

Parallel Distributional Deep Reinforcement Learning for Mapless Navigation of Terrestrial Mobile Robots

Victor A. Kich¹, Alisson H. Kolling², Junior Costa de Jesus², Gabriel V. Heisler³, Hiago Jacobs⁴, Jair A. Bottega¹, André L. da S. Kelbouscas⁴, Akihisa Ohya¹, Ricardo B. Grando^{2,4}, Paulo L. J. Drews-Jr², Daniel F. T. Gamarra³

¹ Intelligent Robot Laboratory, University of Tsukuba, (victorkich98@gmail.com) * Corresponding author

² Federal University of Rio Grande,

³ Federal University of Santa Maria,

⁴ Technological University of Uruguay

Abstract: This paper introduces novel deep reinforcement learning (Deep-RL) techniques using parallel distributional actor-critic networks for navigating terrestrial mobile robots. Our approaches use laser range findings, relative distance, and angle to the target to guide the robot. We trained agents in the Gazebo simulator and deployed them in real scenarios. Results show that parallel distributional Deep-RL algorithms enhance decision-making and outperform non-distributional and behavior-based approaches in navigation and spatial generalization.

Keywords: Parallel Distributional Deep Reinforcement Learning, Terrestrial Mobile Robot, Mapless Navigation

1. INTRODUCTION

Deep Reinforcement Learning (Deep-RL) has shown significant potential in engineering and robotics for controlling discrete and continuous systems [1]. Initially applied to stable environments [2], its complexity increases with non-stationary robots like terrestrial mobile robots due to environmental interactions.

To address this, new Deep-RL techniques focusing on action discretization have been developed [3], achieving success in mapless navigation for various mobile robots [4], [5], [6]. Distributed Deep-RL approaches offer promising solutions to the training time issue in limited simulated environments [7]. However, autonomous navigation of terrestrial mobile robots in complex environments remains challenging.

We present two new Deep-RL approaches using parallel distributional techniques: Parallel Distributional Deterministic Reinforcement Learning (PDDRL) and Parallel Distributional Stochastic Reinforcement Learning (PDSRL), incorporating prioritized memory replay for enhanced navigation in complex scenarios. As shown in Fig. 1, our methods use 24-dimensional range findings and the relative distance and angle to the target. Multiple agents' simultaneous learning improves performance in both simulation and real-world scenarios. We used the Turtlebot3 Burger robot for extensive evaluations in four increasingly complex scenarios, including an additional real scenario for spatial generalization testing.

This work's contributions include:

- Two new distributional Deep-RL approaches for improving goal-oriented mapless navigation using a simple range-based sensing architecture.
- Demonstration of the feasibility of our approaches through sim-to-real evaluation, addressing challenges such as imprecision and delays.
- Evidence that the stochastic actor-critic technique with prioritized experience replay outperforms non-

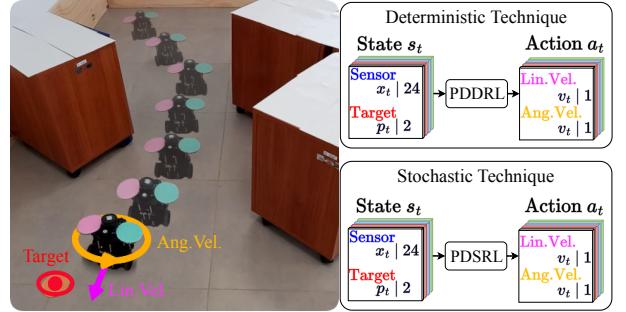


Fig. 1.: Left: Turtlebot3 Burger navigating a real-world obstacle scenario. Right: Input and output structure of our proposed PDDRL and PDSRL approaches.

distributional techniques and classic algorithms, marking the first extensive sim-to-real evaluation for mapless navigation of terrestrial mobile robots using parallel distributional Deep-RL approaches.¹²

2. RELATED WORKS

Traditional Deep-RL models rewards as a single value, but Bellemare *et al.* [8] proposed modeling it as a probabilistic distribution. This idea was extended to the new implementation of Soft Actor-Critic (SAC) algorithm by Duan *et al.* [9], which inspired our approaches.

