training process more efficient. We find that guided reinforcement learning [47] can speed up learning and propose a technique to inject human guidance into reinforcement learning. After training the policies in simulation, we deploy the policies directly on the real robots, and have relative low success rate. We propose the OODSI method to train robust policies to tackle the sim-to-real gap. The main contribution of this paper is two-fold: first, we demonstrate how to use multi-agent RL to train real-world mobile robots to develop competitive and cooperative team strategies through self-play. Secondly, due to the Sim2Real gap of asynchronous actions of multi-robots, the learned cooperation strategies are infeasible during deployment. We propose OODSI method to combat the Sim2Real gap in the multi-agent RL settings.

## II. Robot Craft Arena

In the video game MineCraft, players can build various customized constructions using the unit blocks. Inspired by this, we build a real-world robot craft arena, which is essentially a robotic construction platform [48], [49]. In this arena, the robot can interact with different objects and build constructions. The goal of the agents is to build a target construction as a team, or as two teams to compete with each other. The robot craft arena consists of three elements, i.e., the craft robots, blocks, and slopes. The craft-robots are mobile manipulation vehicles, which can transport the blocks and slopes. The robot utilizes a front camera and ApirlTags for localization. The block is a hollow cube. The craft robots can move underneath the block, lift it, move it, and place it at any accessible location in the arena. The slope is a folding ramp block. With the slope as the building material, the craft robots can move to higher floors of the construction.

Based on the real-world specifications, we build a simulation using pyBullet for training. In order to train the RL agents efficiently, we discretize the observation space and the action space. The agent perceives a list of attributes as observation, which describes the position, orientation, and categorical information of all the objects in the arena. After processing the observation, the agent can choose one out of eleven actions, including moving forward, back, left, or right, turning left or right, lifting the object, i.e., block or slope, dropping the object, folding or unfolding the slope, or stop i.e., no-operation.

### A. Real-World Robot Constrains

In the simulation, we consider the real-world constraints of the robots. For example, when the robot wants to move into the slope and move it, it needs to first adjust its orientation to make the camera face outwards and then roll back into the slope. This is to avoid losing track of AprilTags, as the slope has an opening only on one side. Similarly, in the case of a block, when the robot is underneath it and wants to move out, it needs to first turn to face forward and then move out of the block. It cannot move directly left or right to exit the block. There are other similar constraints. To avoid actions that do not comply with constraints, we utilize action masks to block unfeasible actions and replace them with stop actions. However, with action masks, we can avoid unfeasible and dangerous actions. However, these real-world constraints make the task more difficult to solve. For instance, to find the solution to entering the slope, the robot needs to first explore the orientation of its rear part facing the entrance of the slope and then move back in. Given an initial random policy, the product probability of this consecutive action sequence is relatively low, which means it is difficult for the agent to explore the solution. These real-world robot constraints make the training process challenging.

### B. The Multi-Agent Sim-to-Real Gap

Multi-robot cooperation and competition pose unique sim-to-sim and sim-to-real challenges. In the training simulation, the observation and action space are discrete. For each time step, given the current state, multiple agents choose valid actions from the action space. The simulation takes in all the actions and produces the next state all at once. The action execution of multiple agents in simulation is synchronized. However, in reality, in different situations, agents' actions take different amounts of time. For example, if the robot

**Algorithm 1** Out of Distribution State Initialization (OODSI)

1: Train policy $\pi_\theta$ in the training simulation (pyBullet)
2: Use $\pi_\theta$ to interact with the more realistic simulation (Gazebo) or the real-world environment to collect trajectories $\mathcal{T}$, which contains OOD states $s^{OOD}$
3: Sample initial states $s_0 \sim \mathcal{T}$ for training
4: Retrain policy $\pi_\theta$ in the training simulation

