MACT: Multi-agent Collision Avoidance with Continuous Transition Reinforcement Learning via Mixup*

Pu Feng[1], Xin Yu[1], Junkang Liang[1], Wenjun Wu[2(✉)], and Yongkai Tian[1]

[1] School of Computer Science and Engineering, Beihang University, Beijing 100083, China
[2] Institute of Artificial Intelligence, Beihang University, Beijing 100083, China
{wwj09315}@buaa.edu.cn

Abstract. Autonomous collision avoidance is a critical component in various multi-robot applications. While deep reinforcement learning (RL) has demonstrated success in some robotic control tasks, it remains challenging to apply it to real-world multi-agent collision tasks due to poor sample efficiency. The limited amount of transition data available and its strong correlation with multi-agent task characteristics have restricted recent research in this area. In this paper, we propose Multi-agent Collision Avoidance with Continuous Transition Reinforcement Learning via Mixup (MACT) to address these challenges. MACT generates new continuous transitions for training by linearly interpolating consecutive transitions. To ensure the authenticity of constructed transitions, we develop a discriminator that automatically adjusts the mixup parameters. Our proposed approach is evaluated through simulations and real-world swarm robots consisting of E-pucks, demonstrating its practical application. Our learned policies outperform existing collision avoidance methods in terms of safety and efficiency.

Keywords: Multi-agent Collision Avoidance · Reinforcement Learning · Continuous Transition. · Data Augmentation

1 Introduction

With the increasing use of unmanned vehicles, mobile robots are becoming more common for a variety of tasks, including rescue operations, final-mile delivery, and clustering tasks. However, navigating swarm robots safely and efficiently in complex and ever-changing real-world environments presents significant coordination challenges. Unlike single robots, swarm robots must take into account the movements of other robots in the cluster. To address these challenges, researchers have extensively studied distributed control, which can be categorized into two main approaches. The first approach is the sensor-based approach, which

* Supported by National Key R&D Program of China 2022ZD0116401.

includes methods such as Artificial Potential Field (APF) [7] and Optimal Reciprocal Collision Avoidance (ORCA)[1]. These methods generate policies based on local observations, but they may fall into local optimum points and have low efficiency. The second approach is the distributed control method based on end-to-end reinforcement learning. This approach has shown promise in dealing with the deadlock problem and achieving better task completion efficiency. Various researchers have employed reinforcement learning for collision avoidance [3, 2], with these methods typically taking the robot's own position, velocity information, and information from other agents in the surrounding domain as input to the neural network. However, one of the main challenges with reinforcement learning is its low sample efficiency.

Efforts have been made to improve sample efficiency in reinforcement learning (RL) for multi-agent collision avoidance. One method is to use inverse reinforcement learning to model the environment, which more fully utilizes environmental information [14]. Another classic approach for improving sample efficiency is off-policy RL, which includes methods such as Deep Deterministic Policy Gradient (DDPG) [8], Twin Delayed Deep Deterministic Policy Gradient (TD3) [4], and Soft Actor-Critic (SAC) [5]. These methods collect samples during the training process and store them in a replay buffer. During policy updates, batch-sized data (s, a, r, s', d) are randomly selected from the buffer. Compared to on-policy methods, off-policy methods reuse collected data and have a higher sample utilization rate. Furthermore, selecting effective data from the replay buffer is also an important research point to further improve sample efficiency [6]. Prioritized sampling methods, which select samples with greater differences for policy learning, have achieved some improvement. However, these methods analyze the discrete transitions collected in training for continuous action tasks. Dividing a segment into several discrete transition pairs still has limited sample efficiency improvement, as the number of transitions is often small. Transforming the trajectories of a group of agents in obstacle avoidance navigation into continuous transitions may lead to better performance.

