Proceedings of the 13th International Workshop on Robot Motion and Control, Poznan University of Technology, Poznan, Poland, July 2-4, 2024

Deep Reinforcement Learning for Multi-Robot Local Path Planning in Dynamic Environments

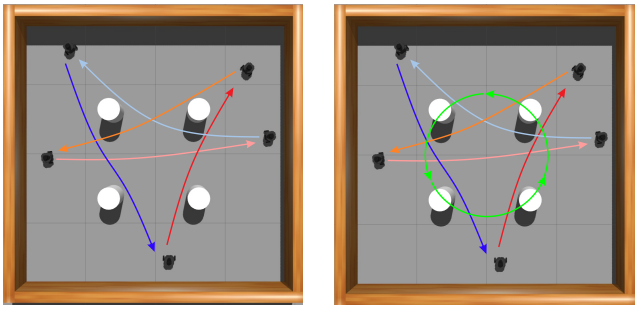
Cong-Thanh Vu and Yen-Chen Liu

***Abstract*— The transportation and logistics sectors are experiencing a remarkable increase in the utilization of autonomous mobile robots, playing a pivotal role in the efficient management and distribution of merchandise and resources. Particularly in large-scale systems, finding effective solutions for coordinating and planning paths is a critical factor influencing overall system performance. Addressing the challenges of dynamic environments and working in uncertain conditions, this study introduces a novel local path planning approach for a multi-agent system based on a decentralized framework, employing deep reinforcement learning. The methodology incorporates Proximal Policy Optimization (PPO) and the Dynamic Window Approach (DWA) to analyze situations based on environmental information. We propose a new cost function for DWA by developing two subfunctions based on the information between agents and the goal. The observation space for deep reinforcement learning is designed by integrating velocity space and incorporating the cost function derived from DWA. The effectiveness of this method is evaluated in four simulated environments and experimented with four TurtleBot Burger robots. Additionally, the approach is compared with state-of-the-art multi-robot path planning methods, revealing significant improvements in success rates, reaching up to 30%, in dynamic environments, and reductions in path lengths.**

I. Introduction

Nowadays, the extensive incorporation of autonomous mobile robots in transportation and logistics is experiencing a significant rise, playing a crucial role in the effective management of goods and resource allocation [1]. This trend is particularly pronounced in large-scale systems, necessitating effective solutions for coordinating and planning multi-robot systems. The challenge becomes even more complex when addressing multi-agent pathfinding, which is a problem with applications extending beyond transportation, encompassing monitoring, mapping, exploration, search, and rescue operations [2]. Individual agent path planning usually focuses on creating collision-free, efficient routes, and smooth paths [3]. However, in a multi-agent system, the paths must not only ensure smooth movement but also proactively avoid conflicts.

Traditionally, path planning encompasses two main approaches: offline planning and online planning [4]. Offline planning operates under the assumption of static obstacles and complete environmental knowledge, whereas online planning adjusts to dynamic obstacles and partial information. However, offline planning often proves inadequate in dynamic workspaces due to the presence of dynamic obstacles [5]. Another strategy involves initial route planning followed by online adjustments, but this can lead to inefficiencies due to frequent re-planning [6]. To tackle path planning challenges, navigation frameworks are divided into global and local planning tasks [7]. Global planning, a traditional offline method, guides robots along a predetermined path, while local planning enables robots to avoid dynamic obstacles encountered in real-time while adhering to the global path. In multi-agent systems, local navigation frameworks treat other robots as dynamic obstacles, but conflicts may arise with increasing robot numbers sharing the same path. Prioritized planning methods have been introduced to manage multi-agent conflicts, assigning priority orders to each agent and planning paths accordingly [8]. However, as the number of agents increases, real-time system performance may be impacted, necessitating optimization for time efficiency.



(a) (b)

Fig. 1: Training environment for the local planner: (a) Static training environment with four obstacles represented as cylindrical blocks arranged at the intersection of five robot trajectories. (b) Dynamic training environment, where four obstacles move along circular trajectories.