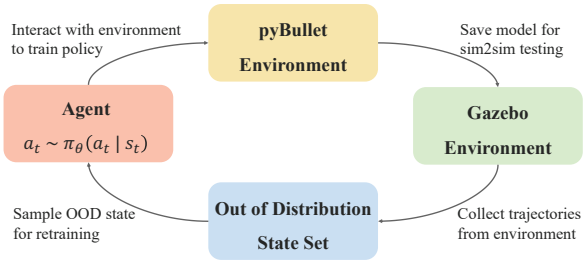


Fig. 1. **Out of Distribution State Initialization**

car starts from standing still and moves forward by one block distance, it might take one second. If the robot car is moving forward and keeps moving forward by one block, it takes less than one second. Compared to consecutively moving forward by a number of blocks, the action sequence of the same length consisting of moving forward and turning takes a longer time. As a consequence, in the real world, for a time step, some of the agents finished the commanded actions and others could not. For the robots that have not finished the actions, they will first finish the current action and then take in a new action. This causes a unique sim-to-real gap in the setting of multi-robot cooperation and competition. The dynamics have changed. For the same starting state, after executing the same agents' actions, the agents might perceive different next states due to action delays and the asynchronization of multi-agent actions. In reality, the action execution process is asynchronous, while in the training simulation, it is synchronous. This sim-to-real gap makes the deployment of simulation-trained multi-agent policy difficult to transfer to the real world. To reproduce this sim-to-real gap in simulation for quick testing and evaluation, we build a Gazebo-based simulation, which has continuous action space as in the real world. The Gazebo-based simulation also utilizes the same asynchronous action execution process as the real-world robots. Although the Gazebo-based simulation is more realistic it is significantly slower compared to the discrete pyBullet-based simulation, which makes the Gazebo-based simulation inadequate for training. We only use the Gazebo-based simulation to test the sim-to-real gap before deploying the policies on the real-world robots. For simplicity, we refer to the sim-to-sim gap between the pyBullet-based simulation and the Gazebo-based simulation as the sim-to-real gap in the rest of the paper.

## III. Preliminaries

A Markov decision process (MDP) is represented by a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, T)$, where $\mathcal{S}$ is a state set, $\mathcal{A}$ an action set, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is a transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_+$ is a start state distribution, and $T$ is the horizon. For multi-agent settings, the transition probability distribution and the reward function are depend on the actions from all agents, i.e., $\mathcal{P} : \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2 \ ... \times \mathcal{A}^N \times \mathcal{S} \to \mathbb{R}_+$ and $r : \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2 \ ... \times \mathcal{A}^N \to \mathbb{R}$, where $i \in \{1, 2, ..., N\}$ is the index for the $i$-th agent. For the multi-team settings, the definition of the reward function implies the game is cooperative or competitive. For example, in a two-team zero-sum game, if we define $r^1 + r^2 = 0$, then the game is competitive, as one team wins and the other loses. The goal of RL is to learn a policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ parametrized by $\theta$ that maximizes the expected accumulated reward $R(\pi, s_0) := \mathbb{E}_{\tau|s_0}[\sum_{t=0}^{T} r(s_t, a_t)]$, i.e., return, $\eta_{\rho_0}(\pi_\theta) = \mathbb{E}_{s_0 \sim \rho_0} R(\pi, s_0)$, starting from a $s_0 \sim \rho_0$, where $\tau = (s_0, a_0, , \ldots, a_{T-1}, s_T)$ denotes a complete trajectory, with $a_t \sim \pi_\theta(a_t|s_t)$, and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$. Policy optimization methods, such as PPO [50], iteratively collect trajectories and then use them to improve the current policy [51]. In practice, action masking is utilized as a common technique to avoid generating invalid actions in discrete action spaces [1], [3], [52]–[54].

### A. Guided RL

In real-world robotics applications, RL still faces challenges. Guiding the learning process with additional knowledge offers potential solutions, which leverage the data-driven approaches and knowledge-based approaches. The goal of guided RL is to accelerate learning efficiency and improve the success rate for real-world robotics deployment via the integration of additional knowledge into the learning process [47]. Recent advances in RL have testified the advantages of utilizing prior knowledge, such as natural languages [55], [56] and programs [57], [58] to accelerate policy learning.

### B. Modifying Start State Distribution in RL

In the literature of RL, researchers directly modify the start state distribution $\rho_0$ to accelerate learning in an MDP. Popov et al. [59] propose to modify the start state distribution $\rho_0$ to be uniform among states visited by expert demonstrations to speed up learning. Florensa et al. [60] utilize a set of start states increasingly far from the goal to gradually train robots to conduct navigation and manipulation tasks in simulation. In our work, we propose a method based on modifying the start state distribution $\rho_0$ to make the policies more robust against the sim-to-real gap.

