applied sciences

Article

Mapless Collaborative Navigation for a Multi-Robot System Based on the Deep Reinforcement Learning

Wenzhou Chen [1,†], Shizheng Zhou [1,†], Zaisheng Pan [1,2], Huixian Zheng [2] and Yong Liu [1,*]

1 Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China; wenzhouchen@zju.edu.cn (W.C.); 21632051@zju.edu.cn (S.Z.); panzs@zju.edu.cn (Z.P.)
2 Zhejiang Chipkong Technology Co., Ltd., Hangzhou 310000, China; zhenghuixian@nz-ic.com
* Correspondence: yongliu@iipc.zju.edu.cn
† These authors contributed equally to this work.

Received: 25 August 2019; Accepted: 24 September 2019; Published: 9 October 2019

Abstract: Compared with the single robot system, a multi-robot system has higher efficiency and fault tolerance. The multi-robot system has great potential in some application scenarios, such as the robot search, rescue and escort tasks, and so on. Deep reinforcement learning provides a potential framework for multi-robot formation and collaborative navigation. This paper mainly studies the collaborative formation and navigation of multi-robots by using the deep reinforcement learning algorithm. The proposed method improves the classical Deep Deterministic Policy Gradient (DDPG) to address the single robot mapless navigation task. We also extend the single-robot Deep Deterministic Policy Gradient algorithm to the multi-robot system, and obtain the Parallel Deep Deterministic Policy Gradient (PDDPG). By utilizing the 2D lidar sensor, the group of robots can accomplish the formation construction task and the collaborative formation navigation task. The experiment results in a Gazebo simulation platform illustrates that our method is capable of guiding mobile robots to construct the formation and keep the formation during group navigation, directly through raw lidar data inputs.

Keywords: multi-robot; collaborative navigation; deep reinforcement learning

1. Introduction

Autonomous navigation for mobile robots is one of the most practical and essential challenges in robotics. The navigation systems for mobile robots mainly rely on the localization in a given map and the decision-making system. The relative technique for localization is called Simultaneous Localization and Mapping (SLAM) [1,2], which can obtain the map of the environment and get the robot poses simultaneously. In addition, the corresponding decision-making system which consists of planning [3–7] and control [8–10] would generate a safety trajectory and control the mobile robot to follow it until reaching the goal. The decision-making system plays an important role to connect the preceding localization stage and the following manipulation stage. This paper provides a decision-making method to address the core problem of robot navigation. We particularly focus on dealing with the motion planning and control problems with a navigation method based on the deep reinforcement learning.

The traditional decision-making system of the mobile robot can be hierarchically structured into four components, the route planning, the behavioral decision-making, the motion planning and the robot control [11]. For indoor navigation tasks, the decision-making system can be simplified into the motion planning part and the navigation control part. The mobile robot is supposed to plan a trajectory from its current position to the target destination with the specific indoor environment. Then, the motion controller will guide the mobile robot precisely follow the trajectories to the target position. The required efficient motion planning strategies and stable motion controllers are mainly based on the mathematical computations and geometric relationships. Although useful in many situations, the applicability of traditional decision-making systems is limited by their flexibility and versatility. The complexity of the environment and the dynamic obstacles can both increase the computational efficiency and decrease the navigation performance. In addition, the errors of each step will be accumulated to the end. Furthermore, most of the traditional methods can't address the mapless navigation tasks in the complex environment. However, there are many practical situations in which the mobile robots can't obtain the accurate map of the environment and only have the relative position relationship between the mobile robots and the targets. Thus, we proposed an end-to-end policy module with the raw sensor inputs to address these issues in mapless navigation tasks.

The deep reinforcement learning techniques have been greatly developed and widely applied in various fields of study. This kind of technical training the agent is conducted through the interaction trajectories between the agent and the environment. Intuitively, the character of interactive learning is quite similar to humans. When we consider the mapless navigation tasks, with a given target value and the current position, one person is likely to find the target position heuristically with his intuition. If one person is informed about the relative position values or the polar relationship to the target, he can easily find the target position based on the prior knowledge and the basic navigation strategies in an indoor environment. Compared with the traditional navigation methods, one human has a more efficient way to navigate without computing the precise mathematical module of the unknown environment. In addition, the deep reinforcement learning technique paves the way to accomplish the robot mapless navigation tasks like humans. With the aim of teaching the robot to navigate like humans, this work presents an approach to training the robot to navigate with the human intuition. By utilizing the proposed deep reinforcement learning method, the policy module of the mobile robot can make decisions through the raw 2D lidar sensor data and navigate to the target position in an indoor environment.

When we extend the navigation tasks to the multi-robot system, the coordination between a group of mobile robots becomes more complex. Similarly, we analyze the human intuition to get the inspiration. If there are a group of humans navigating in the unknown indoor environment, they would communicate and collaborate with each other to attain the goal. There are several forms of communications such as sharing the experience and observations. For a multi-robot navigation system, the robots can share the sensor observations, the training data and the relative state parameters. Inspired by the analysis of human intuition, our work provides a new insight into the multi-robot collaborative navigation task with deep reinforcement learning. By sharing the training experience and observing the states of other robots, the multi-robot system can keep the formation during group navigation.

In our proposed method, we assume that the relative position values of the targets are easily acquirable for the robots via cheap localization solutions such as WiFi [12] and QR code [13]. By improving the classical deep reinforcement learning methods, the proposed algorithm can accomplish the single robot mapless navigation task with human intuition. We also extend the method into the multi-robot navigation cases with some useful training strategies, and then evaluate the performance in the simulation platform. Particularly, the contributions of this paper can be summarized as follows:

1. We improve the classical Deep Deterministic Policy Gradient (DDPG) to address the end-to-end single robot mapless navigation problems directly through the raw lidar data inputs.
2. The Parallel Deep Deterministic Policy Gradient (PDDPG) is proposed to accomplish the multi-robot formation construct and the formation keeping during collaborative navigation. We also evaluate the proposed methods in the Gazebo simulation platform.