Distributed Deep-RL, introduced to speed up training, involves distributing computation across multiple processors [10]. Mnih *et al.* [11] used asynchronous actor-critic methods, while Horgan *et al.* developed the *Ape-X* method, which decouples the actor from the learner [10]. Barth-Maron *et al.* [7] further improved this with the Deep Deterministic Policy Gradient (DDPG) algorithm.

Tai *et al.* [12] demonstrated Deep-RL's application in mobile robotics, inspiring advancements in mapless navi-

¹<https://youtu.be/cOVOijEwLUA>

²<https://github.com/victorkich/Parallel-Turtle-DRL>

gation for terrestrial robots [3]. Jesus *et al.* [4] explored deterministic and stochastic approaches in mapless navigation without parallel networks.

Our work introduces parallel Deep-RL approaches for mapless navigation of terrestrial mobile robots, addressing more complex scenarios with sim-to-real evaluation. We propose PDDRL (deterministic) and PDSRL (stochastic) approaches, compared with the traditional Behavior-based algorithm (BBA) [13] and parallel versions of DDPG and SAC using both classic and prioritized memory replay.

3. METHODOLOGY

In this section, we introduce the central concept of Deep-RL, encompassing the details of both deterministic and stochastic approaches. Additionally, we present our simulated and real scenarios, addressing specific issues such as the rewarding system and the network architecture.

3.1 Deep Reinforcement Learning

The objective of standard reinforcement learning is to maximize the expectation of the sum of discounted rewards. The state-action value function is a mathematical model that describes the expected return when taking an action a from a state s , and then acting according to the policy π [14]. This function, known as

$$Q_\pi(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \quad (1)$$

is typically utilized to evaluate the quality of a policy.

At each time step, the networks predict the behavior for the present state and generate a signal of temporal difference (TD) error. The Bellman operator

$$(T_\pi Q)(s, a) = r(s, a) + \gamma \mathbb{E}[Q(s', \pi(s')) | s, a] \quad (2)$$

can minimize this TD error, whose expectation is computed with respect to the next state s' .

This work employs two distinct neural networks, namely an actor and a critic, to evaluate the TD error. The TD error is assessed under a separate target policy and value network, where the networks have distinct parameters (θ', ω') to stabilize learning. The critic-network generates the Q-value for action, whereas the output of the actor-network is a real value that represents the selected action.

3.2 Parallel Distributional Deterministic RL

The DDPG architecture, as proposed by [15] and extended in D4PG [7], serves as a cornerstone in Deep-RL applications for mobile robots in continuous observation spaces [16]. It adopts an actor-critic framework that utilizes approximation functions to learn policies in continuous spaces.

The primary distinction between DDPG and D4PG lies in the latter's incorporation of the distributional Bellman operator, which is crucial for the optimization process. This operator is defined as follows:

$$(T_\pi Z)(s, a) = r(s, a) + \gamma \mathbb{E}[Z(s', \pi(s')) | s, a], \quad (3)$$

where $Q_\pi(s, a) = \mathbb{E} Z_\pi(s, a)$ and Z_π returns a distributional variable, specifically a categorical distribution.

In the D4PG framework, the categorical distribution models the output of the critic network, which predicts a vector of probabilities over predefined reward bins, each corresponding to a range of potential reward values. This distribution allows the network to estimate a full probability distribution of expected returns, rather than a single expected value, capturing the variability in possible outcomes:

$$\text{Categorical}(Z_\pi(s, a)) = [p_1, p_2, \dots, p_k], \quad (4)$$

where p_i is the probability of the return falling into the i -th bin and k is the number of bins. The training of the critic involves minimizing the divergence between the predicted and target distributions, enhancing the policy robustness by better representing the uncertainties in dynamic environments.