Recently, learning-based approaches have been explored to tackle the challenge of online scheduling in dynamic environments by learning from the training environment, as shown in Fig. 1, through trial and error to gain experience [9], [10]. Although reinforcement learning (RL) has showcased remarkable performance across various applications, several challenges hinder its effectiveness in the path planning problem. Firstly, in expansive environments, the scarcity of reward values results in heightened training efforts, diminishing the overall efficacy of the learning process. Another challenge pertains to the issue of overfitting, where robots often struggle to generalize beyond their training environments to unseen surroundings [11]. Consequently, the efficiency, generality, and scalability of current RL-based

* This work was supported in part by the National Science and Technology Council (NSTC), Taiwan, under Grant NSTC 112-2636-E-006-001 and NSTC 112-2628-E-006-014-MY3.

Cong-Thanh Vu and Yen-Chen Liu are with the Department of Mechanical Engineering, National Cheng Kung University, Tainan 70101, Taiwan. Email: vuthanh.cdt@gmail.com, yliu@mail.ncku.edu.tw.

2024 13th International Workshop on Robot Motion and Control (RoMoCo) | 979-8-3503-9396-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ROMOCO60539.2024.10604387

planning tools remain insufficient to meet the demands of many applications.

We propose a local planning approach for multi-robot systems based on a decentralized framework using deep reinforcement learning (DRL) with observations based on velocity space from the Dynamic Window Approach (DWA), replacing traditional local navigation frameworks that can integrate with other global planning methods (such as RRT, A*, etc.). The main contribution of our work includes:

- **New Cost Functions for DWA:** Introducing two new cost functions for multi-robot systems to enhance the adaptability of DWA in complex scenarios.
- **Observation Space and Reward Functions for DRL:** We propose a novel observation space grounded in velocity space and introduce corresponding cost functions for DRL. Additionally, we present reward functions aligned with Proximal Policy Optimization (PPO), enabling the DRL model to optimize velocity selections within the permissible velocity space. This approach ensures stability and smooth motion, reducing collision rates by up to 30% in dynamic scenarios involving multiple robots, as compared to [12].
- **Sim-to-Real Capability:** Demonstrating the transition from simulation to the real world, the method is experimented with four Turtlebot Burger robots and static obstacles to showcase zero-shot transfer capabilities.

The rest of this paper is organized as follows: Section II begins with a review of related works, followed by the background provided in Section III, and the methodology presented in Section IV. Section V covers the results and discussion. Finally, Section VI serves as the conclusion, summarizing the findings and outlining future directions.

II. Related Work

The field of multi-robot path planning has witnessed remarkable advancements in recent years, with approaches falling into three primary categories: coupled, decoupled, and dynamically-coupled approaches [13]. In coupled approaches [14], each robot is treated as a constituent within a unified space, and path planning is subsequently conducted within this space. While theoretically capable of identifying optimal solutions for multi-robot planning challenges, these methods encounter complexity limitations as the number of robots grows, thus diminishing their practicality for larger groups. On the other hand, decoupled approaches involve finding individual paths for each agent, and then adjusting these paths based on the movements of other agents to avoid collisions [15]. Therefore, adjustments to individual paths can be made in a low-dimensional search space, making decoupled approaches suitable for large-scale multi-agent systems. Nevertheless, the main limitation of decoupled methods is that the low-dimensional search spaces they use only represent a small portion of the overall configuration space, and often do not achieve completeness. Dynamically-coupled approach [16] combines both coupled and decoupled methods. The method enables agents to expand the search space when necessary in the shared search space without exploring the entire space. However, the dynamic environment remains a challenge for these methods.