In recent years, data augmentation techniques have achieved tremendous success in augmenting data and improving sample efficiency. Various data augmentation methods have been extensively researched and demonstrated to be effective in computer vision applications. Among them, mixup has gained increasing attention in recent years. Mixup was first introduced as a way to generate new training examples by linearly interpolating pairs of randomly selected samples and their labels [15]. The resulting mixed samples and labels are then used to train a deep neural network. Since its introduction, mixup has been shown to improve the performance of various neural network architectures across a range of tasks, such as image classification, object detection, and natural language processing. However, most of these strategies have been developed for image inputs. To the best of our knowledge, only a few papers have investigated mixup for single-agent reinforcement learning training [9], and no research has been conducted on mixup for multi-agent collision avoidance tasks. Therefore, there is a need for further research to explore the potential of mixup for multi-agent

collision avoidance tasks, which may improve sample efficiency and enhance the performance of the trained models.

For multi-agent collision avoidance tasks, as the number of agents increases, the probability of mutual influence among agents in the field increases. In the process of heading towards the target point, agents not only are affected by fixed obstacles but also experience greater interference from other moving agents. This is similar to the increased driving difficulty on congested roads. In this paper, we introduce mixup to multi-agent reinforcement learning, where we use the state-action-reward tuples of the Markov decision process as the object for data augmentation. We construct continuous transitions between discrete steps using linear interpolation, where the similarity between continuous states is high, and action states and rewards are close. Through mixup, we create new transitions that are likely to exist in the actual transition manifold or are close to it. To make the transition more consistent with the true manifold, we use interpolation ratios to measure the relative weights between two consecutive transitions. However, when the interpolation ratio approaches 0 or 1, the state will degenerate into the original sample collected in the experiment. The interpolation ratio serves as a coefficient for adjusting the proportion of different parts of the data. By changing the value of the ratio, we can make the continuous transitions as close as possible to the authentic transition manifold. To achieve this, we constructed an energy discriminator to automatically adjust the value of the ratio.

The contributions of this paper are summarized as follows:

- We propose a novel method named MACT by constructing new continuous transitions for multi-agent collision avoidance for optimizing the sampling efficiency and asymptotic performance.
- We construct a multi-agent collision avoidance scenario in the Webots simulation environment, which is suitable for training. We apply planning algorithms such as APF and reinforcement learning algorithms such as SAC and validate that the performance of MACT is superior to other algorithms in Webots.
- We establish a platform using real robots and validate the proposed algorithm in real-world E-puck2 swarms, thus verify the applicability of MACT in the real-world.

2 Preliminaries

2.1 Problem formulation

In this paper, the multi-agent collision avoidance problem is formulated in the context of a nonholonomic differential drive robot moving on the Euclidean plane and avoiding collision with obstacles and other neighbor robots.

To tackle this problem, we assume all robots in this task are homogeneous. To be specific, all of N agents are modelled as the agents with the same radius R and same max velocity $V_{\max}$. Each robot i has access to its observation that

only provides partial information of the environment and then according to its policy, computes a collision-free action a_i to drive itself towards the target g_i.

$$\mathbf{a}^t \sim \pi_\theta \left(\mathbf{a}^t \mid \mathbf{o}^t \right), \tag{1}$$

In our experiment, θ denotes the policy parameters, and the computed action $\mathbf{a}^t$ is a vector representing the left and right wheel speed of the E-puck2 mobile car. Each robot sequentially makes its decision until it reaches its target, and robots cannot access the states and targets of other robots, which is in line with the real-world scenario. In the initial task setting, robots are placed at the edge of the environment and assigned a goal on the opposite side.

2.2 Mixup

MixUp [15] is a data-augmentation strategy originally developed for image classification. In essence, MixUp improves the training of deep learning models by creating synthetic samples through linear interpolation of pairs of training samples and their corresponding labels. Specifically, given a pair of training samples x^i and x^j, which consist of input data and their corresponding labels, MixUp generates synthetic samples by linearly interpolating between the two samples based on an interpolation ratio ϵ

$$\mathbb{M}_\epsilon \left(x^i, x^j \right) = \epsilon x^i + (1 - \epsilon) x^j \tag{2}$$

where the interpolation ratio ϵ denotes the weights of two samples. The resulting mixed data is a linear combination of the two original data, while the mixed label is a linear combination of the two original labels. This process generates a new training data point that lies on the line connecting the two original data points

3 Methodologies

In the process of training a reinforcement learning algorithm, simulators such as Webots[11] and MPE (Multi-Agent Particle Environment) [10] are commonly used to simulate state transitions at regular intervals known as time-steps. As the duration of these intervals is often quite short, there is typically a strong continuity between the states of adjacent time-steps. In the multi-agent obstacle avoidance task, we implement parameter sharing for each agent. Agents explore the environment to collect new states, and we use the Multi-Agent Continuous Transition (MACT) method to construct previously unknown continuous transitions.

