

Consensus-based Deep Reinforcement Learning for Mobile Robot Mapless Navigation

Wenxing Liu*, Hanlin Niu*, Ipek Caliskanelli*, Zhengjia Xu[†], Robert Skilton*

*Remote Applications in Challenging Environments (RACE), United Kingdom Atomic Energy Authority, Culham, UK

[†]School of Aerospace, Transport, and Manufacturing, Cranfield University, Cranfield, MK43 0AL, UK

Abstract—When using mobile robots to perform data collection about the surroundings, the performance might be dissatisfying since the environments could be unknown and challenging. This situation will pose challenges for mobile robot navigation and exploration. To tackle this issue, we propose a consensus-based deep reinforcement learning (DRL) algorithm for multiple robots to perform mapless navigation and exploration. The proposed algorithm leverages both consensus-based training and DRL, which reduces required training steps while maintaining the same training reward. Once trained with fixed obstacles, the proposed training model can demonstrate adaptability in handling real-world random static obstacles and sudden obstacles. The experimental video is available at: <https://youtu.be/ym2yvbKg4fU>.

Index Terms—Deep reinforcement learning, consensus, obstacle avoidance, multi-robot systems.

I. INTRODUCTION

With the continuous evolution of science and technology, robots have played an indispensable role across various applications, including formation coordination [1], pick-and-place [2], [3], localization [4], etc. Navigation [5] and exploration [6] constitute crucial applications of robotics, where mobile robots are equipped with sensors to gather information about the structure of the environment during the stage of approaching the target positions. If the environments being explored are known and safe, the process of data collection will be simple and rapid. Nevertheless, in many circumstances, the environments being explored are unknown and challenging. To address this issue, we incorporate the utilization of DRL to enable mobile robots to engage in navigation and exploration without reliance on pre-existing maps.

In this paper, we propose a consensus-based DRL algorithm for multiple robots to perform navigation and exploration in unknown environments, as shown

This work is supported by UKAEA/EPSCRC Fusion Grant 2022/2027 No. EP/W006839/1. (Corresponding author: Hanlin Niu. Email address: hanlin.niu@ukaea.uk)

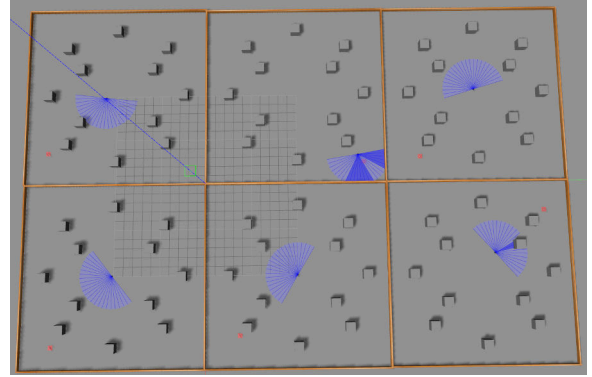


Fig. 1: Consensus-based parallel training in Gazebo environment. The cubes represent obstacles and the red squares are target positions.

in Fig. 1. The main contributions of the paper can be summarized as follows:

- 1) An actor-critic-based DRL method is presented for each mobile robot to deal with collision avoidance tasks. By applying this method, each robot is able to accomplish mapless navigation and exploration.
- 2) Specially, a consensus-based training policy is developed for the multi-robot system to improve the training efficiency of the DRL method. The proposed multi-agent training algorithm takes advantage of both consensus-based training and DRL, which decreases the number of required training steps while ensuring the same training reward.
- 3) The performance of the proposed method is validated in real environments. After being trained with fixed obstacles, the proposed training model can also be applied to deal with obstacles with random locations as well as sudden obstacles.

The structure of this work proceeds as follows. Section II provides an extensive review of related literature. In

Section III, the proposed methodology is expounded in detail. Section IV presents experimental validation to underscore the viability of the proposed approach. Section V summarizes this work.

