mathematics

MDPI

Article

Mobile Robot Wall-Following Control Using Fuzzy Logic Controller with Improved Differential Search and Reinforcement Learning

Cheng-Hung Chen [1], Shiou-Yun Jeng [2] and Cheng-Jian Lin [2,3,*]

[1] Department of Electrical Engineering, National Formosa University, Yunlin 632, Taiwan; chchen.ee@nfu.edu.tw
[2] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan; shiouyun@ncut.edu.tw
[3] College of Intelligence, National Taichung University of Science and Technology, Taichung 404, Taiwan
* Correspondence: cjlin@ncut.edu.tw

Received: 10 June 2020; Accepted: 29 July 2020; Published: 31 July 2020

Abstract: In this study, a fuzzy logic controller with the reinforcement improved differential search algorithm (FLC_R-IDS) is proposed for solving a mobile robot wall-following control problem. This study uses the reward and punishment mechanisms of reinforcement learning to train the mobile robot wall-following control. The proposed improved differential search algorithm uses parameter adaptation to adjust the control parameters. To improve the exploration of the algorithm, a change in the number of superorganisms is required as it involves a stopover site. This study uses reinforcement learning to guide the behavior of the robot. When the mobile robot satisfies three reward conditions, it gets reward +1. The accumulated reward value is used to evaluate the controller and to replace the next controller training. Experimental results show that, compared with the traditional differential search algorithm and the chaos differential search algorithm, the average error value of the proposed FLC_R-IDS in the three experimental environments is reduced by 12.44%, 22.54% and 25.98%, respectively. Final, the experimental results also show that the real mobile robot using the proposed method can effectively implement the wall-following control.

Keywords: fuzzy logic control; wall-following control; mobile robot; reinforcement learning; differential search algorithm

1. Introduction

Wall-following [1,2], navigation [3], path tracking [4], and parallel-parking controls are prevalent in the field of robotics and artificial intelligence research. The design of robot navigation and parallel-parking behavior pushes the robot to move in an unknown environment. Wall-following behavior control is significant for mobile robot behavior. Zadah [5] proposed fuzzy logic in 1965; however, the logic was characterized by a high degree of uncertainty, complexity, and nonlinearity. Nevertheless, it was able to solve authentic world uncertainty by simulating the human experience in the form of rules. Many researchers applied fuzzy logic controllers (FLC) [6] to mobile robot navigation [7] and wall-following tasks [8]. Moreover, an optimization method has been proposed to improve the performance of FLC, such as supervised learning [9], reinforcement learning [10,11], and population-based learning [12].

Traditional supervised learning requires training data, whereas reinforcement learning does not require training data as the rewards or punishment mechanism is only needed for the training. Civicioglu [13] proposed a new heuristic differential search (DS) algorithm in 2012, which was originally used for geodetic transformation as the DS algorithm and other traditional algorithms were similar

and could offer the best local solution. To balance exploration and exploitation, many scholars have proposed methods to improve and optimize the DS algorithm [14–17]. The control parameter setting of the heuristic algorithm is important for evolution. This study proposes an adaptive parameter [18,19] for the IDS algorithm. Reinforcement learning is also widely used in robot training. Frommberger et al. [20] used reinforcement learning to achieve knowledge transfer and simulate the navigation technique in a real-world robotic platform [21–23]. Several scholars use reinforcement to learn the robot navigation problem. From a review of the literature, reinforcement learning can effectively solve the robot navigation problem. In addition, other control methods are used to solve robot-related issues.

The most common types of designed controller in mobile robot are divided into non-optimization [24–27] and optimization [28–30] approaches. Turennout et al. [24] proposed the robot-follow-the wall and discussed its control problem characteristics. Wang et al. [25] presented spiking neural networks to enhance the mobile robot to follow the wall. The PID-fuzzy controller was proposed for handling a remote surgical robot by Haidegger et al. [26]. Farooq et al. [27] used a fuzzy logic controller (FLC) to enable mobile robots to follow the wall effectively. Castillo et al. [28] use the application of ant colony optimization and particle swarm optimization on the optimization of membership functions of FLC for an autonomous mobile robot of a FLC. The new multi-objective of ant colony optimization algorithm is developed to optimize the FLS structure and parameters. This method controls the direction and movement speed of the mobile robot when performing wall-tracking tasks [29]. Grey wolf optimizer-based approaches were proposed for the optimal path planning and optimal tuning of tracking fuzzy controllers for nonholonomic wheeled mobile robots by Precup et al. [30].

In this study, an improved differential search (IDS) algorithm is proposed to optimize parameters of a FLC and to achieve mobile robot wall-following control task. Using reinforcement learning to train the robot controller, three conditions are defined. At the same time, three conditions are presented for a reward value. The IDS algorithm uses self-adaptive parameters to adjust control parameters, changing a chance of a superorganism in a stopover site. The results are compared with the other algorithms regarding their efficiency in designing the FLC of the wall-following task. Simulations and experiments prove that the proposed FLC_R-IDS method effectively implements the mobile robot wall-following control.

The rest of the study is organized as follows: Section 2 describes the FLC for mobile robots; Section 3 presents the proposed reinforcement-based improved differential search algorithm for mobile robot wall-following control; Section 4 presents the training environment, training results, and test environment simulations; finally, Section 5 provides conclusions and future works.

2. Mobile Robot Control Using a Fuzzy Logic Controller

This section discusses the mobile robots and the architecture of FLC. The mobile robot is trained to follow the wall and must be guided by reinforcement learning reward conditions. Figure 1 shows the architecture of the FLC with the reinforcement-based improved differential search algorithm (FLC_R-IDS) for the mobile robot.

