electronics

MDPI

Article

Navigation Based on Hybrid Decentralized and Centralized Training and Execution Strategy for Multiple Mobile Robots Reinforcement Learning

Yanyan Dai, Deokgyu Kim and Kidong Lee *

Robotics Department, Yeungnam University, Gyeongsan 38541, Republic of Korea; yanyan_dai@ynu.ac.kr (Y.D.)
* Correspondence: kdrhee@yu.ac.kr

Abstract: In addressing the complex challenges of path planning in multi-robot systems, this paper proposes a novel Hybrid Decentralized and Centralized Training and Execution (DCTE) Strategy, aimed at optimizing computational efficiency and system performance. The strategy solves the prevalent issues of collision and coordination through a tiered optimization process. The DCTE strategy commences with an initial decentralized path planning step based on Deep Q-Network (DQN), where each robot independently formulates its path. This is followed by a centralized collision detection the analysis of which serves to identify potential intersections or collision risks. Paths confirmed as non-intersecting are used for execution, while those in collision areas prompt a dynamic re-planning step using DQN. Robots treat each other as dynamic obstacles to circumnavigate, ensuring continuous operation without disruptions. The final step involves linking the newly optimized paths with the original safe paths to form a complete and secure execution route. This paper demonstrates how this structured strategy not only mitigates collision risks but also significantly improves the computational efficiency of multi-robot systems. The reinforcement learning time was significantly shorter, with the DCTE strategy requiring only 3 min and 36 s compared to 5 min and 33 s in the comparison results of the simulation section. The improvement underscores the advantages of the proposed method in enhancing the effectiveness and efficiency of multi-robot systems.

Keywords: multiple-robot navigation; hybrid DCTE strategy; reinforcement learning; DQN; effectiveness and efficiency

check for updates

Citation: Dai, Y.; Kim, D.; Lee, K. Navigation Based on Hybrid Decentralized and Centralized Training and Execution Strategy for Multiple Mobile Robots Reinforcement Learning. *Electronics* **2024**, *13*, 2927. https://doi.org/10.3390/electronics13152927

Academic Editors: Chao Zhang, Wentao Li, Huiyan Zhang and Tao Zhan

Received: 28 June 2024
Revised: 19 July 2024
Accepted: 23 July 2024
Published: 24 July 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Navigation for multiple mobile robots is an attractive research topic, with significant implications across various fields, including autonomous vehicles [1,2], robotics [3,4], and logistics [5,6]. Multiple robots can work together to accomplish complex tasks more efficiently than a single robot. Multiple robots can significantly increase efficiency and scalability in environments such as warehouses or manufacturing plants. They can be deployed to handle different tasks simultaneously, from transporting goods to assisting in assembly processes. The multi-robot systems offer greater resilience and redundancy than a single robot system. Integrating machine learning and artificial intelligence with multi-robot systems opens new avenues for autonomous decision making and adaptive learning. Robots can learn from their environment and each other, optimizing their strategies and behaviors for improved performance over time.

Navigation for multiple mobile robots using reinforcement learning (RL) [7–12] involves teaching robots how to optimally move through their environment and interact with each other to achieve goals. This approach leverages the principles of RL, where a robot learns to make decisions through trial and error, guided by rewards for achieving desired outcomes. Implementing RL in the multi-robot system is complex due to the dynamic interactions among robots and their environment. It needs sophisticated algorithms to handle coordination, communication, and obstacle avoidance. In RL, the environment

includes everything that the robots interact with, including other robots, obstacles, and the target location. It is typically modeled as a state space that the robots can observe and act upon. Each robot is considered a robot with the ability to perceive the state of the environment through sensors, make decisions based on its policy, and execute actions to change its state or the environment. The state is the situation of a robot in the environment. In this paper, it includes the robot's location and the positions of other robots and obstacles. Actions are the set of movements a robot can make, such as moving forward, moving backward, turning left, turning right, or stopping. The choice of action at each step is based on the robot's policy. Rewards provide feedback to the robot for arriving at its goal point. Penalties are given for collisions. The policy is a strategy that the robot follows to decide its actions based on the current state. The goal of RL is to learn an optimal policy that maximizes the expected cumulative reward.

Applying RL to multiple robots' navigation is important, since it enhances the capabilities and efficiency of multi-robot systems. RL enables robots to learn optimal paths in complex environments. Using trial and error, robots can discover the most efficient routes to their destinations while avoiding obstacles. This is particularly important in dynamic environments where predefined paths may not be feasible [13]. RL allows robots to make autonomous decisions based on real-time environmental data. This reduces the need for constant human supervision and intervention, making the system more efficient and scalable [14]. Robots can adapt their strategies based on the environment and the behavior of other robots. This adaptability is crucial in dynamic settings where conditions and tasks can change rapidly. RL helps robots to continuously improve their performance over time [15]. In multi-robot systems, RL facilitates coordination and collaboration among robots. Robots can learn to work together to achieve common goals, such as avoiding collisions and optimizing task allocation. This leads to higher overall system efficiency [16]. Therefore, applying RL to multiple robots' navigation is essential for developing intelligent, efficient, and adaptable robotic systems.

The strategy of applying RL in multi-robot systems is of great importance. A well-defined RL strategy ensures that robots can work together well, optimizing their collective performance and achieving complex tasks [17]. Strategies that balance centralized and decentralized learning can optimize the use of computational resources [18]. A scalable RL strategy allows for the deployment of numerous robots in various environments, from small-scale operations to large, complex settings [19]. An effective RL strategy incorporates redundancy and adaptive learning, enabling robots to recover from failures and continue operations [20]. Therefore, this paper designs and applies a Hybrid Decentralized and Centralized Training and Execution (DCTE) Strategy of RL to solve complex path planning problems.