Taking advantage of recent advances in deep learning, RL is being considered a promising approach for solving path planning in dynamic environments through trial and error. Sartoretti et al. [13] introduced a path planning framework that integrates reinforcement and imitation learning. A drawback of this approach is that as agents move toward the goal, they may disappear from the map and no longer be considered part of the state space for other agents. This can easily lead to impacts from peripheral factors, reducing system performance. The research work by Wang et al. [17] proposed a path planning approach for mobile robots in dynamic environments using globally guided reinforcement learning. In this approach, robots make global plans based on traditional algorithms and then apply RL to avoid conflicts during movement as a local planner. However, the workspace and actions of this method are discrete, limiting the robot to only being able to perform basic actions such as going forward, backward, turning left, and turning right, making it unsuitable for diverse practical environments. The use of RL for multi-robot planning in a continuous action space was also presented in [12]. Nevertheless, the instantaneous control velocities generated by this method are often infeasible in terms of dynamics. In other words, the computed velocities do not ensure the constraints of acceleration and non-holonomic motion of the robot, resulting in highly non-smooth and jerky trajectories.

III. Background

In this section, we offer an explanation of the background concepts and components that are utilized in this work.

A. Dynamic Window Approach

DWA [18] operates within the velocity search space $[v, w]$, determining feasible robot movements in the workspace. The approach aims to reduce the state space by imposing dynamic limitations and utilizes an optimal function to maximize the velocity space, thereby achieving the defined objective function.

In the initial stage, a velocity space reachable by the robot within a given time period is defined while considering constraints to avoid collisions during movement. The set V_a of admissible velocities is established according to the following criteria:

$$V_a = \left\{(v, w) | v \leq \sqrt{2\text{dist}(v, w)\dot{v}}, w \leq \sqrt{2\text{dist}(v, w)\dot{w}}\right\}, \tag{1}$$

in which $\text{dist}(v, w)$ denotes the distance to the nearest obstacles, and v, w, $\dot{v}$, and $\dot{w}$ represent linear and angular velocities and accelerations, respectively.

To further reduce the state space, the velocity search space V_d is defined by constraining velocities of the robot within

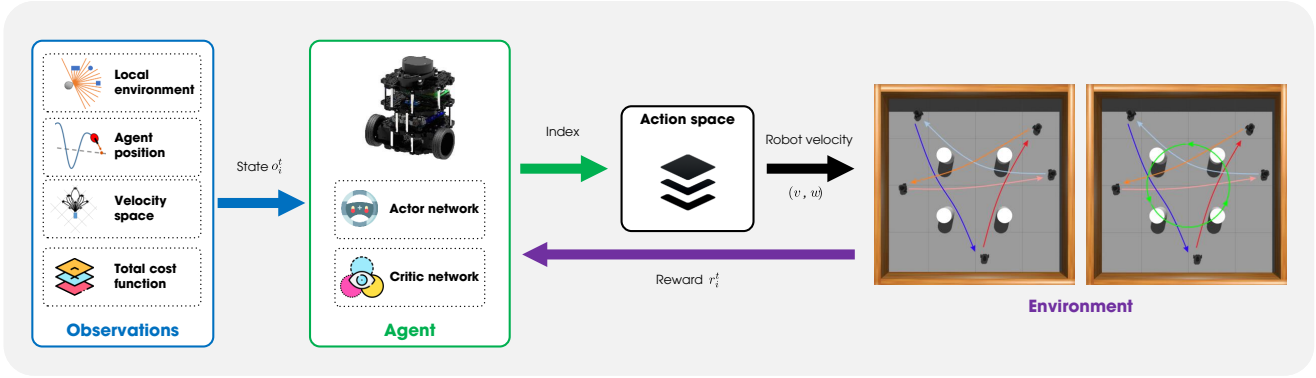


Fig. 2: The overall structure of the system architecture. The ith robot observes environmental states, including obstacle information from the lidar sensor, agent positions, velocity space, and the total cost function for the multi-robot system. PPO uses this state space to determine the action index and evaluates the environment with velocity (v, w), then receives reward r_i^t.

