

Received 7 March 2024, accepted 16 April 2024, date of publication 19 April 2024, date of current version 26 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3391408

RESEARCH ARTICLE

Feedback-Based Curriculum Learning for Collision Avoidance

JEONGMIN CHOI¹, GYUYONG HWANG², AND GYUHO EOH³

¹Department of Mechanical Design Engineering, Tech University of Korea, Siheung-si, Gyeonggi-do 15073, South Korea

²Department of Electronics Engineering, Tech University of Korea, Siheung-si, Gyeonggi-do 15073, South Korea

³Department of Mechatronics Engineering, Tech University of Korea, Siheung-si, Gyeonggi-do 15073, South Korea

Corresponding author: Gyuhoh Eoh (gyuhoh.eoh@tukorea.ac.kr)

This work was supported by the GRRC program of Gyeonggi province [GRRC TUKorea2023-B03, Development of an intelligent inspection system and an autonomous navigation system for the transportation of multi-material parts].

ABSTRACT This paper proposes a novel curriculum learning approach for collision avoidance using feedback from the deep reinforcement learning (DRL) training process. Previous research on DRL-based collision avoidance algorithms has encountered challenges such as long training times and difficulty in convergence due to sparse rewards. To address these issues, curriculum learning has been used to divide the target task into multiple subtasks for training. However, manual or random curriculum design often generates unnecessary subtasks that do not improve performance. Furthermore, a standardized curriculum design method for collision avoidance has not yet been presented. Therefore, this paper introduces a curriculum-based collision avoidance learning method that utilizes feedback during the training phase. The proposed method differs from traditional curriculum learning in that the subtask is not predetermined before training. Instead, the curriculum is modified during training based on feedback obtained from validation environments. If a robot demonstrates high collision avoidance performance in a validation environment, it is then validated in more challenging environments for rigorous evaluation. Conversely, if collision avoidance performance is low in the validation environment, the robot is trained in a new environment to overcome frequent collision situations. Simulations and practical experiments were conducted for the proposed method, which showed better performance compared to the non-curriculum method.

INDEX TERMS Collision avoidance, curriculum learning, deep reinforcement learning, subtask generation.


I. INTRODUCTION

Recently, autonomous mobile robots (AMRs) have found extensive applications in various fields, such as logistics [1], warehousing [2], and service robotics [3]. For the wider use of AMRs, robots should be able to navigate in unknown and complex environments by avoiding obstacles and reaching the target destination. There are several collision avoidance algorithms for AMRs that can generate suboptimal or near-optimal paths to the target [4], [5]. However, practical implementation is challenging because path generation for collision avoidance can be computationally expensive in highly complex environments [6], [7].

Deep reinforcement learning (DRL) has shown promise in addressing these challenges, as it can automatically learn

meaningful features and patterns in the various training environments [8]. Once the training is complete, the robot is able to determine its actions without excessive computation [9], [10], [11]. However, DRL-based collision avoidance methods often suffer from overfitting to specific training environments and require significant computation in the training process [12]. Moreover, the sparsity of rewards can result in long training times or low convergence during learning [13].

To overcome these drawbacks of DRL, recent developments in the field of DRL have introduced curriculum learning to accelerate the training speed and improve the learning performance [14], [15]. In curriculum learning, the robot first learns relatively simple tasks and then builds on the previously acquired knowledge to tackle more challenging tasks. By focusing on the experiences necessary to achieve the target performance, curriculum learning

The associate editor coordinating the review of this manuscript and approving it for publication was P. Venkata Krishna .

reduces training time and improves performance [16], [17]. However, curriculum learning is particularly useful only when the learning order of the training data is well organized; the training difficulty should increase gradually for stable learning; the curriculum should be carefully designed from easy to difficult. If a curriculum is not well designed, the performance is sometimes worse than not using a curriculum at all. In addition, there is no known standardized way to design a curriculum [18].

Therefore, we propose a feedback-based curriculum design method for collision avoidance. The collision avoidance performance is evaluated in *validation environments* at each training phase of the curriculum; the validation environments are different from the training environments used for learning. If the collision avoidance performance in the validation environments is high, i.e., success feedback, the robot is validated in the next validation environment to find the shortcomings of the current learning level. On the other hand, if the collision avoidance performance in the validation environment is low, i.e., fail feedback, the curriculum is redesigned by configuring a new subtask aimed at overcoming situations where collisions frequently occur; a new training environment is created for training. In other words, higher performance compared to a fixed curriculum can be expected by incorporating feedback into the generation of the training environment.

The contributions of this paper are as follows:

- 1) We propose a feedback-based curriculum design method for collision avoidance using DRL.
- 2) To compensate for the deficiencies found during the learning phase, we propose an adaptive method for generating subtasks and training environments.
- 3) We propose DRL-based collision avoidance that can operate in a variety of environments using only local sensing.
- 4) We verify the collision avoidance performance of the robot in both simulation and real environments.

This paper is organized as follows: section II presents the related work on DRL and curriculum learning. Section III defines the problem addressed in this study and highlights its importance. Section IV presents the overall system architecture, and section V describes the proposed curriculum design method. Sections VI and VII verify the proposed method through simulation and real experiments, respectively. The discussion of the proposed method is presented in section VIII, and the conclusion is described in section IX.

II. RELATED WORK

A. DEEP REINFORCEMENT LEARNING-BASED COLLISION AVOIDANCE

The DRL-based collision avoidance algorithms have been applied in various applications such as mobile robots, unmanned aerial vehicles (UAVs), and unmanned surface vehicles (USVs).

In the field of mobile robotics, Chen et al. [19] proposed DRL-based multi-agent collision avoidance by using the positions, velocities, and target locations of all agents. Multiple robots were shown to use DRL without communication to reach their destinations by avoiding collisions. Xin et al. [20] suggested a DRL-based path planning method using deep Q-network (DQN) for end-to-end path planning of mobile robots. Xue et al. [21] proposed a double deep Q-network (DDQN)-based collision avoidance algorithm using target locations and obstacle size and location information as inputs. This showed significant improvements in training speed compared to traditional methods such as CADRL and deep deterministic policy gradient (DDPG). Yao et al. [22] presented multi-robot collision avoidance using the proximal policy optimization (PPO) algorithm by mapping egocentric grid map frames to robot control commands. Srouji et al. [23] presented a collision avoidance approach that outperforms the dynamic window approach (DWA) in terms of safety and time efficiency; this was achieved by integrating autonomous emergency braking signals into the training process.

In a case study involving UAVs, He et al. [24] developed a novel learning framework by combining imitation learning and reinforcement learning to speed up the training process. Singla et al. [25] addressed the issue of partial observability in DRL by proposing a method based on recurrent neural network (RNN) architecture and temporal attention; a robot could avoid collisions using observations collected over specific time intervals. Roghair et al. [26] achieved collision avoidance by combining convolutional neural networks with the soft actor critic (SAC) algorithm to improve uncertain image processing.

In the field of USVs, Chun et al. [27] demonstrated the stability of a DRL-based collision avoidance algorithm in unexpected situations compared to the A* algorithm, using ship information such as position, speed, direction, and collision risk as inputs. Xu et al. [28] proposed a cumulative priority sampling mechanism to address the training data inefficiency of the DDPG algorithm. The resulting path planning and dynamic collision avoidance (PPDC) algorithm was shown to have superior real-time performance in collision avoidance.

The above mentioned DRL-based collision avoidance methods have the advantage of being able to learn to avoid collisions in complex and unpredictable environments. However, challenges remain, such as the need for massive amounts of data for convergence and inefficiencies in the sampling of experience data. In addition, learning requires a long training times due to the sparse reward problem, which can be problematic and requires careful manual tuning of the reward function [21].

B. CURRICULUM LEARNING

Curriculum learning can be divided into two main categories based on how a robot determines the order of the experience samples and tasks.