2. Related Works

There are several strategies for RL in multi-robot navigation. In centralized training with a decentralized execution strategy [21,22], robots are trained together in a centralized manner to learn how to coordinate their actions, but they execute their learned policies independently in the real world. This approach leverages the advantages of shared learning and optimizes the collective knowledge gained during training, while allowing for flexibility and individual decision making in execution. However, training multiple robots in a centralized manner can become computationally intensive as the number of robots increases, requiring significant processing power and memory. The complexity limits scalability and is not feasible for large-scale deployments. In Multi-Robot Reinforcement Learning [23], multiple robots learn and make decisions simultaneously. This strategy considers the interactions between robots in addition to the individual robot–environment interactions. This helps in developing policies that consider not just individual success but also the interdependencies between robots, essential for tasks requiring high coordination. However, the environment's dynamics can change as each robot learns and adapts, making the learning process inherently non-stationary. Algorithms may struggle to converge to a stable policy,

leading to suboptimal performance and longer training times. It can be difficult to determine which actions by which robots led to a particular outcome, especially when actions are taken simultaneously by multiple robots. This complicates the learning process and can slow down the development of effective strategies. About communication mechanisms strategy [24], establishing robust communication channels between robots allows them to share real-time data about their status and the environment, such as detected obstacles or positional data. However, maintaining real-time communication between robots can introduce significant overhead in terms of latency and bandwidth usage. In environments where split-second decisions are critical, delays in communication can lead to inefficiencies or errors. As the number of robots increases, the amount of data that needs to be exchanged can grow exponentially, making it challenging to manage effectively. This can limit the maximum number of robots that can effectively collaborate within a system. The reward shaping and shared rewards strategy [25,26] is designed to motivate not only individual achievements but also collaborative efforts. Shared rewards promote teamwork, whereas individual rewards focus robots on personal efficiency and task completion. Such a dual reward system fosters an environment where robots learn to balance personal objectives with group objectives, enhancing the overall system performance. However, designing reward functions that accurately reflect both individual and collective goals is challenging and can often require extensive tuning and domain knowledge. Poorly designed rewards can lead to undesired behaviors, where robots exploit loopholes in the reward function rather than performing the intended task. If the balance between individual and shared rewards is not well managed, robots might either become too selfish, only optimizing their own performance, or too altruistic, where they fail to effectively optimize their own tasks. This can lead to inefficient overall performance and failure to achieve the collective optimally goal.

The contribution of this paper is described as below. In order to address the disadvantages discussed above, a Hybrid Decentralized and Centralized Training and Execution (DCTE) Strategy is proposed. It provides an effective method for solving complex path planning problems. This strategy adeptly handles collision and coordination issues common in multi-robot environments. The main idea of this method is to reduce computational complexity and improve the system efficiency through staged optimization. If implementing multi-robot navigation, the algorithm complexity and computation time will affect the system's scalability. This strategy can reduce computational complexity, making it feasible to scale up the system. Simulation comparison experiments show that, with this strategy, the computation time for robots to find safe paths is significantly reduced. The strategy implementation includes six steps: initial decentralized path planning, central collision detection, dynamic planning in collision areas, path re-optimization and connection, and execution. In the initial decentralized path planning step, each robot independently learns the optimal path from the starting point to the destination considering static obstacles within a static environment based on DRL [27,28]. In the central collision detection step, after all robots find their optimal paths, the central system performs path analysis to detect potential intersection or collision risks between these paths. If the path of a robot does not intersect with the paths of other robots, the path is confirmed to be safe and can be directly used for execution. At the dynamic planning in collision areas step, if path intersections are detected, the areas where these intersections occur will be marked as collision areas. Robots in these areas need to re-plan their paths and consider other robots as dynamic obstacles to avoid obstacles. During the re-learning process, the confirmed safe path will set an expanded safety area around it, prohibiting other robots from entering, and ensuring the security of the execution path. At the path re-optimization and connection step, once the avoidance paths are determined, they are connected to the initial safe paths to form a complete, secure route for execution. At the execution step, multiple robots will track the optimized paths and detect the environment again. If there are new obstacles, the implementation will return to step 1. This strategy effectively allocates computing resources and ensures the security of the execution process by first optimizing independently and then

adjusting centrally. This hybrid strategy shows a high degree of adaptability in dealing with static and dynamic obstacles, and can flexibly respond to complex and changeable actual environments. The extensive experimental validation of the proposed strategy is conducted using the Gazebo Simulator and Agilex LIMO robots equipped with onboard LiDAR and camera sensors. The results confirm the advantages of the DCTE strategy, showcasing reduced reinforcement learning time (3 min and 36 s) compared to traditional methods (5 min and 33 s) and demonstrating efficient path planning and execution.

3. Hybrid Decentralized and Centralized Training and Execution Strategy for Multiple Mobile Robots Reinforcement Learning

This section will introduce the hybrid decentralized and centralized training and execution (DCTE) strategy for multiple-mobile-robots reinforcement learning (MRRL). In MRRL, robots must cooperate to achieve optimal collective behavior. Hybrid training and execution strategies combine decentralized and centralized approaches to optimize learning efficiency and strategy implementation. The hybrid DCTE strategy includes six steps: initial decentralized path planning based on DQN, central collision detection, dynamic planning in collision areas, path re-optimization and connection, execution, and checking whether there are new obstacles or not in the environment. The processing flowchart is shown in Figure 1.