a time step Δt, and let v_a, w_a be the actual velocity:

$$V_d = \Big\{(v,w)|v \in [v_a - \dot{v}\Delta t, v_a + \dot{v}\Delta t], \\ w \in [w_a - \dot{w}\Delta t, w_a + \dot{w}\Delta t]\Big\}. \tag{2}$$

The dynamic window search space is obtained as the intersection of the restriction fields with V_s, the space of possible velocities:

$$V_r = V_s \cap V_a \cap V_d. \tag{3}$$

In the second stage, an objective function is formulated to maximize the velocity space V_r:

$$G(v,w) = \alpha\ \text{dist}(v,w) + \\ \beta\ \text{heading}(v,w) + \gamma\ \text{vel}(v,w), \tag{4}$$

where heading(v, w) measures the angle of progress towards the goal position, and vel(v, w) represents the robot speed supporting fast movements to a goal. The weighting constants α, β, and γ are utilized to balance the contributions of each term in the objective function.

B. Deep Reinforcement Learning

Instead of following a predetermined sequence of actions, as is characteristic of traditional methods, DRL capitalizes on environmental observations for the purpose of exploiting rewards and then deciding the action. Assuming there are N robots in the system controlled by a decentralized framework, each ith robot (where $i = 1, ..., N$) is equipped with a proposed local navigation framework. The ith robot acquires information about its surrounding environment, such as obstacles, constituting the observation space denoted as o_i^t. The evaluative framework of the system relies upon the reward function r_i^t to inform decision-making. The action space a_i^t of DRL includes a linear and angular velocity that the robot can achieve. The primary goal during training is to maximize the performance of the robot by encouraging actions that lead to rewards and avoiding actions that result in penalties. This process continues until the robot consistently achieves maximum rewards after multiple training iterations. In this work, we employ PPO [19] algorithm for robot training, ensuring that the learning process enables the robot to consistently achieve optimal outcomes with each training cycle.

IV. Methodology

This section introduces two new cost functions for DWA with multiple agents, along with the observation space, reward function, and network architecture for PPO. The structural diagram of the method is illustrated in Fig. 2.

A. Dynamic Window Approach for Multi-agents

DWA algorithm is expanded with two new cost functions to optimize performance in a multi-agent setting. Firstly, in order to prevent conflicts between the paths of different agents, we introduce the function $\text{dist}_{agent}(v, w)$, which represents the smallest distance from the simulation trajectory to the agents nearby. Although the $\text{dist}_(v, w)$ function exists in DWA to address obstacle avoidance during robot movement, its activation is contingent upon the robot being within the observation range of the sensor. Consequently, this cost function provides support for assisting agents in generating optimal paths while operating in the same environment.

The second cost function $\text{dist}_{goal}(v, w)$ is the distance from the simulated trajectory to the goal. This helps reduce the time taken to reach the goal, especially when there are obstacles around the trajectory. After adding the evaluation sub-functions, the improved cost function for the ith robot can be shown as follows with two new weighting constants λ, δ:

$$G(v,w) = \alpha\ \text{dist}(v,w) + \beta\ \text{heading}(v,w) \\ + \gamma\ \text{vel}(v,w) + \lambda\ \text{dist}_{agent}(v,w) + \delta\ \text{dist}_{goal}(v,w). \tag{5}$$

Based on the newly established cost function, by selecting appropriate weights, a pair of velocities (v, w) can be obtained. However, in the real world, environmental information is uncertain and dynamic, necessitating the adaptation of these weights to different circumstances. Leveraging the learning capabilities of DRL, PPO is employed to determine the position of the velocity pair within the space V_r.