II. RELATED WORK

Consensus control in navigation and exploration:

Consensus control has been harnessed to manage and resolve navigation and exploration tasks [7]. A position consensus controller which was based on the sliding mode control was introduced in [8] to solve the problem of unmanned aerial vehicle (UAV) navigation. A decentralized control approach was developed in [9] to allow underwater vehicles to reach a goal with large time delays. In [10], a two-level consensus approach with the concept of a virtual leader was proposed for the formation control and tracking of multiple autonomous agents. In [11], a decentralized estimation strategy on the basis of a consensus-extended Kalman filter was designed for multiple fixed-wing UAVs to improve observability. In the works mentioned above, the authors demonstrate a predominant focus on theoretical analysis which relies on fairly accurate modelling of dynamic models. If the dynamic model of any agent changes, the proposed control input should also be adjusted to satisfy the stability analysis. Our proposed algorithm is model-free, which is versatile and suitable for a broader spectrum of applications.

Deep reinforcement learning in navigation and exploration: DRL plays an important role in addressing the challenges associated with exploration and navigation tasks [12]. The aim of applying DRL in these tasks is to enhance intelligence in determining decision boundaries, allowing for the handling of unpredictable or sudden changes in the environment [13]. A DRL-based decision algorithm was developed in [14] to address the problem of automatic exploration under an unknown environment. In [15], a discrete state space DRL algorithm was introduced to reduce the training time for mobile robot navigation. In [16], a successor-feature-based DRL algorithm was proposed to enable a mobile robot to transfer knowledge from previous navigation tasks to new environments. A combined DRL and imitation learning method was mentioned in [17] to deal with motion planning in cluttered and crowded environments. Nevertheless, the aforementioned studies were primarily trained with data obtained from one simulated agent, which could be regarded as a constraint. Our algorithm engages multiple agents in the training process, which facilitates efficiency in training stages while ensuring the same reward.

III. METHODOLOGY

In this section, we introduce an innovative multi-agent training algorithm that leverages the benefits of both consensus-based training and DRL. Initially, we present the actor-critic-based [18] DRL approach for each mobile robot, and then the consensus-based training method for multi-agent systems is introduced. Through the utilization of actor-critic-based DRL with consensus-based training, the proposed algorithm decreases the number of required training steps while ensuring the same reward.

A. Deep Reinforcement Learning Setup

The goal of the experiment is to train each mobile robot to reach a randomly assigned target position while ensuring collision avoidance throughout the reaching process, as shown in Fig. 2.

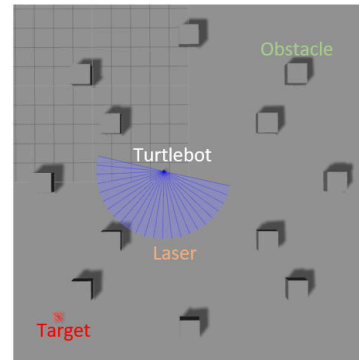


Fig. 2: Training environment of each mobile robot.

1) *Action Space and State Space:* For each mobile robot, the action space a is a 2-dimensional vector that consists of two different velocities, which are the linear velocity V_l and the angular velocity V_a . These values are computed in polar coordinates.

The state space s is a vector of dimension 28 which includes the linear velocity V_l , the angular velocity V_a , 24 laser values, the distance between the mobile robot and the target, and the angle difference between the mobile robot and the target. The 24 laser values are normalized by the maximum range.

2) *Reward Space:* Each mobile robot is designed to reach its target position while performing collision avoidance. The reward function for each mobile robot is defined as

$$r_i = r_{di} + r_{ai} + r_{oi} + r_{ei} + r_{ni} \quad (1)$$

where r_i stands for accumulated reward of the i^{th} mobile robot, r_{di} represents the distance reward of the i^{th}

mobile robot, r_{ai} describes the arrive reward of the i^{th} mobile robot, r_{oi} is the collision reward of the i^{th} mobile robot, r_{ei} represents the linear punishment reward of the i^{th} mobile robot, r_{ni} is the angular punishment reward of the i^{th} mobile robot.