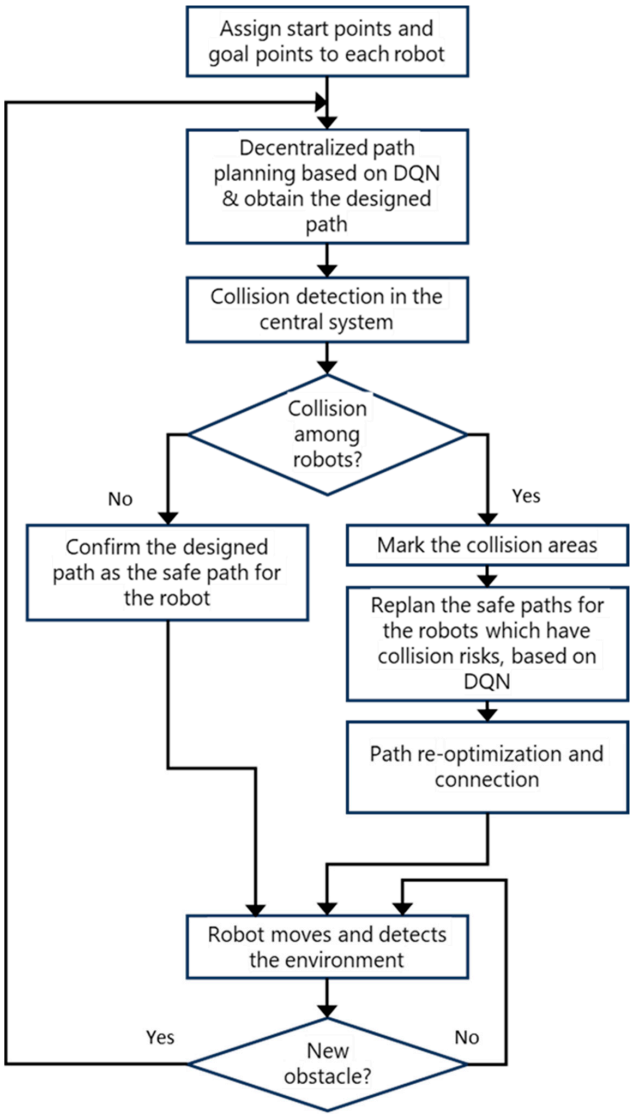


Figure 1. Hybrid DCTE strategy processing flowchart.

Initially, each robot is assigned with the start point and goal point. In the decentralized path planning step, each agent independently learns to find the optimal path from the starting point to the goal point, avoiding static obstacles, based on DRL. The states of the robot are the inputs of the neural network. There are two layers of the neural network. The first layer has 10 neurons, and the second layer has 4 neurons. The outputs of the neural network are the values of all actions. Each robot selects the action with the highest value among its possible actions (up, down, left, and right). After learning, the optimal safe path is obtained for each robot, and sent to the central system. The central system checks whether there is a collision risk or not among the robots. As shown in Figure 2, the black blocks represent static obstacles. After the decentralized path planning step, the red robot, blue robot, yellow robot, and orange robot obtain the solid black path, dash black path, dash–dot black path, and dot black path, separately. The arrow directions represent the moving directions. The yellow robot's path does not intersect with any other paths, so it is a safe path. Although the dot black path and the solid black path do interact, there is no collision risk, because the orange robot passes firstly. Therefore, the orange robot's dot path is confirmed to be safe and can be directly used for execution. However, the solid path and the dash path do interact, and the red robot and the blue robot also meet each other in the green block. Therefore, the green block is the collision area. The path in the collision area should be replanned.

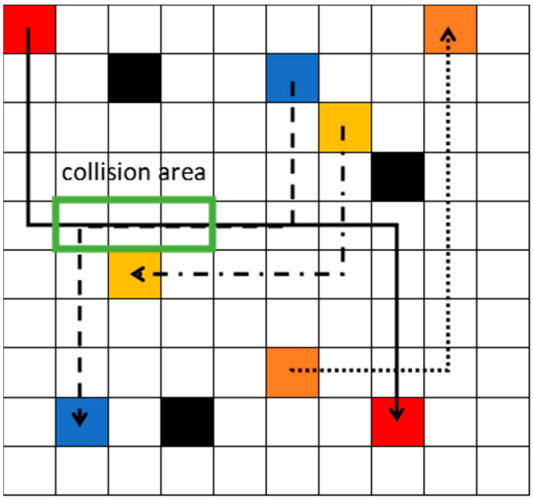


Figure 2. Three robots obtained optimal path after decentralized path planning step.

As shown in Figure 2, the collision area is marked. Robots in the collision area need to re-plan their paths. The robots should consider other robots as dynamic obstacles to avoid obstacles. As shown in Figure 3, the blue robot's start point (x_{bs}, y_{bs}) and goal point (x_{be}, y_{be}) are assigned to the robot. The red robot's start point (x_{rs}, y_{rs}) and goal point $(x_{\Re}, y_{\Re})$ are given to the robot. Because the path for the yellow robot is confirmed, the safe path of the yellow robot, which is marked as black blocks in Figure 3, will set as the static obstacles for the red and the blue robot. The blue and red robot should learn and find the optimal paths, avoiding static obstacles and collision with each other, based on [22]. Each robot has five possible actions: up, down, left, right, and stop. The Deep Q-Network (DQN) receives inputs based on the states of multiple robots. It then estimates the Q-value for each possible action of the robots. Each robot selects the action a associated with the highest Q-value, and then dispatches these actions to all robots. The function $f(a, n, k)$ is used to calculate the specific action a_k for robot $k(k = 1, 2, 3, \ldots)$. a is the selected action with the highest Q-value. $n(n \geq 2)$ is the number of the robot. For example. If there are three robots, then n is three. k (1,2,3, ...) is the index number of the robot. The pseudo code is used to calculate a_k as Algorithm 1, where 5 means that the robot has four possible actions.

Algorithm 1. Function: : $f(a, n, k)$ is used to calculate the specific action a_k for a robot.

Define a as the action with the highest Q-value.
Define n (n $\geq$ 2) as the number of the robot.
Define k (1, 2, 3, . . .) as the index number of the robot.
Define one variable **base**, which equals the integer part of the equation $a/5^{n-1}$.
if k equals n:
  a_k equals **base**.
else if k equals n $-$ 1:
  a_k equals the integer part of **base**/5.
else if k equals n $-$ 2:
  get the integer part of base/5 as **base1**.
  a_k equals the reminder part of **base1**/5.
else if k equals n $-$ 3:
  get the integer part of base/5 as **base1**.
get the reminder part of **base1**/5 as **base2**.
a_k equals the integer part of **base2**/5.

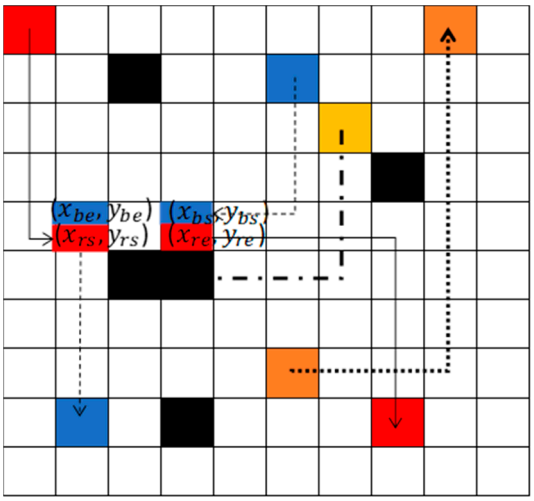


Figure 3. Red robot and blue robot re-planned the trajectory to avoid the collision.

