applied sciences

MDPI

Article

FGRL: Federated Growing Reinforcement Learning for Resilient Mapless Navigation in Unfamiliar Environments

Shunyu Tian [1], Changyun Wei [1,*], Yajun Li [1] and Ze Ji [2]

[1] College of Mechanical and Electrical Engineering, Hohai University, Changzhou 213200, China; 221319010023@hhu.edu.cn (S.T.); liyajun0908@gmail.com (Y.L.)
[2] School of Engineering, Cardiff University, Cardiff CF24 3AA, UK; jiz1@cardiff.ac.uk
* Correspondence: c.wei@hhu.edu.cn

Featured Application: This work is motivated by practical applications, such as smart factories and warehouses, where unmanned ground vehicles (UGVs) are required to efficiently navigate to desired destinations without accurate environmental maps, and they may encounter unfamiliar obstacles with various sizes and shapes. Reinforcement learning (RL)-based mapless navigation approaches have shown promising results in dealing with the lack of precise global maps. However, it often takes a long time for a single agent to converge to the optimal model and still struggles with out-of-distribution observations. To ensure the generation and adaptability of unfamiliar environments, our proposed federated growing reinforcement learning (FGRL) approach can allow multiple learning agent to achieve knowledge aggregation so as to obtain an adaptive and resilient navigation model.

Abstract: In this paper, we propose a federated growing reinforcement learning (FGRL) approach for solving the mapless navigation problem of unmanned ground vehicles (UGVs) facing cluttered unfamiliar obstacles. Deep reinforcement learning (DRL) has the potential to provide adaptive behaviors for autonomous agents through interactive learning, but standard episodic DRL algorithms often struggle with out-of-distribution observations. For navigation tasks, UGVs often encounter unfamiliar situations where novel obstacles differ from prior experience. To address this problem, the proposed FGRL approach enables multiple agents to obtain their individual navigation models in diverse scenarios, and achieves online knowledge aggregation to obtain an adaptive and resilient model that copes with unfamiliar uncertain obstacles. Specifically, during the learning process of navigation tasks, we introduce the growth rate of each agent's local model based on the performance of consecutive learning rounds. Then, we weight the local model of each agent based on the growth rate to achieve knowledge aggregation in a shared model. We also consider a growth threshold to eliminate the interference of low-quality local models. We carry out extensive simulations to validate the proposed solution, and the results show that our approach can learn resilient behaviors of collision avoidance for UGVs to cope with never encountered and cluttered unfamiliar obstacles.

Keywords: collision avoidance; navigation; deep reinforcement learning; federated learning

Citation: Tian, S.; Wei, C.; Li, Y.; Ji, Z. FGRL: Federated Growing Reinforcement Learning for Resilient Mapless Navigation in Unfamiliar Environments. *Appl. Sci.* **2024**, *14*, 11336. https://doi.org/10.3390/app142311336

Received: 29 October 2024
Revised: 1 December 2024
Accepted: 4 December 2024
Published: 5 December 2024

1. Introduction

Unmanned ground vehicles (UGVs) have been widely used in various applications such as manufacturing [1], warehousing [2], and carrying support for terminals and hospitals [3]. In these scenarios, UGVs are required to be capable of efficiently navigating to desired destinations without colliding with obstacles. Many studies have addressed the navigation problem of UGVs, but one remaining tough challenge is how to plan collision-free and efficient paths if the accurate environmental map is not available and the obstacles are unfamiliar to UGVs, e.g., in a complex and ever-changing factory environment, as depicted in Figure 1.

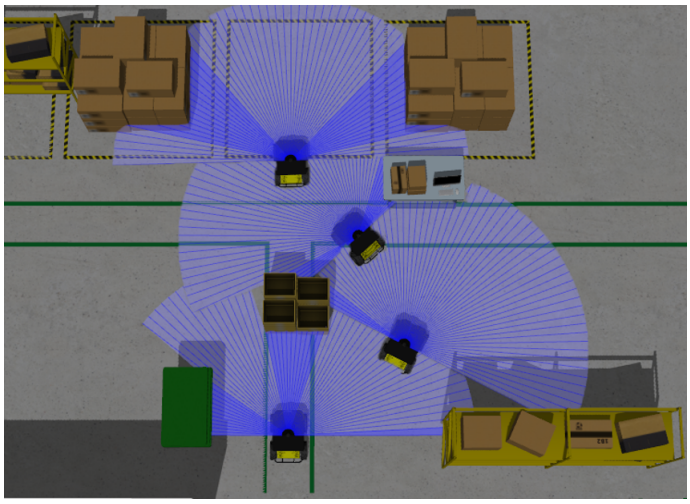


Figure 1. UGVs may encounter unfamiliar obstacles with various sizes and shapes in a smart factory.

To handle navigation tasks, the popular Simultaneous Localization and Mapping (SLAM) and its variants [4,5] can use sensors to generate the obstacle maps of the navigation environment. Planning algorithms (e.g., rapid-exploration random tree [6]) can then be applied to find the collision-free paths. However, building an accurate map can be expensive due to the need for sensor fusion [7], and map accuracy directly affects the quality of generated paths.

Our work focuses on finding solutions for navigation in unknown environments without relying on maps. Traditional navigation methods, which depend on precise global maps, are often impractical in such environments. Therefore, we aim to develop methods that allow robots to navigate effectively without needing maps. We believe robots can learn navigation policies using reinforcement learning (RL), a technique that has been successful in fields like gaming [8,9] and robotic manipulation [10,11]. In RL, an agent learns to make decisions based on observations, with the goal of maximizing long-term rewards. Recently, RL has also been applied to navigation tasks in robotics [12,13]. Although RL has shown promising results in navigation tasks, a major challenge is that the optimal policy must be learned through extensive interactions with the environment. It often takes a long time for a single agent to converge to the optimal solution, and it is nearly impossible for one agent to explore all possible states in real-world robotic tasks. To address this, distributed and parallel RL algorithms [14,15] offer a potential solution. However, these methods require a central server to collect data from multiple agents for model training, raising concerns about information leakage and privacy [16]. Moreover, standard episodic RL algorithms often struggle with out-of-distribution observations and the adaptability of unfamiliar environments. For example, UGVs may face unfamiliar situations in warehouses, but novel obstacles may differ from prior learned experience. These challenges highlight the need for novel RL algorithms that can address these issues and enable effective navigation in unknown environments.

This work employs the idea of federated learning (FL), which is a distributed machine learning technique that allows multiple agents to be trained simultaneously [17]. In FL systems, each agent can train its local model separately and send back the model-related parameters to an aggregation center. Each agent's local model is obtained based on its own observations, but it can be aggregated so as to construct a shared model. Most importantly, each agent only needs to process its own collected raw data, and send the parameters of its local model to the aggregation center.

In this paper, we propose an FL-based reinforcement learning approach for the mapless navigation problem of UGVs for unfamiliar environments. In particular, we address how the navigation capabilities of UGVs can be learned based on sensor-level observations, how multiple agents can be used to speed up the learning procedures, and how knowledge aggregation can be developed to ensure the generalization of new environments and obstacles

never seen before. To this end, the performance of each learning round will be evaluated by the growth rate of each agent's local model, and the knowledge aggregation mechanism will be achieved based on such an evaluation. We also take account of the elimination of the interference of low-quality local models during the knowledge aggregation process. The main contributions of this work are summarized as follows.