## IV. METHODOLOGY

In this section, we propose two innovations to tackle the aforementioned two challenges, respectively, i.e., efficient learning under real-world robot constraints and the multi-agent sim-to-real gap. The first innovation is guided RL via action masking, which can be considered a practical technique. The second one is a method named Out of Distribution State Initialization (OODSI) for sim-to-real transfer.

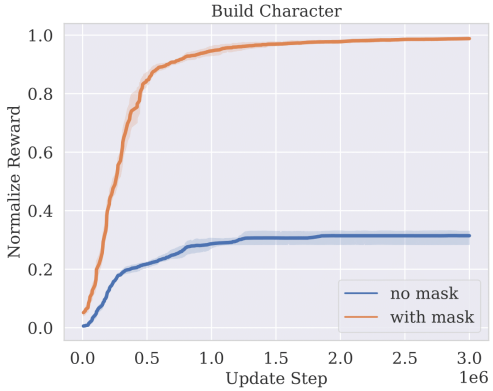


Fig. 2. Training curves w/o action masking

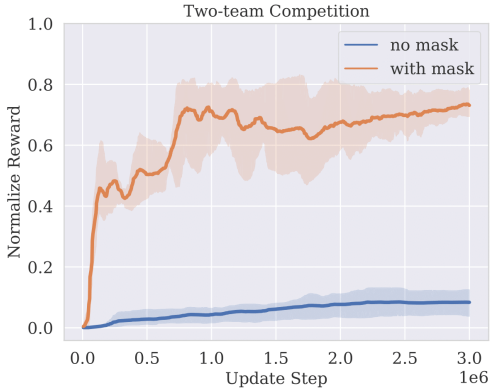


Fig. 3. Training curves w/o action masking

### A. Guided RL via Action Masking

In the robot craft arena, there are real-world constraints, see Section II-A, such as the specific orientation for the robot to move into the slope or move out of the block. We implemented rule-based sub-routine planners through the action masks to guide the robot to efficiently explore the environment, such as moving into the slope, etc. To introduce the guidance, given a state, along with the set of invalid actions, we further block the actions that are not beneficial for the downstream task. This reduces the exploration space for the agents. The implemented rule-based sub-routine planners can also be seen as parts of the environment. Similar to the game environments, the agent does not need to consider low-level details, when performing actions, such as picking up a box in MineCraft or walking toward the enemies in StarCraft. This design choice simplifies the task for the RL agents. The RL agent can focus on learning high-level strategies. The combination of rule-based sub-routine local planners and the data-driven RL policies enable the agents to efficiently learn complex real-world robotics tasks.

### B. Out of Distribution State Initialization

We propose Out of Distribution State Initialization (OODSI) to tackle the multi-agent sim-to-real gap. As mentioned in Section II-B, in the training environment, i.e., the pyBullet-based environment, the action execution of multiple agents is synchronous and the actions are discrete. Whereas in the testing environment, i.e., the Gazebo-based environment and the real-world robot craft arena, the action execution is asynchronous and continuous. This discrepancy between the training and the testing environment leads to the sim-to-real gap. Under the sim-to-real gap, the dynamics are different between the training environment and the testing environment. To be more specific, under the same current state, $s_t$, executing the same multi-agent actions, $a_1, a_2, \ ...a_n$, the next states in the training environment, $s_{t+1}^{train}$, could be different as the next states in the testing environments, $s_{t+1}^{test}$. With respect to the trained agents, these unseen states from the testing environments are Out of Distribution (OOD) states $s^{OOD}$. We propose to add these OOD states $s^{OOD}$ collected from the testing environments to the start state distribution, $\rho_0$, in the training process to train the agent to learn robust policies against OOD states. After training with the OOD states, when encountering the OOD states, the agents would still know how to handle the situation and complete the task. Our approach can be understood as sequentially composing locally stabilizing policies by growing trajectories from an OOD state to the next OOD state, of which the idea bears similarity with the work of Tedrake et al. [61] and the work of Burridge et al. [62]. We summarize OODSI method in Algorithm 1 and Figure 1.