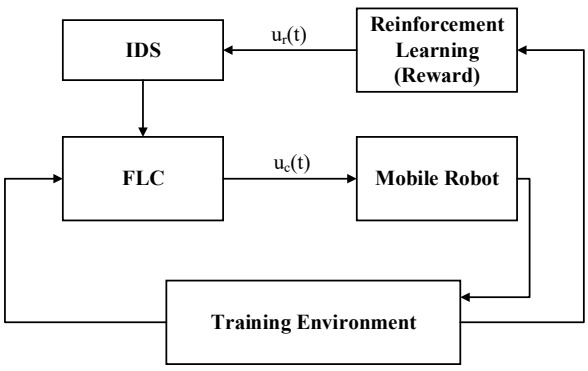

Figure 1. Architecture of the proposed FLC_R-IDS for mobile robot.

2.1. Description of Mobile Robots

The experiments carried out by a mobile robot (PIONEER 3-DX). In many navigation design and robot movement problems, the Pioneer 3-DX robot is a pony lightweight, two-wheel, two-motor differential drive ideal for indoor environmental laboratory or smaller classroom use. The robot itself is equipped with eight front ultrasonic sensors, batteries, motor encoders, microcontrollers with ARCOS firmware and a Pioneer Mobile Robot Software Development Kit. The robot ultrasonic sensor measures a range between 0.15 m and approximately 4.75 m. The ultrasonic sensor positions in Pioneer 3-DX were fixed in the following configuration: two on the side and six facing outward at 20° intervals of 180° forward coverage, as shown in Figure 2.

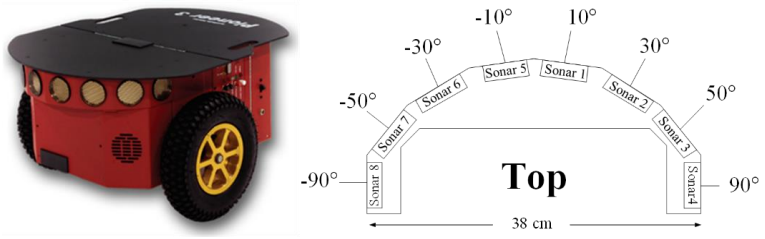

Figure 2. Pioneer 3-DX robot.

2.2. Architecture of Fuzzy Logic Controller

In the architecture of a FLC, the right four ultrasonic sensors (S_1, S_2, S_3, and S_4) are FLC inputs. The left-wheel and right-wheel speeds of the robot are FLC outputs. The FLC realizes a fuzzy model in the following form:

$$\begin{gathered} \textit{Rule } j: \left[\textit{IF } x_1 \textit{ is } A_{1j} \textit{ and } x_2 \textit{ is } A_{2j} \textit{ and } x_3 \textit{ is } A_{3j} \textit{ and } x_4 \textit{ is } A_{4j}\right] \\ \textit{THEN } y_l \textit{ is } u_j \textit{ and } y_r \textit{ is } v_j \end{gathered}$$

where x_1 is an ultrasonic sensor value of S_1; x_2 is an ultrasonic sensor value of S_2; x_3 is an ultrasonic sensor value of S_3; x_4 is an ultrasonic sensor value of S_4; A_{ij} is the linguistic term of the precondition part; y_l is the left-wheel speed of the robot; y_r is the right-wheel speed of the robot; u_j and v_j are the weights of consequent parts.

A fuzzification operation serves as the Gaussian membership function:

$$\mu_{A_{ij}} = \exp\left(\frac{-\left[xi - m_{ij}\right]^2}{\sigma_{ij}^2}\right) \tag{1}$$

where m_{ij} represents the mean of the Gaussian membership function of the fuzzy set, whereas σ_{ij} represents the variance of the Gaussian membership function of the fuzzy set.

In the fuzzy implication operation using product operation, the fuzzy implication evaluates the consequent part of each rule as follows:

$$\mu_{A_j} = \left(\prod_i \mu_{A_{ij}}\right) \tag{2}$$

In the defuzzification operation, the center of the area is used in this study, and it is described as:

$$y_l = \frac{\sum\limits_j \mu_{A_j} u_j}{\sum\limits_j \mu_{A_j}},\ y_r = \frac{\sum\limits_j \mu_{A_j} v_j}{\sum\limits_j \mu_{A_j}} \tag{3}$$

where y_l is the left-wheel speed of the robot, and y_r is the right-wheel speed of the robot.

3. The Proposed Reinforcement-Based Improved Differential Search Algorithm

The DS algorithm is a new heuristic algorithm [13]. The main concept of the DS algorithm is the simulation of the migration behavior of the creature. Numerical optimization algorithms have some common problems because they are characterized by poor exploration ability, and it is easy to fall into the local solution. To improve the performance of the algorithm, this study proposed the IDS algorithm. The IDS algorithm is inspired by parameter adaptability, in which the controller adjusts the parameters to the most suitable value during the evolution. In addition, adjusting the probability of parameter exchange enables the algorithm to be more diverse and to perform better than the original DS algorithm. On the mobile robot wall-following control, the IDS algorithm associated with the FLC is discussed in Section 2.1. This study proposes a fuzzy logic controller with the reinforcement-based improved differential search algorithm (FLC_R-IDS) for solving the mobile robot wall-following control problem. The proposed IDS is an evolutionary algorithm to optimize parameters of the FLC. Therefore, all parameters of the FLC are encoded into each individual of IDS. Moreover, each individual (i.e., FLC) is evaluated by reinforcement learning—that is, the performance of each individual is defined as the reward value of the mobile robot wall-following control in the training environment.

The DS algorithm that simulates the migration behavior of the creature is discussed. The DS algorithm simulates the biological process of the food energy migration. The superorganism often migrates to areas of abundant resources during seasonal changes. The Brownian-like random-walk movement is used to determine the migration of the superorganism; its pseudocode is shown in Algorithm 1.