This paper is organized as follows: Section 2 describes the deep reinforcement learning algorithms and their applications in robot navigation tasks. In Section 3, an overview of the proposed algorithm is presented. The methods for different situations and the training details are described separately. Section 4 presents the evaluations of the proposed methods in the simulated environment and analyzes the experiment results. Section 5 concludes this work.

2. Related Works

Benefiting from the development of the deep neural networks, the deep reinforcement learning techniques show great potential for solving the decision-making tasks. By deploying deep neural networks as powerful nonlinear function approximators, the deep reinforcement learning algorithms can handle the complex decision-making problems with high-dimensional state and action spaces.

2.1. Deep Reinforcement Learning

With the aim of maximizing the expected return of behaviors, deep reinforcement learning considers the agent learning a good policy by interacting with its environment. Mnih et al. [14] first kickstarted the revolution in deep reinforcement learning. They proposed the deep Q network (DQN) that could learn to play Atari 2600 games at a superhuman level only based on the image inputs. This work convincingly demonstrates that deep reinforcement learning agents can be trained with high-dimensional observations. The later research [15–20] updated these kinds of methods and promoted the performance in the subproblems of the Atari games.

The second standout success of the deep reinforcement learning was the AlphaGo [21]. The AlphaGo merged the supervised learning and the Monte Carlo Tree Search (MCTS) [22] techniques into the deep reinforcement learning framework, and defeated the world champion human in Go by learning from human knowledge. After the AlphaGo received widespread attention, the AlphaGo Zero [23] defeated the AlphaGo completely. Unlike utilizing the human knowledge in the training stage, the AlphaGo Zero mastered the game of Go through self-play. Furthermore, the researchers extended the AlphaGo Zero to the other board games and proposed the AlphaZero [24].

2.2. Deep Reinforcement Learning for Navigation

Besides the well-known works in Games, the DRL algorithms also have been successfully applied to a wide range of problems, such as manipulators [25] and mobile robots [26,27]. In this section, we discuss the research about the mobile robot navigation tasks.

With the progress of the deep learning techniques, the powerful representation capabilities shed a new light on learning control policies directly from raw sensor inputs with the reinforcement learning framework. In recent years, there have been lots of proposed methods to tackle the autonomous navigation tasks with deep reinforcement learning algorithms. To address the navigation problems in reinforcement learning ways, these methods formulate the navigation process as the Markov decision process(MDP) 111 or partially observable Markov decision process(POMDP), and stack the observations from sensor readings as the states. Then, the methods will find the optimal policy module that is capable of guiding the robot to the target position.

Kretzschmar et al. [28] and Pfeiffer et al. [29] used the maximum entropy inverse reinforcement learning (IRL) methods to learn interaction models for pedestrians from demonstration in occupied environments. Zhu et al. [30] used the image of the target object and the current observation as the input to the Siamese actor–critic model. Then, they formulated their task as a target-driven navigation problem, and evaluated the performance in an indoor simulator [31]. Zhang et al. [32] proposed a deep reinforcement learning algorithm based on the successor features, which can transfer knowledge from previous navigation problems to new situations. By using additional supervision signals from auxiliary tasks, Mirowski et al. [33] greatly boosted the training and improved the task performance of their DRL algorithm in 3D mazes navigation tasks. Unlike addressing the navigation tasks in the static environment, Chen et al. [34] developed a time-efficient navigation method in dynamic environments with pedestrians. Moreover, Long et al. [35,36] extended the robot navigation task to the multi-robot case, and focused on addressing the collision avoidance problem.

2.3. Multi-Agent Deep Reinforcement Learning

If we ignore the the kinematics and only consider the behavior strategies of the group of robots, there have been plenty of novel works called multi-agent reinforcement learning (MARL) in recent years. Raileanu et al. [37] proposed a new approach for learning in self other-modeling. This method used its own policy to estimate the other agent's actions and updated its belief of the hidden state. Then, the estimations were used to choose new actions. Yang et al. [38] simplified the communication of agents into an average effect. By introducing the mean-field theory, they mainly studied one agent with the average effect of the others. Wang et al. [39] estimated an opponent's future behavior by utilizing the history information. Tacchetti et al. [40] proposed the relational forward models to address the MARL tasks. They added the relational graph model in the action making stage, and used the recurrent neural network (RNN) to model the relationships between agents. However, with the ideal assumptions, there is still a lot of work to do if we want to apply these methods to the multi-robot navigation tasks.

3. Methods

In this section, we first formulate the problem for Multi-Robot collaborative navigation. Then, we describe our Parallel Deep Deterministic Policy Gradient (PDDPG) method for this task.

3.1. Problem Formulation

This paper aims to provide an end-to-end navigation method for multi-robot systems. We try to find such a learnable policy module π:

$$a_t^i = \pi(x_t^i, \varphi_t^i, a_{t-1}^i), \tag{1}$$

where x_t^i is the observation from the raw lidar sensor information of robot i at timestep t, φ_t^i is the relative parameters about the target, and a_{t-1}^i is the control action in the last timestep. In the multi-robot reinforcement learning system, these inputs can be regarded as the state of whole system $s_t = (x_t, \varphi_t)$. For a single robot at each timestep t, the robot makes observations $s_t \in \mathcal{S}$ and selects actions $a_t \in \mathcal{A}$ with respect to its policy $\pi\colon \mathcal{S} \to \mathcal{P}(\mathcal{A})$, which maps states to a probability distribution over the actions. Then, the robot can receive the reward $r(s_t, a_t)$ and arrive at the next state. The state-action value function describes the expected return of a state-action trajectory according to π. It is commonly used to evaluate the quality of a policy as defined in Equation (2):

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t\right)\right] \quad where \;\; s_t \sim p\left(\cdot | s_{t-1}, a_{t-1}\right), a_t = \pi\left(s_t\right). \tag{2}$$

The recursive form of the state-action value function, known as the Bellman equation, can be defined as Equation (3). The policy module π directly maps the state perception to the control law of robots with the collaborative consideration. E represents the environment. In addition, the notation $\sim$ means the former variable follows the distribution of the latter:

$$Q^\pi\left(s_t, a_t\right) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r\left(s_t, a_t\right) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}\left[Q\left(s_{t+1}, a_{t+1}\right)\right]\right]. \tag{3}$$

In order to get a good policy module π, the multi-robot system has to overcome several challenges. First of all, it is difficult to extract valuable information from the raw sensor data and merge it properly with the relative parameters of robot state. Then, based on the processed information, a policy module needs to learn a stable transition rule between the perceptions and the decision to make. Additionally, the robots need to avoid the collision and consider the collaboration in some situation.