3.1 Reinforcement Learning Setup

In our research, we present a decentralized partially observable Markov decision process (Dec-POMDP) as a common framework for multi-agent reinforcement learning. In this context, Dec-POMDPs are represented as a tuple, namely,

$(I, S, A, P, R, r, O, Z, \gamma)$. The index set $(I = 1, \ldots, N)$ corresponds to the collection of agents. It is crucial to note that due to the partial observability of the environment state s_t, each agent relies on their individual observation o_i in accordance with the local observation function $o_{i,t} = \mathcal{Z}i\left(s_t\right) : \mathcal{S} \rightarrow \mathcal{O}$. Every agent i selects its action based on its policy $ai, t \sim \pi_i\left(\cdot \mid O_t\right)$. The joint action space A encompasses the union of all agents' action spaces $\bigcup_{i=1}^{N} \mathcal{A}i$. We introduce a deterministic transition function $\mathrm{T} : \mathrm{S} \times \mathrm{A} \rightarrow \mathrm{S}$ and a discount factor $\gamma \in (0, 1)$. Assuming the agent policy $ri, t = \mathcal{R}_i\left(s_t, \mathbf{a}_t\right)$ is parameterized by θ_i, the individual goal of each agent is established as the discounted sum of individual rewards:

$$R_i(\tau) = \sum_{t=t_i^s}^{t_i^e} \gamma^{t-t_i^s} r_{i,t}$$

Reward function In the proposed algorithm, we aim to avoid collisions during navigation and minimize the mean arrival time of all robots. Each agent obtains its own reward at each time step, which is defined as follows:

$$r_i^t = \left(g_r\right)_i^t + \sum \left(c_r\right)_i^t \tag{3}$$

The reward r_i^t received by the robot i at time-step t consist of two terms, including g_r and c_r. In particular, the robot is awarded by $\left(g_\gamma\right)_i^t$ for getting closer to its target:

$$\left(g_r\right)_i^t = \omega_g \left(\left\|\mathrm{p}_i^{t-1} - \mathrm{g}_i\right\| - \left\|\mathrm{p}_i^t - \mathrm{g}_i\right\|\right) + r_{\text{arrival}} \tag{4}$$

$$\begin{cases} r_{\text{arrival}} = 0.05 & \text{if } \left\|\mathrm{p}_i^t - \mathrm{g}_i\right\| < 0.03m \\ r_{\text{arrival}} = 0 & \text{otherwise.} \end{cases} \tag{5}$$

where p_i^t denotes the position of the robot i at time t, g_i denotes the target position of agent i, and ω_g represents the distance reward weight. When the robot collides with other robots or obstacles in the surrounding environment, it is penalized by $\left(c_r\right)_i^t$:

$$\left(c_r\right)_i^t = \begin{cases} c_{\text{collision}} & \text{if } \left\|\mathrm{p}_i^t - \mathrm{c}_i\right\| < R_{\text{agent}} + R_{\text{col}} \\ & \text{or } \left\|\mathrm{p}_i^t - \mathrm{p}_j^t\right\| < 2R_{\text{agent}} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

where c_i denotes the position of collision around agent i. We set the reward for arriving the target point $r_{\text{arrival}} = 0.05$, the weight factor ω_g=100, the penalty for collision $c_{\text{collision}}$=-0.35, the agent's influence radius R_{agent}=0.04, the collision's influence radius R_{col}=0.11 during the training process.