The robot executes the selected action and observes the new state and reward. If the robot reaches the goal point, +1 rewards is obtained. If the robot moves to a free space without any incidents, 0 reward is obtained. If the robot either collides with the other robot, an obstacle, or returns to its starting point, -1 reward is obtained. The total reward r for all robots at each step is calculated by summing the individual rewards rwd_k:

$$r = \sum_{k=1}^{n} rwd_k \tag{1}$$

In the learning process, two neural networks are utilized: the evaluation network and the target network. The evaluation network is updated continuously and is used to predict the Q-values for current states and actions. The target network is a periodic copy of the evaluation network and is used to calculate the target Q-values. Each network has two layers. The first layer has 10 neurons, and the second layer has 4^n neurons. The method uses prioritized experience replay where the absolute temporal difference error is recorded in memory. It is calculated as the difference between the values from the target network and the evaluation network. A larger temporal difference error indicates significant room for improvement in prediction accuracy, thereby assigning higher priority to those samples for further learning. The formula used to train the evaluation network, representing the target Q-value, is given by:

$$Y_j = r_j + \gamma max\hat{Q} \tag{2}$$

r_j is the reward obtained after all robots taking actions $a_k(k = 1, 2, 3, \ldots\ldots)$ in state s_j. s_j is a single array stacking all robots' states. s_{j+1} is the resulting state after taking the action. $\gamma = 0.9$ is the discount factor, which determines the importance of future rewards. $max\hat{Q}$ represents the maximum predicted Q-value for the next state using the target network's parameters. a' represents the next action that maximizes the Q-value for the new state s_{j+1}. θ_j^- represents the parameters of the target network at step j.

After replanning the path, the optimal paths are shown in the green block in Figure 4. The solid black line and dash black are the paths for red robot and blue robot separately. The path re-optimization and connection step will be processed. These paths will be connected to the initial safe paths, obtained from decentralized path planning based on the DQN step, to form a complete, secure route for execution, as shown in Figure 5. Then, each robot will track the path and detect the environment. If new obstacles are detected, it will return to the step of decentralized path planning based on DQN.

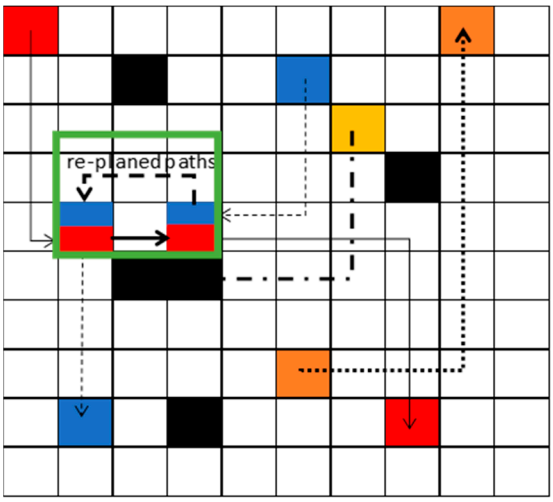


Figure 4. After replanning the path, the optimal paths are shown in the green block.

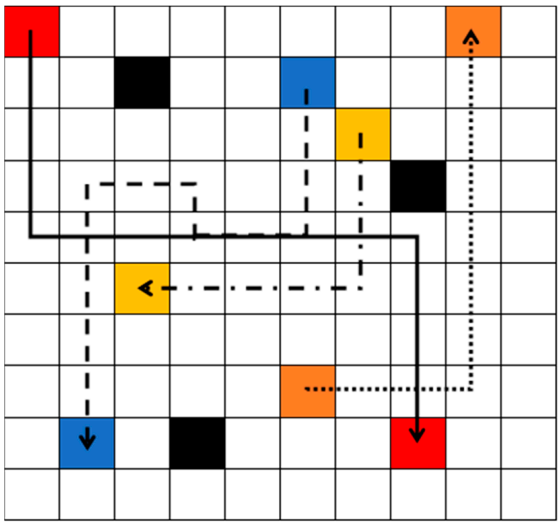


Figure 5. Path re-optimization and connection.

4. Simulations

Based on the proposed Hybrid DCTE strategy, three robots plan and track the safe path from start points to goal points. Initially, three robots plan the path based on DQN separately. There are four actions: forward, backward, left, and right. The parameters for the algorithm are given with a learning rate of 0.01, a reward decay of 0.9, an epsilon-greedy policy with an epsilon set to 0.9, and a target network replacement iteration of 200. The memory size and batch size were set to 2000 and 32. In Figure 6, the unit of each grid is 1 m. The black blocks represent the obstacles. The starting points for robots 1, 2, and 3 are marked by red, blue, and gray blocks, respectively. The goal points for robots 1, 2, and 3 are marked by yellow, green, and purple blocks, respectively. The red path is the planed path for robot 1. The blue path presents the planed path for robot 2. The gray path represents the planned path for robot 3. Since robots do not have the stop action to efficiently reach their goal points, robot 1 and robot 3 have a collision risk between each other. The risk area is shown in the green block.

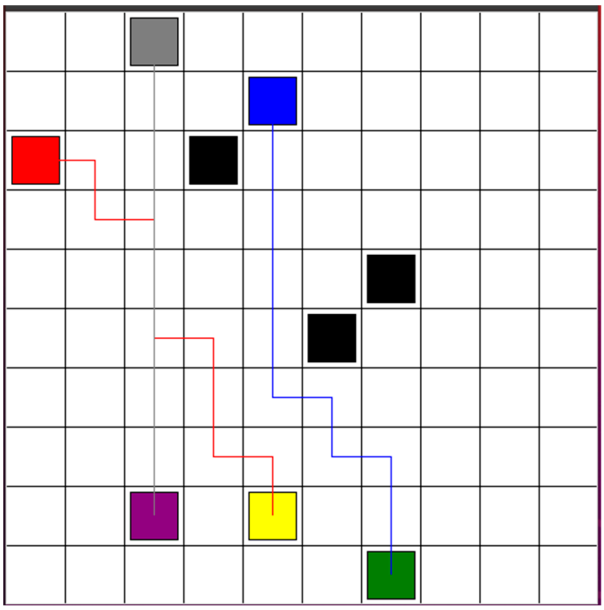


Figure 6. Three robots plan a path after decentralized path planning step, based on DQN.