Furthermore, D4PG introduces the concept of N -step returns to estimate the Temporal Difference (TD) error, expressed as:

$$\begin{aligned} (T_\pi^N Q)(s, a) &= r(s, a) + \\ &\mathbb{E} \left[\sum_{n=1}^{N-1} \gamma^n r(s_n, a_n) + \gamma^N Q(s_N, \pi(s_N)) | s, a \right]. \end{aligned} \quad (5)$$

In our research, we have developed an approach called **PDDRL** based on D4PG, incorporating an extension of N -step returns and a critic value function modeled as a categorical distribution [8]. The deterministic policy is denoted by μ , while noisy actions are represented by μ' , where the noise process involves:

$$\mu' = \mu(s_t) + \mathcal{N}, \quad (6)$$

and \mathcal{N} follows the Ornstein-Uhlenbeck process [17].

Given the inconsistent experimental results reported in the original D4PG study [7], our research has led us to develop two variants: **PDDRL**, which utilizes classical replay memory, and **PDDRL-P**, employing prioritized replay memory [18].

Finally, a target network needs to be established to enhance learning stability. This network is a duplicate of the actor and critic networks but uses “soft” updates. The weights θ' of the target network are gradually adjusted according to the factor τ , as described by the equation:

$$\theta' = \tau \theta + (1 - \tau) \theta'. \quad (7)$$

3.3 Parallel Distributional Stochastic RL

The SAC architecture [19], proposed as a stochastic counterpart to deterministic actor-critic methods like DDPG, employs approximation functions to learn policies in continuous action spaces. SAC introduces a Bellman operator enhanced by entropy addition, aiding in exploration and policy optimization:

$$(T_\pi Q)(s, a) = r(s, a) + \gamma \mathbb{E}[Q(s', a') - \alpha \log \pi(a'|s') | s, a]. \quad (8)$$

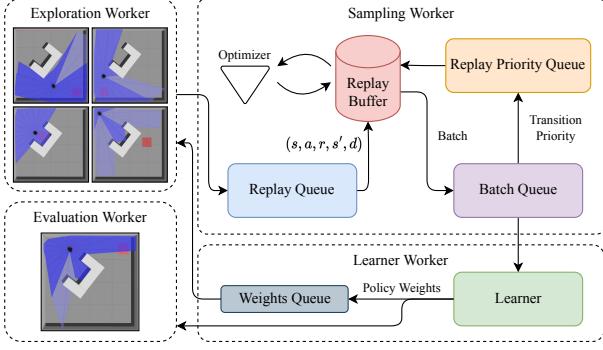


Fig. 2.: Parallel Deep-RL training process structure.

To foster robust exploration, SAC emphasizes maximizing both reward and the entropy of the policy, thereby promoting action diversity and discouraging premature policy convergence. The algorithm assigns equal probability to actions yielding similar Q-values and mitigates failure risks in the Q-function approximation due to uncertain actions.

Building on this, DSAC [20] merges the maximum entropy framework of SAC with the distributional approach of DDPG, leading to a hybrid Bellman operator:

$$(T_\pi Z)(s, a) = r(s, a) + \gamma [Z(s', a') - \alpha \log \pi(a'|s') | s, a], \quad (9)$$

which integrates both stochastic and distributional components, further described as:

$$Z_\pi(s, a) = \sum_{t=0}^{\infty} [r(s, a) + \gamma - \alpha \log \pi(a_{t+1}|s_{t+1}) | s, a]. \quad (10)$$

In our research, the newly proposed **PDSRL** methodology builds upon DSAC, incorporating the strategic use of N-step returns and soft updates—features previously detailed in Section 3.2 under PDDRL—to enhance both prediction accuracy and stability.

We implemented two variants, **PDSRL** using classical replay memory, and **PDSRL-P** with prioritized replay memory, both featuring a 100000-step sized replay memory for consistency across all Deep-RL approaches.