### C. States, Actions and Rewards

We construct the observation of the agent's current position and the positions of other agents. The actions of the craft robots include moving in horizontal and vertical directions and interacting with different objects in the environment. The craft robot is designed to possess the ability to interact with objects in the environment in various ways, such as lifting, moving, and unloading a block or a slope, and folding and unfolding a slope. In the craft arena, we define two tasks. The first one is a cooperative building characters task. The second one is a competitive building two-floor structure task between two teams. For the cooperative task, we first define a target building, which consists of multiple blocks and slopes. When the agents build one more block right, the agents receive a positive reward $r_{build}$. After the agents successfully complete the building, the agents receive a completion reward $r_{completion}$, which is designed to be larger than the building reward $r_{build}$ to encourage the agents to finish the whole building instead of leaving several pieces behind. In the case of the two-team building competition, the resources are limited. There is a block missing, hence only one of the two teams can finish their building and win the game receiving $r_{completion}$. And the other team loses in the game. The two-team competition is a zero-sum game.

### D. Learning Framework

We train the agents to cooperate using the Centralized Training and Decentralized Execution (CTDE) framework [63], [64]. During training, we have a centralized value function, which processes the input information collected by

TABLE I

SUCCESS RATE IN PYBULLET AND GAZEBO

| Method | Build Character | | Two-team Competition | |
|---|---|---|---|---|
| | pyBullet | Gazebo | pyBullet | Gazebo |
| PPO | $96.88 \pm 0.67\%$ | $46.67 \pm 20.5\%$ | $72.75 \pm 3.99\%$ | $43.33 \pm 17.00\%$ |
| PPO+DR | $89.96 \pm 7.89\%$ | $50 \pm 14.14\%$ | $73.29 \pm 2.32\%$ | $46.67 \pm 4.71\%$ |
| PPO+OODSI | $98.46 \pm 0.21\%$ | $66.67 \pm 12.47\%$ | $75.67 \pm 8.07\%$ | $53.33\% \pm 9.43\%$ |
| PPO+DR+OODSI | $99.41 \pm 0.36\%$ | **76.67** $\pm 4.71\%$ | $72.08 \pm 5.69\%$ | **66.67** $\pm 12.47\%$ |