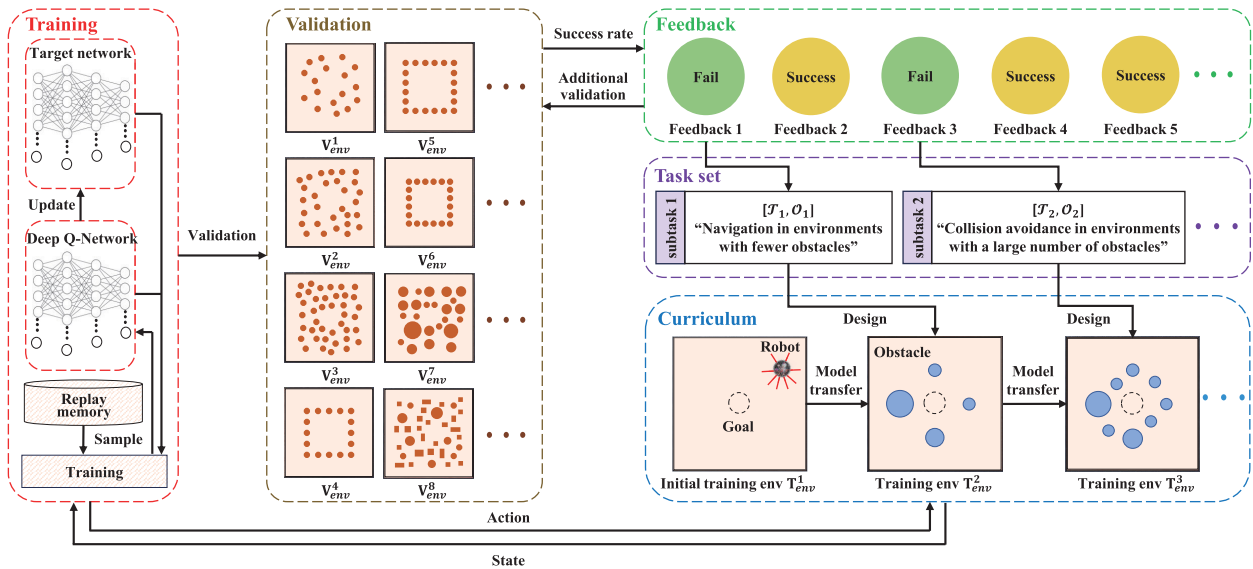


FIGURE 1. Overview of the proposed curriculum learning method using feedback. A robot is trained with the training environment T_{env}^i , and then a robot is evaluated in the validation environment V_{env}^i . If the success rate of the validation environment is higher than $P_{Success}$, the robot is evaluated in the next validation environment V_{env}^{i+1} . Conversely, if the success rate is less than $P_{Success}$, a new subtask is generated and the robot is trained in an environment corresponding to the subtask, T_{env}^{i+1} . This process continues until the robot has passed all validation environments.

First, a method was proposed to perform sampling based on the priority of the experience [29]; they prioritized data samples using the temporal difference (TD) error in DRL, resulting in accelerated learning and improved performance. Niu et al. [30] applied the prioritized experience replay (PER) algorithm to collision avoidance, improving training speed. Ren et al. [31] increased the sampling efficiency and reduced the frequency of excessive transitions by exploiting the relationship between the difficulty of the current curriculum and the TD error; this approach was found to be more effective than the PER algorithm.

Second, there is the manual design for determining the order of tasks. Xiao et al. [32] demonstrated successful navigation through narrow gaps in a real environment by manually designing two curricula. The subtasks were designed incrementally from larger to smaller intervals to allow the robot to pass through the narrow gaps. Zhang et al. [33] designed a curriculum for encircling in a multi-robot system, starting with a small team encircling a target and gradually progressing to a large team. Chen et al. [15] performed curriculum learning by randomly generating obstacles and targets in static and dynamic environments. This method showed superior performance in terms of success rate, arrival time, and average linear speed compared to existing algorithms in simulated and real environments. Narvekar et al. [34] presented a method for dynamically generating task sets for curriculum learning by extracting multiple features related to the target task. In an extended study, the automatic generation methods of easy or difficult tasks to solve in the next order are proposed [35].

As mentioned above, curriculum learning can be an alternative to solve problems such as slow training speed

and sparse rewards [36]. However, manual curriculum design can lead to inappropriate learning due to heuristic design. Random curriculum design sometimes generates disruptive or unnecessary tasks, making it difficult to design an appropriate curriculum [37]. To overcome these problems, automatic curriculum design methods have been proposed. The robot can dynamically adjust the task difficulty based on the robot’s performance, which provides a more appropriate curriculum design compared to the traditional methods [38]. Eoh and Park [39] proposed an automatic curriculum method for object transportation by generating the difficulty map. The curriculum is generated adaptively from easy to difficult environments for object transportation. However, the automatic curriculum is a lack of standardized curriculum design methods, and the curriculum is still highly dependent on manually specified hyperparameters [40].

III. PROBLEM FORMULATION

The problem formulation of this paper is as follows. The subtask \mathcal{T}_i and the order \mathcal{O}_i ($i = 1, \dots, N_{env}^{train}$) are defined as $\chi_i = [\mathcal{T}_i, \mathcal{O}_i]$, where N_{env}^{train} is the total number of training environments. A curriculum \mathcal{C} is a set of χ : $\mathcal{C} = \{\chi_1, \dots, \chi_{N_{env}^{train}}\}$. The approach to designing the curriculum \mathcal{C} involves defining the order \mathcal{O}_i in which certain subtask \mathcal{T}_i is to be learned in the curriculum. Therefore, the problem is formulated as follows:

$$\mathcal{C}^* = \arg \max_{\mathcal{C}} \mathbb{E} [p_{success}^{\mathcal{C}} | \mathcal{C}], \quad (1)$$

where $p_{success}^{\mathcal{C}}$ is the probability that the robot will reach a goal without colliding with obstacles if it follows the curriculum \mathcal{C} . The objective of this study is to design a curriculum \mathcal{C}^* that

maximizes the expected collision avoidance success rate for the robot when learning the task \mathcal{T}_i in its order \mathcal{O}_i during the N_{env}^{train} training phases.

This paper makes the following four assumptions. First, we assume the use of a mobile robot in a 2D Euclidean plane. Second, we assume that the robot has a completely circular shape. Even if the robot is not circular in shape, the proposed method can be applied by approximating it as a circular shape. Third, we assume that the positions of the robot and the target are known in real time. Finally, we assume that the robot has the ability to detect obstacles through its own proximity sensors.

IV. SYSTEM OVERVIEW

The proposed feedback-based curriculum learning consists of 5 components, as shown in Fig. 1: *training*, *validation*, *feedback*, *task set*, and *curriculum*. Initially, a robot is trained using DQN in a training environment T_{env}^1 by the training and curriculum components. The initial training environment T_{env}^1 is an obstacle-free environment; the robot learns how to move to a goal without considering obstacles. After training in T_{env}^1 , the robot's collision avoidance performance is evaluated in the validation environments from V_{env}^1 to $V_{env}^{N_{val}}$ at the validation component. Various validation environments can be used to validate the current learning model. For each validation environment, the robot receives feedback indicating the success rate of collision avoidance; this feedback is used to generate a new training environment. For example, if the success rate in the validation environment V_{env}^i is higher than $\mathbf{P}_{success}$, i.e. success feedback, the robot will be validated in the next validation environment V_{env}^{i+1} to evaluate in different obstacle distributions. The $\mathbf{P}_{success}$ indicates a probability that represents the success criteria of the collision avoidance. We continue in this way until the collision avoidance success rate is less than $\mathbf{P}_{success}$ or the evaluation is complete in all given validation environments. Note that validation environments are not used for training, only for validation. On the other hand, if the collision avoidance success rate in a particular validation environment becomes less than $\mathbf{P}_{success}$, i.e., fail feedback, a new subtask for collision avoidance learning is generated at the task set component. This new subtask is designed to overcome the failures of the validation phase, and is inserted into the curriculum during training. For example, if the robot receives feedback 1 at the feedback component, which is not capable of obstacle avoidance at all, we construct subtask \mathcal{T}_1 for navigating in environments with fewer obstacles; we then generate a training environment T_{env}^2 where there is a fewer obstacle. During this process, the robot can use what it learned in the previous step by transferring the knowledge from the first training environment T_{env}^1 to the next training environment T_{env}^2 . As described so far, the proposed curriculum learning method receives continuous feedback from the validation environments during the training process and uses it to design new training environments.

V. FEEDBACK-BASED CURRICULUM LEARNING DESIGN

This section discusses the details of feedback-based curriculum learning. First, the basic collision avoidance learning method using DQN is described in section V-A. Next, the curriculum-based collision avoidance method is described in section V-B. Finally, a generation method for validation and training environments is presented in section V-C.

A. COLLISION AVOIDANCE LEARNING USING DQN

We deal with situations where a robot moves towards its goal while avoiding obstacles. Since the costs and state transitions caused by collisions depend only on the current state of the robot and the action taken in that state, it can be defined as a Markov Decision Process (MDP) problem; this allows us to apply Q-learning, a standard reinforcement learning approach for MDPs. In Q-learning, the optimal action-value function $Q^*(s, a)$ selects actions such that future rewards are maximized as follows:

$$Q^*(s, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \dots + \gamma^{\infty} r_{\infty} | s_t = s, a_t = a, \pi] \quad (2)$$

where γ and r_t are the discount factor and the reward at time t , respectively. The goal of the Q-learning is to maximize the expected discounted cumulative reward of taking action a_t in state s_t according to the policy π . The DQN has a large number of weight parameters from Q-learning by constructing a deep neural network.