1. We propose a federated growing reinforcement learning (FGRL) approach that allows multiple agents to be trained simultaneously so as to speed up the learning efficiency.
2. The proposed FGRL approach provides a sensor-level navigation and collision avoidance mechanism that does not require a precise global environmental map.
3. We introduce a knowledge aggregation strategy that ensures the generation of a shared model based on the performance of each agent's local model.
4. We carry out extensive experiment studies in Gazebo, and the results show that traditional RL-based mapless navigation algorithms indeed cannot cope with unfamiliar obstacles. Comparatively, our proposed approach can make UGVs fuse and transfer prior knowledge to new and unfamiliar obstacles in navigation tasks.

The paper is organized as follows. We first discuss the state of the art in Section 2, and then we present our proposed approach in Section 3. The experiment study is carried out in Section 4. Finally, we conclude this paper in Section 5.

2. Related Work

2.1. RL-Based Mapless Navigation

Recent advances have proposed sensor-level approaches (e.g., a single monocular camera [18], a 2D LiDAR and a monocular camera [19], and a depth camera [20]) to address the navigation problem, in which a single robot can be trained to obtain the capability of avoiding collisions with obstacles. A critical issue of applying RL-based approaches to solve navigation tasks is that a considerable amount of sensor-level data are required to train a navigation policy. Thus, most work focuses on transferring the learned policy in simulations to real scenarios [21,22].

To reduce the training time, the parallel and distributed paradigm [23,24] can be applied to more efficiently collect samples by multiple agents in parallel. In synchronous RL algorithms (e.g., Parallel Advantage Actor Critic (PAAC) [25]), each agent explores its own environment and collect samples, and then the global model can be updated synchronously. In asynchronous RL algorithms (e.g., Asynchronous Advantage Actor Critic (A3C) [26]), a coordinator is required to update the global model without waiting for all agents to send the gradients. However, privacy protection and communication consumption need to be taken into consideration for parallel RL methods.

Moreover, we also find that the testing environment for most work is almost the same as the training settings such as in [22,23]. A potential problem that still needs to be investigated is whether the learned policy can be flexible and resilient enough when the robot encounters an unknown and unfamiliar environment. Standard episodic RL algorithms often struggle with the generalization and adaptability of out-of-distribution observations [27]. For navigation tasks, UGVs may face a variety of never encountered obstacles in practice, and we cannot enumerate all the scenarios during the training procedure [28].

2.2. Federated Learning

Federated learning is a distributed paradigm that allows multiple agents to train a model simultaneously and does not need to upload any private data to the server during the training process [17]. Each agent only needs to use local data to obtain a local model, and the central server is responsible for the weighted aggregation of local models so as to construct a global shared model. After multiple rounds of iterations, the shared model close to the results of centralized learning can be finally obtained, which can effectively reduce the privacy risks due to data aggregation [29]. Another issue that should be noted is that heterogeneous data affect the convergence of federated learning to some certain

extent. To address this problem, multi-layer federated learning can be used to achieve model selection and aggregation [30,31].

Therefore, model aggregation is essential in FL for agents to share experience, and determining how to achieve a better model is crucial. During the model training, federated models can be biased towards different agents. In [32], an agnostic FL approach is introduced, where a centralized model is optimized for any target distribution formed by a mixture of the client distributions. A reputation-supported aggregation method [33] is presented to measure the user's aggregation weight based on reputation scores. Majcherczyk et al. [34] put forward a data-driven mechanism to synchronize the learning process, in which a robot can contribute the model updates when enough data are collected. The state of the art also shows the possibility of combining RL with FL in data management [35–37].

In this work, we seek to improve the learning efficiency through parallel training by multiple agents, but we also hope to protect data privacy and reduce communication overheads. In typical FL scenarios, the state transitions of all the agents have the same distribution [16]. However, in navigation tasks, each robot may face different environments, so we need to improve the generalization ability of the model in comparison with conventional FL approaches.

3. FGRL for Mapless Navigation

In this section, we will first detail the RL-based framework for mapless navigation of UGVs and then proceed to discuss how to improve the learning efficiency via the proposed FGRL approach. In this work, UGVs only rely on their own sensor readings, i.e., LiDAR sensors, to obtain the situational awareness about the environment. Sensor readings are represented as the state information in RL tasks, and the control policy can provide action commands for UGVs to navigate without the need for mapping.

3.1. RL-Based Mapless Navigation

In discrete RL tasks, the learning of appropriate actions can be achieved by Deep Q-Network (DQN) [38], and in order to resolve the overoptimistic value estimates of the max operator, the Double Deep Q-Network (DDQN) algorithm [39] is introduced. In continuous RL tasks, the policy gradient method Deep Deterministic Policy Gradient (DDPG) [40] is adapted from DQN, and it updates the Q-values in the same way as DQN. Thus, the overestimation bias still needs to be taken into consideration. In this work, we adopt the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [41] to construct the mapless navigation framework for UGVs, as it can further improve DDPG by addressing variance reduction.

3.1.1. General Framework

Figure 2 depicts the RL-based framework adapted from the TD3 algorithm for mapless navigation. We can see that two critic networks $Q_{\theta 1}$ and $Q_{\theta 2}$, and one actor network π_ϕ are parameterized by θ_1, θ_2, and ϕ, respectively. We can also find three corresponding target networks parameterized by θ_1', θ_2' and ϕ'. The target networks allow a stable convergence of the training procedure. In each learning round, the agent selects an action a to perform based on actor network π_ϕ and takes account of exploration noise ϵ such that $a \sim \pi_\phi(s) + \epsilon$, where ϵ is drawn from a normal distribution $\mathcal{N}(0, \sigma)$ with a mean of 0 and standard deviation σ. Similarly, the target actor network π_ϕ' also includes noise ϵ, but it is drawn from a truncated normal distribution $\epsilon \sim clip(\mathcal{N}(0, \bar{\sigma}), -c, c)$, with the noise clipped to the range $[-c, c]$ to ensure stability. State transitions (s, a, r, s') are stored in the replay buffer. In order to smooth the value estimate, a small amount of random noise is added to the target policy, and the noise is clipped so as to stabilize the target within a small range. During

training, a batch of size N is sampled from the replay buffer. The loss function for the critic network is defined as follows:

$$Loss = N^{-1} \sum_{j=1}^{N} (y_j - Q_j(s_j, a_j))^2, \tag{1}$$

where

$$y_j = r_j + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \bar{a}), \tag{2}$$

and $\bar{a} \sim \pi_{\phi'}(s')$. The actor network is updated in the direction guided by the critic network. Specifically, the update of the actor network is carried out by computing the gradient via the chain rule, starting from the initial distribution J:

$$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s). \tag{3}$$

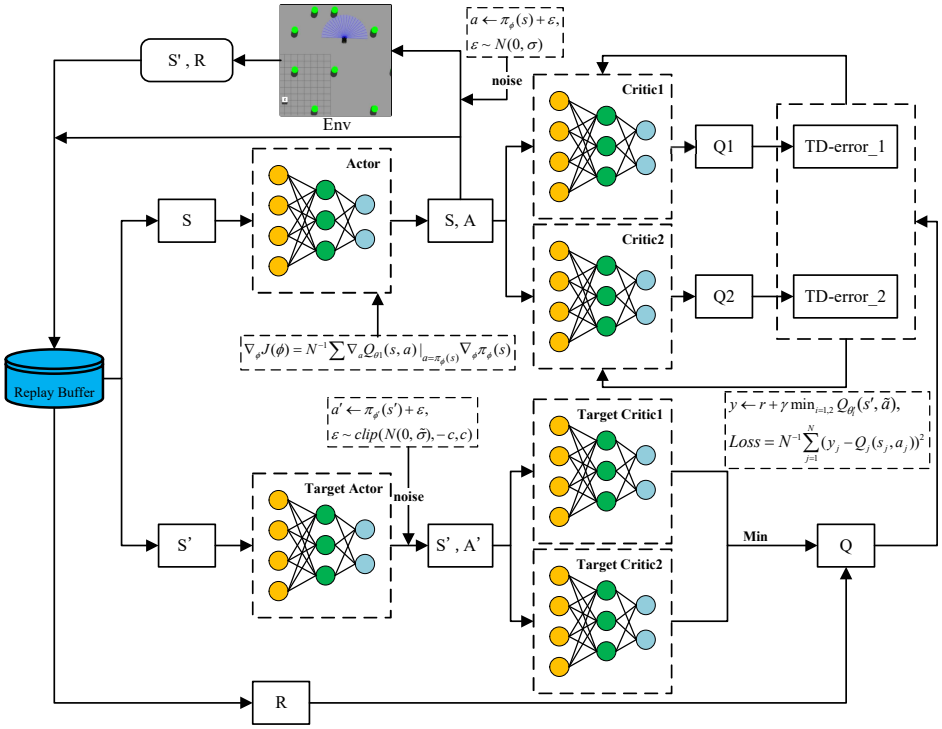


Figure 2. RL-based mapless navigation learning framework adapted from TD3.

3.1.2. Observation Space

In this work, the UGV utilizes low-cost 2D LiDAR sensors to obtain environmental awareness. We deploy the UGV in the Gazebo environment with a real physics engine. At each time step, the observation of the UGV includes three terms $S = [s_1, s_2, s_3]$. s_1 represents the LiDAR readings with 180 degrees, 24 laser beams and a scanning frequency of 30 Hz. s_2 consists of the estimated distance between the UGV and its target location, as well as the offset of the heading angle of the UGV. To estimate the position and orientation of the UGV, we use an Inertial Measurement Unit (IMU) module. The IMU module provides accurate position and orientation data, enabling the calculation of the relative distance between the UGV and the target. The last term s_3 indicates the internal state of the UGV, including the linear velocity v and angular velocity ω.

3.1.3. Action Space

UGVs need to move continuously in the working zone, so we define continuous action space to control the linear velocity v and the angular velocity ω. In this work, the policy directly maps the current observations to the coming action (i.e., the linear velocity and

the angular velocity of the next time step). In order to ensure the stability of the robot movement, the action is limited to a fixed range. Specifically, in this paper, the linear velocity and the angular velocity of UGVs at each time step are limited to $v \in [0, 1]$ (in meters per second) and $\omega \in [-1, 1]$ (in radians per second), respectively.

3.1.4. Reward Function

The objective of the UGV navigation task is to successfully arrive at the target location without colliding with any potential obstacles. To this end, the RL agent aims to obtain the most cumulative rewards, and the reward function is defined as follows:

$$r = r_{\text{collision}} + r_{\text{arrive}} + r_{\text{distance}} + r_{\text{speed}}. \tag{4}$$

During the training period, if a collision event occurs, the learning agent will receive a large negative penalty $r_{\text{collision}}$:

$$r_{\text{collision}} = \begin{cases} -200, & d_{min} \leq d_{lim} \\ -e^{\delta((d_{min}/d_s)-0.5)}, & d_{lim} < d_{min} \leq 2d_{lim} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where d_{min} represents the shortest value of LiDAR readings, and d_s represents the normal detection distance of LiDAR readings without obstacles. d_{lim} defines the limited distance at which collisions can occur, and δ denotes the collision adjustment coefficient used to modify the collision reward function curve within the range $[d_{lim}, 2d_{lim}]$, which is a negative value. According to the above rules, if the distance between the robot and an obstacle is less than the limited value, it will trigger a collision event and receive a large negative reward -200. If the distance between the robot and an obstacle is less than twice the limited distance but larger than this limit, the negative penalty increases as the LiDAR readings become smaller. In this study, we set $d_s = 3.5$ m, $d_{lim} = 0.5$ m, and $\delta = -15$ to allow the reward function to better cover the reward range $[-200, 0]$, effectively reflecting the reward adjustment as the UGV approaches obstacles.

Since the ultimate goal of UGVs is to reach the target location, a positive reward r_{arrive} is obtained if it can successfully arrive at its destination without collisions, with the condition that the distance between the robot's center and the target location is less than 0.5 m. Here, we define a sparse reward to assign a large positive reward:

$$r_{\text{arrive}} = \begin{cases} 200, & \text{if success} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

To solve the problem of sparse rewards during the training process, we also design a continuous reward function, where the agent can obtain feedback r_{distance} at each time step to provide a heuristic for the target location, as follows:

$$r_{\text{distance}} = \mu d_{\text{step}}, \tag{7}$$

where d_{step} indicates the difference between the estimated distance to the target location at current time step and the one at the previous time step. Here, μ denotes the adjustment coefficient of the distance reward. The larger the value of μ, the higher the reward for each step, which encourages the UGV to move toward the target. However, it is important to note that the value of μ is influenced by the final target arrival reward r_{arrive}. If μ is set too high, it can reduce the attractive force of the target, causing the UGV to approach it more slowly as it gets closer, rather than reaching it directly. Therefore, selecting an appropriate value for μ is crucial for balancing the trade-off between approaching the target and reaching the target in the reward function.

During the training process, the moving speed of UGVs may be too fast due to the incentive of distance rewards. For safety concerns, we further consider a penalty $r_{\text{speed}} = r_v + r_\omega$ to regulate the linear and angular velocity, as follows:

$$r_v = \begin{cases} -5, & v \geq v_{max} \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

$$r_\omega = \begin{cases} -5, & |\omega| \geq \omega_{max} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The safety term works when the linear velocity v and the angular velocity ω exceed the maximum permitted values v_{max} and ω_{max}, respectively.

3.2. Federated Growing Reinforcement Learning (FGRL)

As mentioned before, in this work, we borrow the idea of FL to speed up the learning efficiency, and we propose the FGRL approach that will evaluate each agent's performance to calculate the weights of local models. Figure 3 illustrates the model aggregation of the proposed algorithm. Suppose that K agents can be trained in parallel and that a global model exists among all the agents. In the beginning, the global model parameters are sent to each agent to initialize each agent's local model. Here, the parameters specifically configure the actor and critic networks. Then, each agent makes decisions based on its local model and starts to collect new experiences. At the same time, each agent can also update its local model based on new experiences. We define a global update period (also called an aggregation round) with E episodes, and we also define two indicators for performance evaluation of E episodes, i.e., the success rate $\zeta = \frac{n_{\text{success}}}{E}$, and the average steps to the desired location during the successful episodes $\varphi = \frac{\sum n_{\text{steps}}}{n_{\text{success}}}$. Here, n_{success} denotes the number of successful episodes in E, and n_{steps} indicates the moving steps of each successful episode. Thus, in this work, we adopt ζ and φ to evaluate the performance of each agent during its individual learning episodes.