Robot 1 and robot 3 replanned the optimal paths simultaneously, based on DQN RL algorithm. The learning rate, reward decay, and e-greedy were 0.01, 0.9, and 0.9, respectively. The target network is updated every 200 iterations. The memory size and batch size were set to 2000 and 32. The number training steps is 16,000 steps. There are five actions for each robot: forward, backward, left, right, and stop. In Figure 7, the red and gray blocks mark the starting points for robots 1 and 3, respectively, while the yellow and purple blocks designate their goal points, respectively. The black blocks present the static obstacles. The robot is recognized as the dynamic obstacle for the other robot. From the start point, robot 1 moves backward first and then move forward. The robot 3 moves forward directly.

After the path re-optimization and connection step, the optimized paths for robots 1, 2, and 3 are shown in Figure 8. There is no collision risk. If there is no new obstacle, three robots track the path to the goal points. If new obstacles are detected, robots will return to the step of decentralized path planning based on DQN. In total, the reinforcement learning time is 3 min and 36 s for three robots.

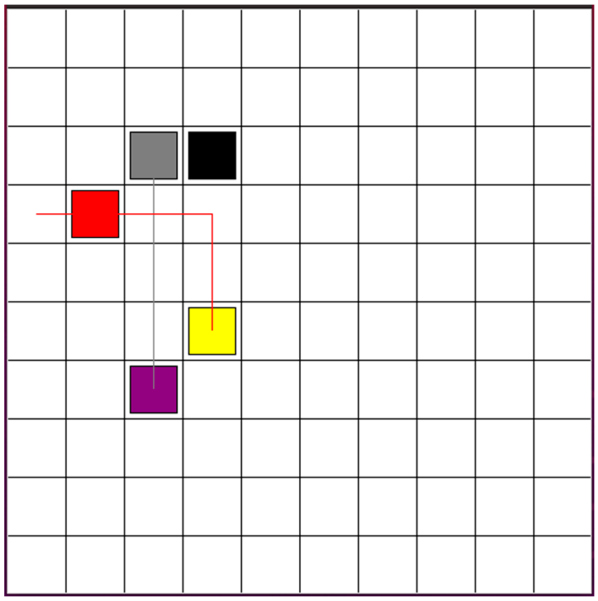


Figure 7. Robot 1 and robot 3 replan the optimal paths simultaneously, based on the DQN RL algorithm.

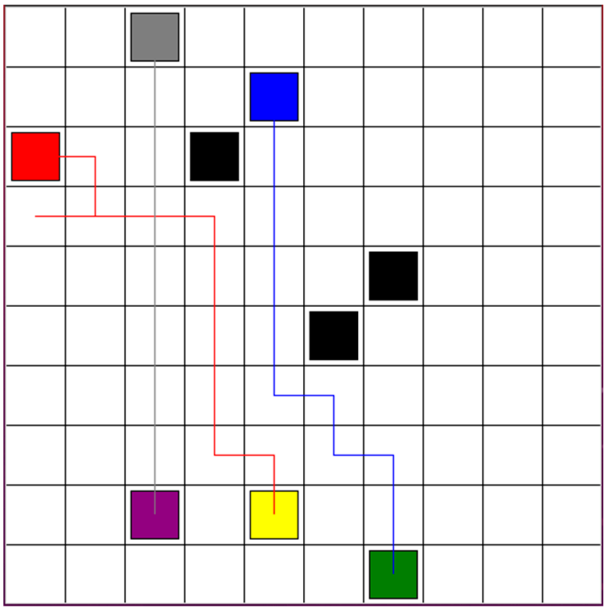


Figure 8. After path re-optimization and connection step, the optimized paths for robot 1, 2, and 3, respectively.

To illustrate the comparison between the hybrid DCTE strategy and the DQN-based approach from ref [22], we present the result where three robots are exploring safe paths in an environment, as depicted in Figure 6. Figure 9 presents the comparative results. The red, blue, and grey lines represent the safe path for robots 1, 2, and 3, separately. The paths are more complex and longer than the paths in Figure 9. In total, the reinforcement learning time is 5 min and 33 s for three robots.

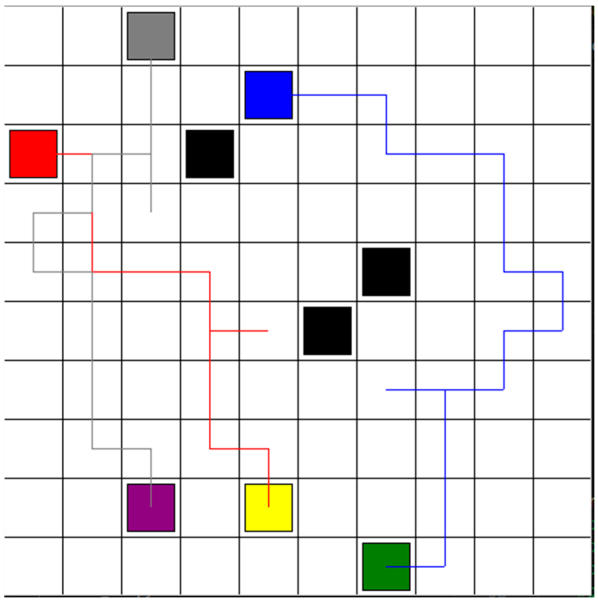
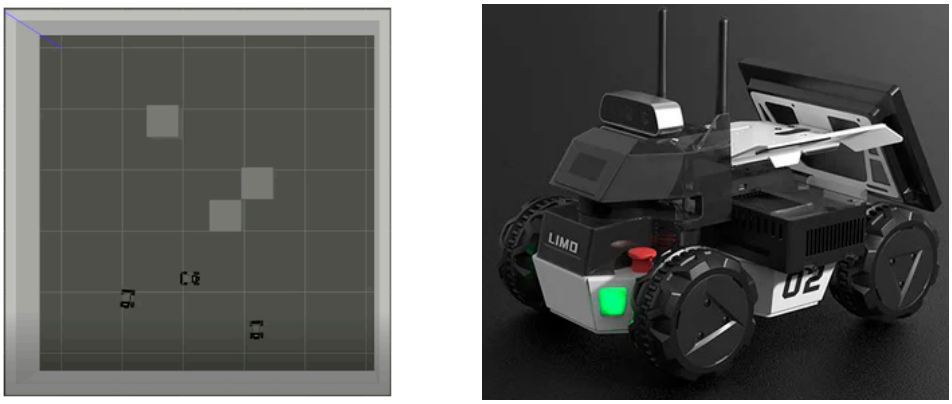


Figure 9. Optimal paths for robots 1, 2, and 3, based on DQN-based approach from ref [22].