B. PPO-based Local Planning

1) Observation Space: The obstacle observations at each timestamp are represented by $(o_{obs})_i^t = [d_1, d_2, d_3, \ldots, d_M]^T$, $(o_{obs})_i^t \in \mathbb{R}^M$, obtained from a 2D lidar sensor. These distances to obstacles, denoted by d, correspond to resolution angles on the sensor, covering a range from the smallest to the largest with uniform resolution. These distances are expressed in the local coordinate frame attached to the robot. The second observation provided is linear and angular velocity observations are represented as $(o_v)_i^t \in \mathbb{R}^H$ and $(o_w)_i^t \in \mathbb{R}^H$, respectively. They are discretized from V_r into separate spaces for linear and angular velocities. The associated cost function is also utilized to define the observation space $(o_c)_i^t \in \mathbb{R}^H$. Each pair of velocities v and w within this space is assigned a distinct cost value, serving as a metric to evaluate the effectiveness of the velocity space. Moreover, the agents observation $(o_a)_i^t \in \mathbb{R}^{2N}$ is used to represent the relative positions between the ith robot and the remaining robots, denoted by $(o_a)_i^t = [x_1^i, y_1^i, x_2^i, y_2^i, ..., x_N^i, y_N^i]^T$.

Hence, the observation space $o_i^t \in \mathbb{R}^K$ with $K = M + 3H + N$ is expressed as:

$$o_i^t = (o_{obs})_i^t + (o_v)_i^t + (o_w)_i^t + (o_c)_i^t + (o_a)_i^t. \tag{6}$$

2) Action Space: The action space is set relative to the velocity space V_r from DWA.

3) Reward Structure: During navigation, the robot should move toward the target while maintaining a clear distance and avoiding collisions with obstacles, as well as preventing conflicts with other robots on the path. Therefore, the reward function considers factors like the distance to obstacles, the distance to the goal, and the distance between agents. The reward function for the ith robot is expressed as follows:

$$r_i^t = (r_g)_i^t + (r_c)_i^t + (r_a)_i^t, \tag{7}$$

where $(r_g)_i^t$ describes the goal reward, $(r_c)_i^t$ denotes the collision reward, and $(r_a)_i^t$ represents the reward for conflicts between agents.

The goal reward $(r_g)_i^t$ is designed:

$$(r_g)_i^t = \begin{cases} r_{\text{arrive}} & \text{if } (d_g)_i^t < D \\ \sigma\left((d_g)_i^{t-1} - (d_g)_i^t\right) & \text{otherwise} \end{cases}, \tag{8}$$

here σ represents the weight assigned to the goal-reaching component of the reward. Meanwhile, $(d_g)_i^{t-1} = \left\|(p_r)_i^{t-1} - p_i^g\right\|_2$ and $(d_g)_i^t = \left\|(p_r)_i^t - p_i^g\right\|_2$ denote the distance between the current position and the goal position of the ith robot at time step t and the previous step $t-1$, respectively. When the distance between the ith robot and the goal position is smaller than the maximum threshold for reaching the target D, the robot will receive a positive reward denoted as r_{arrive}. Conversely, the robot will receive a positive reward proportional to the displacement between consecutive time steps towards the goal. This is designed to motivate the robot to approach the goal as rapidly as possible. Otherwise, if the robot moves farther away from the target, it will incur a negative reward.

To assist the robot in avoiding obstacles during its movement, a penalty function $(r_c)_i^t$ is applied, ensuring that the robot maintains a safe distance R from obstacles. If the robot reaches a safe distance, it incurs a penalty as $r_{\text{collision}}$. The structure of this penalty function is outlined as follows:

$$(r_c)_i^t = \begin{cases} r_{\text{collision}} & \text{if } (d_c)_i^t < R \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

in which $(d_c)_i^t$ represents the nearest distance to the obstacle detected by the lidar sensor, obtained from $(o_{obs})_i^t$.

Additionally, in order to proactively prevent conflict during the local planning process and manage agent positions, a penalty function is introduced as follows:

$$(r_a)_i^t = \begin{cases} r_{\text{conflict}} & \text{if } (d_a)_i^t < S \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

with $(d_a)_i^t$ represents the minimum distance from the current position of the robot to other agents. The robot will receive a penalty r_{conflict} when the distance is less than the allowed distance S.