The weight parameters of the DQN are denoted θ_t and the loss function $\mathcal{L}_t(\theta_t)$ is defined based on the experience sample (s_t, a_t, r_t, s_{t+1}) , as shown in (3). The loss is calculated as the mean squared difference between the predicted Q-value for the current state-action pair (s_t, a_t) and the target Q-value. The target Q-value is the sum of the immediate reward r_t and the discounted maximum Q-value for the next state s_{t+1} using the target Q-network with parameters θ_t^- :

$$\mathcal{L}_t(\theta_t) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1})} \left[r_t + \gamma \max_a Q(s_{t+1}, a; \theta_t^-) - Q(s_t, a_t; \theta_t) \right]^2 \quad (3)$$

The state of a robot is described as follows:

$$s_t = [d_t^{r,g}, \theta_t^{r,g}, ps_t^1, ps_t^2, \dots, ps_t^{N_{ps}}], \quad (4)$$

where s_t is the state including the relative distance $d_t^{r,g}$ and the relative heading angle $\theta_t^{r,g}$ between the robot and the goal at time t . The sensor value ps_t^i is a measured distance obtained from the i^{th} proximity sensor; N_{ps} is the number of proximity sensors attached to the robot. Since proximity sensors can only detect when the robot is near an obstacle, the proposed method is local planning with local sensing. The final state used in the DQN is defined by concatenating three consecutive states: $\tilde{s}_t = [s_{t-2}, s_{t-1}, s_t]$ for $t \geq 3$.

Also, the robot can take 6 actions as follows:

$$a_t = [\mathbf{FF}, \mathbf{SF}, \mathbf{TR}, \mathbf{TL}, \mathbf{OR}, \mathbf{OL}], \quad (5)$$

where the symbols are described in Table 1. For the robot to move freely, we set forward and backward, right and left

TABLE 1. The action descriptions in (5).

Symbol	Description
FF	Fast forward
SF	Slow forward
TR	In-place turn right
TL	In-place turn left
OR	Right wheel turning while the left wheel is stationary
OL	Left wheel turning while the right wheel is stationary

rotation as actions, and added a rotation motion that keeps one wheel stationary to reduce the travel time.

The robot is rewarded in four cases. First, the robot receives a +100 reward for reaching the goal. Second, the robot receives a reward of -100 if it collides with obstacles. Third, the robot is rewarded based on its proximity to the goal, with a reward of $\omega(d_{t+1}^{r,g} - d_t^{r,g})$ for getting further or closer, where ω is a weight parameter. Finally, to discourage aimless wandering, with a time-based reward of -1 for each timestamp.

B. CURRICULUM LEARNING USING FEEDBACK

Algorithm 1 shows the proposed DQN-based curriculum learning method using feedback. Before starting DQN training, we randomly initialize the weights of the active and target action-value functions (lines 1–2). The training and validation indices are also initialized, and the replay memory for DQN is prepared (lines 3–5). There is a maximum number of training environments, N_{env}^{train} , which will stop learning if it is exceeded (lines 7–8). The robot performs an action a_t in the training environment $T_{env}^{id_{train}}$, receives a reward r_t , and transitions to the next state s_{t+1} according to the transition probability p_t (lines 12–14). The experience data (s_t, a_t, r_t, s_{t+1}) are stored in the replay memory (line 15). We perform gradient descent on θ by randomly sampling experience data from the replay memory (lines 17–23). For each C-step, we replicate the network θ to the target network θ^- (line 24). After completing the DQN training for the M_{ep} episodes, we evaluate its performance in the validation environments $V_{env}^i (i = 1, 2, \dots, N_{env}^{val})$ (lines 26–27). If the evaluation success rate exceeds $P_{success}$ in the validation environment V_{env}^i , the robot is evaluated in the next validation environment V_{env}^{i+1} (lines 28–29). On the contrary, if the success rate does not exceed $P_{success}$, the training method is divided into the following two cases. First, the robot is trained again in the current training environment $T_{env}^{id_{train}}$ if the robot shows the low performance in the validation environment where it performed well before (lines 31–33). Second, if the robot only performs poorly in validation environments that were not previously evaluated, create a new training environment $T_{env}^{id_{train}+1}$ that takes into account the problematic collision situation (line 35). In this case, the weight parameters of the Q-network trained in the previous step are transferred to the next training environment (line 36). Meanwhile, the values of M_{ep} , T_{step} , and N_{env}^{val} should be predetermined for the proposed curriculum learning, as shown in Algorithm 1. This means that even in the worst

Algorithm 1 Curriculum Learning Using Feedback

```

1 Initialize active action-value function  $Q$  with random
  weights  $\theta$ ;
2 Initialize target action-value function  $\hat{Q}$  with weights
   $\theta^- \leftarrow \theta$ ;
3 Initialize the training index:  $id_{train} \leftarrow 1$ ;
4 Initialize the validation index:  $id_{val} \leftarrow 1$ ;
5 Initialize replay memory  $\mathcal{D}$ ;
6 while True do
7   if  $id_{train} > N_{env}^{train}$  then
8     break;
9   end
10  for  $ep = 1$  to  $M_{ep}$  do
11    for  $t = 1$  to  $T_{step}$  do
12      With probability  $\epsilon \in [\epsilon_{init}, \epsilon_{final}]$  selects a
        random action  $a_t$ ;
13      otherwise select  $a_t \leftarrow \max_a Q(s_t, a; \theta)$ ;
14      Execute action  $a_t$ , observe reward  $r_t$  and
        next state  $s_{t+1}$  in the environment  $T_{env}^{id_{train}}$ ;
15      Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $\mathcal{D}$ ;
16    end
17    Sample random minibatch of transitions
         $(s_m, a_m, r_m, s_{m+1})$  in  $\mathcal{D}$ ;
18    if  $d_m^{r,g} \leq \epsilon_{goal}$  then
19       $y_m \leftarrow r_m$ ;
20    else
21       $y_m \leftarrow r_m + \gamma \max_{a'} Q(s_{m+1}, a'; \theta)$ ;
22    end
23    Perform a gradient descent step on
         $(y_m - Q(s_m, a_m; \theta))^2$  w.r.t the parameter  $\theta$ ;
24    Every C-steps, copy weights from  $Q$  to  $\hat{Q}$ ;
25  end
26  for  $i = 1$  to  $N_{env}^{val}$  do
27    Validate the robot with weights  $\theta$  in  $V_{env}^i$ ;
28    if  $p_{success} \geq P_{success}$  then
29       $id_{val} \leftarrow i$ ;
30    else
31      if  $id_{val} > i$  then
32        Retrain the robot in the current
          training environment  $T_{env}^{id_{train}}$ ;
33         $id_{val} \leftarrow i$ ;
34      else
35        Create a training environment  $T_{env}^{id_{train}+1}$ 
          that takes into account the failure of
          the validation environment  $V_{env}^i$ ;
36        Transfer  $Q$  with weights  $\theta$  to the next
          training environment  $T_{env}^{id_{train}+1}$ ;
37      end
38      break;
39    end
40  end
41   $id_{train} \leftarrow id_{train} + 1$ ;
42 end