3.2. Single-Robot End-to-End Navigation

To accomplish the final goal, we start at the single robot navigation case in this section. In the end-to-end mapless navigation tasks, the relationship between the perception inputs and the output control law can be very complex. In our work, we consider a number of modifications to the Deep Deterministic Policy Gradient (DDPG) and use it as the basic framework.

3.2.1. Basic Reinforcement Learning Framework

Classical Deep Deterministic Policy Gradient (DDPG) [41] is a kind of remarkable reinforcement learning algorithm to address the continuous control problem. As we all know, the trade-off between the exploration and the exploitation is one of the most important problems in the reinforcement learning field. To increase the sample efficiency, the DDPG uses the deterministic target policy $\mu : \mathcal{S} \leftarrow \mathcal{A}$ rather than the stochastic policy π, and the corresponding exploration decrease is made up of the off-policy technique. The Bellman equation can be rewritten as Equation (4), where the γ term represents the discount factor of the equation:

$$Q^{\mu}\left(s_t, a_t\right) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r\left(s_t, a_t\right) + \gamma Q^{\mu}\left(s_{t+1}, \mu\left(s_{t+1}\right)\right)\right]. \tag{4}$$

At each timestep, the actor networks and the critic networks are updated by sampling a minibatch uniformly from the memory buffer. The algorithm also creates a copy of the actor and critic networks, $Q'\left(s, a|\theta^{Q'}\right)$ and $\mu'\left(s|\theta^{\mu'}\right)$, respectively; these target networks are then updated softly by the learned networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ with $\tau \ll 1$. This trick can greatly improve the stability of learning.

The loss function for the critic networks can be formulated as Equation (5):

$$L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2, \; where \; y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right). \tag{5}$$

In addition, the actor networks can be updated by using the sampled policy gradient as shown in Equation (6):

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a|\theta^Q\right)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}}\mu\left(s|\theta^{\mu}\right)\Big|_{s_i}. \tag{6}$$

3.2.2. Network Structure

Owing to the structure of the 2D lidar data, we add one-dimensional, convolutional neural network (CNN) layers to the feature extraction part. To enrich the perception information of the policy module, some relative parameters about the target are added to our framework. It enables the policy module to simultaneously consider the sensor observation and the robot state information. Both of them are essential for efficient navigation.

As shown in Figure 1, the raw lidar sensor data are filtered into 180 dimensions as inputs. As mentioned before, there are two kinds of neural networks in our module. One is the actor network and the other is the critic network. Both of them use three one-dimensional, CNN layers for feature extraction. After the data feature extraction, the networks use residual building blocks with shortcut connections [42] to reduce the training complexity. Then, we pack the target distance, the target angle and the actions at the last timestep as the state parameter set. The state parameter set and the lidar data feature extraction will be fused together and fed to the actor and critic networks at the same time. For the actor network, the fusion data will pass three fully connected layers and output the linear velocity and angular velocity through different activation functions. Besides controlling the robot navigation, the output of the actor network will be transmitted to the critic network. The output action, the state parameter set and the feature extraction of the critic network will be fused into one data stream. After passing through four fully connected layers, the data stream will finally become a Q value to evaluate the policy and train the networks.

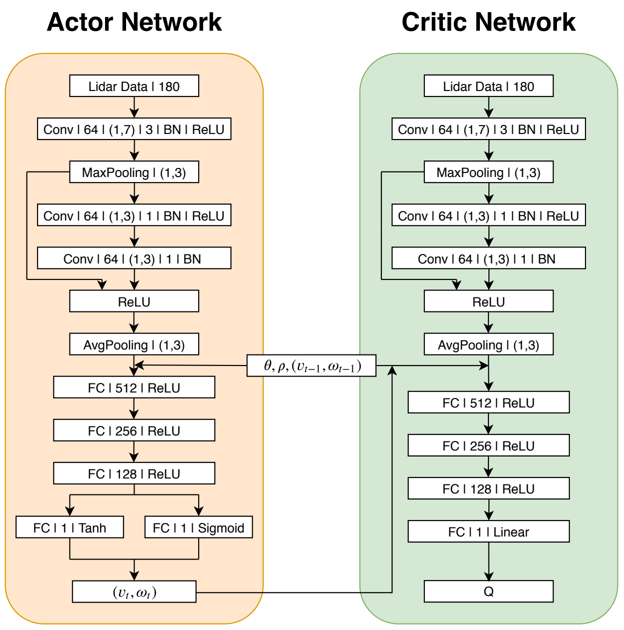

Figure 1. The structure of the modified Deep Deterministic Policy Gradient for single robot mapless navigation tasks.