The distance reward r_{di} can be defined as

$$r_{di} = r_{pi} - r_{ci} \quad (2)$$

where r_{pi} is the distance between the i^{th} mobile robot and the target with the previous action, r_{ci} stands for the distance between the i^{th} mobile robot and the target with the current action. If r_{pi} is larger than r_{ci} , the i^{th} mobile robot will obtain a positive distance reward since it gets closer to the target position.

The arrive reward r_{ai} is computed by

$$r_{ai} = \begin{cases} r_{al} & \text{if } |r_{ci}| < r_{dh} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where r_{al} represents a positive large reward, r_{dh} stands for the distance threshold.

The collision reward r_{oi} is given by

$$r_{oi} = \begin{cases} -r_{ol} & \text{if } r_{limit} \leq |r_{si}| < 2r_{limit} \\ -2r_{ol} & \text{if } |r_{si}| < r_{limit} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where r_{ol} denotes a positive collision reward, r_{si} is the distance between the i^{th} mobile robot and the obstacles, r_{limit} represents the laser threshold limit. As a result, r_{oi} will receive a punishment if $|r_{si}|$ is above the laser threshold. By introducing r_{oi} in the accumulated reward, the mobile robot is encouraged to avoid obstacles.

The linear punishment reward r_{ei} is computed by

$$r_{ei} = \begin{cases} -r_{ep} & \text{if } V_l < V_{lmin} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where r_{ep} represents a positive reward, V_{lmin} stands for the minimum linear velocity threshold. This linear punishment reward stimulates the mobile robot to explore the environment.

The angular punishment reward r_{ni} is obtained by

$$r_{ni} = \begin{cases} -r_{np} & \text{if } |V_a| > 0.5|V_{amax}| \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where r_{np} represents a positive reward, V_{amax} stands for the maximum angular velocity threshold. This angular punishment reward prevents the mobile robot from excessive rotation speed, which encourages it to investigate the surroundings.

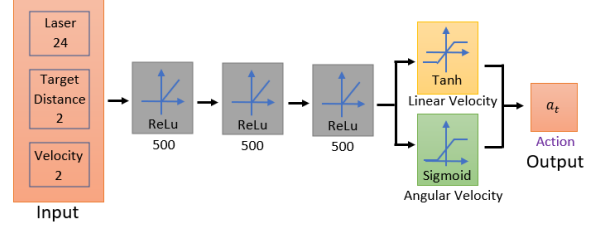


Fig. 3: The structure of the actor network.

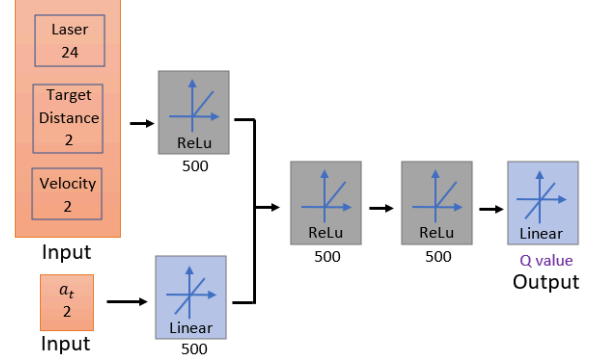


Fig. 4: The structure of the critic network.

3) *Network Structure*: The input of the actor network passes the data through three dense layers whose activation functions are ReLu [19] with 500 nodes, as shown in Fig. 3. This input comprises 28 elements, encompassing parameters such as linear velocity, angular velocity, 24 laser readings, the distance between the mobile robot and its target, and the angular difference between the mobile robot and the target. The output of the actor network consists of 2 elements, which are the linear velocity V_l and the angular velocity V_a . This output is also utilized as an input for the critic network, as illustrated in Fig. 4. The other input of the critic network shares the same set of elements as the actor network's input. The output of the critic network quantifies the performance of the actor network, which is a Q value derived through a linear activation function.