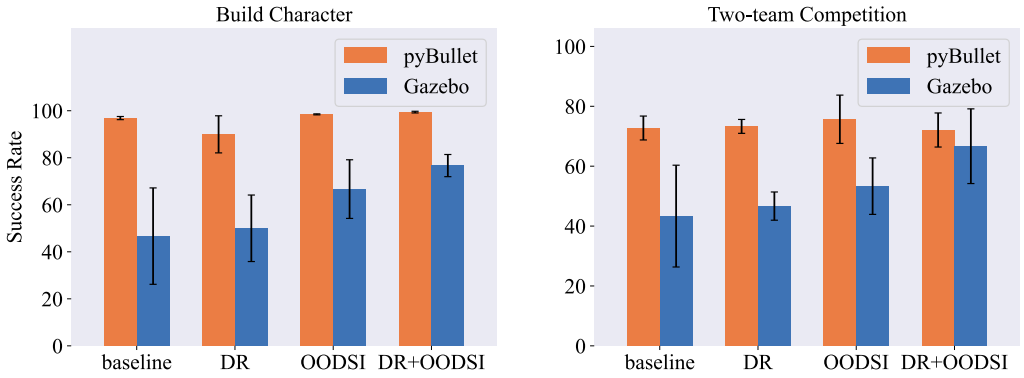


Fig. 4. Sim2Real Performance Comparison

Fig. 5. Blocking behavior in the pyBullet environment

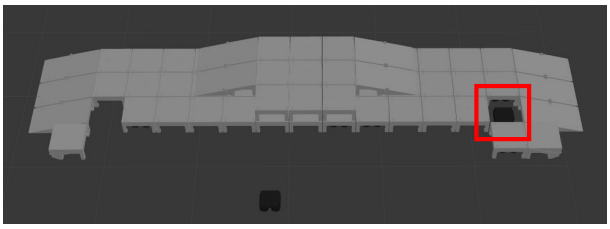

Fig. 6. Blocking behavior in the Gazebo environment

all the agents and estimates the value i.e., estimated return, and assigns the credits to each agent in the team. During execution, each agent only needs to complete its own subtasks. The agent's policy is represented by a multi-layer perceptron (MLP) network. To improve learning efficiency, the agents in the same team share the weights of the policy networks. We know that RL requires a large amount training samples to work well. To efficiently collect training samples, we adapt the large-scale distributed training framework in game AI [3], [54] and run the training process using Kubernetes. For distributed training, we have two kinds of pods, i.e., the learner pods and the actor pods. The actor pods receive the policy weights from the learner pods and run rollouts in parallel to collect training samples. After collecting the samples, the actor pods send these samples to the learner pods. The leaner pods calculate the gradient using these samples and update the weights using Proximal Policy Optimization (PPO) [50], [65]. To avoid gradient explosion and stabilize the training process, we clip the gradient norm [66], [67]. To train the two teams to compete with each other, after one team reaches a certain success rate threshold, we pause training for the leading team, fix its policies, and only train the other team. For the learning team, the paused team can be seen as a part of the environment. Since the policy of the paused team is fixed, the environment is stationary, which is suitable for learning. If the environment is non-stationary, it is very difficult to train the agents to find optimal policies. When the weaker team finds a counter policy and its success rate is higher than the threshold, we will switch the role for pause. This enables the two teams to stably evolve their policies during self-play.

## V. EXPERIMENTAL RESULTS

We evaluate the proposed method in the robot craft arena, as introduced in Section II. The robot craft arena consists of robot cars, blocks, and slopes. The robot cars can lift and move the blocks and slopes. Additionally, the robot car can also fold and unfold the slopes. We design two different kinds of tasks, including a collaboration task and a

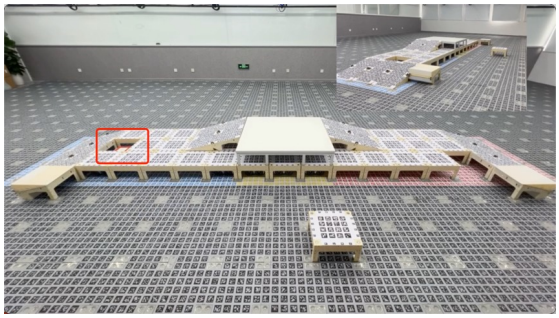

Fig. 7. Blocking behavior in the real world

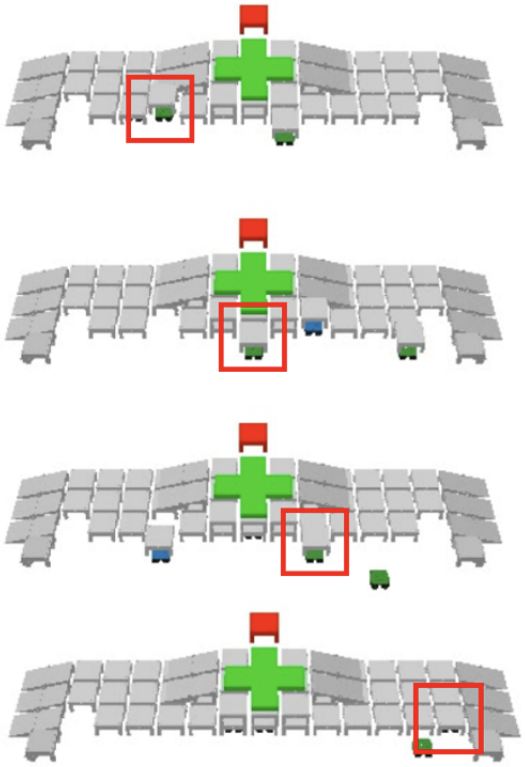

Fig. 8. Stealing behaviour

competition task. For the collaboration task, the robot cars cooperate with each other to build characters "TPX". For the competition task, the robot cars are divided into two teams. These two teams compete with each other to build two-floor structures with limited resources. We use pyBullet simulation for training and test the performance in the more realistic Gazebo simulation. To improve the training efficiency, we employ a distributed reinforcement learning training framework named TLeague [54]. In the experiments, we first conduct ablation studies on action masking. Secondly, we verify the effectiveness of the proposed OODSI method for Sim2Sim or Sim2Real transfer, compared to baseline methods, including PPO [50] and Dynamics Randomization (DR) [43].