3.2.3. Reward Shaping

The reward function plays an important role in reinforcement learning: it greatly influences the navigation of the robot system and serves as the truth holder of the learning process. For the single robot mapless navigation task, the reward function shown in Equation (7) consists of three different parts: the arrival award, the collision penalty and the approaching award. If the distance value between the robot and the target point is less than the arrival threshold, the robot can get a positive reward r_{arrive} if the robot collides with the obstacles during navigation. In other words, one of the lidar distance observations is less than the safety threshold. The robot will get a negative reward $r_{collision}$. To enable the intuition of approaching the target, we add the approaching award to the robot, where the reward is noted as $k(\rho_{t-1} - \rho_t)$. The k term is a constant to adjust the amplitude of the approaching reward. The ρ term represents the distance between the target and the mobile robot at timestep t. To unify the range of different parameters, they will be normalized before utilizing:

$$r(s,a) = \begin{cases} r_{arrive} & (\rho < d_{goal}) \\ r_{collision} & (\min(d_1, d_2, ..., d_{180}) < d_{collision}) \\ k(\rho_{t-1} - \rho_t) & (At\ each\ time\ step\ t). \end{cases} \tag{7}$$

3.2.4. The Switchable Controller

Even if we formulate the reward function to encourage the robot to navigate efficiently and safely, the state space of the robot navigation is still large. Utilizing the switchable controller is an intuitive way inspired by imitation learning. For example, if the robot is close to the obstacle, the basic navigation rules will give a large angle velocity to control the robot in order to avoid crashing.

When the robot starts training, the switchable controller can guide the robot to follow some basic navigation rules, rather than randomly exploring in the unknown environment. In addition, the probability of choosing the basic controller will decrease slowly. This means that the robot mainly learns the navigation policy from the basic controller at the beginning, and the basic controller plays an important role as an expert. The trajectories saved in the experience memory will teach the basic navigation rules to the robot. After learning the basic navigation rules, the robot will gradually increase the probability of choosing the learned navigation policy. With that practical framework, the distribution of the policy module can converge to the appropriate shape faster than random exploration.

3.3. Multi-Robot Collaborative Navigation

In this section, we would extend the single navigation algorithm to the multi-robot situation. In the multi-robot system, the observations of each robot are contained by the other robots, the others can be regarded as the dynamic obstacles. If the group of robots don't know how to collaborate with others, they will interfere during navigation.

3.3.1. Parallel Deep Deterministic Policy Gradient

To address the aforementioned issues, we proposed a parallel deep deterministic policy gradient (PDDPG). Compared with the improved DDPG for a single robot mapless navigation task, the PDDPG algorithm trains the robots in parallel. The algorithm attains the target of a whole multi-robot system by sharing the experience memory data and the navigation policy. The robots of the system use the same policy module to make decisions, and the trajectories of robots will be saved in the shared experience replay buffer. The basic network structure unit of the PDDPG algorithm is the same as improved DDPG. We only modify the input dimension of lidar observations and the robot states.

3.3.2. Curriculum Design for Reward Shaping

In the last section, we modified the network structure and proposed the PDDPG to address multi-robot navigation tasks. Besides the good network structure like PDDPG, proper reward formulation is one of the most essential parts of enabling a collaborative capability for a multi-robot navigation system.

When the robots navigate in a mapless environment, the reward would be very sparse. PDDPG would took a long time to converge to a satisfying policy module. A direct training with PDDPG on the collaborative navigation task with eight robots does not yield acceptable performance. To address this issue, we use the curriculum learning [43] which trains the robot with a sequence of progressively more difficult tasks. It paves the way to train the agent to accomplish the difficult tasks. In our proposed method, we only divide the collaborative navigation task into two stages. First, the group of robots needs to construct the formation. In addition, then the robots would be trained to keep the initial formation during navigation.

For the formation construction task, we adjust the reward function of the single robot navigation task to the multi-robot version. The approaching reward of each robot would be summed up as a group approaching reward, as shown in Equation (8). When any of the robots collide, the environment will be reset and the episode will be ended. In the view of training a good reinforcement learning agent, terminating the episode at an appropriate step can efficiently accelerate the agent training. After that, we train the robot to accomplish the collaborative navigation task based on the formation construction policy module. Besides the aforementioned reward, we add the formation constraints in the group reward term $r_{formation}$. This term represents some deformation penalties in our system. Specifically, the algorithm calculates the distances between the robots and save their ratios at the first timestep. When the group of robots are navigating, the algorithm would check the distance ratios and give relative rewards. Moreover, if the speed of one robot is much higher than the mean, the formation keeping constraints would penalize the robot:

$$r(s,a) = \begin{cases} r_{arrive} & (\rho < d_{goal}) \\ r_{collision} & (\min(d_1, d_2, ..., d_{180}) < d_{collision}) \\ \Sigma_1^n k(\rho_{t-1}^i - \rho_t^i) & (At\ each\ time\ step\ t) \\ r_{formation} & (Only\ at\ the\ formation\ keeping\ stage). \end{cases} \quad (8)$$

4. Experiments

To evaluate the performance of our PDDPG algorithm, sufficient experiments are illustrated in the simulation platform. To construct the whole navigation environment of the mobile robots, the simulation is built by a robot simulator named Gazebo, which is used for fast moduling and validating the proposed PDDPG algorithm. The proposed method is implemented on a PC with 15.6 G memory, i7-7700 CPU and Geforce GTX1080Ti on an Ubuntu16.04 operating system.

4.1. Mobile Robot Construction

Robot Operating System (ROS) is a robotics middleware; it provides various tools and services designed for robot research, such as package management, hardware abstraction and the low-level device control. This work utilizes the Gazebo simulation platform to construct the simulation environment. Gazebo is a well-designed simulator with a robust physics engine and ROS support. It offers the ability to rapidly design robots and test algorithms. By using the gym-Gazebo toolkit [44], the deep reinforcement learning agent could be trained on the Gazebo platform efficiently.

As shown in Figure 2a, we build differential driven robot modules in simulation, and add the 2D lidar sensor to perceive the environment. The perception angle of 2D lidar is 270° with 0.25° resolution. The measurement distance of the lidar is from 0.1 m to 30 m, and the frequency of it is 40 Hz. As shown in Figure 2b, the blue lines represent the lasers. When the lasers are blocked by the barriers, each laser will return a distance measurement to the lidar sensor. If the distance measurement is longer than 30 m, the return value will be restricted at 30 m.