Algorithm 1. A comparison of different identifiers in terms of dynamic system identification

Input: *Superorganism*
Output: *Stopover site*
Initialization of the population of *Superorganism*, where *Superorganism* = [*Artificial Organism*];
Evaluation of the *Superorganism*;
while *cycle* **do**
  Randomly selected *donor*;
  Calculate the p1, p2 and *Scale*;
  Generate the *Stopover site*, where *Stopover site* = *Superorganism* + *Scale*×(*donor* − *Superorganism*);
  //The Superorganism participating in the search process is determined through random scheme
  Evaluation of the *Stopover site*;
  if *Stopover site* is better than *Superorganism* **then**
    Superorganism is replaced by *Stopover site*;
  end
end
return *Stopover site*;

In the DS algorithm, the initial position of the superorganism is expressed as follows:

$$X_{i,j} = rand \cdot \left(up_j - low_j \right) + low_j \tag{4}$$

where *rand* is the uniform random number distribution between 0 and 1.

Randomly selected individual of superorganism is $donor = X_{rand\_selecte(i)}$, and superorganism uses the donor to find the stopover sites. However, the size of the stopover site depends on the scale value. The scale is expressed as follows:

$$Scale = rand[2 \cdot rand_1] \cdot (rand_2 - rand_3) \tag{5}$$

where $rand_1$, $rand_2$, and $rand_3$ are gamma random number distribution, and rand is a uniform random number distribution between 0 and 1.

In the DS algorithm, the stopover site position is expressed as follows:

$$Stopovesite = Superorganism + Scale \times (donor - Superorganism) \quad (6)$$

In the DS algorithm, the members of the superorganism are randomly assigned to stopover site search process. In assessing the stopover site, if the stopover site is better than the superorganism, the superorganism will migrate to the stopover site. With the constant migration of position, the superorganism will migrate to the best global solution.

3.1. Improved Differential Search Algorithm for Optimizing FLC Parameters

Although the original DS algorithm can solve the problem, they lack the diversity of the algorithm and the exploration ability [14]. To improve the diversity of algorithms and exploration capabilities and to avoid falling into the best local solution in the evolutionary process, the IDS algorithm adjusts the number of times a random scheme participates in the stopover site. For some heuristic algorithms, no fixed control parameters are better suited to solve different problems [18,19]. The original DS algorithm has two control parameters (p1 and p2) that affect the proportion of the superorganism in the stopover site. A flowchart of the IDS is shown in Figure 3 and explained as follows:

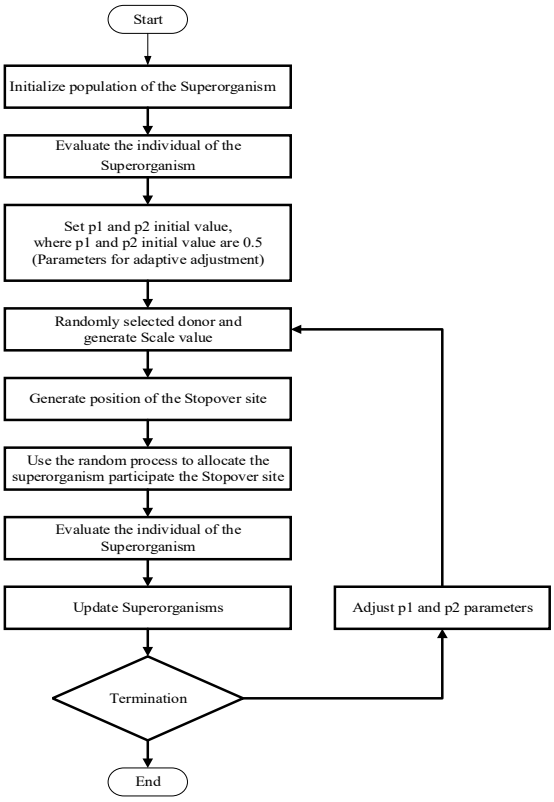

Figure 3. Flowchart of the IDS algorithm.

Step 1. Initialize the Superorganism

To initialize the Superorganism individually, the superorganism is defined as $X_{ij}(i = 1,2,3 \ldots N, j = 1,2,3 \ldots D)$, where N is the population size, and D is the dimension of the problem. The superorganism is composed of FLC fuzzy rules. Figure 4 shows the FLC coding principle in the IDS algorithm, where m_{ij} is the mean of the Gaussian membership function and σ_{ij} is the variance of the Gaussian membership function. u_j and v_j are the corresponding weight parameters of the consequent part. Four inputs and two outputs in the FLC are identified.

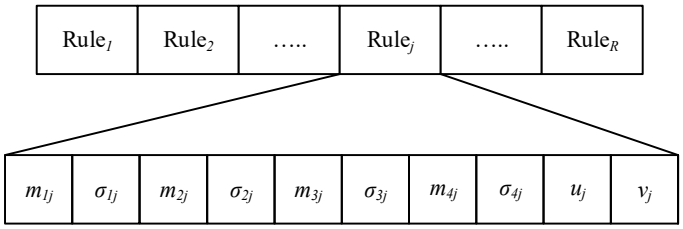

Figure 4. The FLC coding in the IDS.

The input parameter is the value of the ultrasonic sensor with the right wall (S_1, S_2, S_3, and S_4), and the output parameter is the rotation speed of the left wheel (*LW*) and right wheel (*RW*). The abovementioned control parameters must be defined by the user in advance. The FLC parameters are initialized as follows:

$$m_{ij} = rand_i(0,1) \times (0.8 - 0.2) + 0.2 \tag{7}$$

$$\sigma_{ij} = rand_i(0,1) \times (0.8 - 0.2) + 0.2 \tag{8}$$

$$u_j = rand[0,10] \tag{9}$$

$$v_j = rand[0,10] \tag{10}$$

where each ultrasonic sensor value has a range of 0.2 to 0.8 m. The left-wheel and right-wheel speeds have a range of 0 m/s to 10 m/s in the simulations.