### A. Action Masking Ablation Study

To verify the effectiveness of guided RL via action masking, introduced in Section IV-A, on improving training efficiency, we compare the performance of baseline method PPO with and without action masking in two different tasks. The agents in both of the tasks are trained for $3 \times 10^6$ steps. Figure 2 and Figure 3 show the learning curve of the normalized reward during training. The shadow area in the figures represents the variance. We can observe that guided RL via action masking can greatly improve the convergence speed and reduce the training time.

### B. Sim2Real Performance Comparison

In this section, we conduct experiments to compare the Sim2Real performance of different methods in two different tasks, including collaboration and competition. We compared the success rates of four methods: PPO, PPO+DR, PPO+OODSI, and PPO+DR+OODSI in pyBullet and Gazebo environments. For all methods, we only train the models in the pyBullet environments and evaluate the Sim2Real performance in the Gazebo environments. For DR, we randomly replace the actions using the stop action with 30% probability to simulate action executing failure to improve robustness. For OODSI method, we use the models trained for $3 \times 10^6$ steps to execute several episodes in the Gazebo environment to collect trajectories. Then, we sample 3 trajectories and divide them equally into 5 segments, collecting the first state of each segment to construct the initial state set for training. For each method, we selected models trained with 3 different seeds and tested them in the Gazebo environment for 10 episodes each. In the end, we calculated the success rate of each method over 30 episodes. The detailed results are provided in the table I. In Table I under Gazebo, we can see that, in both tasks, DR performs better than the baseline PPO. OODSI outperforms DR. The best performer is the combination of DR and OODSI. We also represent the experimental results in bar charts, see Figure 4. Figure 4 left part shows the success rate of the building character task. DR slightly improves the success rate in the Gazebo environment. OODSI method brings higher performance improvements. DR+OODSI significantly increases the success rate, which is $30\%$ higher compared to the baseline method. Figure 4 right part shows the success rate of the two-team competition task. In this task, the success rate is the sum of the winning rates of both teams. Similar to the results in the building character task, the DR and OODSI methods improve the Sim2Real performance. DR+OODSI still brings a significant increase, i.e., $23.34\%$, in the success rate. We can see that OODSI helps the agents learn robust multi-agent cooperative and competitive policies against the Sim2Real gap.

### C. Qualitative Results

In this section, we present the learned high-level strategies that we observed from the agent's policy. In the two-floor competition task, the robot car learns the blocking behavior. The robot car of the green team, which is highlighted with a red rectangle in Figure 5, invades the blue team's territory

and raises a block to prevent the left team from completing the task. Similarly, in both the Gazebo environment, see Figure 6, and the real-world robot craft arena, see Figure 7, we also observe the blocking strategy. The robot car learns to go to the opponent's field and assist their own team to win the game by blocking the opponent team from completing the task. Another interesting strategy the agent learns is 'stealing' the block, as shown in Figure 8. Figure 8 shows that a robot car of the green team 'steals' a block from the blue team's structure brings it to its own team and completes the building task. The complete real-world process of building the character and the two-team competition tasks are shown in the video: `https://youtu.be/G_oLeJfXIyU`.

## VI. CONCLUSIONS

In this work, to push the boundary of Multi-Agent Reinforcement Learning (MARL) in the real world. We build a robot craft arena and propose collaborative and competitive tasks to train and evaluate the agent policies in different simulations and on the physical robots. We conduct ablation studies to show the effectiveness of guided RL via action masking in real-world robotic tasks. To tackle the multi-agent Sim2Real challenge, we propose Out of Distribution State Initialization (OODSI) method and show its superior performance in combating the Sim2Real gap, compared to baseline methods. We conduct multiple experiments in simulations and in the real world, demonstrating that the proposed method exhibits favorable Sim2Real performance and sheds light on practical applications of MARL in robotics.

## APPENDIX

We summarize the hyperparameters that we used in the experiments in Table II.

TABLE II
HYPERPARAMETERS

| Parameter | Value |
|---|---|
| Number of Layers | 4 |
| Number of Units | 256 |
| Activation Function | ReLU |
| Learning Rate | 0.00001 |
| batch size | 4096 |
| Reward Discount | 0.95 |
| Optimizer | Adam |
| GAE $\lambda$ | 0.95 |
| Replay Buffer Size | 256000 |
| Episode max step | 500 |

## REFERENCES


[1] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[2] 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.