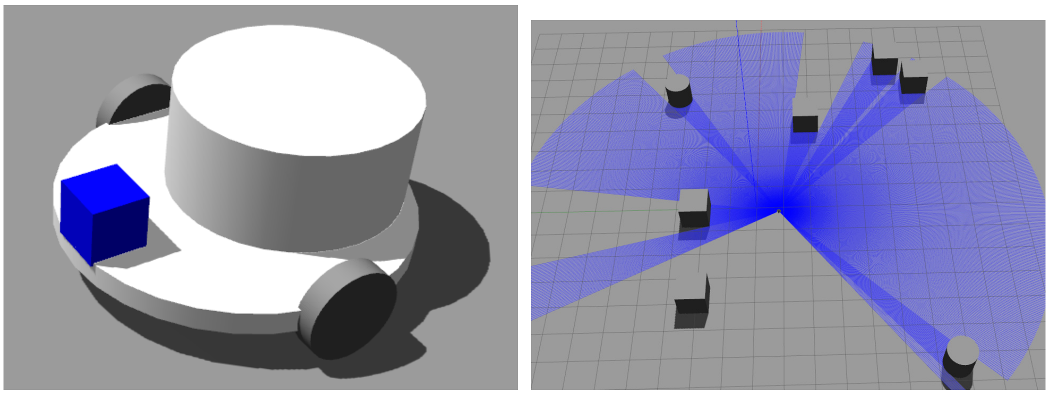
(**a**) Robot Module (**b**) Laser Visualization

Figure 2. (**a**) the module of mobile robots in our Gazebo simulation platform; (**b**) the visualization of the lasers.

4.2. Single-Robot Experiments

In this section, we discuss the single robot mapless navigation task. The training procedure of our robot is implemented in the indoor scene simulated by Gazebo. We train our robot in several indoor scenes with different obstacle placement. We randomly set the robot at the start position with different initial directions. To guarantee the variety of the navigation trajectories and overcoming the overfitting, the targets are randomly chosen outside the restricted zone within the indoor scenes. It can reduce the gap between the training and testing of the policy module.

In the simulation environment, the training procedure can be sped up 10 times. With the simulation acceleration, the control frequency of the mobile robot can reach 100 Hz. The safety threshold of the mobile robot is defined as 0.3 m. In addition, the arrival threshold is also defined as 0.3 m. In other words, if the distance between the mobile robot and the target is less than 0.3 m, this episode will be terminated. Other parameters of the single robot experiments are illustrated in Table 1.

Table 1. Hyperparameters of the networks on a single-robot navigation task.

Parameter	Batch Size	Max Step	LR (Actor)	LR (Critic)	Discount	Buffer Size	Explore Decay
Value	128	800	0.001	0.0001	0.99	100,000	0.998

To compare the performance between the improved DDPG and the classical DDPG, we trained both of them with the same hyperparameters. As shown in Figure 3, the negative Q value of the improved DDPG illustrated by the orange line has a steady decline, while the negative Q value of the classical DDPG illustrated by the blue line increases slightly. Since the Q value evaluates the performance of the policy module, this means the improvement of DDPG becomes better during training, and the classical DDPG falls into local optimum. By analyzing the experiment results, we can infer that the modification in the improved DDPG can address the single robot mapless navigation tasks properly.

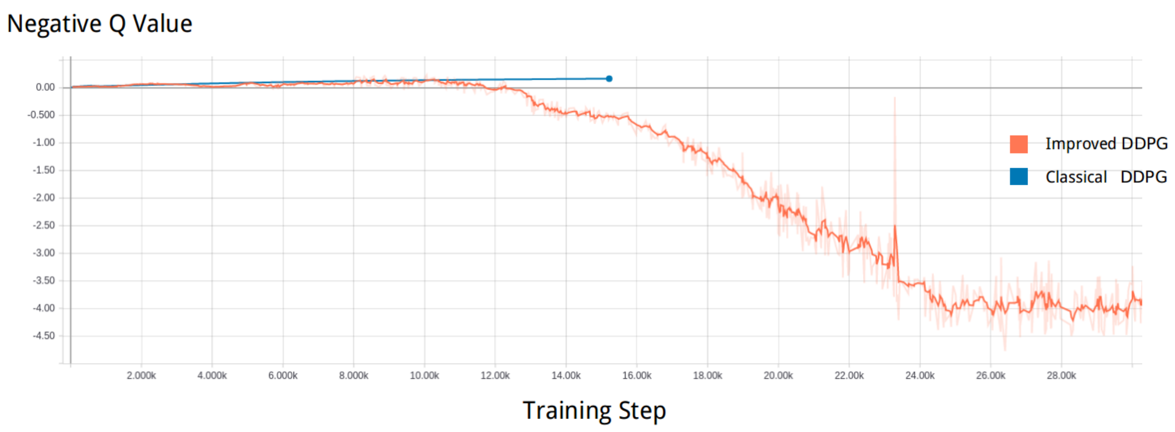

Figure 3. The negative Q value comparison between the improved deep deterministic policy gradient (DDPG) and the classical deep deterministic policy gradient(DDPG) and the classical DDPG of the single robot navigation experiments.

For the navigation task that has a high-dimensional decision space, the improved DDPG with the switchable controller and the prioritized experience replay [17] technique can quickly constrain the searching field of policy module. Then, the policy module will be improved gradually. In the simulation environment, the robot with classical DDPG has the possibility to go around in circles. In addition, the robot with improved DDPG can arrive to the target position in most cases. We tested the improved DDPG policy module in the training environments 80 times, and its arrival rate can even reach 95%. When we move it to the unseen environment, the arrival rate declines to 87.5%. Figure 4 shows the single robot navigation experiments with the improved DDPG. The left part shows the simulation environment, and the right part illustrates the navigation trajectories of the mobile robot. The trajectories are separately visualized by the Rviz toolkit. The robot starts to navigate from the star and ordinally traverses through target 1 to target 6. When the robot arrives at target 6, it will set target 1 as the next destination and repeat this cycle.