The proposed hybrid DCTE strategy for multi-robot navigation was evaluated using the Gazebo Simulator. The simulation environment, consistent with the environment shown in Figure 10a, was designed to rigorously test the algorithm's effectiveness. LIMO robots from Agilex Robotics, as shown in Figure 10b, is equipped with onboard LiDAR and camera sensors. The obstacles are represented by gray boxes. Each cell measured 2*2 m. Three robots are added to the Gazebo simulation environment. Each robot is assigned a goal point. Robots applied the proposed DCTE Strategy to explore optimal paths. Robots subscribed to the "/odom" topic to acquire their x and y positions and the "/imu" topic to obtain the yaw angle value. Robots published the "/cmd_vel" topic to control movement using linear and angular velocities. Robots use linear velocity 0.02 m/s for forward actions, Robots use linear velocity −0.02 m/s for backward action. Robots calculate angular velocity, based on the difference of the angle between two waypoints and the robot's yaw angle.



(**a**) (**b**)

Figure 10. (**a**) Gazebo simulator environment; (**b**) Agilex LIMO robot.

Figure 11 demonstrates the safe path exploration and tracking process. At t = 0, robots initiate safe path exploration. At t = 39 s, three robots track their safe paths. At t = 66 s, robot 1 turns left to avoid collision with robot 3. At t = 96 s, robot 1 moves backward to track its optimal path. At t = 133 s, robot 2 avoids the static obstacle. At t = 162 s, three robots track their safe paths. At t = 201 s, robot 3 reaches its goal point. At t = 252 s, robot 2 arrives at its goal point. At t = 308 s, robot 1 finally reaches its goal point. The video can be

seen in the link: https://www.dropbox.com/scl/fi/vq6tf7onj66vv9fc4625x/gazebo-result-DCTE-strategy.webm?rlkey=8hz2o8zn3fx9p7k5tqsx2lk9q&st=hxk66req&dl=0 (accessed on 26 June 2024).

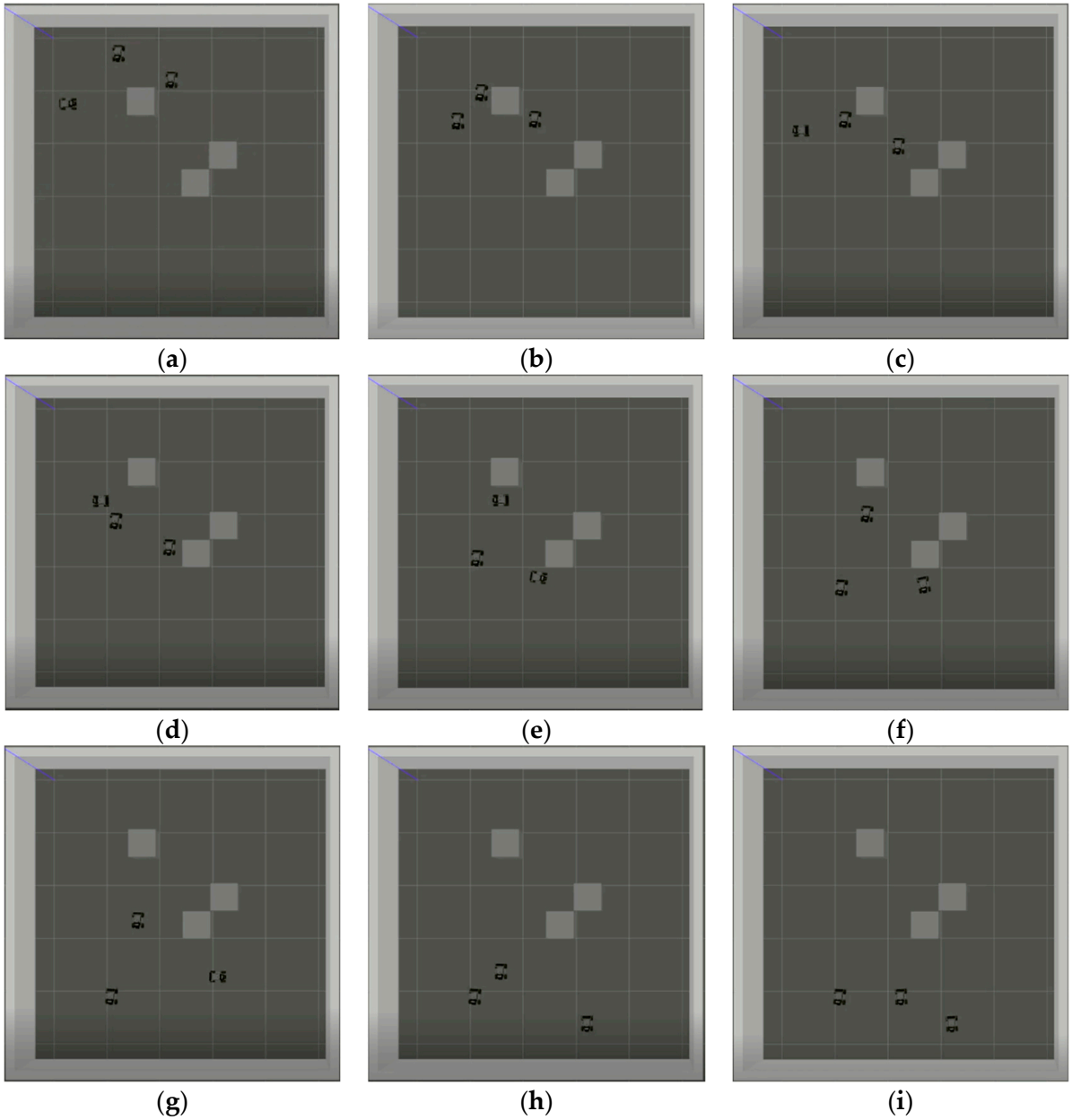


Figure 11. Three robots explore and track the optimal paths, based on a hybrid DCTE strategy: (**a**) t = 0 s; (**b**) t = 39 s; (**c**) t = 66 s; (**d**) t = 96 s; (**e**) t = 133 s; (**f**) t = 162 s; (**g**) t = 201 s; (**h**) t = 252 s; (**i**) t = 308 s.