We use m to indicate an aggregation round, where not all agents can perform better than the previous round. Thus, we define a growth value g_m to quantify the performance of each agent in the m-th aggregation round as follows:

$$g_m = \psi \frac{\Delta\zeta_m}{\zeta_m} + (\psi - 1)\frac{\Delta\varphi_m}{\varphi_{m-1}}, \tag{10}$$

where $\Delta\zeta_m = \zeta_m - \zeta_{m-1}$, ζ_m and ζ_{m-1} represent the success rates of the current m-th and the previous aggregation round, while $\Delta\varphi_m = \varphi_m - \varphi_{m-1}$, φ_m and φ_{m-1} indicate the average steps of the current m-th and the previous aggregation round. $\psi \in (0, 1)$ is a normal constant. Thus, we can use the value of g_m to express how much an agent has grown in this m-th aggregation round compared to the previous round.

In order to reduce the fluctuation of the growth value for model aggregation, we use η_m to replace g_m as follows:

$$\eta_m = \frac{1}{1 + e^{-g_m}}. \tag{11}$$

We also set a threshold Ω for the growth value η_m to measure whether the agent grows enough or degrades in an aggregation round. If an agent's performance of the current m-th aggregation round does not exceed the threshold, its local model will not participate in the model aggregation.

To update the global model, we compare two consecutive rounds. For the first update round, we assume that all the agents have the same growth value, $\eta_1 = 1$, and we can average the local models to update the parameters of the global model. After the first

update, we will calculate their respective growth values, using Equations (10) and (11), to weight the contribution of each agent:

$$w_m^{\text{global}} = \sum_{k=1}^{K'} \frac{\eta_m^k}{\sum_{k=1}^{K'} \eta_m^k} w_m^k, \tag{12}$$

where w_m^{global} denotes the global network parameter, and w_m^k represents the local model of k-th agent in the m-th aggregation round. K' indicates the number of agents that are filtered by threshold Ω and participate in the model aggregation. Once the global model is upgraded, it will be distributed to each agent for the soft update of each local model:

$$w_{m+1}^k = \lambda w_m^k + (1-\lambda) w_m^{\text{global}}, \tag{13}$$

where w_{m+1}^k represents the local model of k-th agent in the $(m+1)$-th aggregation round. To realize the sensor-level mapless navigation problem, the proposed FGRL approach combines TD3 with our model aggregation strategy, and Algorithm 1 details the computing pseudo-code.

Algorithm 1: Federated growing reinforcement learning algorithm. (K agents are indexed by k, M global aggregation rounds are indexed by m, E is the number of local episodes.)

Initialize $2K$ critic networks $Q_{\theta_{11,\cdots,1K}}$, $Q_{\theta_{21,\cdots,2K}}$ and K actor networks $\pi_{\phi_{1,\cdots,K}}$, initialize $2K$ target critic networks $Q_{\theta'_{11,\cdots,1K}}$, $Q_{\theta'_{21,\cdots,2K}}$ and K target actor networks $\pi_{\phi'_{1,\cdots,K}}$, and initialize replay buffer $B_{1,\cdots,K}$.
for *each global aggregation round* $m = 1, 2, \cdots, M$ **do**
  for *each agent* k *in parallel* **do**
    for $t = 1, 2, \cdots$ **do**
      Select action with noise $a_k \sim \pi_{\phi_k}(s) + \epsilon, \epsilon \sim N(0, \sigma)$ and store transition tuple (s_k, a_k, r_k, s'_k) in B_k
      Sample mini-batch of N transition (s_k, a_k, r_k, s'_k) from B_k
      $\tilde{a_k} \leftarrow \pi_{\phi'_k}(s') + \epsilon, \epsilon \sim clip(N(0, \tilde{\sigma}), -c, c)$
      $y_k \leftarrow r_k + \gamma min_{i=1,2} Q_{\theta'_{ik}}(s'_k, \tilde{a_k})$
      Update critics $\theta_{ik} \leftarrow argmin\theta_{ik} N^{-1} \sum (y_k - Q_{\theta_{ik}}(s_k, a_k))^2$
      if t *mod* d **then**
        Update ϕ_k by the deterministic policy gradient: $\nabla_{\phi_k} J(\phi_k) = N^{-1} \sum \nabla_a Q_{\theta_{1k}}(s_k, a_k)|_{a=\pi_{\phi_k}(s)} \nabla_{\phi_k} \pi_{\phi_k}(s_k)$
        Update target networks $\theta'_{ik} \leftarrow \tau\theta_{ik} + (1-\tau)\theta'_{ik}$, $\phi'_k \leftarrow \tau\phi_k + (1-\tau)\phi'_k$
  for *each agent* k *in parallel* **do**
    if $m = 1$ **then**
      After E local episodes calculate $\zeta_m^k, \varphi_m^k, \eta_m^k = 1$
    else
      After E local episodes calculate ζ_m^k, φ_m^k
      Update growth value
      $g_m^k = \psi \frac{\varphi_m^k - \varphi_{m-1}^k}{\varphi_m^k} + (1-\psi) \frac{\zeta_{m-1}^k - \zeta_m^k}{\zeta_{m-1}^k}$
      $\eta_m^k = \frac{1}{1+e^{-g_m^k}}$
    Global model aggregation $w_m^{\text{global}} = \sum_{k=1}^{K'} \frac{\eta_m^k}{\sum_{k=1}^{K'} \eta_m^k} w_m^k$
    Local model soft update $w_{m+1}^k = \lambda w_m^k + (1-\lambda) w_m^{\text{global}}$

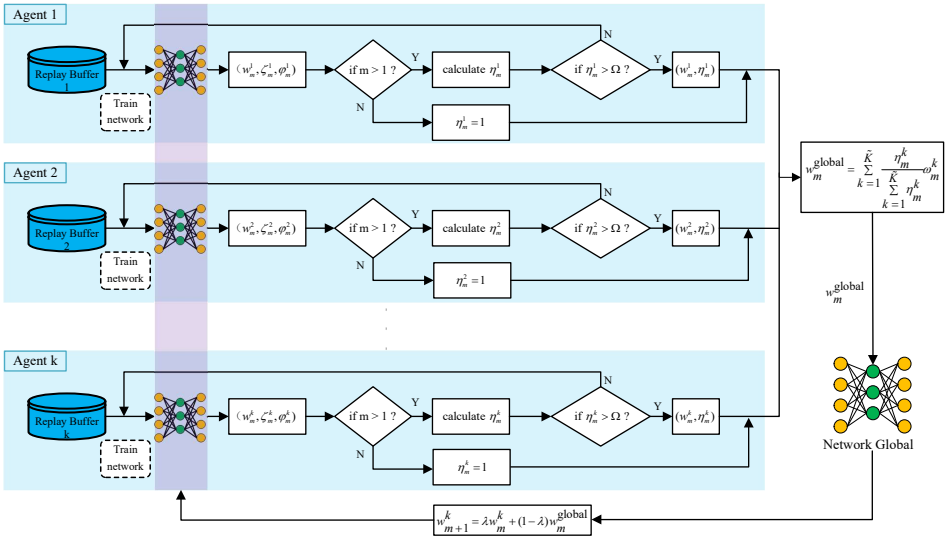
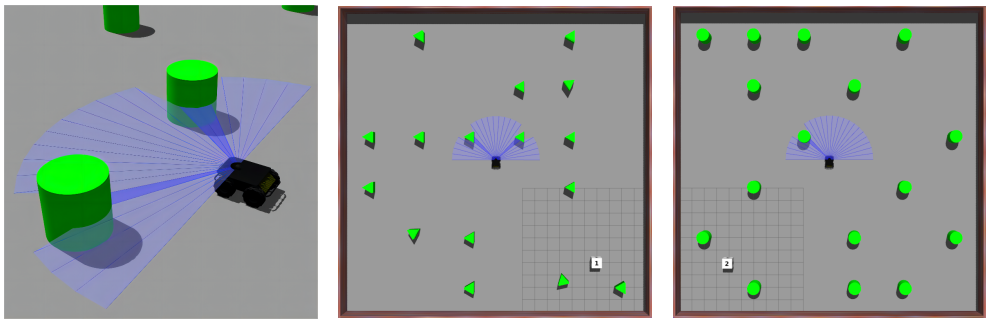