In every iteration, each mobile robot engages in environment exploration, introducing random noise into its actions. The accumulated reward obtained in this process is denoted as r_j . In each training session, the state space, action space, next state space, and the accumulated reward are recorded and stored in the replay buffer. When this buffer reaches its capacity, the training model can draw and learn from its contents in a random manner.

B. Consensus-based Training Strategy

In order to boost the training efficiency in the multi-robot system, a consensus-based training strategy is proposed. Therefore, our initial focus is on introducing the consensus network structure that aids in this training process.

Define the interaction topology of P agents as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \dots, P\}$ is the vertex set, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represents the edge set. Denote $(p, i) \in \mathcal{E}$ if the p^{th} agent and i^{th} agent are connected [20]. For undirected graph $\mathcal{G}$, define the adjacency matrix $\mathcal{A}$ as $\mathcal{A} = [a_{pi}] \in \mathbb{R}^{P \times P}$, where $a_{ip} = a_{pi} > 0$ if $(p, i) \in \mathcal{E}$, otherwise $a_{ip} = a_{pi} = 0$. If $\mathcal{D} = \text{diag}\{\sum_{p \neq 1} a_{1p}, \dots, \sum_{p \neq P} a_{Pp}\} \in \mathbb{R}^{P \times P}$, then the Laplacian matrix $\mathcal{L}$ can be defined as $\mathcal{L} = \mathcal{D} - \mathcal{A}$. It is easy to obtain that $\mathcal{L} \geq 0$ and $\mathcal{L}\mathbf{1}_P = 0$, where $\mathbf{1}_P = [1, \dots, 1]^T$. The rank of $\mathcal{L}$ equals $P - 1$ if the graph $\mathcal{G}$ has a spanning tree [21].

In the context of an undirected graph $\mathcal{G}$, if $\hat{\varphi}_i \in \mathbb{R}^n$ is the updated training parameter of $\varphi_i \in \mathbb{R}^n$ after a single consensus step and φ_i represents the row vector of the training parameter for agent i within the graph, the consensus training step for each agent i can be depicted as follows

$$\hat{\varphi}_i = \varphi_i + u_i \quad (7)$$

$$u_i = \sum_{p=1}^P a_{ip}(\varphi_p - \varphi_i) \quad (8)$$

where a_{ip} is an element of the graph adjacency matrix and u_i represents the input of the agent i . a_{ip} can be generated from the undirected graph $\mathcal{G}$.

By combining (8) and (7), the consensus algorithm can be employed to update an agent i using the subsequent approach:

$$\begin{aligned} \hat{\varphi}_i &= \varphi_i + \sum_{p=1}^P a_{ip}(\varphi_p - \varphi_i) \\ &= \varphi_i - \sum_{p=1}^P l_{ip}\varphi_p \\ &= \mathcal{C}_i(\varphi_p, l_{ip}) \end{aligned} \quad (9)$$

where l_{ip} stands for an element of the Laplacian matrix $\mathcal{L}$ and $\mathcal{C}_i$ is the consensus protocol of the i^{th} agent.

The training parameter update for all P agents after a single consensus step can be concluded as follows:

$$\begin{aligned} \hat{\varphi} &= ((I_P - \mathcal{L}) \otimes I_n)\varphi \\ &= \mathcal{C}(\varphi, \mathcal{L}) \end{aligned} \quad (10)$$

where $\mathcal{C}$ is the consensus protocol for all agents, I_P and I_n stand for the $P \times P$ and $n \times n$ identity matrix, $\mathcal{L}$ denotes the Laplacian matrix. By iteratively computing

(10), this consensus algorithm drives all agents to converge towards their weighted average [22].