To validate the effectiveness of the proposed strategy, we deployed three robots in a more complex environment. By applying the hybrid DCTE strategy, the robots were able to reach their target points efficiently, quickly, and safely. In Figure 12, the robots need to avoid the black block obstacles. Robots 1 and 3 have the risk of colliding with each other and need to re-plan their optimal paths to avoid both static and dynamic obstacles.

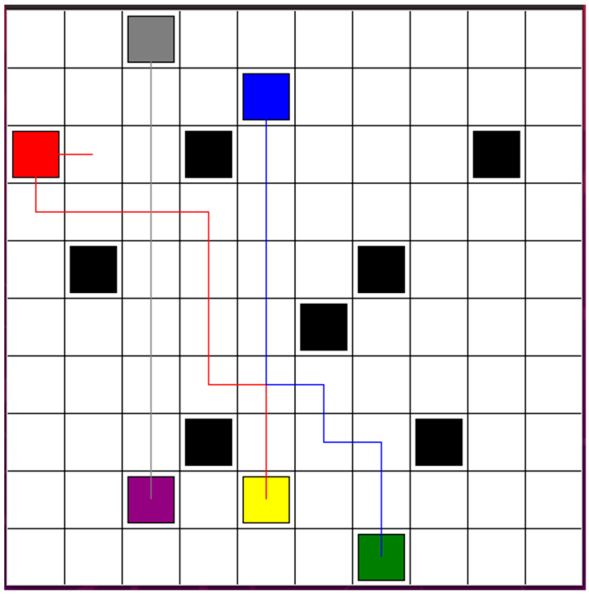


Figure 12. Three reach their target points efficiently, quickly, and safely, in a complex environment.

The complexity of the proposed control algorithm in real-time implementation can be discussed through theoretical analysis based on existing research and simulation results. The proposed Hybrid Decentralized and Centralized Training and Execution (DCTE) Strategy includes decentralized path planning and dynamic re-planning phases, which are expected to require substantial computational power. Each robot's Deep Q-Network (DQN) involves multi-layer neural network computations, typically necessitating powerful CPUs or GPUs. In large-scale deployments, distributed computing resources might be needed to ensure real-time performance. The DQN requires the storage of extensive training data, replay buffers, and neural network parameters, leading to significant memory usage. Especially when handling multiple robots, the memory requirements will grow exponentially. Based on the structure and training process of DQNs, the time complexity of the decentralized path planning and dynamic re-planning phases mainly depends on the number of robots and the complexity of the environment. In practical applications, the algorithm needs to complete path planning and adjustments within milliseconds. Although DQNs can theoretically achieve efficient computations, its actual performance will be influenced by hardware conditions and environmental complexity. The real-time processing of LiDAR sensor data requires efficient data stream management and rapid computational capability to ensure minimal latency. Results from the Gazebo simulator can provide valuable references for practical applications [29,30]. Therefore, the DCTE strategy performs well in simulated environments, suggesting it could achieve similar outcomes in real-world scenarios. The goal of the proposed algorithm is to reduce the computational load, improve the real-time effectiveness, and enhance scalability. We believe that the algorithm has great potential for practical applications. Through theoretical analysis, we preliminarily validate the feasibility and potential complexity of the proposed DCTE strategy in real-time multi-robot navigation. Further practical experiments and optimizations will help comprehensively evaluate and enhance the performance of this algorithm.

5. Conclusions

This paper has introduced a novel Hybrid Decentralized and Centralized Training and Execution (DCTE) Strategy to address the complex challenges of path planning in multi-robot systems. The initial decentralized approach, based on Deep Q-Networks (DQN), allows each robot to independently find a path in a static environment. The centralized collision detection step ensures that only non-intersecting paths proceed to execution, while paths at risk of collision are dynamically replanned, considering other robots as dynamic

obstacles. Those robots in a collision area replan safe paths simultaneously, based on the DQN RL algorithm. After the re-optimization and connection step, all robots obtain the optimization paths. The hybrid nature of the DCTE strategy ensures that computing resources are allocated efficiently, with initial independent optimizations followed by necessary centralized adjustments. This approach not only maintains continuous operation but also adapts flexibly to both static and dynamic obstacles, ensuring robust performance in varied and unpredictable environments. Simulation results demonstrate that the DCTE strategy significantly improves the system performance by reducing the collision risks and optimizing path planning in a computationally efficient manner.

Author Contributions: Conceptualization, Y.D.; methodology, Y.D.; software, Y.D.; validation, Y.D.; formal analysis, Y.D.; investigation, Y.D.; resources, D.K.; data curation, Y.D.; writing—original draft preparation, Y.D.; writing—review and editing, Y.D. and K.L.; visualization, Y.D.; supervision, K.L.; project administration, K.L.; funding acquisition, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (RS-2024-00406796, HRD Program for Industrial Innovation).