Step 2. Evaluate the individual of the Superorganism

Each individual of the superorganism is a controller of the mobile robot. Thus, reinforcement learning is used to train a mobile robot to the wall-follow task. Moreover, reinforcement learning offers rewards or penalties when training the mobile robot controller. After several trainings, the mobile robot controller will gradually learn the correct action to get rewards. The reward value is used to evaluate the controller in the training process. Section 3.2 introduces reinforcement learning conditions designed for the mobile robot wall-following control task.

Step 3. Adjust the control parameters

In many studies, the setting of control parameters affects the performance of the algorithm [18,19]. The IDS algorithm has two control parameters: p1 and p2. In this study, the self-adaptive parameter adjusts the different learning processes. Figure 5 shows that the parameter adaptation is different from the fixed parameter encoding.

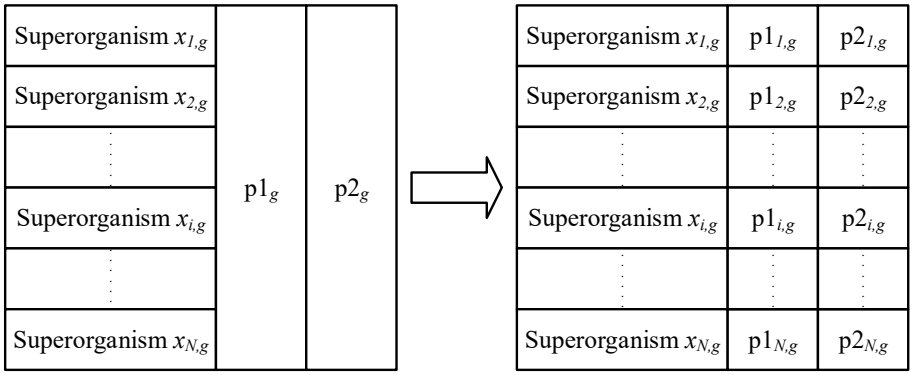

Figure 5. Parameter self-adaptation encoding.

The p1 and p2 initial values are 0.5. At each generation g, the control parameters p1 and p2 of each individual of the superorganism are generated by a normal distribution of mean μp1 and μp2, and the standard deviation is 0.1 truncated to [0, 1]:

$$p1_{i,g} = randn(\mu p1, 0.1) \tag{11}$$

$$p2_{i,g} = randn(\mu p2, 0.1) \tag{12}$$

The mean μp1 and μp2 are initialized to 0.5 and updated to each generation as follows:

$$\mu p1 = (1 - x) \cdot \mu p1 + x \cdot mean(Sp1) \tag{13}$$

$$\mu p2 = (1 - x) \cdot \mu p2 + x \cdot mean(Sp2) \tag{14}$$

where x is a positive constant 0 and 1, and the mean is the arithmetic mean. The S_{p1} and S_{p2} are the successful control parameters p1 and p2 at generation g.

Step 4. Select the Donor and Generate Scale

Randomly selected individuals of the superorganism to the donor are shown as follows:

$$donor = X_{rand\_selecte} \tag{15}$$

The size of the stopover site depends on the scale factor where the scale is expressed as follows:

$$Scale = randg[2 \cdot rand1] \cdot (rand2 - rand3) \tag{16}$$

where *randg* is gamma random number distribution, and rand is a uniform random number distribution between 0 and 1.

Step 5. Generate Position of the Stopover Site

In the IDS algorithm, the position of the stopover site is expressed as follows:

$$Stopover\ site = Superorganism + Scale \times (donoer - Superorganism) \tag{17}$$

The stopover site must be limited in scope. The limit is expressed as follows:

$$\begin{gathered} if\ Stopover\ site \langle 0.2\ or\ Stopover\ site \rangle 0.8 \\ Stopover\ site = rand \cdot (0.8 - 0.2) + 0.2 \end{gathered} \tag{18}$$

where *rand* is the uniform random number distribution between 0 and 1.

Step 6. Adjust the Number of Superorganisms in the Search

In this step, the superorganism participating in the search process is determined through random scheme. Its random process pseudocode is shown in Algorithm 2. Then, the procedure of the random number will produce $N \times D$ random number matrix, called the R matrix. The size of the R matrix corresponds to the population size. If the value within the R matrix is greater than 0, the individual of the Superorganism is involved in the Stopover site.

Step 7. Evaluate the Stopover Site

Similarly, the reinforcement learning reward that assesses the stopover site uses the same reward condition to evaluate the stopover site.

Step 8. Update Superorganism

If the solution of the Stopover site is superior to that of the superorganism, the superorganism is replaced by the stopover site:

$$Superorganism = \begin{cases} Stopover\ site & if(Stopover\ site \geq Superorganism) \\ Superorganism & otherwise \end{cases} \quad (19)$$

Algorithm 2. The random program is used to adjust the number of superorganisms in the search

```
// N is the size of the Superorganism population, where i = (1,2,3…N)
// D is the size of the problem dimension
for i = 1: N do
    if rand 1 < rand2 then
        if rand 3 < p1 then
            R = rand(1,D) ;
            R(i,:) = R(i,:) < rand 4;
        else
            R = ones(1,D) ;
            R(i,randi(D)) = R(i, randi(D)) < rand 5;
        end
    else
        R = ones(1,D) ;
        d = randi(D,1, 「p2 · rand · D」 ) ;
        for k = 1:size(d) do
            R(1,d(k)) = 0;
        end
    end
end
return R;
```

3.2. Reward of Reinforcement Learning

Reinforcement learning is a learning method in machine learning. Contrary to supervised learning, reinforcement learning has no training data. The concept of reinforcement learning comes from reward and punishment. This involves training the learner with rewards. When the learners behave correctly, they are rewarded. After some periods, the learner learns to behave correctly in different situations. Since the correct behavior can get a reward, it is essential to discuss the reward conditions for training mobile robots for the wall-following behavior.