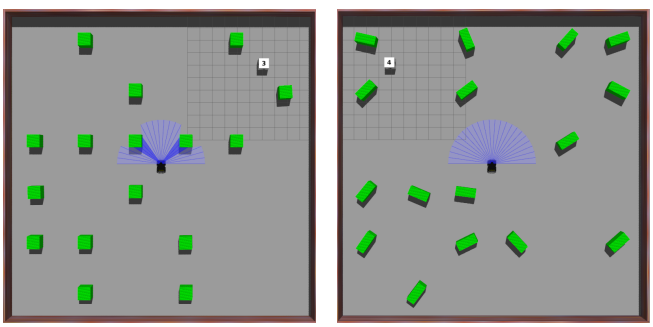
Figure 3. Illustration of model aggregation of the proposed FGRL approach.

4. Experiments and Discussion

To assess the efficiency and resilience of our proposed FGRL, we have implemented and compared it to a baseline, the TD3-based mapless navigation approach. In addition, we enhanced the baseline by integrating federated learning (FL), resulting in the FL-TD3 approach. This approach facilitates the simultaneous training of multiple agents and establishes a global model by averaging local models. Therefore, this section aims to evaluate the performance of the three approaches, i.e., TD3, FL-TD3, and our FGRL. The experiments are conducted in the Gazebo environment, utilizing UGVs equipped with a 2D LiDAR sensor with a 180-degree field-of-view and 24 laser beams, as depicted in Figure 4a. The laser scanner updates at a rate of 30 Hz, providing situational awareness of the environment.



(**a**) UGV with a 2D LiDAR sensors. (**b**) Environment 1 (env1). (**c**) Environment 2 (env2).

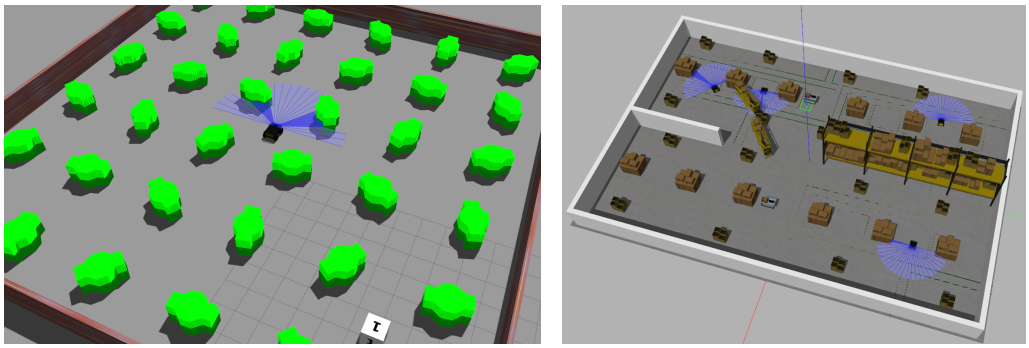


(**d**) Environment 3 (env3). (**e**) Environment 4 (env4).

Figure 4. Training and testing environments for mapless navigation tasks.

4.1. Experiment Setup

Our experiments involve the design of six distinct scenarios, which are categorized as either familiar or unfamiliar. The familiar group comprises four scenarios, as presented in Figure 4. During the training period, UGVs are trained in these environments before evaluating the performance of mapless navigation. Therefore, UGVs are familiar with the scenarios in this group. Conversely, the unfamiliar group (Figure 5a,b) consists of environmental configurations that the UGVs have never encountered before. This enables the evaluation of the knowledge transfer and generalization capabilities in unknown and unfamiliar environments. As illustrated in Figure 4, each UGV will be initialized at the center of each scenario during the training period, with a randomly assigned target location at one of the corners. The obstacle locations are also randomly generated for each run. Specifically, the positions of the obstacles are uniformly distributed across the map, and their orientations are randomly sampled from the interval $[0, \pi]$ to introduce variability in obstacle configurations. The target location is represented by the white square, the green objects depict the obstacles, and the blue lines indicate the laser beams of each robot.



(**a**) Plain setting (env5). (**b**) Factory setting (env6).

Figure 5. Trained models are evaluated in unfamiliar environments.

For the creation of simulation environments used in training and evaluation, we utilize ROS Melodic and Gazebo 9. The workstation used for these tasks is equipped with an Intel Xeon 3.8 GHz processor and an Nvidia GeForce RTX 3060 GPU. PyTorch is employed for the implementation of neural networks, while the hyper-parameters used in this work are listed in Table 1.

Table 1. Hyper-parameters.

Symbol	Definition	Value
α	learning rate	0.0004
γ	discount factor	0.99
δ	adjustment coefficient of the collision	-15
μ	adjustment coefficient of the distance reward	500
E	number of the local episodes	10
M	number of the global update periods	40
α	adjustment coefficient of the growth value	0.5
Ω	threshold of the growth value	0.1
λ	adjustment coefficient of the soft update	0.9

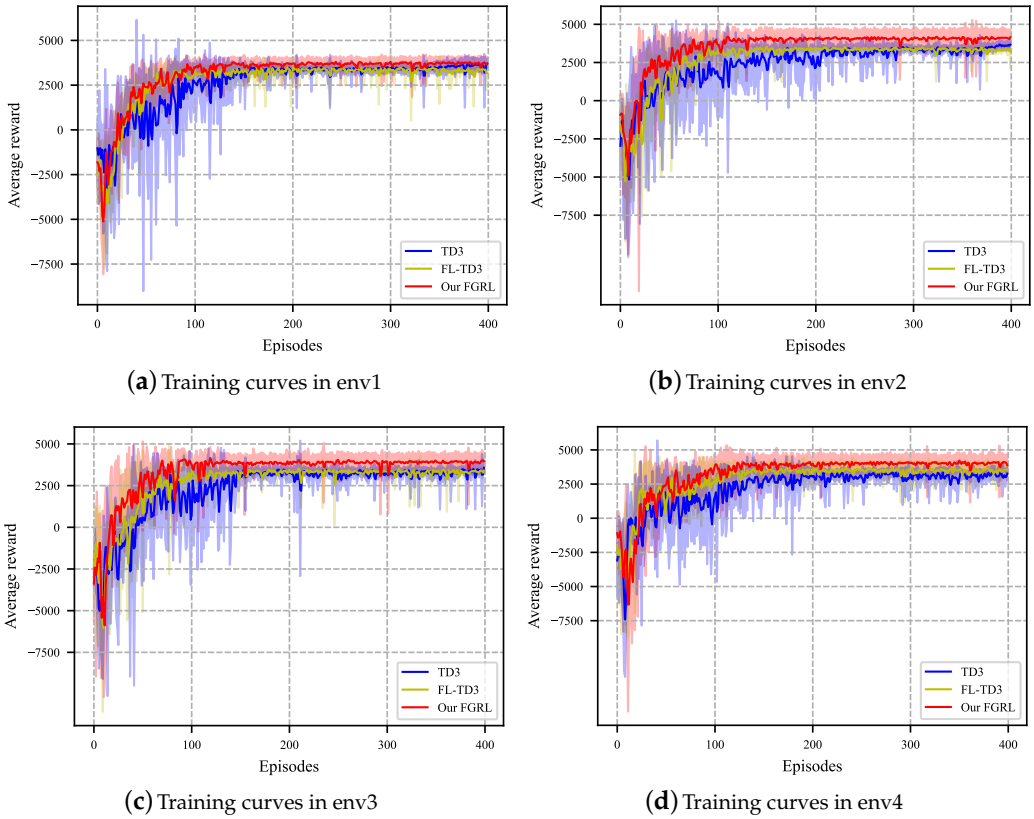
To evaluate the performance of the approaches, we consider several metrics, namely the success rate, collision rate, timeout rate, and average successful steps. The success rate indicates the percentage of total rounds in which the robot successfully reaches its target location without experiencing any collisions. On the other hand, the collision rate measures the percentage of total rounds where the robot collides with an obstacle. The timeout rate indicates the percentage of total rounds in which the robot cannot reach its target