4.3. Multi-Robot Experiments

Based on the single robot navigation results, we discuss the experiments of the multi-robot navigation tasks in this section. The proposed Parallel Deep Deterministic Policy Gradient (PDDPG) algorithm will be evaluated and analyzed carefully. With the curriculum learning setup, the experiments would be illustrated in a two-stage arrangement: the formation construction and the collaborative navigation.

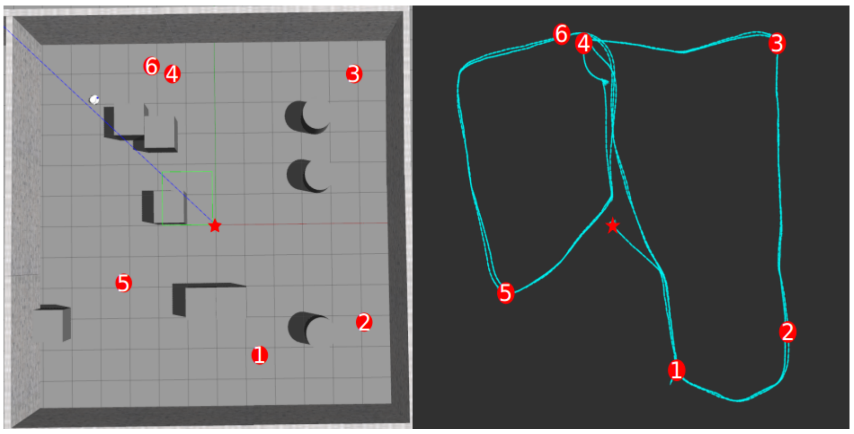

Figure 4. The trajectory of the mobile robot in the single robot mapless navigation tasks. The left part is the navigation environment in Gazebo, the right part is the trajectory of the mobile robot.

4.3.1. Formation Construction

In the formation construction experiments, eight mobile robots are deployed in the indoor scene. The robots are initialized at different positions with various environmental characteristics. We set the target formation of the robots as a rectangle; then, the eight robots will be trained together with the proposed Parallel Deep Deterministic Policy Gradient (PDDPG) algorithm. The hyperparameters of the PDDPG are listed in Table 2. In the training stage, we add the random bias to the target position values in each episode. It can enrich the formation data and reduce the overfitting problem of multi-robot training. Figure 5a,b illustrate one of the formation construction experiments in the testing stage. Figure 5a shows the initialization of the multi robot system; the robots are deployed in the indoor scene with various poses. Figure 5b shows the final results of the formation construction task. The curves with different colors represent the navigation trajectories of different mobile robots. As illustrated in the figures, all the mobile robots arrive at the target and construct the formation properly.

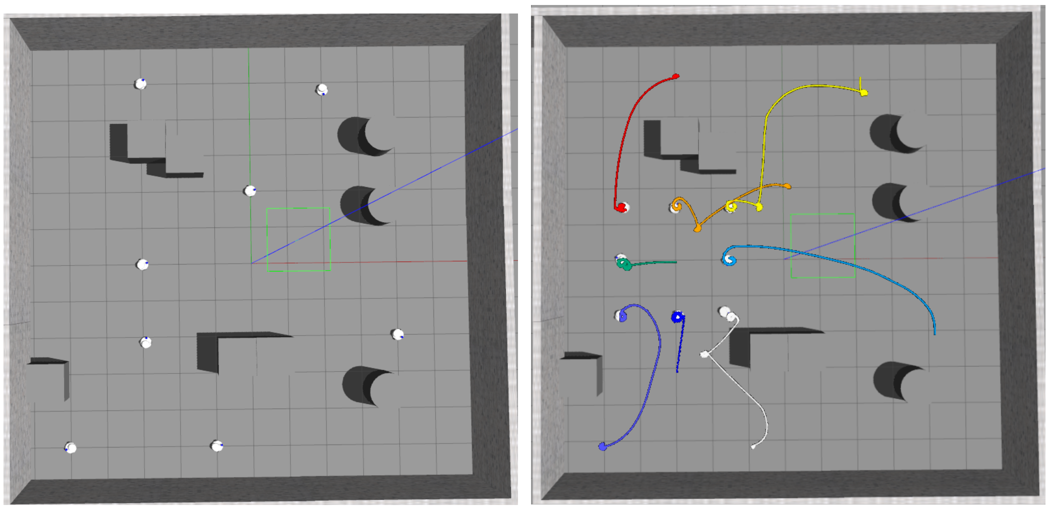
(**a**) Initial State (**b**) Formation Construction

Figure 5. (**a**) the initial state of the group of mobile robots; (**b**) the terminal state of the group of mobile robots after accomplishing the formation construction navigation.

Table 2. Hyperparameters of the networks on multi-robot navigation tasks.

Parameter	**Batch Size**	**Max Step**	**LR (Actor)**	**LR (Critic)**	**Discount**	**Buffer Size**	**Explore Decay**
Value	256	2000	0.0001	0.0001	0.99	500,000	0.998

4.3.2. Collaborative Navigation

After obtaining a good policy module in the formation construction task, we use it as the pretrained module in this section. The collaborative reward functions and the constraints are added in these experiments. As shown in Figure 6, the group of mobile robots can keep the rectangle formation during navigation. The trajectories of different mobile robots are illustrated by the colorful curves. In addition, the mobile robots at several timesteps are merged into one figure after getting the trajectory.