```

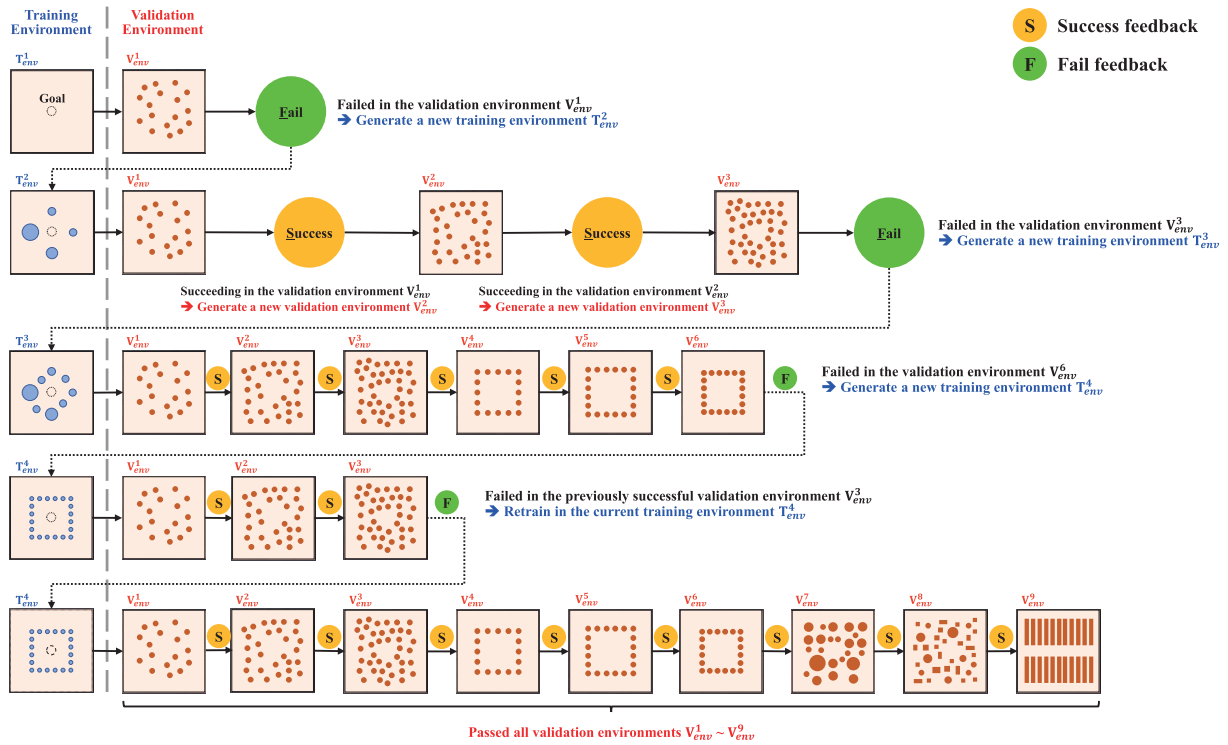


FIGURE 2. An example of designing a collision avoidance curriculum using feedback from the validation environment. If the robot succeeds in avoiding collisions in the validation environment, a new validation environment is created and validated again. Conversely, if a robot fails in a validation environment, a new training environment is created and the robot is trained in the environment to compensate for the failure. On the other hand, during validation, the robot sometimes fails to avoid collisions in environments where it has previously completed validation. This means that the robot has not learned enough, so it is retrained in the existing training environment.

case, the final training time is bounded, since we can set the variables involved in the training to be bounded.

Figure 2 shows an example of the curriculum design process using the Algorithm 1. In the first row, the robot trained in the obstacle-free training environment T_{env}^1 is evaluated in the validation environment V_{env}^1 . Since the success rate is low, a new training environment T_{env}^2 is created to overcome this challenge. In the second row, the robot trained in the new training environment T_{env}^2 is evaluated again in the previously evaluated validation environment V_{env}^1 . Since the success rate is high, the next validation environment V_{env}^2 is prepared for additional evaluation. In this way, each time the collision avoidance performance in the validation environment V_{env}^i is sufficiently high, the next validation environment V_{env}^{i+1} is prepared and the performance in that environment is evaluated. In the fourth row, the robot fails in a validation environment that it has passed before. In this case we assume that the robot has forgotten the knowledge it gained in the training environment T_{env}^4 and the robot is trained again in the existing training environment T_{env}^4 . This training process continues until the robot has passed all validation environments.

C. THE GENERATION METHOD OF VALIDATION AND TRAINING ENVIRONMENT

Validation environments are designed according to the following four principles. First, we create validation

environments with obstacles that vary in number, placement, size, and spacing. Since the ultimate goal is to perform robot collision avoidance in various environments, it is important to set up a variety of possible collision situations for validation. Second, the validation environments are set up by the user prior to training. To ensure the above diversity, it is important for the user to create a variety of validation environments that consider different collision situations. Third, the validation order of the validation environments should be such that it becomes increasingly difficult to avoid collisions. If collision avoidance fails in a validation environment, this feedback is used to create a new training environment, so it is important to organize environments that gradually increase the difficulty of collision avoidance. Finally, the maximum number of validation environments that can be created is theoretically unlimited, but for simplicity we limit the number of environments in practice.

The training environments are created based on the feedback from the validation environments. The robot considers the situations where it failed in the validation environment and creates a similar training environment. For example, if the robot failed to avoid collisions in the validation environment V_{env}^1 with a small number of obstacles, such as the first row in Fig. 2, the training environment T_{env}^2 is created with a slightly different number of obstacles. In this case, the placement and size of the obstacles are different from the validation environment V_{env}^1 , and the new training environment T_{env}^2 is

Algorithm 2 The Generation of Training Environment

```

1 if  $id_{val} > id_{val}^{max}$  then
2   | Retrain in the previous training environment  $T_{env}^{prev}$ ;
3 else
4   |  $n_{obs} \leftarrow n_{obs}^{prev} + [\mathcal{N}(0, \sigma_n^2)]$ ;
5   | Duplicate or eliminate  $(n_{obs} - n_{obs}^{prev})$  obstacles and
   | place them randomly;
6   | for  $i = 1$  to  $n_{obs}$  do
7     |  $\begin{bmatrix} x_i \\ y_i \end{bmatrix} \leftarrow \begin{bmatrix} x_i + \mathcal{N}(0, \sigma_x^2) \\ y_i + \mathcal{N}(0, \sigma_y^2) \end{bmatrix}$ ;
8     |  $r_i \leftarrow r_i + \mathcal{N}(0, \sigma_r^2)$ ;
9   | end
10  | Validates if there is any overlap between obstacles;
11  | Generate the next training environment  $T_{env}^{next}$  with
   |  $n_{obs}$  obstacles;
12 end

```

created by adjusting the placement and size of the obstacles based on V_{env}^1 . Similarly, if collision avoidance fails due to dense placement in the validation environment V_{env}^6 , train in a new training environment with dense obstacles, T_{env}^4 . Thus, the robot can improve its collision avoidance performance for situations where it failed in a particular validation environment by learning in a new training environment. Meanwhile, the number of training environments is limited because the training should be completed within a limited time.

Algorithm 2 shows the method for generating training environments. This algorithm works only when a robot has failed to navigate in the validation environment; the only condition for generating a new training environment is that the validation environment fails. If the current validation index id_{val} exceeds the maximum index of the validation environment, the robot will be trained again in the same training environment (lines 1–2). If the robot has failed in a new validation environment that it has not experienced before, the next training environment is generated based on the failed validation environment. In this case, we first scale the number of obstacles with a Gaussian distribution $\mathcal{N}(0, \sigma_n^2)$ (line 4), where σ_n^2 is the variance of the number of obstacles. Then the obstacles are duplicated or eliminated, taking into account the obstacle placement of the failed validation environment. Also, the positions and radii of the obstacles are modified using Gaussian distributions with variances $(\sigma_x^2, \sigma_y^2, \sigma_r^2)$ (lines 6–9). After the number, positions and radii of all obstacles have been adjusted by Gaussian distributions, the overlap validation is performed (line 10). If the overlap validation passes, the next training environment T_{env}^{next} with n_{obs} obstacles is generated (line 11). On the other hand, the algorithm 2 is executed again if the overlap between obstacles occurs.

Table 2 shows a comparison of the generation methods for validation and training environments. The validation

TABLE 2. A comparison of the generation methods for validation and training environments.

	Validation Environment	Training Environment
Creator	User	Automatic
Considerations for creation	Diversity in the number, placement, size, and spacing of obstacles	A validation environment that failed to avoid obstacles
Maximum # of environment	Unlimited	Limited
Environmental size	Same as training environment	Same as validation environment
Fixed or not	Fixed	Not fixed

environments are user-designed and fixed to account for various collision situations. The training environments are designed by the robot itself in a similar form to the validation environments, reflecting feedback from the validation process. The reason why the training environments are not the same as the validation environments is that if they are designed the same, the validity of the validation is not guaranteed.

VI. SIMULATION**A. SIMULATION ENVIRONMENT**

To verify the proposed curriculum design method, we used the Webots simulator to constitute training and testing environments [41]. The Webots simulator allows the construction of environments with obstacles and supports a two-wheeled mobile robot. We used the e-puck robot as a two-wheeled differential robot, with a diameter of 7 cm [42]. We trained a robot on Geforce RTX 3080 and Intel i7-12700 3.6 Ghz. The duration of a step in an episode is 0.1 seconds. The fast (**FF**) and slow (**SF**) forward velocities are 0.157 m/s and 0.035 m/s, respectively, and the right (**TR**) and left (**TL**) in-place rotations are -0.66 rad/s and 0.66 rad/s, respectively. The values **OR** and **OL** in Table 1 are (0 m/s, 0.035 m/s) and (0.035 m/s, 0 m/s), respectively. The parameters used in the training process are summarized in Table 3.

1) TRAINING ENVIRONMENTS

Figure 3 shows four training environments used in the simulation. All training environments have dimensions of 2 m × 2 m, and there are walls along the edges of the map. The mobile robot navigates through environments with static obstacles, avoiding all obstacles, to reach the goal of the map at coordinates (1 m, 1 m). The robot is considered as to have reached the goal when it is within 10 cm of the goal. An episode ends when the robot collides with obstacles or reaches the goal within T_{step} steps or exceeds the T_{step} steps, where T_{step} is set to 1000. When the episode ends, it is reset to a random position and heading within the training environment.