We set $r_{\text{arrive}} = 100$, $r_{\text{collision}} = -1000$, $r_{\text{conflict}} = -1000$ and $\sigma = 0.5$.

C. Network architecture

The architecture of the actor network is employed as illustrated in Fig. 4. The input consists of a K-dimensional observation space o_i^t aggregated from sub-observation spaces. The input data is fed through 3 fully-connected layers, each with 512 nodes. As the action space represents a set of robot velocity vectors, the activation function of the last layer of the actor network is a Softmax function, providing probabilities for sampling actions. The other hidden layers in the network use the ReLU activation function. Similarly, the critic network is equipped with 3 fully-connected layers, each with 512 nodes, and utilizes ReLU activation, followed by the last layer of the critic network employs a linear activation function.

V. Results and Discussion

In this section, we explain the deployment of our method in the simulation environment and experiments, then we evaluate the proposed local planner by comparing it with another method.

A. Simulation Results

Our proposed method is implemented using PyTorch with CUDA and employs a distributed training framework for acceleration. The training simulation is conducted using Turtlebot Burger on ROS Humble within the Gazebo simulation environment, running on a workstation equipped with an Intel I9-10900K CPU and RTXA4000 GPU. The training process consists of two stages involving five robots. In the first stage, a static obstacle environment is created with four obstacles surrounding the trajectory of the robot (Fig. **??**). The goal points are strategically generated and dynamically change as the targets are achieved, providing adaptability to

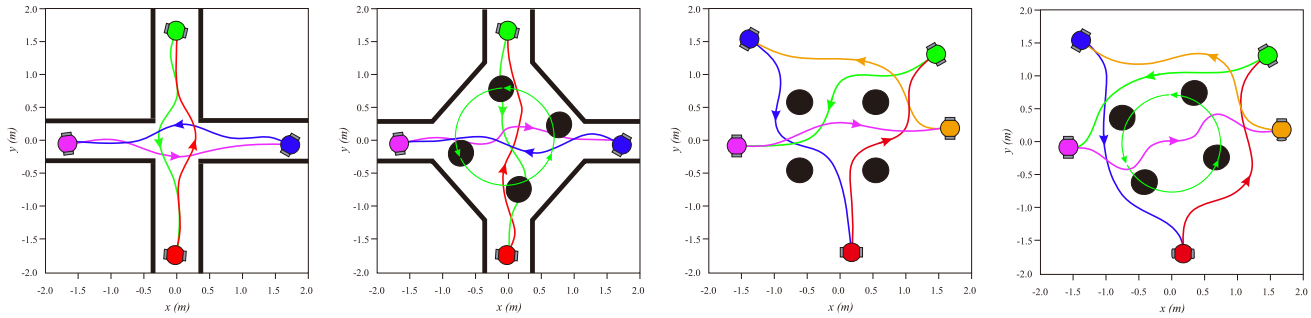


Fig. 3: The test scenarios for evaluating the approach: (a) Scenario 1 with four robots; (b) Scenario 2 featuring a moving black obstacle; (c) Scenario 3 involving five robots and static obstacles; (d) Scenario 4 comprising five robots and dynamic obstacles.

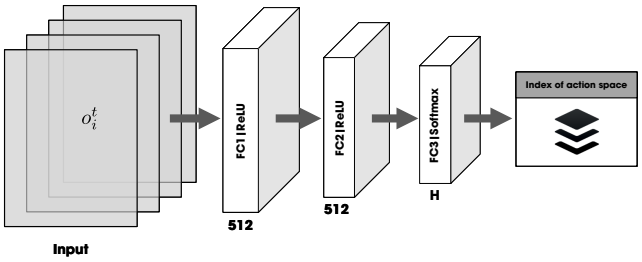


Fig. 4: The actor network architecture for training takes K-dimensional observation states.