In reinforcement learning, in training robots along the wall, the reward value evaluates the FLC performance. This study designed three conditions to train the robot to learn to follow the right wall. When the mobile robot satisfies the three conditions at the same time, the mobile robot controller can get the reward value +1. Otherwise, when the robot violates any of the conditions during the move, the controller stops accumulating the reward value, and the next controller is tested. However, when the controller gets a reward value which has accumulated to 6000, the mobile robot will stop learning and will be upgraded to the next run of training.

To enable the robot to learn to follow the wall, it is essential to define the distance between the robot and the wall in advance. However, the first condition is to keep distance. The robot must be kept between 0.3 and 0.8 m from the wall using the sensor (S_4) to measure the distance between the robot and wall.

The first condition is defined as:

$$0.3m \leq S4 \leq 0.8m \quad (20)$$

where S_4 denotes the ultrasonic sensor values.

We also use the front and right front sensors (S_1 and S_3) to determine whether any obstacles are present in the front. While sensing the right front of the mobile robot with or without walls, the second condition is defined as:

$$\sqrt{(S1)^2 + (S3)^2 - 2 \cdot S1 \cdot S3 \cdot \cos(40^\circ)} \quad \geq \quad 0.4 \tag{21}$$

where S_1 and S_3 are ultrasonic sensor values, and cos (40°) is the angle between the sensor S_1 and S_3.

Finally, the third condition is the speed of the mobile robot. To stop the robot from moving forward in the training process, we used the third condition to increase the wheel speed of the robot to more than 1 m/s. In this way, the mobile robot must move forward in the training. The third condition is defined as:

$$\left(\frac{RM + LM}{2}\right) \geq 1.0 m/s \tag{22}$$

where RM is the mobile robot right-wheel speed, and LM is the mobile robot left-wheel speed.

3.3. Stability Analysis of the FLC_R-IDS

The global stability of the control system is a basic requirement for solving mobile robot wall-following control problems. Since the general evolutionary algorithm is characteristic of random search, some search points may make the learning process unstable. In this subsection, the supervisory control $u_r(t)$ is designed (Figure 1) to guarantee the global stability of the closed-loop system in the sense that the error state variables must be uniformly bounded: $\left|e(t)\right| \leq M < \infty$ for all $t \geq 0$, where M is a design parameter that is specified by the designer. Therefore, using the supervisory control $u_r(t)$ in Figure 1 always yields $V(t) \to 0$ (i.e., $s(t) \to 0$), which in turn implies $\left|e(t)\right| \leq M$.

4. Experimental Results

To demonstrate the proposed FLC_R-IDS for solving the mobile robot wall-following control problem, this section describes the results of wall-following control simulations performed using the PIONEER 3-DX and compares these experimental results with those of other algorithms. The FLC has four inputs, defined as the right four ultrasonic sensors of the mobile robot and two outputs defined as the left-wheel and right-wheel speeds of the mobile robot in Section 2. The IDS is designed to optimize the FLC parameters. Therefore, the FLC parameters are encoded into individuals of the IDS algorithm. Furthermore, each individual of IDS is evaluated by the reward of reinforcement learning in Section 3. In this study, three reward conditions are defined as follows: to maintain a user-defined robot-wall distance, to avoid robot-wall collision, and to ensure that the robot can successfully move along the wall to go round the stadium.

During the learning process, mobile robots are allowed to learn along the wall in a simple environment. This training environment comprises of straight lines, right angles, and obtuse walls. A simple environment to train a mobile robot demonstrates that the incentive conditions effectively learn the wall-following behavior. It can also be completed along the wall in a more complex test environment. The Webot robotic simulation software is used to train the mobile robot to learn along the wall. At the same time, we compared this with the DS algorithm [13] to optimize the FLC_R-DS. We used the chaotic DS algorithm [14] to optimize the FLC_R-CDS. Figure 6 shows the training environment. The proposed method has five parameters: a population size (PS), a number of rules, p1 and p2 and a learning success reward value. In general, the larger PS, the more robust the search will be, with increased computational cost. The number of rules depends on the complexity of the problem. In the IDS algorithm, there are two control parameters: p1 and p2. The self-adaptive parameters (p1 and p2) were adjusted on the different learning processes. A single performance measurement in terms of failure and success can be used to determine the control policy that produces a maximal learning success reward value by trial-and-error tests. However, the selection of these parameters will critically affect the simulation results. The population size, which uses the range [20, 50], the number of rules, which uses the range [5, 10], the p1 and p2, which use the range [0, 1], and the learning success

reward value, which uses the range [5000, 8000], were carefully examined in extensive experiments. Table 1 presents the initial parameters set before the learning process. Mobile robots will learn along the wall in a training environment. If the robot satisfies the reward conditions, the controller gets the reward. Conversely, the inability of the robot to satisfy the reward conditions is considered a failure. When a failure occurs, the accumulated value of the reward is used to evaluate the FLC.

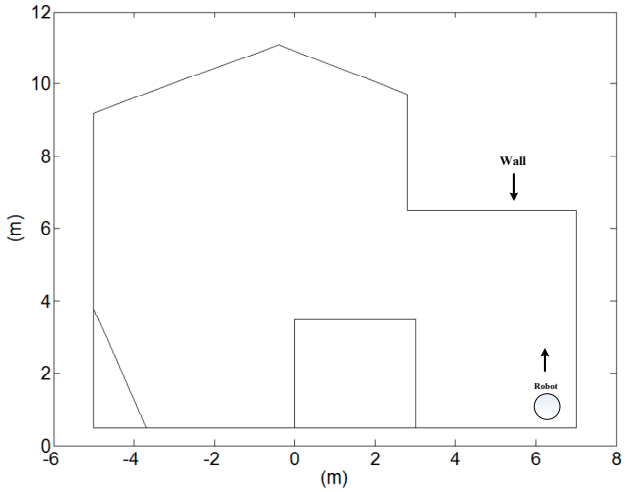

Figure 6. The training environment.