Figure 3(a) shows an initial training environment where there are no obstacles, only the robot (phase 1). In phase 1, the robot cannot learn to avoid obstacles. Figure 3(b) shows both the robot and four circular obstacles with radii of

TABLE 3. Parameters in the training process.

Source	Description	Symbol	Value
Training (Algorithm 1, 2)	The maximum number of training environments	N_{env}^{train}	4
	Episode size	M_{ep}	40
	The maximum timestamp	T_{step}	1000
	The radius of goal	ϵ_{goal}	0.1 m
	The maximum number of validation environments	N_{env}^{val}	9
	The threshold probability of success rate	$P_{success}$	0.99
	The number of proximity sensors	N_{ps}	8
	The variance of the number of obstacles	σ_n^2	64
	The variance of the positions of obstacles	σ_x^2, σ_y^2	$0.1m^2, 0.1m^2$
	The variance of the radii of obstacles	σ_r^2	$0.01 m^2$
	Deep Q-network (Algorithm 1, Equation (3), and (4))	Initial exploration probability	ϵ_{init}
Final exploration probability		ϵ_{final}	0.1
Update period of target Q-network		C	1000
Depth of Q-network		-	3
Width of each layer		-	1024 / 512 / 6
Learning rate		-	0.001
Discount factor		γ	0.95
Batch size		-	1024
Replay memory size		The size of \mathcal{D}	1.0×10^6

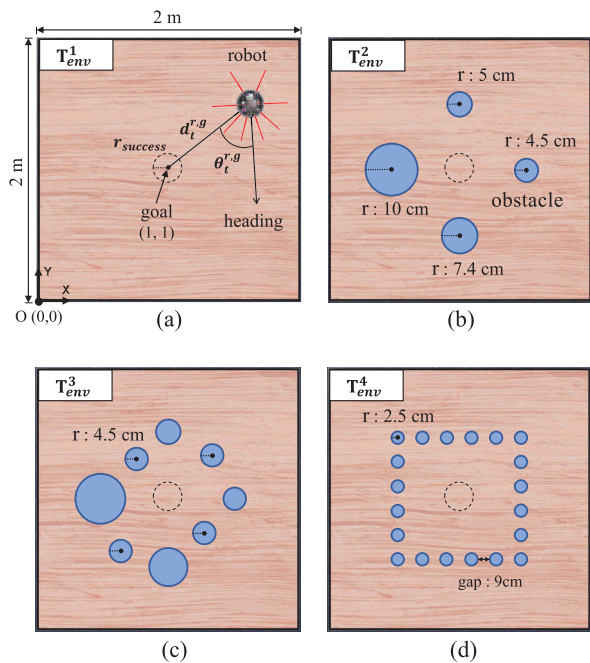


FIGURE 3. Training environments in the simulation. (a) The initial training environment of the curriculum (phase 1). (b) The second training environment with four circular obstacles (phase 2). (c) The third training environment with the addition of circular obstacles with a radius of 4.5 cm (phase 3). (d) The training environment with circular obstacles of 2.5 cm radius placed at 9 cm intervals (phase 4).

4.5 cm, 5 cm, 7.4 cm, and 10 cm, respectively (phase 2). The distance between the obstacles is at least 35 cm. In the phase 2, the robot can move to a goal while avoiding four circular obstacles of different sizes. Starting with phase 2, four additional circular obstacles with a radius of 4.5 cm have been added to the training environment, as shown in Fig. 3(c) (phase 3). The distance between these obstacles is at least 13 cm. The objective of phase 3 is that the robot learns to avoid as many obstacles as possible. Finally, we construct a training environment with 20 identical circular

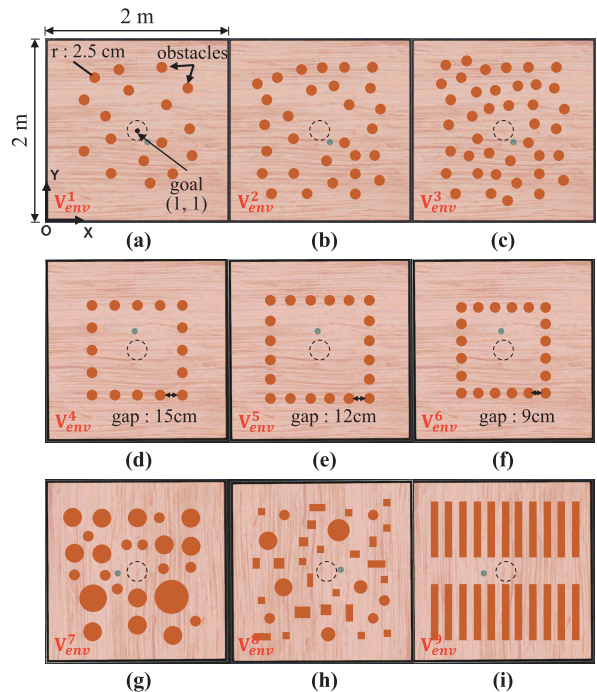


FIGURE 4. Validation environments used in the simulation. (a), (b), (c) Environments designed to evaluate collision avoidance performance by varying the number of obstacles. (d), (e), (f) Environments designed to evaluate collision avoidance by varying the distance between obstacles. (g), (h), (i) Environments designed to evaluate the robot's performance in situations not encountered in the training environment.

obstacles, as shown in Fig. 3(d) (phase 4). The robot learns how to navigate in a complex environment where the distance between obstacles is small.

2) VALIDATION ENVIRONMENTS

Figure 4 shows 9 validation environments for the simulation. The size of the environment is 2 m × 2 m the same as the training environment, and the origin is in the lower left corner. In the following results, we will omit units,

i.e., meters, for simplicity. The coordinate of the goal is the center of the environment: (1, 1). The robot's start position varies clockwise around the center, dividing the area between relative coordinates (0,0), (0,2), (2,2), (2,0) into 100 equal parts, resulting in a total of 400 start positions. The robot starts from each position with a random direction and moves towards the goal without colliding with obstacles. An episode ends when the collision occurs, when the goal is reached, or when the robot reaches the immobilized state. After an episode ends, the next episode starts immediately from the next start position. If the robot does not reach the goal within T_{step} , we consider this situation as *local minima* [43]; this case is counted as a failure.

Figure 4(a), (b), and (c) are designed to validate the robot's collision avoidance performance by varying the number of obstacles, with radii of 2.5 cm for static obstacles; the number of obstacles is 17, 29, and 40, respectively. We tried to place all obstacles at least 12 cm apart. However, when there are more than 40 obstacles, it is impossible to place obstacles more than 12 cm apart in a 2 m × 2 m environment, so we placed a maximum of 40 obstacles, as shown in Fig. 4(c). The validation environments of the second row in Fig. 4 are for varying the distances between obstacles. The distances in Fig. 4(d), (e), and (f) are 15 cm, 12 cm, and 9 cm respectively; the distances between obstacles were equally spaced in all directions. We set the minimum distance between obstacles to 9 cm because the diameter of the e-puck robot is about 7 cm; the margin for considering sensor error is 2 cm. Finally, Fig. 4(g), (h), and (i) are validation environments to evaluate the collision avoidance performance of the robot in situations with different obstacle sizes, displacements, and shapes. The radii of the circular obstacles vary from 5 cm to 20 cm, and the distance between each obstacle is 9 cm or more, as shown in Fig. 4(g). The environment in Fig. 4(h) has a variety of sizes of circular obstacles, and it also includes rectangular obstacles with shapes not experienced in the training environment, to assess whether the robot can avoid collisions. The length of one side of the rectangular obstacles is randomly set between 4 cm and 10 cm, and the obstacles are placed at least 9 cm apart. The environment in Fig. 4(i) is set up similar to a warehouse in a logistics center to evaluate whether the robot can successfully avoid collisions in such an environment. The distance between each rectangular obstacle is set to 10 cm.

B. RESULTS

We define TE(n) as the collision avoidance success rate of a robot trained sequentially from the training environment T_{env}^1 to T_{env}^n . For example, TE(3) refers to the collision avoidance performance of a robot trained incrementally in the environments from T_{env}^1 to T_{env}^3 .

The TE(1) in the validation environment V_{env}^1 is 92.3%, as shown in Fig. 5(a); the TE(1) is relatively low because the robot has only learned the simple navigation method in the training environment without obstacles, i.e. T_{env}^1 in Fig. 3(a).