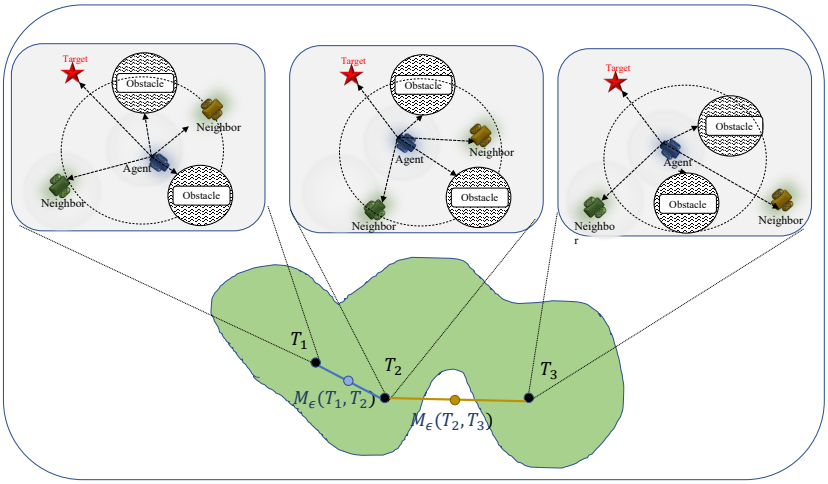


Fig. 1: The sketch of the multi-agent collision avoidance transition manifold.

3.2 Multi-agent Continuous Transition

Reinforcement learning (RL) has emerged as a prevalent approach for training agents to tackle complex tasks. Nevertheless, during the training process, a pervasive challenge is low training efficiency, primarily due to the performance constraints of various simulators employed for training, such as Webots [11] and AirSim [13]. These simulators operate on discrete time intervals for simulation, yielding discrete transitions that form a complete trajectory. For tasks with continuous action states, the volume of data accessible for training is considerably diminished. This issue is further intensified in multi-agent tasks, where state changes are influenced not only by the agent's actions but also by the actions of other agents, leading to an expanded state space that demands broader exploration and heightened sample efficiency. To tackle this challenge, we suggest employing off-policy methods, such as SAC and DDPG, which store the gathered data in a replay buffer and repurpose samples to enhance sample efficiency. In this paper, we present the mixup method for creating a Multi-agent Continuous Transition, which supplements authentic transitions with constructed transitions to bridge gaps between discrete transitions. Our approach offers a solution for boosting sample efficiency and generating a more comprehensive set of training data.

To achieve a smooth and uninterrupted progression, we employed mixup linear interpolation to bridge the gap between two discrete transitions and create new continuous transitions. The new transitions build upon the foundation of the original transitions, ensuring that it changes seamlessly along the actual trajectory. Specifically, in the multi-agent obstacle avoidance task, where parameter sharing is utilized, we constructed continuous transitions T for each agent's sample using mixup interpolation. The new transition $M_\epsilon\left(T_1, T_2\right)$ can be obtained

as following equation:

$$\mathbb{M}_\epsilon\left(T_1, T_2\right) = \epsilon T_1 + (1-\epsilon)T_2, where \epsilon \sim \mathbb{B}(\beta, \beta) \tag{7}$$

We utilize the temperature parameter β to adjust the beta distribution. In continuous control tasks, successive transitions frequently contain similar dynamic information, such as position. Consequently, for multi-agent tasks, we store the continuous samples of each agent separately in the environment and create independent continuous transitions. However, as the similarity between samples from different agents is often low, mixup-generated transitions may not reside within the true transition manifold. As illustrated in Fig. 1, the interpolation of two closely situated transitions is more likely to yield a new transition (blue dot in the figure) that genuinely exists in the task manifold, whereas the interpolation (yellow dot in the figure) produced by T_2 and T_3 is situated outside the manifold. To provide two straightforward examples: consider a scenario where a robot maintains stable forward motion, resulting in relatively consistent transitions. In this case, the constructed transition is likely to remain within the task manifold. Conversely, if the robot experiences a collision during its movement, the state changes between transitions may be more substantial, causing the newly constructed transition to appear outside the manifold.

In the MACT algorithm, the most crucial hyperparameter is the temperature β of the beta distribution. When β approaches 0, the beta distribution resembles a 2-point Bernoulli distribution, and as β approaches 1, it converges towards the uniform [0, 1] distribution. Following the insights from [9], we observe a positive correlation between the expected distance of continuous transitions and authentic transitions manifold with the value of β. To maintain the distance between two transitions below the tolerance threshold M, we can adjust the β accordingly.