3.4 Network Structure

All the approaches in our work employ a neural network with 26 inputs and two outputs. The inputs comprise 24 range findings of a Lidar, as well as the relative position and relative angles to the target. The sensor samples range from 0° to 360° , and are equally spaced by 15° . The input angles serve to direct the vehicle towards the target and enhance the learning process, while the target distance encourages the network to minimize it. Meanwhile, the outputs are the linear and angular velocities that enable control of the vehicle. As a point of comparison with the Deep-RL approaches, we employed a traditional technique BBA, which uses hardcoded navigation procedures. It is important to note that the BBA uses the same amount of sensor distance information as the other approaches for a fair comparison.

The neural networks employed in all approaches have three hidden fully-connected layers, with 256 neurons

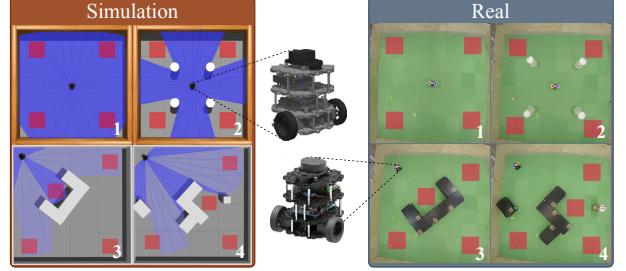


Fig. 3.: Simulated and real setups.

each. These layers are connected through ReLU activation. The range of action is between -1 and 1 , and the activation function for the actor-network is the hyperbolic tangent function (*Tanh*). The outputs for the linear velocity are scaled between -0.12 and 0.15 meters, while the outputs for the angular velocity are scaled between $-0.1 m/s$ and $0.1 m/s$. In both approaches, the critic network predicts the Q-value of the current state, while the actor-network predicts the current state. During the training phase of the parallel approaches, four agents were employed for training, and one agent was employed for evaluation. The training process is depicted in Fig. 2.

3.5 Reward Function

In order to facilitate learning, it is essential to establish a reward system that encourages good actions and penalizes poor ones by the agent. This system has been designed based on empirical knowledge obtained through empirical evaluation. The reward system implemented for this work is provided below:

$$r(s_t, a_t) = \begin{cases} r_{arr} & \text{if } d_t < c_d \\ r_{coll} & \text{if } \min_x < c_o \\ r_{idle} & \text{if } \min_x \geq c_o \text{ and } d_t \geq c_d \end{cases} \quad (11)$$

A simple rewarding function is defined, providing only three types of rewards: one for successfully completing the task, another for failing to complete the task, and a final one in the event that neither of the previous two occurred. A reward of 200 is given to the agent for suc-

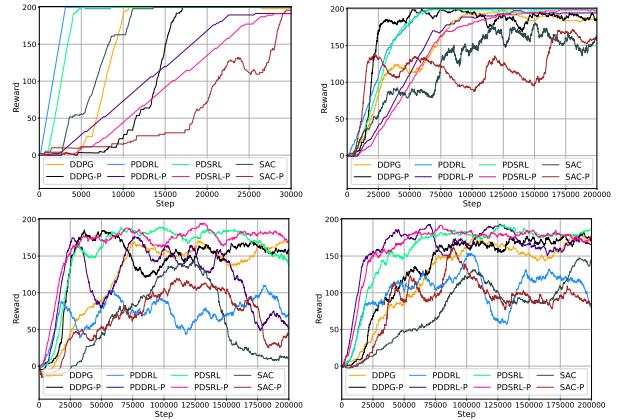


Fig. 4.: Moving average of the agent’s reward at each training step for all parallel approaches in each scenario. Scenarios organized by following the respectively order, from left to right and from top to bottom: 1, 2, 3, and 4.