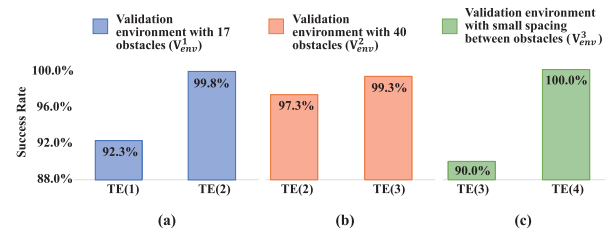


FIGURE 5. The robot's collision avoidance success rate obtained from the validation environments. Feedback was obtained from three environments: one with fewer obstacles (V_{env}^1), another with many obstacles (V_{env}^2), and a third with close spacing between obstacles (V_{env}^3). This feedback was then used to generate new subtasks and training environments.

Therefore, we designed a new training environment T_{env}^2 for learning obstacle avoidance, as shown in Fig. 3(b). A robot was incrementally trained in two training environments, T_{env}^1 and T_{env}^2 , resulting in TE(2). The TE(2) in V_{env}^1 is 7.5% higher than the TE(1), as shown in Fig. 5(a). The TE(2) performed well in the environment with fewer obstacles (V_{env}^1), but performed 2.5% worse in an environment with many obstacles (V_{env}^2), as shown in Fig. 5(b). To overcome this obstacle-rich environment, additional training was performed in an environment with more and different obstacles (T_{env}^3), as shown in Fig. 3(c): TE(3). As a result, the robot was able to avoid collisions 99.3% of the time in the validation environment with 40 obstacles, as shown in Fig. 5(b). As we can see from the TE(3), the robot was able to learn how to avoid collisions in an environment with many obstacles. However, when the obstacles were close together, such as V_{env}^3 , the TE(3) was only 90.0%, as shown in Fig. 5(c). This is because the robot has no opportunity to learn collision avoidance in an environment with dense obstacles. To compensate for this, the robot continues learning in a new training environment T_{env}^4 with obstacles spaced 9 cm apart, as shown in Fig. 3(d): TE(4). The performance of TE(4) in Fig. 5(c) shows a 100.0% success rate in avoiding obstacles and reaching the goal in a narrow-gap environment. No further training was needed because TE(4) had completed the training for a variety of obstacle displacements and a large number of obstacles.

Figure 6 shows the average reward over episodes in each training environment; the red line represents the moving average of the robot's reward, calculated from the last 25 data points, and the purple area shows the volatility around the average, representing the standard deviation range. First, the robot learned collision avoidance in an obstacle-free environment (T_{env}^1), as shown in Fig. 6(a); the average reward per episode increases as training progresses. Second, the robot learned a navigation method in an environment with a small number of obstacles to avoid collisions and reach the goal (T_{env}^2), as shown in Fig. 6(b). In this step, the reward was partially modified to detect collisions; the reward of +5 was given when there were no obstacles around the robot, indicating a safe state. Finally, Fig. 6(c) and (d) show the reward results of training in an environment with obstacles

TABLE 4. Success rates of reaching the goal with collision avoidance. The values in the table represent the following: success rate (# of success/# of trials).

	APF [44]	w/o curriculum				Proposed
		T_{env}^1	T_{env}^2	T_{env}^3	T_{env}^4	
V_{env}^1	98.5% (394/400)	92.3% (369/400)	100.0% (400/400)	10.5% (42/400)	100.0% (400/400)	100.0% (400/400)
V_{env}^2	95.3% (381/400)	92.3% (369/400)	100.0% (400/400)	13.5% (54/400)	100.0% (400/400)	100.0% (400/400)
V_{env}^3	78.5% (314/400)	92.3% (369/400)	100.0% (400/400)	4.5% (18/400)	98.8% (395/400)	99.8% (399/400)
V_{env}^4	95.0% (380/400)	92.3% (369/400)	100.0% (400/400)	9.0% (36/400)	100.0% (400/400)	100.0% (400/400)
V_{env}^5	19.5% (78/400)	92.3% (369/400)	100.0% (400/400)	3.5% (14/400)	100.0% (400/400)	100.0% (400/400)
V_{env}^6	4.3% (17/400)	86.8% (347/400)	99.8% (399/400)	2.3% (9/400)	100.0% (400/400)	99.8% (399/400)
V_{env}^7	75.8% (303/400)	86.8% (369/400)	99.8% (399/400)	1.8% (7/400)	100.0% (400/400)	100.0% (400/400)
V_{env}^8	15.8% (63/400)	74.0% (296/400)	96.5% (386/400)	0.0% (0/400)	96.3% (385/400)	100.0% (400/400)
V_{env}^9	5.0% (20/400)	77.5% (310/400)	98.5% (394/400)	0.0% (0/400)	82.3% (329/400)	100.0% (400/400)

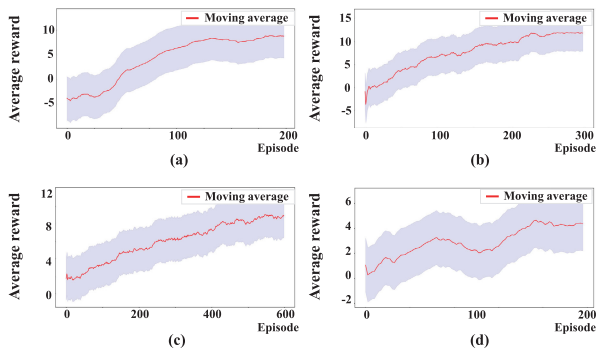


FIGURE 6. The average reward per episode received by the robot at each training step in the training environments. (a) obstacle-free environment (T_{env}^1 in Fig. 3). (b) the training environment with the small number of obstacles (T_{env}^2 in Fig. 3). (c) the training environment with many obstacles (T_{env}^3 in Fig. 3). (d) narrow-gap environment (T_{env}^4 in Fig. 3).

of different sizes and spacing (T_{env}^3 and T_{env}^4). The gradual increase of the reward in all p has confirmed that the training was normal.

In Table. 4, we compared the proposed method with other collision avoidance methods, including a traditional method such as artificial potential field (APF) [44] and DQN-based methods in trained T_{env}^i ($i = 1, \dots, 4$) environments without curriculum learning. Using APF, a robot was mostly successful in the simple validation environments such as V_{env}^1 , V_{env}^2 , and V_{env}^4 where the obstacles are scattered. However, the success rate of reaching the goal drops dramatically when the obstacles are densely packed as in V_{env}^5 , V_{env}^6 , V_{env}^8 , and V_{env}^9 . This is due to the typical drawback of APF, local minima, which makes the robot unable to move in the crowded environment. In the DQN-based collision avoidance methods without curriculum learning, we found that the performance is worse than the proposed method in most of the validation environments. Although the model trained in T_{env}^2 generally shows high generalization performance, but it shows a slight decrease in the validation environments with shape heterogeneity of obstacles, such as V_{env}^8 and V_{env}^9 . The results of the model trained in T_{env}^3 show that the advantages of the proposed method are obvious. The significantly low success rates for T_{env}^3 indicate that it may not converge to the desired performance in challenging environments without curriculum learning. The model trained in T_{env}^4 shows lower success rates

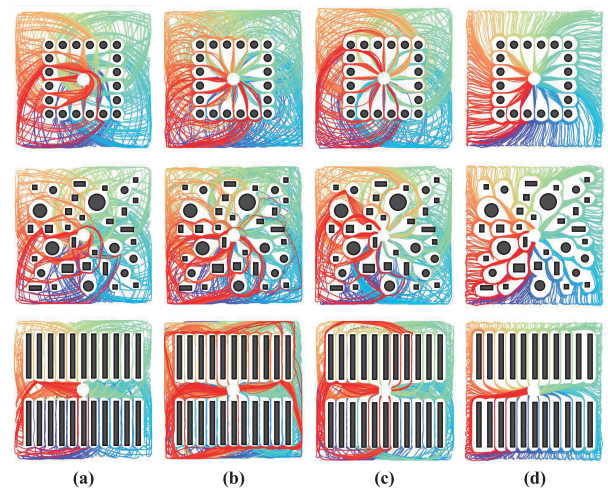


FIGURE 7. Trajectory results in validation environments with different training methods (a) Robot trajectory within the validation environment $V_{env}^{5,8,9}$ trained only in the training environment T_{env}^1 without curriculum learning (b) Robot trajectory in the validation environment $V_{env}^{5,8,9}$ trained only in the training environment T_{env}^2 without curriculum learning (c) Robot trajectory in the validation environment $V_{env}^{5,8,9}$ trained only in the training environment T_{env}^3 without curriculum learning (d) Robot trajectory in the validation environment $V_{env}^{5,8,9}$ trained with the proposed curriculum learning.

than T_{env}^2 in validation environments not experienced during training. Meanwhile, it can be observed that the proposed method achieves a collision avoidance performance close to 100% in all validation environments. This result shows that the proposed method can continuously compensate the robot's shortcomings of the robot without losing the knowledge learned in previous training environments. This also suggests that the proposed method can maximize the generalization performance.