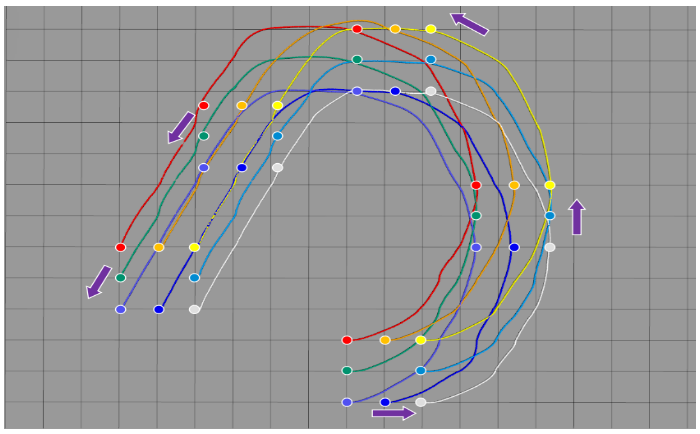
Figure 6. The trajectory of the multi-robot collaborative navigation.

To evaluate the versatility of the multi-robot navigation module, we also evaluate the module with the arrow formation. Figure 7 consists of four different figures, and they are recorded at the different timesteps. The intersection point of three colorful lines represents the origin of the environment. The group of robots navigated from the bottom left to the top right. This experiment illustrates that the mobile robots can keep the arrow formation during navigation.

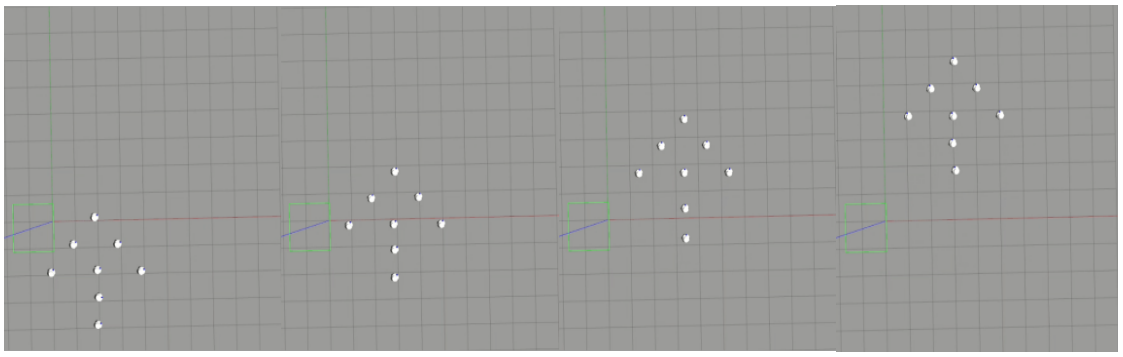

Figure 7. The arrow formation of the multi-robot navigation system.

As illustrated in the collaborative navigation experiments, we mainly evaluate the formation keeping navigation tasks in the simulation environment without obstacles, since the distance of the obstacles can influence the performance of formation keeping. Generally, the multi-robot collaborative navigation task with formation keeping has two levels: one is the formation keeping navigation; the other is the multi-robot obstacle avoidance in a complex environment with constant formation. In this paper, we mainly focus on the former. In our work, the policy module of a collaborative navigation task utilizes the pre-trained policy module in the formation keeping task. There is lots of prior knowledge about obstacle avoidance that has been learned in the pre-trained module. Thus, the collaborative navigation module is sensitive about the obstacles. If there are obstacles in the navigation path, the formation of mobile robots can't keep the neat shape. The size of the obstacles can also influence the performance of formation keeping. In this paper, we mainly illustrate the formation keeping navigation tasks in the simulation environment without obstacles. In addition, we will address the multi-robot obstacle avoidance in a complex environment with constant formation keeping in future work.

To discuss the execution time of our method, we add the experiments to compute the execution time for the proposed Parallel Deep Deterministic Policy Gradient (PDDPG) algorithm. Since our end-to-end policy module can directly obtain the linear velocity and angular velocity through the raw sensor data, we can infer the execution time by computing the navigation velocity of the multi-robot system. There are two kinds of times in the Gazebo simulation platform: the simulation time and real time. The platform can accelerate the simulation or slow it down to fit specific tasks. To speed up the experimental efficiency, our work accelerates the simulation during training. Thus, we convert the simulation time to real time and compute the velocity of our multi-robot navigation system. As shown in Table 3, the group of robots navigates in four different kinds of trajectories with rectangular formation. The navigation distances, the time durations and the execution velocities are listed as follows. According to the results, we can infer that our method has an acceptable execution time.

Table 3. Experiments related to the navigation velocities of the multi-robot system.

Trajectory Type	**Straight Line**	**L Shape**	**Rectangular Shape**	**S Shape**
Distance (m)	12.73	18.50	35.50	35.50
Time (s)	5.01	8.45	18.49	18.98
Velocity (m/s)	2.54	2.19	1.92	1.87

5. Conclusions

This work mainly studied the collaborative formation and navigation of multi-robot system by using the deep reinforcement learning algorithm. By taking the raw 2D lidar sensor data and the relative target positions as inputs, the proposed Parallel Deep Deterministic Policy Gradient (PDDPG) algorithm could directly control the group of mobile robots to construct the formation and maintain it during navigation. The end-to-end policy module for mapless navigation was evaluated both in the single-robot situation and the multi-robot situation. Our experimental results demonstrated that the proposed method could teach the multi-robot system to learn human intuition and accomplish the collaborative navigation tasks with a high arrival rate.

Author Contributions: Conceptualization, W.C. and S.Z.; Data curation, S.Z.; Formal analysis, S.Z.; Funding acquisition, Y.L.; Investigation, S.Z. and H.Z.; Methodology, W.C. and S.Z.; Project administration, W.C., S.Z. and Y.L.; Resources, Z.P. and H.Z.; Software, S.Z.; Supervision, Z.P. and Y.L.; Validation, W.C. and S.Z.; Visualization, W.C.; Writing—original draft, W.C.; Writing—review and editing, W.C.

Funding: This work is supported by the National Natural Science Foundation of China under Grant U1509210, 61836015.