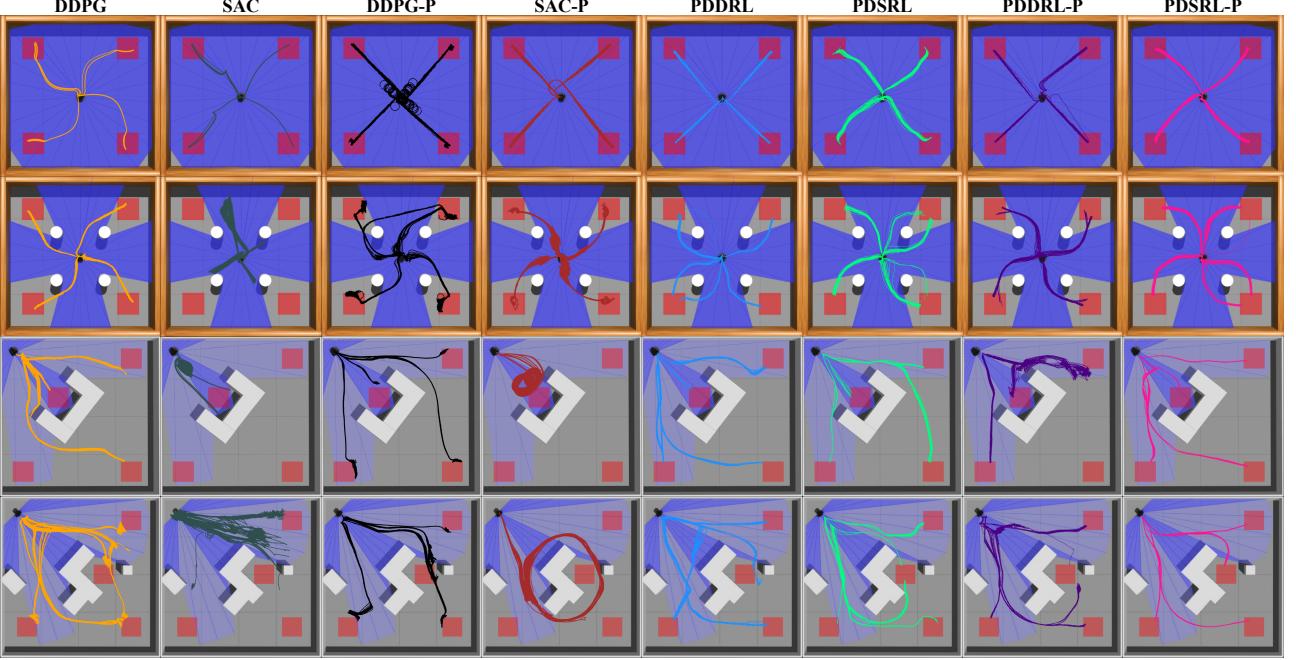


Fig. 5.: The behavior of each parallel approach was evaluated by conducting 100 navigation trials in each simulated scenario. The lines illustrate the paths taken by the agents, with each agent given 25 attempts to capture each target.

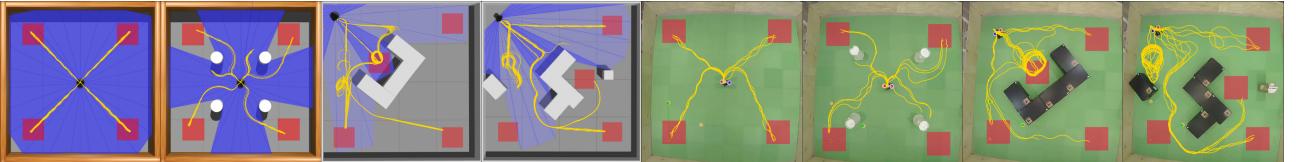


Fig. 6.: BBA approach evaluated in simulated and real scenarios over 100 and 12 trials, respectively. Lines represent agents' paths, with 25 (simulated) and 3 (real) attempts to capture each target.

cessfully reaching the goal within a margin of c_d meters, which was set at 0.25 meters. If the agent collides with an obstacle or reaches the limits of the scenario, a negative reward r_{coll} of -20 is given. Collisions are detected when distance sensor readings are less than a distance of c_o , which is set at 0.12 meters. If the agent's Lidar distance is greater than c_o and the target distance is greater than c_d , a reward r_{idle} of 0 is given. This function enables a focus on the Deep-RL approaches themselves, their similarities and differences, rather than the scenario.