Table 1. Initialization parameters before the training.

Parameter	Value
Population size (PS)	30
Number of rules	5
p1 and p2 (DS and CDS)	0.3 × rand
p1 and p2 (IDS self-adaptive)	0.5
Learning success reward value	6000

4.1. Training Results of Mobile Robot Wall-Following Control

The mobile robot controller will continue to train until a successful condition is reached. The success condition is the accumulated reward value that reaches 6000. When the mobile robot reaches the success condition, it performs the wall-following task. In the learning process, the FLC_R-IDS, FLC_R-DS and FLC_R-CDS methods have 30 independent training runs. Table 2 shows the FLC_R-IDS, FLC_R-DS and FLC_R-CDS methods and compares the evaluation numbers in the learning process.

Table 2. Comparison of the evaluation numbers of various existing models in the learning process.

	FLC_R-IDS	FLC_R-DS	FLC_R-CDS
Minimum number of evaluations	26	121	70
Maximum number of evaluations	1851	2554	2716
Average number of evaluations	484.86	936.6	1047.1
Standard deviation	471.47	671.85	707.68

Figure 7 shows the learning curve of 30 independent training runs of the FLC_R-IDS, FLC_R-DS and FLC_R-CDS methods. In Figure 7a, the average number of evaluations of the FLC_R-IDS method learning success is 484. Using the FLC_R-IDS method to complete the training along the wall, the minimum number of evaluations is 26. The maximum number of evaluations is 1851. In Figure 7b, the average number of evaluations of the FLC_R-DS learning success is 936. Using the FLC_R-DS method to complete the training along the wall, the minimum number of evaluations is 121.

The maximum number of evaluations is 2554. Figure 7c shows that the average number of evaluations for the FLC_R-CDS method learning success is 1047. Using the FLC_R-CDS method to complete the training along the wall, the minimum number of evaluations is 70. The maximum number of evaluations is 2716. From the above, the FLC_R-IDS method learns faster than the other algorithms in the learning process.

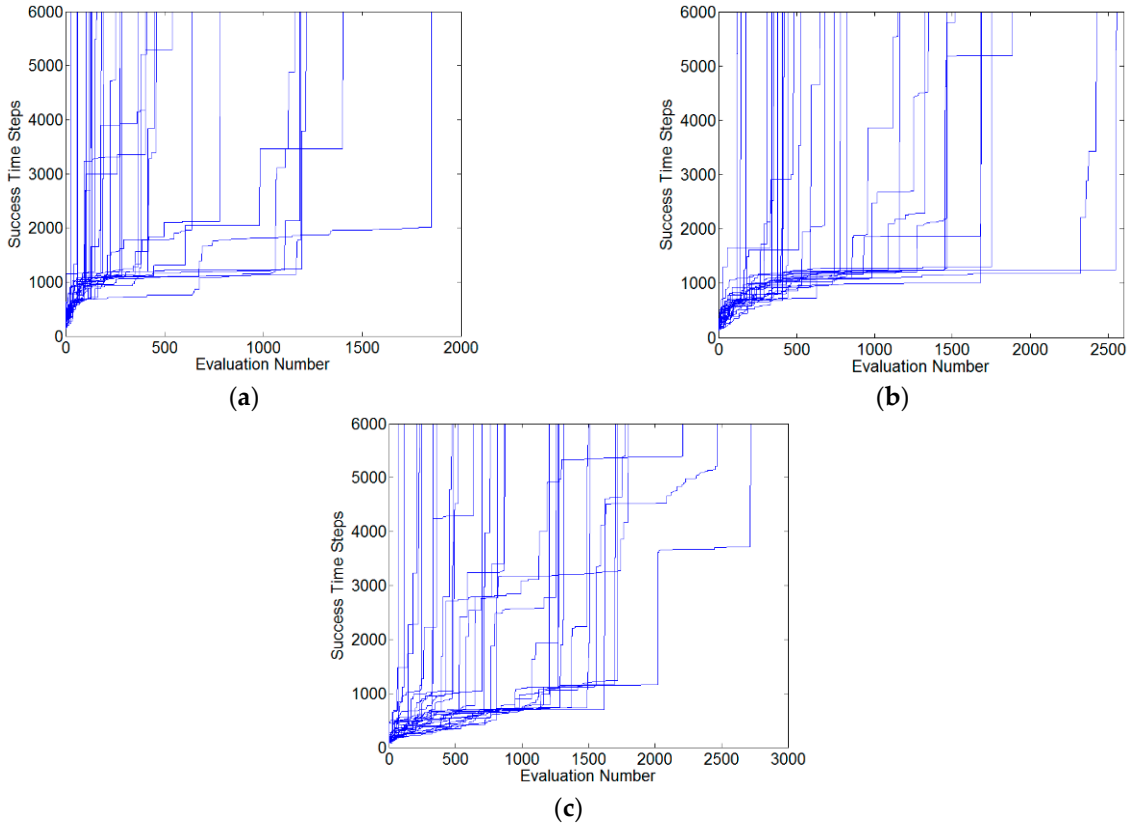

Figure 7. Mobile robot wall-following controller learning curve. (**a**) FLC_R-IDS method, (**b**) FLC_R-DS method and (**c**) FLC_R-CDS method.