C. Consensus-based Deep Reinforcement Learning

Given the consensus training strategy in the previous subsection, the proposed consensus-based DRL algorithm is now introduced. As stated in [23], the process of updating the actor training parameter ω^i and the critic training parameter θ^i for the i^{th} agent can be given as:

$$\delta_t^i = r_{t+1}^i + \lambda g_\theta(s_{t+1}^i, a_{t+1}^i) - g_\theta(s_t^i, a_t^i) \quad (11)$$

$$\theta_{t+1}^i = \hat{\theta}_t^i + \alpha \delta_t^i \nabla_\theta g(s_t^i, a_t^i) \quad (12)$$

$$\omega_{t+1}^i = \hat{\omega}_t^i + \beta \nabla_\omega \pi(s_t^i) \nabla_a g_\theta(s_t^i, a^i)|_{a^i=\pi_\omega(s^i)} \quad (13)$$

where α and β represent the critic and the actor learning rate, respectively. δ_t^i stands for the temporal difference error and λ is the discounted factor. $g_\theta(s_{t+1}^i, a_{t+1}^i)$ denotes the linearized value function with respect to θ and $\nabla_\theta g(s_t^i, a_t^i)$ represents the gradient of the value function. $\pi_\omega(s^i)$ is the linearized policy function with respect to ω and $\nabla_\omega \pi(s_t^i)$ represents the gradient of the policy function.

By applying (9), the consensus-based critic training parameter update for the i^{th} agent can be summarised as:

$$\hat{\theta}_t^i = \theta_t^i + \sum_{k=1}^P a_{ik}(\theta_t^k - \theta_t^i) \quad (14)$$

$$\theta_{t+1}^i = \hat{\theta}_t^i + \alpha \delta_t^i \nabla_\theta g(s_t^i, a_t^i) \quad (15)$$

Substituting (14) into (15), we have

$$\theta_{t+1}^i = \theta_t^i + \sum_{k=1}^P a_{ik}(\theta_t^k - \theta_t^i) + \alpha \delta_t^i \nabla_\theta g(s_t^i, a_t^i) \quad (16)$$

Similarly, the consensus-based actor training parameter update for the i^{th} agent can be obtained as

$$\omega_{t+1}^i = \omega_t^i + \sum_{k=1}^P a_{ik}(\omega_t^k - \omega_t^i) + \beta d_t^i \quad (17)$$

where $d_t^i = \nabla_\omega \pi(s_t^i) \nabla_a g_\theta(s_t^i, a^i)|_{a^i=\pi_\omega(s^i)}$.

Let $\Theta_t = [\delta_t^1 \nabla_\theta g(s_t^1, a_t^1), \dots, \delta_t^P \nabla_\theta g(s_t^P, a_t^P)]^T$ and $\Omega_t = [d_t^1, \dots, d_t^P]^T$, for P agents, the update of the critic and actor training parameters at iteration t can be illustrated as

$$\begin{aligned} \theta_{t+1} &= ((I_P - \mathcal{L}) \otimes I_n)\theta_t + \alpha \Theta_t \\ &= \mathcal{C}(\theta_t, \mathcal{L}) + \alpha \Theta_t \end{aligned} \quad (18)$$

$$\begin{aligned} \omega_{t+1} &= ((I_P - \mathcal{L}) \otimes I_n)\omega_t + \beta \Omega_t \\ &= \mathcal{C}(\omega_t, \mathcal{L}) + \beta \Omega_t \end{aligned} \quad (19)$$

Algorithm 1 summarizes our proposed consensus-based DRL algorithm.