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

References

1. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
2. Bresson, G.; Alsayed, Z.; Yu, L.; Glaser, S. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Trans. Intell. Veh.* **2017**, *2*, 194–220. [CrossRef]
3. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
4. Schmerling, E.; Janson, L.; Pavone, M. Optimal sampling-based motion planning under differential constraints: The driftless case. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 2368–2375.
5. Li, Y.; Littlefield, Z.; Bekris, K.E. Sparse methods for efficient asymptotically optimal kinodynamic planning. In *Algorithmic Foundations of Robotics XI*; Springer: New York, NY, USA, 2015; pp. 263–282.
6. Janson, L.; Schmerling, E.; Clark, A.; Pavone, M. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *Int. J. Robot. Res.* **2015**, *34*, 883–921. [CrossRef]
7. Webb, D.J.; Van Den Berg, J. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 5054–5061.
8. Kim, E.; Kim, J.; Sunwoo, M. Model predictive control strategy for smooth path tracking of autonomous vehicles with steering actuator dynamics. *Int. J. Automot. Technol.* **2014**, *15*, 1155–1164. [CrossRef]
9. Gáspár, P.; Szabó, Z.; Bokor, J. LPV design of fault-tolerant control for road vehicles. *Int. J. Appl. Math. Comput. Sci.* **2012**, *22*, 173–182. [CrossRef]
10. Doumiati, M.; Sename, O.; Dugard, L.; Martinez-Molina, J.J.; Gaspar, P.; Szabo, Z. Integrated vehicle dynamics control via coordination of active front steering and rear braking. *Eur. J. Control* **2013**, *19*, 121–143. [CrossRef]
11. Paden, B.; Čáp, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55. [CrossRef]
12. Biswas, J.; Veloso, M. Wifi localization and navigation for autonomous indoor mobile robots. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; pp. 4379–4384.
13. Nazemzadeh, P.; Fontanelli, D.; Macii, D.; Palopoli, L. Indoor localization of mobile robots through QR code detection and dead reckoning data fusion. *IEEE/ASME Trans. Mechatron.* **2017**, *22*, 2588–2599. [CrossRef]
14. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef]
15. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
16. Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; De Freitas, N. Dueling network architectures for deep reinforcement learning. *arXiv* **2015**, arXiv:1511.06581.
17. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.

18. Fortunato, M.; Azar, M.G.; Piot, B.; Menick, J.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; Pietquin, O.; et al. Noisy networks for exploration. *arXiv* **2017**, arXiv:1706.10295.
19. Dabney, W.; Rowland, M.; Bellemare, M.G.; Munos, R. Distributional reinforcement learning with quantile regression. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
20. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
21. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Driessche, G.V.D.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484. [CrossRef]
22. Browne, C.B.; Powley, E.; Whitehouse, D.; Lucas, S.M.; Cowling, P.I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; Colton, S. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intell. AI games* **2012**, *4*, 1–43. [CrossRef]
23. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* **2017**, *550*, 354. [CrossRef] [PubMed]
24. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv* **2017**, arXiv:1712.01815.
25. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv* **2018**, arXiv:1806.10293.
26. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
27. Tai, L.; Zhang, J.; Liu, M.; Boedecker, J.; Burgard, W. A survey of deep network solutions for learning control in robotics: From reinforcement to imitation. *arXiv* **2016**, arXiv:1612.07139.
28. Kretzschmar, H.; Spies, M.; Sprunk, C.; Burgard, W. Socially compliant mobile robot navigation via inverse reinforcement learning. *Int. J. Robot. Res.* **2016**, *35*, 1289–1307. [CrossRef]
29. Pfeiffer, M.; Schwesinger, U.; Sommer, H.; Galceran, E.; Siegwart, R. Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 2096–2101.
30. Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J.J.; Gupta, A.; Fei-Fei, L.; Farhadi, A. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3357–3364.
31. Kolve, E.; Mottaghi, R.; Gordon, D.; Zhu, Y.; Gupta, A.; Farhadi, A. Ai2-thor: An interactive 3d environment for visual ai. *arXiv* **2017**, arXiv:1712.05474.
32. Zhang, J.; Springenberg, J.T.; Boedecker, J.; Burgard, W. Deep reinforcement learning with successor features for navigation across similar environments. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 2371–2378.
33. Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A.J.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; et al. Learning to navigate in complex environments. *arXiv* **2016**, arXiv:1611.03673.
34. Chen, Y.F.; Everett, M.; Liu, M.; How, J.P. Socially aware motion planning with deep reinforcement learning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1343–1350.
35. Long, P.; Liu, W.; Pan, J. Deep-learned collision avoidance policy for distributed multiagent navigation. *IEEE Robot. Automat. Lett.* **2017**, *2*, 656–663. [CrossRef]
36. Long, P.; Fanl, T.; Liao, X.; Liu, W.; Zhang, H.; Pan, J. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6252–6259.
37. Raileanu, R.; Denton, E.; Szlam, A.; Fergus, R. Modeling others using oneself in multi-agent reinforcement learning. *arXiv* **2018**, arXiv:1802.09640.

38. Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; Wang, J. Mean field multi-agent reinforcement learning. *arXiv* **2018**, arXiv:1802.05438.
39. Wen, Y.; Yang, Y.; Luo, R.; Wang, J.; Pan, W. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv* **2019**, arXiv:1901.09207.
40. Tacchetti, A.; Song, H.F.; Mediano, P.A.; Zambaldi, V.; Rabinowitz, N.C.; Graepel, T.; Botvinick, M.; Battaglia, P.W. Relational forward models for multi-agent learning. *arXiv* **2018**, arXiv:1809.11044.
41. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
42. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
43. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
44. Zamora, I.; Lopez, N.G.; Vilches, V.M.; Cordero, A.H. Extending the openai gym for robotics: A toolkit for reinforcement learning using ros and gazebo. *arXiv* **2016**, arXiv:1608.05742.