In this section, we introduce the IDS algorithm without the self-adaptive method. This part removes the self-adaptive method of the IDS algorithm. To compare the difference between the IDS algorithm and IDS algorithm without the self-adaptive method, we used the IDS algorithm without the self-adaptive method to train 30 times independently. Compared with the FLC_R-DS method and FLC_R-IDS method, Table 3 shows the FLC_R-IDS method without the self-adaptive method and compares the results. The average value of the evaluation without the self-adaptive method was 626.5. Without the self-adaptive method, the control parameters (p1 and p2) cannot be adjusted during the learning process. By adjusting the number of the superorganisms in the search, the method can be learned faster than the FLC_R-DS method.

Table 3. Comparison between the evaluation numbers of various existing models in the learning process.

	FLC_R-IDS Without the Self-Adaptive Method	FLC_R-IDS	FLC_R-DS
Minimum number of evaluations	30	26	121
Maximum number of evaluations	3030	1851	2554
Average number of evaluations	626.5	484.86	936.6
Standard deviation	656.6	471.47	671.85

4.2. Testing Results of Mobile Robot Wall-Following Control

To show the proposed FLC_R-IDS method, we describe the results of the wall-following control simulations performed using the Webots robotic simulation software and we compare the results of the performance with those of other algorithms. In the learning process, the FLC undergoes reinforcement learning to get the best controller. Then, the trained FLC will be tested in the three experimental environments used for the simulations. First, the controller performs the test in the original training environment. The terrain of the training environment is relatively simple. Most of the terrain is a straight line, right angle, and an obtuse angle. The second experimental environment terrain is a combination of a right angle and a straight line. The third experimental environment is more complex than the previous two environments. The terrain is a straight line, right angle, and acute angle. However, the acute angle never appeared in the training environment. The arc-shaped terrain is also used for the experiment. Several best controllers in the experimental environment for testing and analysis are discussed, which are used to compare the performance of the FLC_R-IDS method with those of other methods. This example aims to design and analyze the FLC for a wall-following task. With 30 independent trainings runs, we get 30 mobile robot wall-following controllers. Thirty wall-following controllers are tested one by one in the experimental environment.

(1) Comparison of results of various methods in testing environment 1

The experiment is performed to demonstrate the best-performing FLC_R-IDS controller in the training environment. Figure 8a shows that the trained controller can complete the task of following the wall. Figure 8b shows the distance values according to the ultrasonic sensors S_1, S_3, and S_4 and the left-wheel and right-wheel speeds of the robot. When the robot moved along the wall to point A, the robot encountered a right angle. To avoid collision with the wall, the robot quickly turned left. At this time, the ultrasonic sensor values of S_1, S_3, and S_4 were 0.78, 0.53, and 0.32, respectively. The left-wheel and right-wheel speeds were 1.76 and 2.94 m/s, respectively. When the robot moved to point B along the wall, the robot slowly turned left in a straight line. At this time, the ultrasonic sensor values of S_1, S_3, and S_4 were 0.76, 0.61, and 0.42 m, respectively. The left-wheel and right-wheel speeds were 1.83 and 3.25 m/s, respectively. When the robot in the C point environment was in a straight line, the robot continued to move straight ahead. At this time, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 0.42, and 0.3 m, respectively. The left-wheel and right-wheel speeds were 1.81 and 1.81 m/s, respectively. At point D, when the robot encountered the outer corner, the robot must turn right; otherwise, the robot will move away from the wall. In this case, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 1.0, and 1.0 m, respectively. The left-wheel and right-wheel speeds were 2.76 and 2.3 m/s, respectively. At point E, when the robot turned over the outer corner, the robot must continue along the wall. In this case, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 0.75, and 0.46 m, respectively. The left-wheel and right-wheel speeds were 2.64 and 2.1 m/s, respectively. Figure 9 shows the comparison of the path tracking best performances of the proposed FLC_R-IDS, FLC_R-DS and FLC_R-CDS methods in test environment 1.

The mobile robots learn along the wall. The excellent performance along the wall is characterized by a robot's capability to maintain a distance from the wall. Thus, we analyzed the distance between the robot and wall using the mean absolute error (MAE), which evaluates the performance of the FLC in the wall-following task. When the mobile robot's S_4 sensor value is 0.3, the robot d_{wall} error is zero. The smaller the value of MAE, the better the performance of the controller:

$$MAE = \frac{\sum_{i=1}^{Step_{total}} \left| S_4(i) - d_{wall} \right|}{Step_{total}} \tag{23}$$

where $S_{4(i)}$ is the value of the S_4 sensor for each step of the robot; the d_{wall} is 0.3 of the distance between the robot and wall; the $Setp_{total}$ is the robot that completes the total number of steps to walking along the wall.