An energy-based discriminator is employed to estimate the distance. We utilize a Multilayer Perceptron (MLP) f_ϕ, parameterized by ϕ, to encapsulate the information of state s_t and action a_t. We denote (s_t, a_t) as x_t and (s_{t+1}, r_t, d_t) as y_t. Consequently, the distance estimator $D(T_t)$ can be expressed as follows:

$$D\left(T_t\right) = \left\|E\left(x_t, y_t\right)\right\| = \left\|f_\phi\left(s_t, a_t\right) - y_t\right\|_2^2 \tag{8}$$

The discriminator is enhanced by optimizing ϕ to minimize Equation 8. The new transition can be represented as x'_t and y'_t. By incorporating the constructed new transition $\mathbb{M}\epsilon\left(T_t, Tt+1\right)$ into the estimator, we can compute the loss between the current D and the actual discrete transitions $E(x_t, y_t)$ and $E(x_{t+1}, y_{t+1})$. By constraining the loss (calculated using Equation 9) within the tolerance range M, we transform the problem into a constrained optimization problem.

$$Loss\left(\mathbb{M}_\epsilon\left(T_t, T_{t+1}\right)\right) = \left\|E(x'_t, y'_t) - \mathbb{M}_\epsilon\left(E(x_t, y_t), E(x_{t+1}, y_{t+1})\right)\right\|_2^2 \tag{9}$$

Formally, we can solve the constrained optimization problem to optimize the β, as shown in Equ. 10.

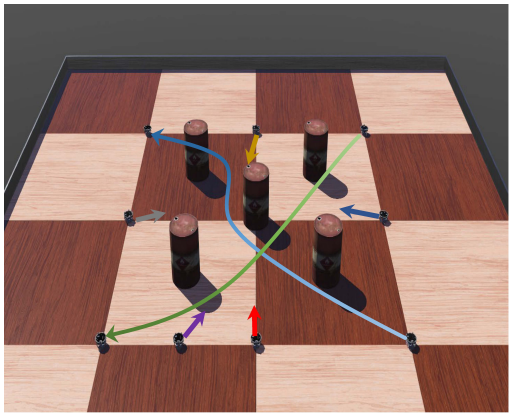


Fig. 2: Multi-robots collision avoidance scenario: Each agent is placed around the field, and the target point is set on the opposite side of the field. The agent aims to go to the target point in the shortest time while avoiding static obstacles and other agents.

$$\max \beta, \text{ s.t. } \mathbb{E}\left[Loss\left(\mathbb{M}_{\epsilon}\left(T_t, T_{t+1}\right)\right)\right] \leq M, 0 < \beta \leq 1 \\ \text{where } \epsilon \sim \mathbb{B}(\beta, \beta) \tag{10}$$

4 Experiments

In this study, our primary focus is on multi-agent navigation within confined spaces containing swarm agents and static obstacles, a scenario that is anticipated to become increasingly prevalent in the practical applications of UAVs and unmanned vehicles. Each agent is positioned around the field, with the target point set on the opposite side of the field. The agents strive to reach the target point in the shortest possible time while evading static obstacles and other agents, as depicted in Fig. 2. In this section, we conduct simulations and experiments to showcase the effectiveness of the proposed MACT algorithm. Through these simulations, we verify the benefits of MACT by comparing it with APF [7], ORCA [1], SAC [5], and PPO [12]. ORCA and APF represent two state-of-the-art traditional planning methods.

4.1 Simulation

We train our methods using Webots, a three-dimensional simulation platform. We configure E-puck2 robots, obstacles, and destinations as depicted in Fig. 2. In the simulation environment, we establish a 2m x 2m area, mirroring the real-world environment dimensions, and place five cylinders as static obstacles with a radius of 0.1m. Corresponding to the real vehicle, we employ the E-puck2 model as the agent in Webots. The E-puck2 robot has a diameter of 0.08m and a maximum speed of 0.14m/s. The action space of MACT is continuous