Algorithm 1 Consensus-based DRL

- 1: Initialize initial state s_0^i , fixed behaviour policy ξ , discounted factor λ , maximum time step T , exploration noise O_t , actor training parameter ω_0^i , critic training parameter θ_0^i , critic learning rate α , actor learning rate β of each agent.
 - 2: Obtain initial state s_1^j of each agent.
 - 3: Reset the environment.
 - 4: **while** $t = 1, T$ **do**
 - 5: Select action $a_t^i = \xi(s_t^i) + O_t$ for action exploration based on the fixed behaviour policy ξ and the exploration noise O_t .
 - 6: Execute action a_t^i , compute reward r_{t+1}^i and acquire new state s_{t+1}^i .
 - 7: Store transition $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ in the replay buffer R_b .
 - 8: Sample a minibatch $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ from transitions in R_b .
 - 9: Calculate the temporal difference error:
10: $\delta_t^i = r_{t+1}^i + \lambda g_\theta(s_{t+1}^i, a_{t+1}^i) - g_\theta(s_t^i, a_t^i)$.
 - 11: Update the critic training parameter:
12: $\theta_{t+1}^i = \hat{\theta}_t^i + \alpha \delta_t^i \nabla_{\theta} g(s_t^i, a_t^i)$.
 - 13: Update the actor training parameter:
14: $\omega_{t+1}^i = \hat{\omega}_t^i + \beta \nabla_{\omega} \pi(s_t^i) \nabla_{a} g(s_t^i, a_t^i)|_{a^i = \pi_{\omega}(s^i)}$.
 - 15: Deploy consensus-based training strategy on the critic and the actor training parameter of each agent:
16: $\hat{\theta}_{t+1}^i = \mathcal{C}_i(\theta_{t+1}^k, l_{ik})$.
 - 17: $\hat{\omega}_{t+1}^i = \mathcal{C}_i(\omega_{t+1}^k, l_{ik})$.
 - 18: **end while**
-

IV. EXPERIMENTS AND RESULTS

This section validates the practicality of the multi-agent training algorithm. The training process was carried out on a laptop with an Intel Core i7 CPU (2.6 GHz) with 16 GB RAM and Nvidia GTX 2070 super.

A. Training Details

The proposed method was trained in Robot Operating System (ROS) [24] and Gazebo [25]. Experiments were carried out with Turtlebot3 Burger mobile robots in both simulated and real-world environments. For each mobile robot, the maximum laser range is set to 3.5 m. The actor learning rate β and the critic learning rate α have a fixed value of 0.0001. The discounted factor λ is set at a constant value of 0.9. The action exploration adheres to the ϵ -greedy exploration strategy [26]. We initialize ϵ at 0.9 and employ a decay rate $\iota = 0.99995$. The maximum time step T is 60000.

B. Comparison with the Baseline Method

We applied actor-critic-based DRL to train a single mobile robot as a baseline for comparison with the proposed multi-agent training algorithm, as can be seen from Fig. 5. The benchmark for comparison is derived from the average reward achieved during training. The average reward is determined by taking the mean of rewards collected within a predefined batch size. In comparison to the baseline method, our proposed algorithm exhibits a quicker training speed. With the proposed algorithm, each mobile robot attains a training reward of 60 at approximately 7000 training steps, whereas the baseline method requires around 45000 training steps to achieve the same training reward. Compared to single-agent DRL, the proposed algorithm reduces the number of required training steps while ensuring the same training reward.

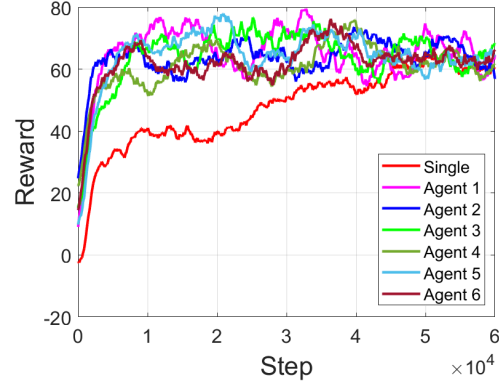


Fig. 5: Average reward graph of the proposed method with six mobile robots compared with single-agent baseline method.