Scenarios	Method	Success Rate	Avg. Speed (m/s)	Avg. Trajectory (m)
Scenarios 1	DWA-RL	1.0	0.17	4.32
	DRL	0.92	0.2	5.16
Scenarios 2	DWA-RL	0.72	0.15	4.52
	DRL	0.42	0.19	5.56
Scenarios 3	DWA-RL	1.0	0.18	4.42
	DRL	0.96	0.21	5.32
Scenarios 4	DWA-RL	0.68	0.15	4.76
	DRL	0.36	0.18	5.92

TABLE I: The performance of the proposed approach is compared with Wen et al.'s method [12].

the training process. Upon successful training in the static environment, this model is utilized to enhance perception in the second environment, as illustrated in Fig. 1b. In the second stage, four static obstacles are transformed into dynamic obstacles as they follow circular trajectories. This approach aims to improve perception in dynamic environments and enhance its ability to handle complex situations.

The performance of the proposed method is evaluated across four different scenarios, conducted over 50 trials. Furthermore, the proposed method is compared to the approach presented by Wen et al. [12], using three evaluation metrics:

- **Success Rate:** This metric determines the percentage of successful goal achievements without collisions with obstacles and other agents to the total number of robots.
- **Average Speed:** This parameter calculates the average speed of all robots in the system.
- **Average Trajectory Length:** This metric indicates the average distance traveled during the trials required to reach the goal.

The comparative results are presented in Table I. It can be observed that both methods achieve high success in scenarios with static obstacles. However, in environments with dynamic obstacles, the proposed approach exhibits significantly fewer collisions. This is attributed to the method of observing the motion of obstacles in velocity space, coupled with information about the positions of agents obtained from DWA. Moreover, the application of velocity space constraints contributes to a reduction in the trajectory length compared to DRL, despite having a lower average speed. The reason behind this lies in the abrupt changes in angular and linear velocity in the DRL approach to avoid obstacles and agents without control within the allowed velocity space, causing oscillations in the movement process. The trajectories of agents in four different environments are also illustrated in Fig. 3. Notably, in environments with static obstacles, the robots chose the shortest paths while ensuring obstacle avoidance and interaction with other agents. In contrast, in environments with dynamic obstacles, the robots prioritize paths that facilitate effective obstacle avoidance over the shortest possible routes.

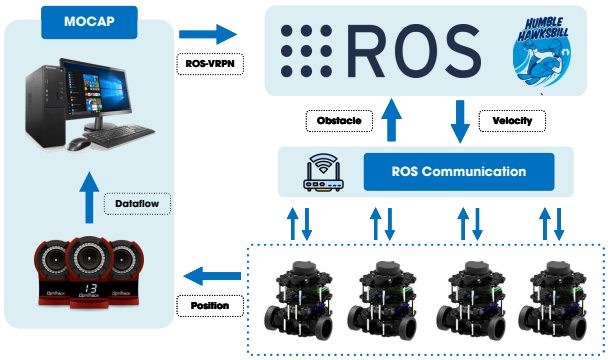
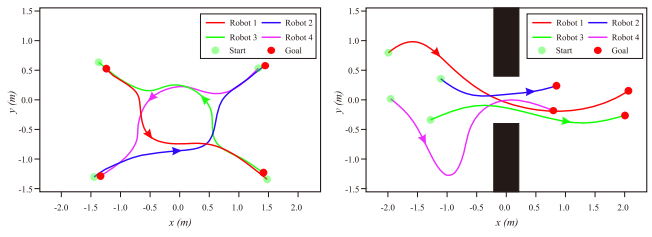


Fig. 5: The connection of hardware in the experiment.