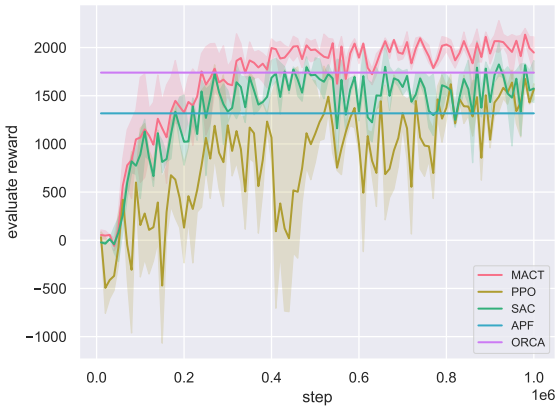


Fig. 3: The evaluate reward of 10 agents task

and represents the speed of both wheels. Each episode runs for a maximum of 1000 steps. We consider a collision to occur when agents come within 0.08m of other agents or within 0.15m of obstacles. We conducted experiments in a dense environment with 5 and 10 agents. In the task involving five agents, we observed that both traditional methods and reinforcement learning methods were effective in completing the task and achieving satisfactory policy performance due to the relatively low task difficulty.

As shown in Fig. 3, the task becomes increasingly challenging as the number of agents grows. Under this reward function design, MACT exhibits superior performance compared to planning methods. MACT is capable of learning a more effective policy when facing difficult task scenarios. Additionally, MACT demonstrates faster sample efficiency and higher reward returns compared to reinforcement methods such as SAC and PPO.

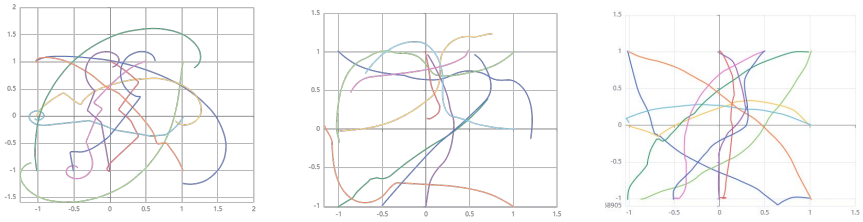
As shown in Table 1, all methods perform well in the task involving 5 agents, which has a lower difficulty level. However, in the more challenging task with 10 agents in a limited area, reinforcement learning methods SAC and PPO are more likely to encounter dangerous states. The APF and ORCA algorithms exhibit excellent safety and rarely collide, but the task requires a larger number of steps and a longer distance to be completed due to frequent stalemates. As seen in Fig. 4, MACT outperforms both reinforcement learning and planning methods in terms of travelled distance and success rate. It achieves better performance in the challenging task of 10 agents, demonstrating its effectiveness in improving multi-agent collision avoidance.

4.2 Real World

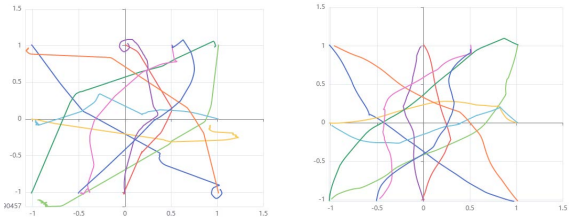
To validate our approach in a real-world setting, we utilized a motion capture system to obtain the coordinates of each E-puck robot in physical space and convert them into positions within the virtual space of Webots. Position and velocity values from the physical space were transmitted to the E-puck2 model

Table 1: Performance analysis of APF, ORCA, SAC, PPO, MACT in collision avoidance task

Agents Number	5					10				
Methods	APF	ORCA	SAC	PPO	MACT	APF	ORCA	SAC	PPO	MACT
Extra Travelled Distance(m)	2.801	2.724	2.655	2.865	**2.615**	2.881	2.946	2.912	3.021	**2.739**
Extra Travelled Steps	985	873	735	980	**733**	1850	1240	1190	1505	**964**
Danger State Rate(%)	**0.0**	**0.0**	1.1	2.1	1.0	**0.0**	**0.0**	2.0	3.3	0.9
Success Rate in 1000 Steps(%)	**100**	**100**	**100**	**100**	**100**	20	60	90	60	**100**