location within a specified time but remains functional. Finally, the average successful steps measure the average path length taken by the robot to reach the target location successfully.

4.2. Training Performance

During the training episodes, the TD3 approach is executed for 400 episodes each for env1 to env4. The resulting converged models are named TD3-env1, TD3-env2, TD3-env3, and TD3-env4, respectively. For the FL-TD3 approach, we enable four UGVs to operate simultaneously in env1 to env4. After 400 episodes, we can obtain a global model. Similarly, we train the FGRL approach with four UGVs running in parallel in the same way. After 400 training episodes, the global model for FGRL is also obtained.

In order to evaluate the training performance, we record the average rewards per round for both of them across env1 to env4. Figure 6 presents the training curves of three approaches, where the x-axis denotes the number of rounds and the y-axis denotes the average reward. The solid line and shaded regions depict the mean and standard deviation, respectively, derived from ten independent runs. In terms of the convergence of the average reward, FL-TD3 exhibits similar performance to TD3 in env1, env2, and env3 but outperforms TD3 in env4. Comparatively, our FGRL demonstrates the best performance across all four environments. As for the convergence speed, TD3 achieves convergence in approximately 150 episodes for env1 and 200 episodes for env2, env3, and env4. Conversely, both FL-TD3 and our FGRL converge in approximately 100 episodes.



(**a**) Training curves in env1 (**b**) Training curves in env2

(**c**) Training curves in env3 (**d**) Training curves in env4

Figure 6. Average reward during the training episodes.

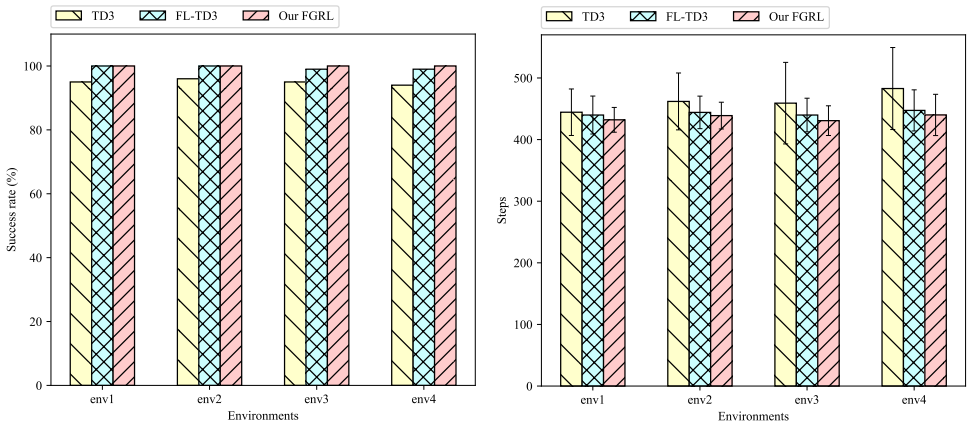
4.3. Evaluation in Familiar Environments

To evaluate the performance of the approaches in familiar environments, the trained models are tested 100 times across env1 to env4, and the results are presented in Table 2 and Figure 7. We can find that our FGRL and FL-TD3 achieve the success rate close to 100%, while TD3 achieves about 95%. This indicates that the combination of federated learning can indeed improve the success rate of obstacle avoidance in navigation tasks. Moreover,

Figure 7b shows that TD3 has the largest average successful steps, indicating that during successful runs, it takes longer steps to reach the target location in comparison with other approaches. The average successful steps of our FGRL are slightly smaller than those of FL-TD3. Table 2 shows that our FGRL can decrease the average successful steps by 5.7% compared to TD3, and by 1.7% compared to FL-TD3 across the four environments.

Table 2. The results of three approaches in familiar environments.

Algorithm	Metric	env1	env2	env3	env4
TD3	Success rate	95%	96%	95%	94%
	Average successful steps	444	461	459	482
FL-TD3	Success rate	100%	100%	99%	99%
	Average successful steps	440	444	439	447
FGRL	Success rate	100%	100%	100%	100%
	Average successful steps	432	438	430	440



(**a**) Success rate. (**b**) Average successful steps.

Figure 7. The success rate and average successful steps of three approaches in familiar environments.

4.4. Evaluation in Unfamiliar Environments

Here, we will further investigate the generalization ability and the performance of trained models in unfamiliar environments, in which the UGVs have never encountered these types of obstacles during training runs. Figure 5a shows a new dense setting (env5), where the obstacles have different shapes and sizes compared to env1 to env4, while Figure 5b presents another unfamiliar factory setting (env6), where UGVs may encounter more complex obstacles, such as shelves with slender legs.

4.4.1. Unfamiliar Plain Environment

After the training process, we can obtain 6 models because TD3 has trained separately across env1 to env4. Each model was tested 100 times in env5, and the results are presented in Figure 8 and Table 3.

During the training and testing experiments, it should be noted that the shape and size of the obstacles gradually become more and more complex from env1 to env5. As shown in Figure 8, when the trained models are applied to resolve unfamiliar scenarios, the success rate of TD3 and FL-TD3 tends to decline. The experimental results show that although the trained models of TD3 and FL-TD3 can perform well in the training scenarios (i.e., env1 to env4), they are still prone to collide with unfamiliar obstacles in env5.

According to the performance metrics, the trained model obtained from TD3-env1 performs the worst and cannot complete the task at all in env5. The success rate of TD3-env2 is 76%, while TD3-env3 and TD3-env4 can achieve about 90%, with the collision rate being less than 5%. Comparatively, the success rate of FL-TD3 and our FGRL is higher than

that of TD3. In particular, our FGRL is successful in all 100 testing runs. The results indicate that the combination of federated learning can indeed improve the generalization ability of UGVs to avoid unknown and unfamiliar obstacles. With regard to the average successful steps, our FGRL reduces the average successful steps by 13.6% and 30.9%, in comparison with FL-TD3 and TD3-env4, respectively. Fewer successful steps can indicate that the UGV takes fewer steps to reach the target location. In other words, the navigation path is shorter and more efficient. Therefore, we can conclude that our proposed FGRL reduces the path length without compromising the success rate.