B. Experiment Results

We conduct experiments in a real-world environment with four Turtlebot Burger robots to ensure sim-to-real capability with two scenarios. The experimental setup and system

connectivity details can be found in Fig. 5. During the experiment, the position and velocity of the robots are captured using a motion capture system (MOCAP). The motion tracking data, sampled at a frequency of 120Hz, is then transmitted to the navigation frame on ROS Humble. Within this framework, decisions based on local navigation information are made, and control velocities are communicated to the Burger robots. Initially, we deploy four robots to traverse diagonally towards intersecting positions in an obstacle-free setting. In the second scenario, the objective is for four robots to navigate through a narrow environment, with interspersed goal locations to evaluate the performance of the local planner. The motion trajectories for the first and second scenarios are illustrated in Fig. 6a and 6b, respectively. In the first scenario, the robots demonstrate adaptability by achieving their goals and adjusting their trajectories sequentially to avoid conflict along the way. In the second scenario, despite operating in a narrow space, the motion trajectories of the robots revealed collaborative interactions to prevent collisions while ensuring they reached their destinations. Notably, in the trajectory of the fourth robot, there is a noticeable divergence in the movement path right from the beginning. This divergence occurred as the robot selected optimal paths toward the goal. However, during movement, it perceives the positions of the first and second robots, prompting it to move farther away. Furthermore, influenced by the reward function, the robot did not come to a halt to wait for others but rather continued moving towards the goal.



(a) (b)

Fig. 6: The trajectory of the robots using the proposed local planner in the experiment: (a) Scenarios 1; (b) Scenarios 2

Moreover, the results also demonstrate that our approach can be applied in real-world environments with zero-shot transfer, whereas the method employing DRL is affected by observational differences between the training simulation space and the real world.

VI. Conclusion

In this paper, we presented a novel approach to local path planning for distributed multi-robot systems through the combination of DWA and DRL using the PPO algorithm. By utilizing velocity space and cost functions for observation space, the local planner guaranteed the generation of kinematically feasible robot velocities for the multi-robot system. The proposed method was validated in both simulated and experiments, showcasing the ability of multiple robots to collaborate in navigating without collisions and achieving goals efficiently. Future work encompasses conducting experiments in more diverse environments with an increased number of robots to comprehensively evaluate the versatility of the approach. Moreover, we will investigate the estimation of motion behaviors of other agents as dynamic obstacles to reduce communication overhead in the network.

References

[1] Z. Liu, H. Wang, W. Chen, J. Yu, and J. Chen, "An incidental delivery based method for resolving multirobot pairwised transportation problems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1852–1866, 2016.

[2] J.-S. Peng, V.-T. Ngo, and Y.-C. Liu, "Toward distributed control of networked omnidirectional automated ground vehicles in complex environment with delay communication," *Control Engineering Practice*, vol. 134, p. 105464, 2023.

[3] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, p. 120254, 2023.

[4] R. Smierzchalski and Z. Michalewicz, "Path planning in dynamic environments," in *Innovations in Robot Mobility and Control*. Springer, 2005, pp. 135–153.

[5] M. Govindaraju, D. Fontanelli, S. S. Kumar, and A. S. Pillai, "Optimized offline-coverage path planning algorithm for multi-robot for weeding in paddy fields," *IEEE Access*, 2023.

[6] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE international conference on robotics and automation*. IEEE, 1994, pp. 3310–3317.

[7] P. K. Mohanty, A. K. Singh, A. Kumar, M. K. Mahto, and S. Kundu, "Path planning techniques for mobile robots: A review," in *International Conference on Soft Computing and Pattern Recognition*. Springer, 2021, pp. 657–667.

[8] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835–849, 2015.

[9] Y.-C. Liu and C.-Y. Huang, "Ddpg-based adaptive robust tracking control for aerial manipulators with decoupling approach," *IEEE Transactions on Cybernetics*, vol. 52, pp. 8258–8271, 8 2022.

[10] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6057–6063.

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

[12] S. Wen, Z. Wen, D. Zhang, H. Zhang, and T. Wang, "A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning," *Applied Soft Computing*, vol. 110, p. 107605, 2021.

[13] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[14] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2005, pp. 430–435.

[15] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Autonomous Robots*, vol. 32, pp. 385–403, 2012.

[16] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[17] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.

[18] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

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