C. Evaluation of Consensus Training

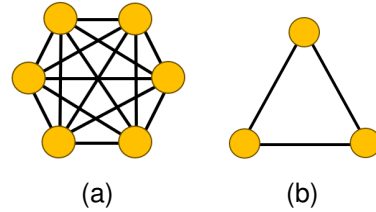


Fig. 6: Interaction topology of varying numbers of mobile robots: (a) Six robots; (b) Three robots

In this section, we aim to investigate the impact on consensus speed as the number of training agents

varies. The interaction topology of varying numbers of mobile robots is depicted in Fig. 6. Fig. 7 illustrates the average reward obtained when employing the proposed method with three mobile robots. Each mobile robot reaches a training reward of 60 at roughly 34000 training steps with the proposed algorithm, while the baseline method demands approximately 45000 training steps to attain the same training reward. Compared with Fig. 5, a larger number of mobile robots participating in the proposed method can expedite the consensus speed, thereby reducing the number of required training steps while maintaining the same training reward.

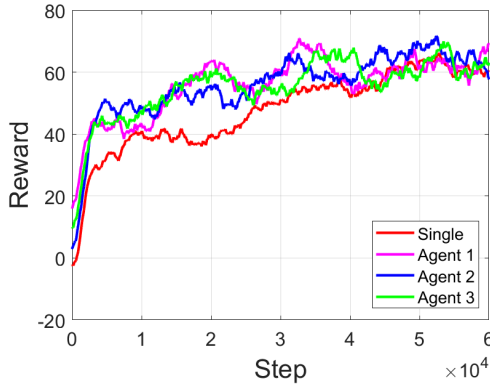


Fig. 7: Average reward graph of the proposed method with three mobile robots compared with single-agent baseline method.

D. Evaluation of Real-world Environment



Fig. 8: Real-world testing environment of the Turtlebot with three random static obstacles.

The training model derived from the proposed method has been deployed on the Turtlebot3 Burger for real-world evaluation, as shown in Fig. 8. The objective is to accomplish mapless navigation to the goal while ensuring random obstacle avoidance. As can be seen from Fig. 9, three random static obstacles are successfully

detected by the Turtlebot since there are laser rays (green dots) around each obstacle. The green line in Fig. 9 represents the trajectory of the Turtlebot. The proposed training model successfully guides the Turtlebot to the designated goal without colliding with any random static obstacle, which validates the effectiveness of our proposed algorithm.



Fig. 9: Trajectory map of the Turtlebot with three random static obstacles.

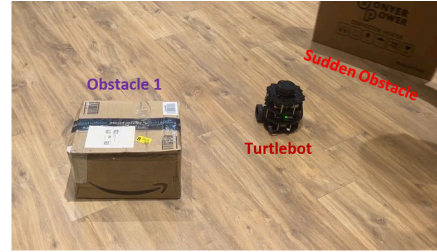


Fig. 10: Real-world testing environment of the Turtlebot with sudden obstacles.

In order to demonstrate that our algorithm does not rely on maps for Turtlebot navigation, we remove obstacle 2 and obstacle 3 without updating the map and introduce an unexpected obstacle during the ongoing Turtlebot navigation process, as depicted in Fig. 10. It can be observed from Fig. 11 that obstacle 2 and obstacle 3 are no longer in their original position since there are no laser rays (green dots) around them. Nevertheless, a sudden obstacle has been successfully identified and avoided by the Turtlebot during its navigation process. To sum up, the proposed method demonstrates its capability in both mapless navigation and handling sudden obstacles.

V. CONCLUSION

In this paper, a consensus-based DRL algorithm is introduced to enable multiple robots to conduct map-



Fig. 11: Trajectory map of the Turtlebot with sudden obstacles.

less navigation and exploration. The proposed algorithm leverages both consensus-based training and DRL, resulting in a reduction in the required training steps while maintaining the same training reward. Following training with fixed obstacles in simulation, the proposed training model can effectively handle random static obstacles and sudden obstacles in the real world. In the future, further optimization of the proposed approach will be explored to address more complex scenarios.