To get a qualitative analysis of how the robot gets to its goal, we looked at the trajectories for a few cases. Figure 7 shows the trajectories of each robot in the validation environments V_{env}^5 , V_{env}^8 , and V_{env}^9 for different training methods, i.e. DQN-based methods without curriculum learning in the T_{env}^1 , T_{env}^2 , and T_{env}^4 , and the proposed method; the colors represent the trajectories of the robot starting from a coordinate and reaching the goal. In a validation environment, 400 starting positions were specified, and only the trajectories that reached the goal among the 400 episodes are shown. The

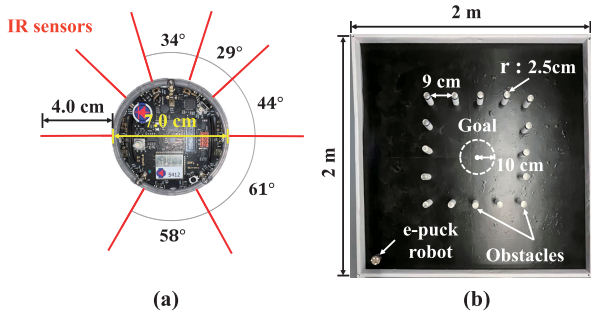


FIGURE 8. Configuration of real experiments. (a) The e-puck robot is equipped with 8 IR sensors with a maximum sensing distance of 4 cm. (b) The real environment shows circular obstacles with a radius of 2.5 cm and a distance of 9 cm between them.

performance trained in T_{env}^3 has been excluded from Fig. 7 due to its low success rate. From Fig. 7 it can be seen that models without curriculum learning tend to take longer paths instead of going straight to the goal. This is because the robot adapts to specific environments that it has learned. However, the proposed curriculum learning tends to take the shorter path to the goal in various environments, even if it does not go straight to the goal; the proposed method has a high generalization performance in collision avoidance.

VII. PRACTICAL EXPERIMENT

A. EXPERIMENTAL ENVIRONMENT

To validate the proposed method, real experiments were conducted. We used an e-puck robot with a diameter of 7.0 cm and equipped with 8 IR sensors ($N_{ps} = 8$ in Eq. (4)), with a maximum sensing distance of 4 cm, as shown in Fig. 8(a). We used Vicon system for robot localization [45]; the Vicon can measure the absolute coordinate of the robot under 2 mm error. The relative distance and angle between the robot and the goal, which are needed for validation, can be calculated from the absolute coordinate of the robot. The experimental configuration was similar to the validation environment V_{env}^6 , as shown in Fig. 8(b). Circular obstacles with a diameter of 5 cm were covered with paper material to facilitate IR detection. The evaluation procedure was similar to the simulations described in section VI-B, with a total of 32 start positions along the perimeter. The goal position is the center of the environment. Similar to the simulation, the radius of the goal ϵ_{goal} is set within 10 cm from the center of the environment. The training of the robot is already done in the simulation, and we only did real experiments for validation.

B. RESULTS

The success rates of each model in the practical experiments are generally lower than in the simulation, as shown in Fig. 9. The result of training only on T_{env}^3 did not perform well in both the simulation and the real environment, resulting in a significantly lower success rate (9.4%). The proposed method showed that the success rate of the practical experiment (63.0%) was lower than that of the simulation (99.8%),

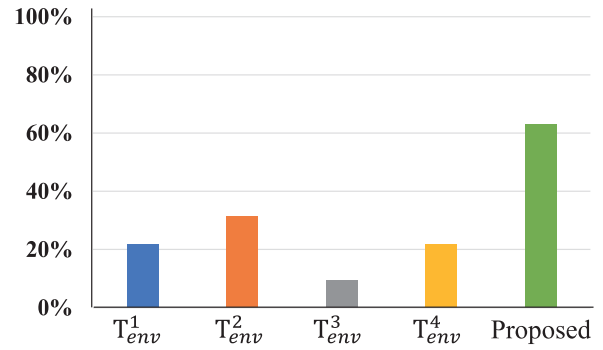


FIGURE 9. The collision avoidance success rates in the practical environment for different training methods. The proposed method shows the highest collision avoidance performance (63.0%) compared to the one trained on a single training environment without curriculum learning, such as T_{env}^1 , T_{env}^2 , T_{env}^3 , or T_{env}^4 .

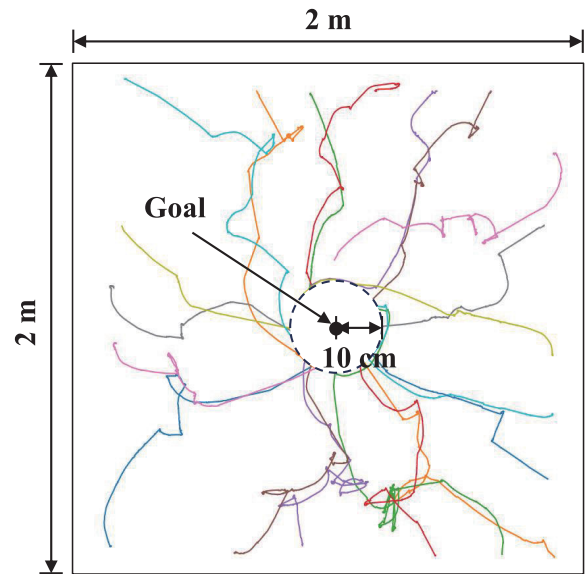


FIGURE 10. The trajectory in the validation environment V_{env}^6 for the proposed method. The e-puck robot starts at 32 peripheral points in the environment and moves to the goal in the center, avoiding obstacles along the way. If the robot gets within 10 cm of the goal without colliding, it is considered successful.

but still outperformed the models without curriculum. This means that the proposed method shows high generalization performance in real environments as well as in the simulation.

Figure 10 shows the robot trajectories of the proposed method. Although there are some unnecessary movements in the process of avoiding the obstacles, we can see that the robot avoids them and successfully reaches the goal.

VIII. DISCUSSION

In this study, we propose a novel curriculum design approach for DRL-based collision avoidance. Unlike the traditional fixed curriculum approach, our approach receives feedback on the robot's collision avoidance performance at each training phase and uses it to modify the curriculum to improve performance. By receiving feedback from various validation

environments and modifying the curriculum accordingly, we were able to generate the subtasks and training environments needed to succeed in various environments, resulting in a model with generalized performance.

In addition, given the lack of standardized obstacle avoidance curriculum design methods in previous studies, the proposed approach can provide a standardized curriculum design method. In the proposed method, multiple validation environments considering different obstacle avoidance situations are determined in advance, and each environment is validated to find the current level of vulnerability. Based on this, we modify the curriculum to improve collision avoidance performance; a new training environment is prepared for efficient learning, which is the modified version of the validation environment. This approach has the advantage of being more intuitive, similar to how humans learn through feedback, and can be applied to a wide variety of collision situations.

A limitation of our study is that the collision avoidance performance is highly dependent on the validation environments. Since the diversity of feedback is highly dependent on the validation environments, it is necessary to generate validation environments that consider various obstacle avoidance situations. Moreover, the number of validation environments and the order of validation may also affect each other, so it is necessary to carefully consider how to organize the validation environments appropriately.

The other limitation is how much we modify the training environment that is built based on the validation environment, i.e., when generating a training environment, how much variance in the Gaussian distribution related to the number of obstacles, positions, and radii should be adjusted? Recent research has proposed to automatically adjust the difficulty of curriculum environments based on the robot's learning performance in the training environment [39]. Integrating such advances into our study by combining methods for generating appropriate subtasks and automatically generating curriculum environments for each task could address the limitations of our approach. This integration could potentially lead to a standardized curriculum design method for collision avoidance.

Another limitation is that the proposed method implies a potential overfitting problem. The proposed method determines whether enough training has been done only in the validation environment, which means that there is a possibility that the overfitting problem may occur due to certain validation environments. If the collision avoidance performance remains low in the particular validation environment, the training environment will be configured to continuously resemble that environment, increasing the likelihood of overfitting. To solve this problem, we need an adaptive design method for both the validation environment and the training environment. Therefore, future research is needed on how to design both validation and training environments to be adaptive, which can be called fully automatic curriculum learning.

IX. CONCLUSION

This paper proposes a feedback-based curriculum design method using DRL. Collision avoidance performance is evaluated in various validation environments for each training phase, which provide feedback of success or failure. This feedback provides the collision avoidance performance of the current DRL model and helps to create adaptive training environments that are well designed for incremental learning; the feedback allows the robot to identify deficiencies in its current level of collision avoidance. Simulation and real-world experiments comparing models without curriculum learning show that the proposed method has an advantage in terms of generalization performance and collision avoidance performance. Although the proposed method is limited to the collision avoidance domain, the proposed design method of curriculum learning using validation results can be applied to various navigation domains such as exploration, target capturing, and formation control, which we leave as future work.