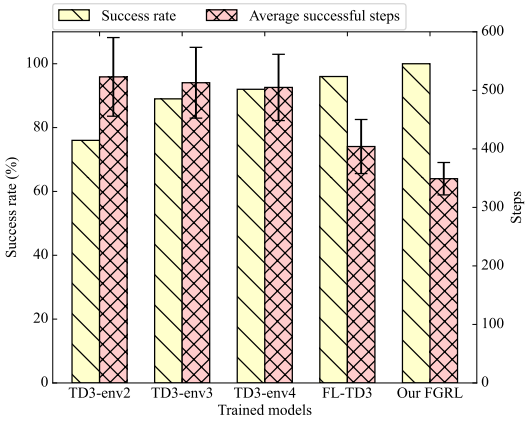


Figure 8. The success rate and average successful steps in env5.

Table 3. The evaluation results in unfamiliar plain environment.

Trained Model	Success Rate	Crash Rate	Timeout Rate	Average Successful Steps
TD3-env1	0%	84%	16%	-
TD3-env2	76%	8%	16%	523
TD3-env3	89%	3%	8%	513
TD3-env4	92%	2%	6%	505
FL-TD3	96%	0%	4%	404
FGRL	100%	0%	0%	349

4.4.2. Unfamiliar Factory Environment

In this scenario (named env6), all the trained models are also tested 100 times, and the results are presented in Figure 9 and Table 4.

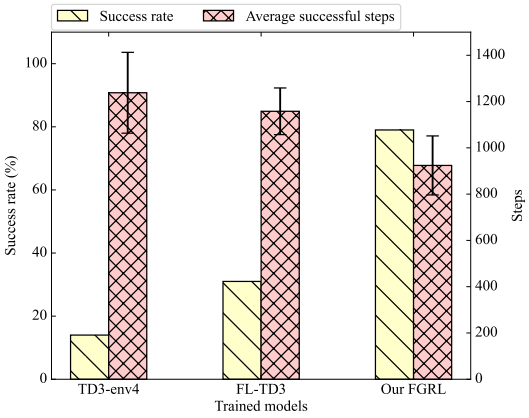


Figure 9. The success rate and average successful steps in env6.

Table 4. The evaluation results in unfamiliar factory environment.

Trained Model	Success Rate	Crash Rate	Timeout Rate	Average Successful Steps
TD3-env1	0%	0%	100%	-
TD3-env2	0%	0%	100%	-
TD3-env3	1%	39%	60%	1449
TD3-env4	14%	37%	49%	1238
FL-TD3	31%	69%	0%	1158
FGRL	79%	21%	0%	924

We can find that the trained models of TD3-env1 and TD3-env2 are completely incapable of accomplishing the navigation task in env6. In addition, the success rate of TD3-env3 and TD3-env4 is only 1% and 14%, respectively. In comparison, the success rate of FL-TD3 has greatly improved, but it is still very low: only 31%. Our FGRL achieves a success rate of 79%, which greatly exceeds the other approaches. With respect to the average successful steps, compared to FL-TD3 and TD3-env4, our FGRL reduces the path length by 20% and 25%, respectively.

We can conclude that the trained models of TD3 from env1 to env4 can hardly generalize to a new and unfamiliar factory environment, and it is difficult for UGVs to select reasonable actions based on the trained policy. Although our FGRL model was also trained across env1 to env4, it still has good generalization ability when encountering unfamiliar and complex obstacles.

4.5. Evaluation in Real-World Environments

To evaluate the effectiveness of the proposed method in real-world environments, we deploy control models trained using FGRL on two TurtleBot3 robots equipped with a 2D LiDAR sensor. The experiment is conducted with four obstacles of varying shapes, while the robots also serve as dynamic obstacles for each other.

The robots start at opposite ends of the map, facing each other. The target locations are positioned at the lower-left and upper-right corners of the map. A snapshot of the experiment is shown in Figure 10.

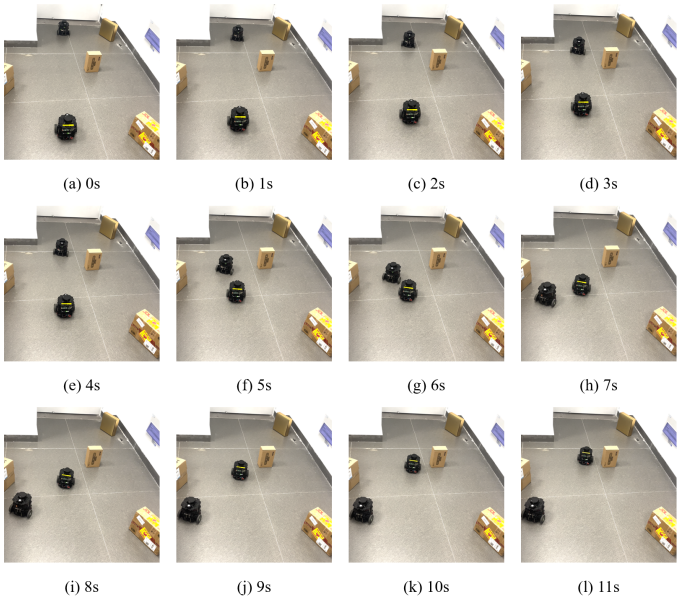


Figure 10. Snapshots of real robot experiments.

Both robots begin moving from their starting positions. After 5 s, they encounter the obstacles and begin to avoid them. By 7 s, both robots successfully navigate around

the obstacles and head toward their target locations. By 11 s, both robots reach their respective targets.

To assess the reliability of the method, we conducted 20 repeated trials using the same setup. The navigation success rate and the time taken for each robot to reach its target location were recorded. Across all 20 trials, the navigation success rate was 85%, with the average time for successful navigation being 11.89 s. These results demonstrate that the proposed method is effective in real-world environments and that the FGRL approach is adaptable and robust in unfamiliar settings.

5. Conclusions

In this work, we propose a federated growing reinforcement learning (FGRL) approach to address the resilient mapless navigation problem. One of the benefits of this approach is that it enables UGVs to improve training efficiency through asynchronous parallel training. Moreover, by aggregating models through federated learning, the generalization ability of the global model can be improved, which can enable robots to make rational decisions based on sensor-level perceptions when encountering unknown and unfamiliar obstacles.

To verify the advantages of our FGRL in comparison with TD3 and FL-TD3, we first evaluate the performance of three approaches in familiar environments that are the same as the training settings. Then, we create two completely different settings with complex obstacles that the UGVs have never encountered during the training process. The results indicate that our FGRL can indeed greatly improve the generalization ability of UGVs facing unknown situations with unfamiliar obstacles, while also taking account of the consumption of path costs.

Although our FGRL approach has demonstrated promising results in UGV navigation and obstacle avoidance, it has certain limitations. A primary challenge lies in its dependence on a single 2D LiDAR for obstacle detection and environmental perception. Since 2D LiDAR is limited to detecting obstacles within its scanning plane, its effectiveness in complex environments is constrained. Moreover, the training and testing environments used in our study included only convex objects, leaving the method's performance with concave objects untested.

In future research, we will aim to integrate LiDAR and depth image data to improve UGVs' environmental perception and navigation capabilities. We also plan to include concave obstacles in training and testing environments to evaluate their impact on navigation strategies and to explore optimization methods for handling such scenarios effectively.

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

Funding: This research was funded by the National Natural Science Foundation of China, grant number 52371275.