REFERENCES

- [1] K. Wu, J. Hu, Z. Ding, and F. Arvin, "Distributed bearing-only formation control for heterogeneous nonlinear multi-robot systems," in *2023 IFAC World Congress*. Elsevier BV, 2023.
- [2] W. Liu, H. Niu, W. Pan, G. Herrmann, and J. Carrasco, "Sim-and-real reinforcement learning for manipulation: A consensus-based approach," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3911–3917.
- [3] W. Liu, H. Niu, R. Skilton, and J. Carrasco, "Sim-to-real deep reinforcement learning with manipulators for pick-and-place," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2023, pp. 240–252.
- [4] K. Wu, J. Hu, Z. Ding, and F. Arvin, "Finite-time fault-tolerant formation control for distributed multi-vehicle networks with bearing measurements," *IEEE Transactions on Automation Science and Engineering*, 2023.
- [5] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Fault-tolerant cooperative navigation of networked uav swarms for forest fire monitoring," *Aerospace Science and Technology*, vol. 123, p. 107494, 2022.
- [6] A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, "Decentralized multi-agent exploration with online-learning of gaussian processes," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 4222–4229.
- [7] K. D. Listmann, M. V. Masalawala, and J. Adamy, "Consensus for formation control of nonholonomic mobile robots," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3886–3891.
- [8] P. Kolaric, C. Chen, A. Dalal, and F. L. Lewis, "Consensus controller for multi-uav navigation," *Control Theory and Technology*, vol. 16, pp. 110–121, 2018.
- [9] M. A. Joordens and M. Jamshidi, "Consensus control for a system of underwater swarm robots," *IEEE Systems Journal*, vol. 4, no. 1, pp. 65–73, 2010.
- [10] M. Porfiri, D. G. Roberson, and D. J. Stilwell, "Tracking and formation control of multiple autonomous agents: A two-level consensus approach," *Automatica*, vol. 43, no. 8, pp. 1318–1328, 2007.
- [11] B. Gong, S. Wang, M. Hao, X. Guan, and S. Li, "Range-based collaborative relative navigation for multiple unmanned aerial vehicles using consensus extended kalman filter," *Aerospace Science and Technology*, vol. 112, p. 106647, 2021.
- [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 International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 31–36.
- [13] 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 *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.
- [14] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 6, pp. 2064–2076, 2019.
- [15] E. Marchesini and A. Farinelli, "Discrete deep reinforcement learning for mapless navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 688–10 694.
- [16] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2371–2378.
- [17] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5671–5677.
- [18] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.
- [19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.
- [20] K. Wu, J. Hu, B. Lennox, and F. Arvin, "Mixed controller design for multi-vehicle formation based on edge and bearing measurements," in *2022 European Control Conference (ECC)*. IEEE, 2022, pp. 1666–1671.
- [21] C. Godsil and G. F. Royle, *Algebraic graph theory*. Springer Science & Business Media, 2013, vol. 207.
- [22] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.
- [23] W. Liu, H. Niu, I. Jang, G. Herrmann, and J. Carrasco, "Distributed neural networks training for robotic manipulation with consensus algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
- [25] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*(*IEEE Cat. No. 04CH37566*), vol. 3. IEEE, 2004, pp. 2149–2154.
- [26] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.



DSSPACE

<https://dspace.org/>

Consensus-based deep reinforcement learning for mobile robot mapless navigation

Liu, Wenxing; Niu, Hanlin; Caliskanelli, Ipek; Xu, Zhengjia; Skilton, Robert

2024-06-05

Institute of Electrical and Electronics Engineers (IEEE)

<https://doi.org/10.1109/ICIT58233.2024.10540762>

Liu W, Niu H, Caliskanelli I, et al., (2024) Consensus-based deep reinforcement learning for mobile robot mapless navigation. In: 2024 IEEE International Conference on Industrial Technology (ICIT), 25-27 March 2024, Bristol, UK