ACKNOWLEDGMENT

The authors would like to thank Jongwon Won for his help and valuable suggestions.

REFERENCES

- [1] R. Bernardo, J. M. C. Sousa, and P. J. S. Gonçalves, "Survey on robotic systems for internal logistics," *J. Manuf. Syst.*, vol. 65, pp. 339–350, Oct. 2022.
- [2] N. V. Kumar and C. S. Kumar, "Development of collision free path planning algorithm for warehouse mobile robot," *Proc. Comput. Sci.*, vol. 133, pp. 456–463, Jan. 2018.
- [3] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39830–39846, 2020.
- [4] J. D. Gammell and M. P. Strub, "Asymptotically optimal sampling-based motion planning methods," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 4, no. 1, pp. 295–318, May 2021.
- [5] J. Guo, C. Li, and S. Guo, "A novel step optimal path planning algorithm for the spherical mobile robot based on fuzzy control," *IEEE Access*, vol. 8, pp. 1394–1405, 2020.
- [6] B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technol.*, vol. 15, no. 4, pp. 582–606, Aug. 2019.
- [7] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
- [8] E. S. Low, P. Ong, and K. C. Cheah, "Solving the optimal path planning of a mobile robot using improved Q-learning," *Robot. Auto. Syst.*, vol. 115, pp. 143–161, May 2019.
- [9] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
- [10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
- [11] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6252–6259.
- [12] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*.
- [13] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 3207–3214.
- [14] G. Chen, L. Pan, Y. Chen, P. Xu, Z. Wang, P. Wu, J. Ji, and X. Chen, "Deep reinforcement learning of map-based obstacle avoidance for mobile robot navigation," *Social Netw. Comput. Sci.*, vol. 2, no. 6, pp. 1–14, Nov. 2021.

- [15] G. Chen, S. Yao, J. Ma, L. Pan, Y. Chen, P. Xu, J. Ji, and X. Chen, "Distributed non-communicating multi-robot collision avoidance via map-based deep reinforcement learning," *Sensors*, vol. 20, no. 17, p. 4836, Aug. 2020.
- [16] K. Li, Y. Lu, and M. Q.-H. Meng, "Human-aware robot navigation via reinforcement learning with hindsight experience replay and curriculum learning," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2021, pp. 346–351.
- [17] H. Xue, B. Hein, M. Bakr, G. Schilbach, B. Abel, and E. Rueckert, "Using deep reinforcement learning with automatic curriculum learning for mapless navigation in intralogistics," *Appl. Sci.*, vol. 12, no. 6, p. 3153, Mar. 2022.
- [18] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 7382–7431, 2020.
- [19] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 285–292.
- [20] J. Xin, H. Zhao, D. Liu, and M. Li, "Application of deep reinforcement learning in mobile robot path planning," in *Proc. Chin. Autom. Congr. (CAC)*, Oct. 2017, pp. 7112–7116.
- [21] X. Xue, Z. Li, D. Zhang, and Y. Yan, "A deep reinforcement learning method for mobile robot collision avoidance based on double DQN," in *Proc. IEEE 28th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2019, pp. 2131–2136.
- [22] S. Yao, G. Chen, L. Pan, J. Ma, J. Ji, and X. Chen, "Multi-robot collision avoidance with map-based deep reinforcement learning," in *Proc. IEEE 32nd Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2020, pp. 532–539.
- [23] M. Srouji, H. Thomas, Y.-H.-H. Tsai, A. Farhadi, and J. Zhang, "SAFER: Safe collision avoidance using focused and efficient trajectory search with reinforcement learning," in *Proc. IEEE 19th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2023, pp. 1–8.
- [24] L. He, N. Aouf, J. F. Whidborne, and B. Song, "Deep reinforcement learning based local planner for UAV obstacle avoidance using demonstration data," 2020, *arXiv:2008.02521*.
- [25] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 107–118, Jan. 2021.
- [26] J. Roghair, A. Niaraki, K. Ko, and A. Jannesari, "A vision based deep reinforcement learning algorithm for UAV obstacle avoidance," in *Proc. Intell. Syst. Appl. (IntelliSys)*, vol. 1. Amsterdam, The Netherlands: Springer, 2021, pp. 115–128.
- [27] D.-H. Chun, M.-I. Roh, H.-W. Lee, J. Ha, and D. Yu, "Deep reinforcement learning-based collision avoidance for an autonomous ship," *Ocean Eng.*, vol. 234, Aug. 2021, Art. no. 109216.
- [28] X. Xu, P. Cai, Z. Ahmed, V. S. Yellapu, and W. Zhang, "Path planning and dynamic collision avoidance algorithm under COLREGs via deep reinforcement learning," *Neurocomputing*, vol. 468, pp. 181–197, Jan. 2022.
- [29] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.
- [30] H. Niu, Z. Ji, F. Arvin, B. Lennox, H. Yin, and J. Carrasco, "Accelerated sim-to-real deep reinforcement learning: Learning collision avoidance from human player," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2021, pp. 144–149.
- [31] Z. Ren, D. Dong, H. Li, and C. Chen, "Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2216–2226, Jun. 2018.
- [32] C. Xiao, P. Lu, and Q. He, "Flying through a narrow gap using end-to-end deep reinforcement learning augmented with curriculum learning and Sim2Real," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2701–2708, May 2023.
- [33] T. Zhang, Z. Liu, S. Wu, Z. Pu, and J. Yi, "Multi-robot cooperative target encirclement through learning distributed transferable policy," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.
- [34] S. Narvekar, J. Sinapov, and M. Leonetti, "Source task creation for curriculum learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, May 2016, pp. 566–574.
- [35] S. Narvekar, J. Sinapov, and P. Stone, "Autonomous task sequencing for customized curriculum design in reinforcement learning," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2536–2542.
- [36] A. Karpathy and M. Van De Panne, "Curriculum learning for motor skills," in *Proc. Intell. Syst. Appl. (IntelliSys)*. Toronto, ON, Canada: Springer, 2012, pp. 325–330.
- [37] Y. Bengio, J. Louradour, and R. Collobert, "Curriculum learning," in *Proc. Int. Conf. Mach. Learn.*, Aug. 2009, pp. 41–48.
- [38] J. Kang, M. Liu, A. Gupta, C. Pal, X. Liu, and J. Fu, "Learning multi-objective curricula for robotic policy learning," in *Proc. Conf. Robot Learn.*, 2023, pp. 847–858.
- [39] G. Eoh and T.-H. Park, "Automatic curriculum design for object transportation based on deep reinforcement learning," *IEEE Access*, vol. 9, pp. 137281–137294, 2021.
- [40] F. L. D. Silva and A. H. R. Costa, "Object-oriented curriculum generation for reinforcement learning," in *Proc. Int. Conf. Auto. Agents MultiAgent Syst.*, 2018, pp. 1026–1034.
- [41] O. Michel, "Cyberbotics Ltd. WebotTM: professional mobile robot simulation," *Int. J. Adv. Robotic Syst.*, vol. 1, no. 1, p. 5, Mar. 2004.
- [42] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klapotocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proc. 9th Conf. Auton. Robot Syst. Competitions*, vol. 1. Castelo Branco, Portugal: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.
- [43] A. Orthey, B. Frész, and M. Toussaint, "Motion planning explorer: Visualizing local minima using a local-minima tree," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 346–353, Apr. 2020.
- [44] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, "The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations," *Ocean Eng.*, vol. 223, Mar. 2021, Art. no. 108709.
- [45] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "A study of vicon system positioning performance," *Sensors*, vol. 17, no. 7, p. 1591, Jul. 2017.



JEONGMIN CHOI is currently pursuing the bachelor's degree in mechanical design engineering with the Tech University of Korea, South Korea. His current research interest includes deep reinforcement learning-based robotics.



GYUYONG HWANG is currently pursuing the bachelor's degree in electronic engineering with the Tech University of Korea, South Korea. His current research interests include multi-robot, collision avoidance, and sim-to-real.



GYUHO EOH received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, South Korea, in 2009, 2011, and 2016, respectively. From March 2016 to December 2020, he was a Senior Engineer with the LG Electronics CTO Division (Advanced Robotics Laboratory), where his research focused on fusion SLAM. From January 2021 to January 2022, he was a Research Professor with Chungbuk National University. He is currently an Assistant Professor with the Tech University of Korea. His current research interests include deep reinforcement learning-based robotics, multi-robot systems, and sensor fusion-based SLAM.