4. EXPERIMENTS

In this section, we present the experimental evaluation of our proposed approaches. The experiments are designed to assess the performance of the methods in both simulated and real-world environments. We describe the setup of the simulated and real scenarios used for training and testing, followed by a detailed analysis of the experimental results.

4.1 Simulation and Real Scenarios

The simulations of the robot and the scenario use the Gazebo Simulator. The connection between the robot and the agents was made using Robot Operating System (ROS). For distributing the agents, multiple instances of

the Gazebo simulator are created in parallel and the data acquired are published to the replay buffer. The robot chosen here is the TurtleBot3 Burger version. The real and simulated Turtlebot3 Burger used can be seen in Fig. 3.

Four simulation scenarios were employed in this study. The first scenario represents a navigable area for the robot to move, with the walls being the only obstacle that could cause the robot to collide. In the event of a collision with the wall or any obstacle, a negative reward is issued for the action, and the current episode is terminated. The second scenario features four fixed cylinder-shaped obstacles with a radius of 25 centimeters each. The third and fourth scenarios were built in order to create more challenging paths for the robot to reach the final goal. The third scenario includes a "U"-shaped object that presents the agent with two possible trajectories of equal cost, with the obstacle creating a dead-end in the middle. The fourth scenario is asymmetrical and more complex, requiring the intelligent agent to develop better strategies to avoid collisions. Both real and simulated scenarios are labeled in Fig.3.

After training the approaches through simulation, we evaluated our approaches in real scenarios. Some of the necessary data in the real scenario were obtained by image processing using OpenCV. All four real scenarios resembled the simulated ones, but their geometry is not equal. The differences of each one are more visually demon-

Table 1.: Precision over 100 (for simulation) and 12 (for real) navigation trials in four different scenarios for all approaches.

Scenario	BBA	DDPG	SAC	DDPG-P	SAC-P	PDDRL	PDSRL	PDDRL-P	PDSRL-P
First Sim.	100%	100%	100%	100%	100%	100%	100%	100%	100%
Second Sim.	62%	100%	0.0%	85%	64%	100%	100%	100%	100%
Third Sim.	75%	74%	0.0%	75%	0.0%	100%	100%	70%	100%
Fourth Sim.	73%	81%	25%	92%	0.0%	83%	98%	76%	100%
First Real	100%	100%	100%	100%	100%	100%	100%	100%	100%
Second Real	83.3%	100%	0.0%	100%	41.6%	100%	91.6%	100%	100%
Third Real	75%	100%	0.0%	100%	0.0%	100%	100%	0.0%	100%
Fourth Real	75%	66.6%	25%	100%	25%	25%	75%	75%	100%

strated in Section 4.

4.2 Experimental Results

We evaluated the navigation ability and spatial generalization of all approaches in both simulation and real-world scenarios. After the training phase, consisting of 30000 steps for the first scenario and 200000 steps for the others, we analyzed the moving average of rewards (Fig. 4). The results showed that stochastic distributional approaches outperformed others in all scenarios, despite a slower start. DDPG had similar reward means to the top approaches. Each approach was evaluated over 100 episodes with pre-determined target coordinates.

Table 1 shows the results from the simulation experiments. Figs. 5 and 6 illustrate the behavior of Deep-RL and classical approaches. In scenario 1, all achieved 100% accuracy. In more complex scenarios, distributional algorithms maintained 100% accuracy, while others varied. DDPG matched the distributional algorithms.

In Scenario 3, distributional approaches outperformed others, except for PDDRL-P, which had 70% accuracy. This issue is known in the literature, where prioritized memory may limit the generalization capacity of the deterministic distributional agent based on the D4PG algorithm. In Scenario 4, only the PDSRL-P approach captured all points, followed by PDSRL with 98% accuracy. Notably, DDPG-P achieved 92% accuracy, surpassing deterministic distributional algorithms.