[3] L. Han, J. Xiong, P. Sun, X. Sun, M. Fang, Q. Guo, Q. Chen, T. Shi, H. Yu, X. Wu *et al.*, "Tstarbot-x: An open-sourced and comprehensive study for efficient league training in starcraft ii full game," *arXiv preprint arXiv:2011.13729*, 2020.

[4] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," *arXiv preprint arXiv:1909.07528*, 2019.

[5] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[6] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[7] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[8] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Robotics: Science and Systems*, 2022.

[9] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.

[10] G. Feng, H. Zhang, Z. Li, X. B. Peng, B. Basireddy, L. Yue, Z. Song, L. Yang, Y. Liu, K. Sreenath *et al.*, "Genloco: Generalized locomotion controllers for quadrupedal robots," in *Conference on Robot Learning*. PMLR, 2023, pp. 1893–1903.

[11] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning*. PMLR, 2023, pp. 138–149.

[12] Y. Li, J. Li, W. Fu, and Y. Wu, "Learning agile bipedal motions on a quadrupedal robot," *arXiv preprint arXiv:2311.05818*, 2023.

[13] L. Han, Q. Zhu, J. Sheng, C. Zhang, T. Li, Y. Zhang, H. Zhang, Y. Liu, C. Zhou, R. Zhao *et al.*, "Lifelike agility and play on quadrupedal robots using reinforcement learning and generative pre-trained models," *arXiv preprint arXiv:2308.15143*, 2023.

[14] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *arXiv preprint arXiv:2306.14874*, 2023.

[15] C. Zhang, J. Sheng, T. Li, H. Zhang, C. Zhou, Q. Zhu, R. Zhao, Y. Zhang, and L. Han, "Learning highly dynamic behaviors for quadrupedal robots," in *International Conference on Robotics and Automation*, 2024.

[16] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik, "Adapting rapid motor adaptation for bipedal robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1161–1168.

[17] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Learning humanoid locomotion with transformers," *arXiv preprint arXiv:2303.03381*, 2023.

[18] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner *et al.*, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *arXiv preprint arXiv:2304.13653*, 2023.

[19] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *arXiv preprint arXiv:2401.16889*, 2024.

[20] C. F. Touzet, "Distributed lazy q-learning for cooperative mobile robots," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 1, 2004.

[21] E. Tuci, M. H. Alkilabi, and O. Akanyeti, "Cooperative object transport in multi-robot systems: A review of the state-of-the-art," *Frontiers in Robotics and AI*, vol. 5, p. 59, 2018.

[22] A. Goldhoorn, A. Garrell, R. Alquézar, and A. Sanfeliu, "Searching and tracking people with cooperative mobile robots," *Autonomous Robots*, vol. 42, no. 4, pp. 739–759, 2018.

[23] R. Mitchell, J. Fletcher, J. Panerati, and A. Prorok, "Multi-vehicle mixed-reality reinforcement learning for autonomous multi-lane driving," *arXiv preprint arXiv:1911.11699*, 2019.

[24] J. J. Koh, G. Ding, C. Heckman, L. Chen, and A. Roncone, "Cooperative control of mobile robots with stackelberg learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7985–7992.

[25] E. Candela, L. Parada, L. Marques, T.-A. Georgescu, Y. Demiris, and P. Angeloudis, "Transferring multi-agent reinforcement learning policies for autonomous driving using sim-to-real," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8814–8820.

[26] R. J. Torbati, S. Lohiya, S. Singh, M. S. Nigam, and H. Ravichandar, "Marbler: An open platform for standardized evaluation of multi-robot reinforcement learning algorithms," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2023, pp. 57–63.

[27] P. Werner, T. Seyde, P. Drews, T. M. Balch, I. Gilitschenski, W. Schwarting, G. Rosman, S. Karaman, and D. Rus, "Dynamic multi-team racing: Competitive driving on 1/10-th scale vehicles via learning in simulation," in *7th Annual Conference on Robot Learning*, 2023.

[28] H. Shi, G. Liu, K. Zhang, Z. Zhou, and J. Wang, "Marl sim2real transfer: Merging physical reality with digital virtuality in metaverse," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 4, pp. 2107–2117, 2022.

[29] M. Yang, G. Liu, Z. Zhou, and J. Wang, "Partially observable mean field multi-agent reinforcement learning based on graph attention network for uav swarms," *Drones*, vol. 7, no. 7, p. 476, 2023.