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

DCTE	Decentralized and Centralized Training and Execution
DQN	Deep Q-Network
RL	Reinforcement Learning
MRRL	Multiple Mobile Robots Reinforcement Learning

References

1. Ort, T.; Paull, L.; Rus, D. Autonomous Vehicle Navigation in Rural Environments Without Detailed Prior Maps. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2040–2047. [CrossRef]
2. Trinh, L.A.; Ekström, M.; Cürüklü, B. Dependable Navigation for Multiple Autonomous Robots with Petri Nets Based Congestion Control and Dynamic Obstacle Avoidance. *J. Intell. Robot. Syst.* **2022**, *104*, 69. [CrossRef]
3. Yew, C.H.; Ksm, S.; Leong, W.; Hong, F.; Nur, A.R.; Nurul, A.; Raad, H. Development of collision avoidance system for multiple autonomous mobile robots. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1–15. [CrossRef]
4. Qin, J.; Du, J. Robust adaptive asymptotic trajectory tracking control for underactuated surface vessels subject to unknown dynamics and input saturation. *J. Mar. Sci. Technol.* **2022**, *27*, 307–319. [CrossRef]
5. Elsanhoury, M.; Mäkelä, P.; Koljonen, J.; Välisuo, P.; Shamsuzzoha, A.; Mantere, T.; Elmusrati, M.; Kuusniemi, H. Precision Positioning for Smart Logistics Using Ultra-Wideband Technology-Based Indoor Navigation: A Review. *IEEE Access* **2022**, *10*, 44413–44445. [CrossRef]
6. Wang, L.; Liu, G. Research on multi-robot collaborative operation in logistics and warehousing using A3C optimized YOLOv5-PPO model. *Front. Neurorobot.* **2024**, *17*, 1329589. [CrossRef] [PubMed]
7. Wen, S.H.; Wen, Z.T.; Zhang, D.; Zhang, H.; Wang, T. A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning. *Appl. Soft Comput.* **2021**, *110*, 107605. [CrossRef]
8. Marchesini, E.; Farinelli, A. Enhancing Deep Reinforcement Learning Approaches for Multi-Robot Navigation via Single-Robot Evolutionary Policy Search. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 5525–5531. [CrossRef]
9. Jestel, C.; Surmann, H.; Stenzel, J.; Urbann, O.; Brehler, M. Obtaining Robust Control and Navigation Policies for Multi-robot Navigation via Deep Reinforcement Learning. In Proceedings of the 2021 7th International Conference on Automation, Robotics and Applications (ICARA), Prague, Czech Republic, 4–6 February 2021; pp. 48–54. [CrossRef]
10. Escudie, E.; Matignon, L.; Saraydaryan, J. Attention Graph for Multi-Robot Social Navigation with Deep Reinforcement Learning. *arXiv* **2024**, arXiv:2401.17914.

11. Parnichkun, M. Multiple Robots Path Planning based on Reinforcement Learning for Object Transportation. In Proceedings of the 2022 5th Artificial Intelligence and Cloud Computing Conference (AICCC '22), Osaka, Japan, 17–19 December 2022; Association for Computing Machinery: New York, NY, USA, 2023; pp. 229–234. [CrossRef]
12. Wenzel, P.; Schön, T.; Leal-Taixé, L.; Cremers, D. Vision-Based Mobile Robotics Obstacle Avoidance with Deep Reinforcement Learning. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 14360–14366. [CrossRef]
13. Giusti, A.; Guzzi, J.; Cireşan, D.C.; He, F.L.; Rodríguez, J.P.; Fontana, F.; Faessler, M.; Forster, C.; Schmidhuber, J.; Di Caro, G.; et al. A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *IEEE Robot. Autom. Lett.* **2016**, *1*, 661–667. [CrossRef]
14. Nguyen, T.; Nguyen, N.D.; Nahavandi, S. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. [CrossRef] [PubMed]
15. de Zarzà, I.; de Curtò, J.; Roig, G.; Manzoni, P.; Calafate, C.T. Emergent Cooperation and Strategy Adaptation in Multi-Agent Systems: An Extended Coevolutionary Theory with LLMs. *Electronics* **2023**, *12*, 2722. [CrossRef]
16. Busoniu, L.; Babuska, R.; De Schutter, B. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2008**, *38*, 156–172. [CrossRef]
17. Orr, J.; Dutta, A. Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications: A Survey. *Sensors* **2023**, *23*, 3625. [CrossRef] [PubMed]
18. Torbati, R.; Lohiya, S.; Singh, S.; Nigam, M.; Ravichandar, H. MARBLER: An Open Platform for Standardized Evaluation of Multi-Robot Reinforcement Learning Algorithms. In Proceedings of the 2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), Boston, MA, USA, 4–5 December 2023; pp. 57–63. [CrossRef]
19. Zhang, Z.; Jiang, X.; Yang, Z.; Ma, S.; Chen, J.; Sun, W. Scalable Multi-Robot Task Allocation Using Graph Deep Reinforcement Learning with Graph Normalization. *Electronics* **2024**, *13*, 1561. [CrossRef]
20. Gu, S.; Kuba, J.G.; Chen, Y.; Du, Y.; Yang, L.; Knoll, A.; Yang, Y. Safe multi-agent reinforcement learning for multi-robot control. *Artif. Intell.* **2023**, *319*, 103905. [CrossRef]
21. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
22. Dai, Y.; Yang, S.; Lee, K. Sensing and Navigation for Multiple Mobile Robots Based on Deep Q-Network. *Remote Sens.* **2023**, *15*, 4757. [CrossRef]
23. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2022**, *55*, 895–943. [CrossRef]
24. Foerster, J.N.; Assael, Y.M.; Freitas, N.D.; Whiteson, S. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 5–10 December 2016.
25. Grzes, M. Reward Shaping in Episodic Reinforcement Learning. In Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems, São Paulo, Brazil, 8–12 May 2017.
26. Cipollone, R.; Giacomo, G.D.; Favorito, M.; Iocchi, L.; Patrizi, F. Exploiting Multiple Abstractions in Episodic RL via Reward Shaping. In Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; pp. 7227–7234.
27. Lee, M.R.; Yusuf, S.H. Mobile Robot Navigation Using Deep Reinforcement Learning. *Sensors* **2022**, *10*, 2748. [CrossRef]
28. Zeng, J.; Ju, R.; Qin, L.; Yin, Q.; Hu, C. Navigation in unknown dynamic environments based on deep reinforcement learning. *Sensors* **2019**, *19*, 18. [CrossRef] [PubMed]
29. Blum, P.; Crowley, P.; Lykotrafitis, G. Vision-based navigation and obstacle avoidance via deep reinforcement learning. *arXiv* **2022**, arXiv:2211.05243. [CrossRef]
30. Alborzi, Y.; Jalal, B.S.; Najafi, E. ROS-based SLAM and Navigation for a Gazebo-Simulated Autonomous Quadrotor. In Proceedings of the 21st International Conference on Research and Education in Mechatronics (REM), Cracow, Poland, 9–11 December 2020; pp. 1–5. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.