Fig. 5 shows PDSRL and PDSRL-P had smoother trajectories. DDPG showed instability with repeated collisions in Scenario 3. Prioritization reduced collisions in Scenario 4, but instability remained. Real environment experiments used target points similar to simulations. Fig. 7 shows trajectories in all real scenarios. PDSRL-P and DDPG-P achieved 100% accuracy in real environments. Performance differences between PDDRL and PDDRL-P were significant in scenarios 3 and 4. Real scenarios introduced delays, affecting performance.

Table 2.: Precision and distance metrics over 12 navigation trials in the extra scenario for top approaches.

Algorithm	Precision (%)	Crashes (%)
DDPG	58.3%	41.6%
DDPG-P	100%	0.0%
PDDRL	100%	0.0%
PDSRL	75%	8.3%
PDSRL-P	100%	0.0%

We proposed a novel scenario where pre-trained models from scenario 3 were evaluated in an unseen environment. Fig. 7 and Table 2 show the results. DDPG had high crash rates, while PDSRL minimized crashes but didn't always achieve rewards. DDPG-P, PDDRL, and PDSRL-P achieved 100% accuracy, demonstrating strong spatial generalization.

Our extensive validation shows the proposed approaches perform well in all scenarios, including the novel one. Deep-RL approaches outperformed traditional algorithms in real-world challenges. Prioritized versions achieved the best success rates, particularly PDSRL-P. Evaluation in a novel scenario confirmed the approaches' learning capabilities and spatial generalization.

5. CONCLUSION

This work introduces novel Deep Reinforcement Learning (Deep-RL) techniques for terrestrial mobile robots to perform mapless navigation, demonstrating their learning capabilities in simulation and potential for real-world implementation. The results highlight that parallel distributional Deep-RL approaches can effectively tackle complex real-world robotics problems that non-learning-based and non-distributional approaches without prioritized memory replay cannot address.

REFERENCES

- [1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in *ICLR*. PMLR, 2016.
- [2] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, “Continuous deep q-learning with model-based acceleration,” in *ICML*, 2016, pp. 2829–2838.
- [3] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, “Target-driven visual navigation in indoor scenes using deep reinforcement learning,” in *IEEE ICRA*. IEEE, 2017, pp. 3357–3364.
- [4] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. d. S. L. Cuadros, and D. F. T. Gamarra, “Soft actor-critic for navigation of mobile robots,” *Journal of Intelligent & Robotic Systems*, vol. 102, no. 2, pp. 1–11, 2021.