[30] S. Beaussant, S. Lengagne, B. Thuilot, and O. Stasse, "Delay aware universal notice network: real world multi-robot transfer learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1251–1258.

[31] B. Huang, Y. Chen, T. Wang, Y. Qin, Y. Yang, N. Atanasov, and X. Wang, "Dynamic handover: Throw and catch with bimanual hands," *arXiv preprint arXiv:2309.05655*, 2023.

[32] C. Park, G. S. Kim, S. Park, S. Jung, and J. Kim, "Multi-agent reinforcement learning for cooperative air transportation services in city-wide autonomous urban air mobility," *IEEE Transactions on Intelligent Vehicles*, 2023.

[33] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[34] Z. Ren and C. J. Anumba, "Multi-agent systems in construction–state of the art and prospects," *Automation in Construction*, vol. 13, no. 3, pp. 421–434, 2004.

[35] R. D. Alexander, "The evolution of social behavior," *Annual review of ecology and systematics*, vol. 5, no. 1, pp. 325–383, 1974.

[36] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[37] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043*, 2019.

[38] D. Perez-Liebana, K. Hofmann, S. P. Mohanty, N. Kuno, A. Kramer, S. Devlin, R. D. Gaina, and D. Ionita, "The multi-agent reinforcement learning in malm\" o (marl\" o) competition," *arXiv preprint arXiv:1901.08129*, 2019.

[39] S. Liu, G. Lever, Z. Wang, J. Merel, S. A. Eslami, D. Hennes, W. M. Czarnecki, Y. Tassa, S. Omidshafiei, A. Abdolmaleki *et al.*, "From motor control to team play in simulated humanoid football," *Science Robotics*, vol. 7, no. 69, p. eabo0235, 2022.

[40] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[41] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. 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)*. IEEE, 2017, pp. 23–30.

[42] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 734–743.

[43] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[44] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[45] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[46] F. Ramos, R. C. Possas, and D. Fox, "Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators," *arXiv preprint arXiv:1906.01728*, 2019.

[47] J. Eßer, N. Bach, C. Jestel, O. Urbann, and S. Kerner, "Guided reinforcement learning: A review and evaluation for efficient and effective real-world robotics [survey]," *IEEE Robotics & Automation Magazine*, vol. 30, no. 2, pp. 67–85, 2022.

[48] Q. Xu, Y. Zhang, S. Zhang, R. Zhao, Z. Wu, D. Zhang, C. Zhou, X. Li, J. Chen, Z. Zhao *et al.*, "Reccraft system: Towards reliable and efficient collective robotic construction," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8979–8986.

[49] R. Zhao, X. Liu, Y. Zhang, M. Li, C. Zhou, S. Li, and L. Han, "Craftenv: A flexible collective robotic construction environment for multi-agent reinforcement learning," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023, pp. 1164–1172.

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

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

[52] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser *et al.*, "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[53] D. Ye, Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo *et al.*, "Mastering complex control in moba games with deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6672–6679.

[54] P. Sun, J. Xiong, L. Han, X. Sun, S. Li, J. Xu, M. Fang, and Z. Zhang, "Tleague: A framework for competitive self-play based distributed multi-agent reinforcement learning," *arXiv preprint arXiv:2011.12895*, 2020.

[55] R. Kaplan, C. Sauer, and A. Sosa, "Beating atari with natural language guided reinforcement learning," *arXiv preprint arXiv:1704.05539*, 2017.

[56] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[57] S.-H. Sun, T.-L. Wu, and J. J. Lim, "Program guided agent," in *International Conference on Learning Representations*, 2019.

[58] C. Chang, N. Mu, J. Wu, L. Pan, and H. Xu, "E-mapp: Efficient multi-agent reinforcement learning with parallel program guidance," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 154–12 168, 2022.

[59] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *arXiv preprint arXiv:1704.03073*, 2017.

[60] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," in *Conference on robot learning*. PMLR, 2017, pp. 482–495.

[61] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[62] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[63] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[64] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[65] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[66] T. Mikolov *et al.*, "Statistical language models based on neural networks," *Presentation at Google, Mountain View, 2nd April*, vol. 80, no. 26, 2012.

[67] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. Pmlr, 2013, pp. 1310–1318.