(a) APF (b) ORCA (c) SAC

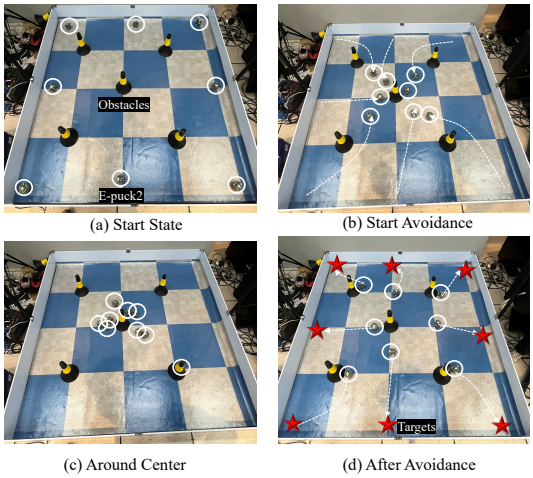


(d) PPO (e) MACT

Fig. 4: Trajectory of five methods

in Webots via ROS communication. We set the perception range for each E-puck2 agent to 0.4m, which means that location coordinates are broadcasted to neighboring robots and obstacles within a 0.4m radius from the agent.

Fig. 5 displays the demonstration of our real-world experiment. In Fig. 5(a), eight agents are positioned at the starting state around the environment's edge. To reach the target point quickly, the agents move towards the center while avoiding static obstacles (safety barrels). As shown in Fig. 5(c), when all eight agents approach the center, they encounter significant mutual interference and dynamic changes, leading to actions such as braking to prevent collisions and successfully avoiding the deadlock problem that often occurs in this region. Ultimately, as depicted in Fig. 5(d), the agents successfully complete obstacle avoidance and approach the target point. Our algorithm's effectiveness is thus validated in a real-world environment.



(a) Start State (b) Start Avoidance

(c) Around Center (d) After Avoidance

Fig. 5: Demonstration of real experiment

5 Conclusion

In this paper, we employed reinforcement learning to tackle a multi-agent collision avoidance task. For continuous tasks like this, we constructed continuous transitions using linear interpolation, effectively leveraging information from the trajectory. To generate transitions closer to the true manifold, we introduced an energy discriminator to adjust the temperature coefficient of mixup, providing more accurate samples for reinforcement learning training. Our experiments demonstrated that, compared to reinforcement learning methods such as SAC, our approach improved both sampling efficiency and training performance in the multi-agent obstacle avoidance task. Furthermore, we conducted experiments on real E-puck robots to validate our method beyond simulations. Our current approach is based on parameter sharing, and in future work, we plan to incorporate mixup data augmentation in other multi-agent algorithms, such as MAPPO.

References

1. Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., Siegwart, R.: Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: Distributed autonomous robotic systems, pp. 203–216. Springer (2013)
2. Chen, Y.F., Everett, M., Liu, M., How, J.P.: Socially aware motion planning with deep reinforcement learning. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1343–1350. IEEE (2017)
3. Everett, M., Chen, Y.F., How, J.P.: Motion planning among dynamic, decision-making agents with deep reinforcement learning. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 3052–3059. IEEE (2018)
4. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International conference on machine learning. pp. 1587–1596. PMLR (2018)

5. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. pp. 1861–1870. PMLR (2018)
6. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
7. Li, G., Yamashita, A., Asama, H., Tamura, Y.: An efficient improved artificial potential field based regression search method for robot path planning. In: 2012 IEEE International Conference on Mechatronics and Automation. pp. 1227–1232. IEEE (2012)
8. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
9. Lin, J., Huang, Z., Wang, K., Liang, X., Chen, W., Lin, L.: Continuous transition: Improving sample efficiency for continuous control problems via mixup. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 9490–9497. IEEE (2021)
10. Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems **30** (2017)
11. Michel, O.: Cyberbotics ltd. webots™: professional mobile robot simulation. International Journal of Advanced Robotic Systems **1**(1), 5 (2004)
12. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
13. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and Service Robotics: Results of the 11th International Conference. pp. 621–635. Springer (2018)
14. Yu, X., Wu, W., Feng, P., Tian, Y.: Swarm inverse reinforcement learning for biological systems. In: 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). pp. 274–279. IEEE (2021)
15. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)