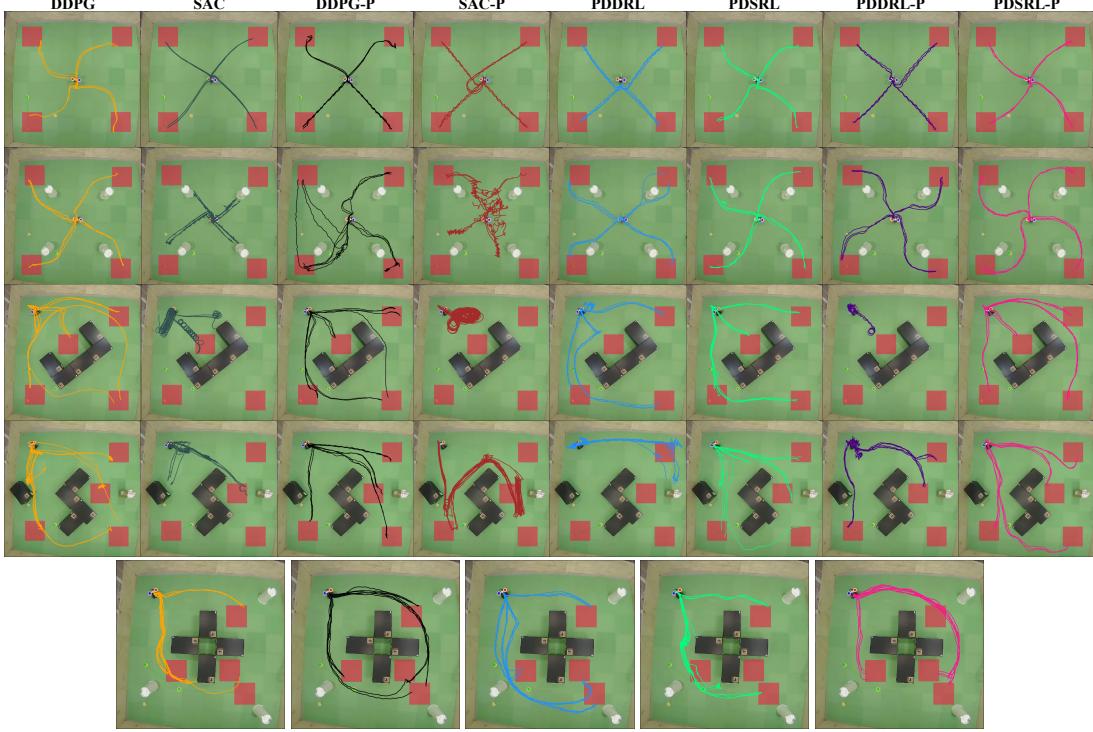


Fig. 7.: (Top) The behavior of each parallel approach was evaluated by conducting 12 navigation trials in each real scenario. The lines illustrate the paths taken by the agents, with each agent given 3 attempts to capture each target. (Bottom) The behavior of the best classified parallel approaches evaluated in the extra real scenario. Organized by the following order: DDPG, DDPG-P, PDDRL, PDSRL, and PDSRL-P.

- [5] R. B. Grando, J. C. de Jesus, V. A. Kich, A. H. Kolling, N. P. Bortoluzzi, P. M. Pinheiro, A. A. Neto, and P. L. Drews, “Deep reinforcement learning for mapless navigation of a hybrid aerial underwater vehicle with medium transition,” in *IEEE ICRA*. IEEE, 2021, pp. 1088–1094.
- [6] R. B. Grando, J. C. de Jesus, V. A. Kich, A. H. Kolling, and P. L. J. Drews-Jr, “Double critic deep reinforcement learning for mapless 3d navigation of unmanned aerial vehicles,” *Journal of Intelligent & Robotic Systems*, vol. 104, no. 2, pp. 1–14, 2022.
- [7] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap, “Distributed distributional deterministic policy gradients,” *ICLR*, 2018.
- [8] M. G. Bellemare, W. Dabney, and R. Munos, “A distributional perspective on reinforcement learning,” in *ICML*. PMLR, 2017, pp. 449–458.
- [9] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, “Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors,” *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2021.
- [10] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, “Distributed prioritized experience replay,” in *ICLR*. PMLR, 2018.
- [11] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *ICML*, vol. 48. New York, New York,

- USA: PMLR, 20–22 Jun 2016, pp. 1928–1937.
- [12] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” in *IEEE/RSJ IROS*. IEEE, 2017, pp. 31–36.
- [13] V. J. Lumelsky and A. A. Stepanov, “Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape,” *Algorithmica*, vol. 2, no. 1, pp. 403–430, Nov 1987.
- [14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *ICML*. PMLR, 2014, pp. 387–395.
- [16] L. Tai and M. Liu, “Towards cognitive exploration through deep reinforcement learning for mobile robots,” 2016.
- [17] G. E. Uhlenbeck and L. S. Ornstein, “On the theory of the brownian motion,” *Physical review*, vol. 36, no. 5, p. 823, 1930.
- [18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” in *ICLR*. PMLR, 2016.
- [19] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, “Soft actor-critic algorithms and applications,” 2018.
- [20] X. Ma, L. Xia, Z. Zhou, J. Yang, and Q. Zhao, “Dsac: Distributional soft actor critic for risk-sensitive reinforcement learning,” 2020.