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

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

References

1. Mabkhot, M.M.; Al-Ahmari, A.M.; Salah, B.; Alkhalefah, H. Requirements of the smart factory system: A survey and perspective. *Machines* **2018**, *6*, 23. [CrossRef]
2. Xue, H.; Hein, B.; Bakr, M.; Schildbach, G.; Abel, B.; Rueckert, E. Using deep reinforcement learning with automatic curriculum learning for mapless navigation in intralogistics. *Appl. Sci.* **2022**, *12*, 3153. [CrossRef]
3. Kriegel, J.; Rissbacher, C.; Reckwitz, L.; Tuttle-Weidinger, L. The requirements and applications of autonomous mobile robotics (AMR) in hospitals from the perspective of nursing officers. *Int. J. Healthc. Manag.* **2022**, *15*, 204–210. [CrossRef]

4. Zhao, Y.L.; Hong, Y.T.; Huang, H.P. Comprehensive Performance Evaluation between Visual SLAM and LiDAR SLAM for Mobile Robots: Theories and Experiments. *Appl. Sci.* **2024**, *14*, 3945. [CrossRef]
5. Blochliger, F.; Fehr, M.; Dymczyk, M.; Schneider, T.; Siegwart, R. Topomap: Topological mapping and navigation based on visual slam maps. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018 ; IEEE: New York, NY, USA, 2018; pp. 3818–3825.
6. Wang, J.; Chi, W.; Li, C.; Wang, C.; Meng, M.Q.H. Neural RRT*: Learning-based optimal path planning. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1748–1758. [CrossRef]
7. Su, Y.; Wang, T.; Shao, S.; Yao, C.; Wang, Z. GR-LOAM: LiDAR-based sensor fusion SLAM for ground robots on complex terrain. *Robot. Auton. Syst.* **2021**, *140*, 103759. [CrossRef]
8. Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature* **2020**, *588*, 604–609. [CrossRef]
9. Perolat, J.; De Vylder, B.; Hennes, D.; Tarassov, E.; Strub, F.; de Boer, V.; Muller, P.; Connor, J.T.; Burch, N.; Anthony, T.; et al. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science* **2022**, *378*, 990–996. [CrossRef]
10. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Proceedings of the Conference on Robot Learning, PMLR, Zürich, Switzerland, 29–31 October 2018; pp. 651–673.
11. Kilinc, O.; Montana, G. Reinforcement learning for robotic manipulation using simulated locomotion demonstrations. *Mach. Learn.* **2022**, *111*, 465–486. [CrossRef]
12. Pintos Gómez de las Heras, B.; Martínez-Tomás, R.; Cuadra Troncoso, J.M. Self-Learning Robot Autonomous Navigation with Deep Reinforcement Learning Techniques. *Appl. Sci.* **2023**, *14*, 366. [CrossRef]
13. Patel, U.; Kumar, N.K.S.; Sathyamoorthy, A.J.; Manocha, D. DWA-RL: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May 30–5 June 2021; IEEE: New York, NY, USA, 2021; pp. 6057–6063.
14. Chen, T.; Zhang, K.; Giannakis, G.B.; Başar, T. Communication-efficient policy gradient methods for distributed reinforcement learning. *IEEE Trans. Control Netw. Syst.* **2021**, *9*, 917–929. [CrossRef]
15. Ma, C.; Zhang, J.; Liu, J.; Ji, L.; Gao, F. A parallel multi-module deep reinforcement learning algorithm for stock trading. *Neurocomputing* **2021**, *449*, 290–302. [CrossRef]
16. Liu, B.; Wang, L.; Liu, M. Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4555–4562. [CrossRef]
17. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
18. Wenzel, P.; Schön, T.; Leal-Taixé, L.; Cremers, D. Vision-based mobile robotics obstacle avoidance with deep reinforcement learning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; IEEE: New York, NY, USA, 2021; pp. 14360–14366.
19. Han, Y.; Zhan, I.H.; Zhao, W.; Pan, J.; Zhang, Z.; Wang, Y.; Liu, Y.J. Deep Reinforcement Learning for Robot Collision Avoidance With Self-State-Attention and Sensor Fusion. *IEEE Robot. Autom. Lett.* **2022**, *7*, 6886–6893. [CrossRef]
20. Jang, Y.; Baek, J.; Han, S. Hindsight Intermediate Targets for Mapless Navigation with Deep Reinforcement Learning. *IEEE Trans. Ind. Electron.* **2021**, *69*, 11816–11825. [CrossRef]
21. Tai, L.; Paolo, G.; Liu, M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; IEEE: New York, NY, USA, 2017; pp. 31–36.
22. Marchesini, E.; Farinelli, A. Discrete deep reinforcement learning for mapless navigation. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: New York, NY, USA, 2020; pp. 10688–10694.
23. Long, P.; Fan, T.; Liao, X.; Liu, W.; Zhang, H.; Pan, J. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; IEEE: New York, NY, USA, 2018; pp. 6252–6259.
24. Hadidi, R.; Cao, J.; Woodward, M.; Ryoo, M.S.; Kim, H. Distributed perception by collaborative robots. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3709–3716. [CrossRef]
25. Clemente, A.V.; Castejón, H.N.; Chandra, A. Efficient parallel methods for deep reinforcement learning. *arXiv* **2017**, arXiv:1705.04862.
26. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York City, NY, USA, 19–24 June 2016; pp. 1928–1937.
27. Xu, M.; Shen, Y.; Zhang, S.; Lu, Y.; Zhao, D.; Tenenbaum, J.; Gan, C. Prompting decision transformer for few-shot policy generalization. In Proceedings of the International Conference on Machine Learning (ICML), PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 24631–24645.

28. Fan, T.; Long, P.; Liu, W.; Pan, J.; Yang, R.; Manocha, D. Learning resilient behaviors for navigation under uncertainty. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: New York, NY, USA, 2020; pp. 5299–5305.
29. Imteaj, A.; Amini, M.H. Fedar: Activity and resource-aware federated learning model for distributed mobile robots. In Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 14–17 December 2020; IEEE: New York, NY, USA, 2020; pp. 1153–1160.
30. Tursunboev, J.; Kang, Y.S.; Huh, S.B.; Lim, D.W.; Kang, J.M.; Jung, H. Hierarchical Federated Learning for Edge-Aided Unmanned Aerial Vehicle Networks. *Appl. Sci.* **2022**, *12*, 670. [CrossRef]
31. Zhou, X.; Liang, W.; She, J.; Yan, Z.; Kevin, I.; Wang, K. Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5308–5317. [CrossRef]
32. Mohri, M.; Sivek, G.; Suresh, A.T. Agnostic federated learning. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 4615–4625.
33. Wang, Y.; Kantarci, B. Reputation-enabled federated learning model aggregation in mobile platforms. In Proceedings of the IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; IEEE: New York, NY, USA, 2021; pp. 1–6.
34. Majcherczyk, N.; Srishankar, N.; Pinciroli, C. Flow-fl: Data-driven federated learning for spatio-temporal predictions in multi-robot systems. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; IEEE: New York, NY, USA, 2021; pp. 8836–8842.
35. Wang, H.; Kaplan, Z.; Niu, D.; Li, B. Optimizing federated learning on non-iid data with reinforcement learning. In Proceedings of the IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; IEEE: New York, NY, USA, 2020; pp. 1698–1707.
36. Zhang, P.; Wang, C.; Jiang, C.; Han, Z. Deep reinforcement learning assisted federated learning algorithm for data management of IIoT. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8475–8484. [CrossRef]
37. Yu, S.; Chen, X.; Zhou, Z.; Gong, X.; Wu, D. When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. *IEEE Internet Things J.* **2020**, *8*, 2238–2251. [CrossRef]
38. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
39. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York City, NY, USA, 19–24 June 2016; pp. 1995–2003.
40. 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.
41. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1587–1596.