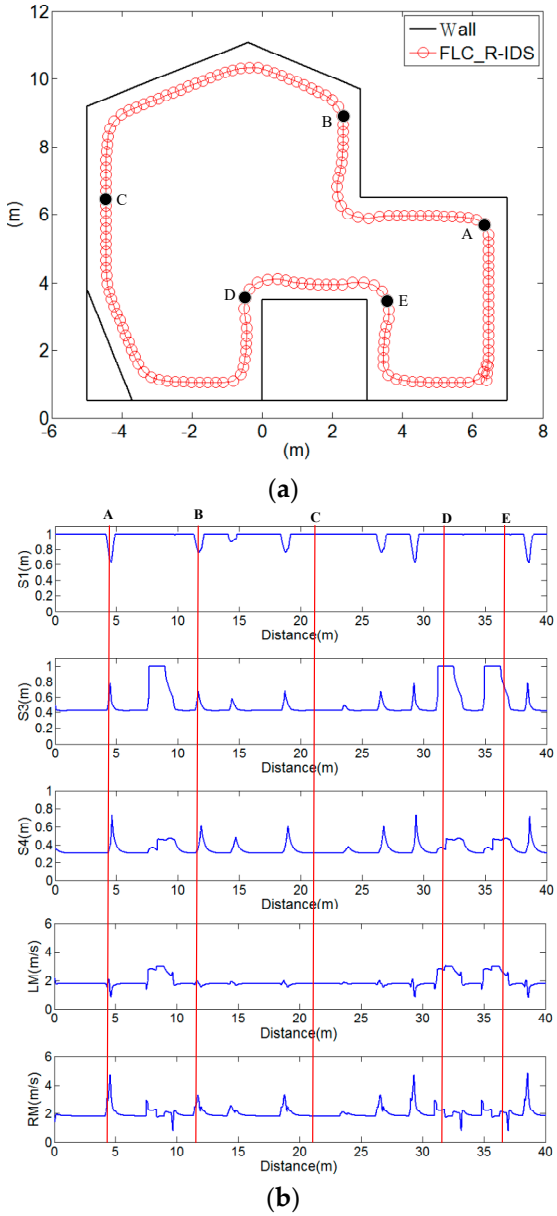

Figure 8. (**a**) Mobile robot path tracking, (**b**) ultrasonic sensor values and the left-wheel and right-wheel speeds of the robot in test environment 1 for the FLC_R-IDS method.

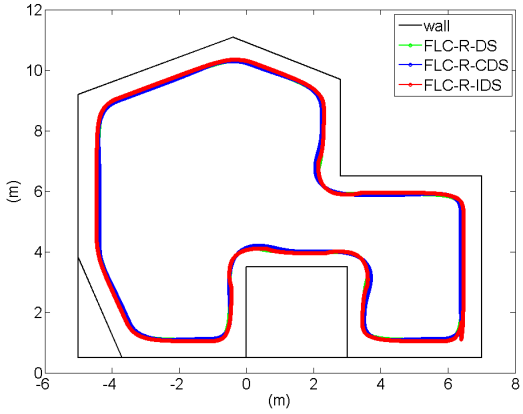

Figure 9. Comparison of the path tracking best performances of the proposed FLC_R-IDS, FLC_R-DS and the FLC_R-CDS methods in test environment 1.

Table 4 represents the FLC_R-IDS, FLC_R-DS and FLC_R-CDS methods for 30 controllers and their MAE values in the test environment. This study also compared the performance of the FLC_R-IDS method with those of other methods. After 30 independent trainings runs, each algorithm got 30 controllers. In IDS, the FLC best controller performance is better than that of the DS and CDS. Regarding the average of 30 controllers, the FLC_R-IDS method performs better than the FLC_R-DS method and FLC_R-CDS algorithms.

Table 4. Comparison of 30 controllers and their MAE values of various existing models in test environment 1.

Algorithms	FLC_R-IDS	FLC_R-DS	FLC_R-CDS
Minimum MAE	0.0491	0.0852	0.1064
Average MAE	0.1358	0.1521	0.1527
Standard deviation	0.0275	0.0262	0.0227

The wall-following controller can train along the right wall using the fuzzy controller with evolutionary reinforcement learning. The mobile robot ultrasonic sensors are symmetrical. So, the inputs of the FLC replaced the mobile robot left ultrasonic sensors. (Sensor S_1 is replaced by S_5; sensor S_2 is replaced by S_6; sensor S_3 is replaced by S_7; sensor S_4 is replaced by S_8.) The mobile robot left- and right-wheel exchange show that mobile robots can complete the task that follows the left wall. Figure 10 shows that the FLC input is replaced by the left side of the ultrasonic sensor. The FLC_R_IDS method is the path of the left wall in the test environment 1.

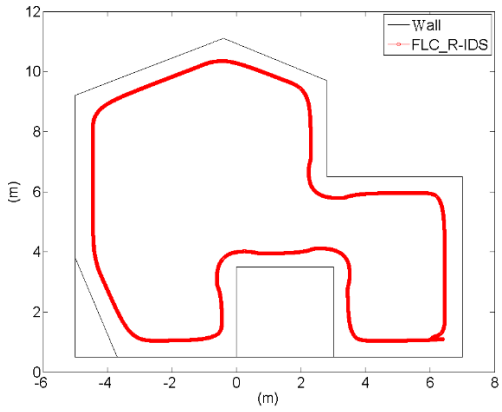

Figure 10. The path tracking of the proposed FLC_R-IDS method, following the left wall in test environment 1.

(2) Comparison of results of various methods in testing environment 2

The results of the mobile robot controller testing in test environment 2 are discussed. This experimental environment terrain is a combination of a right angle and straight line. Figure 11a shows that the trained controller can complete the task in combination with the right angle and straight line environment. Figure 11b shows the distance values according to the ultrasonic sensors S_1, S_3, and S_4 and the left-wheel and right-wheel speeds of the robot at the robot moving distances. When the robot moved along the wall to point A and encountered the outer corner, it must turn right; otherwise, the robot will move away from the wall. At this time, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 1.0, and 0.35 m, respectively. The left-wheel and right-wheel speeds were 2.78 and 2.27 m/s, respectively. When the robot moved along the wall to point B, the front area of the robot was a small corner; the robot must go left and then go right. At this time, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 0.36, and 0.38 m, respectively. The left-wheel and right-wheel speeds were 1.61 and 2.32 m/s, respectively. When the robot moved along the wall to point C, the robot encountered a right

angle. To avoid collision with the wall, the robot quickly turned left. At this time, the ultrasonic sensor values of S_1, S_3, and S_4 were 0.68, 0.71, and 0.37 m, respectively. The left-wheel and right-wheel speeds were 2.11 and 3.39 m/s, respectively. When the robot at point D of the environment is travelling in a straight line, it continues to go straight. In this case, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 0.41, and 0.31 m, respectively. The left-wheel and right-wheel speeds were 1.81 and 1.82 m/s, respectively. Figure 12 shows the comparison of the path tracking best performances of the proposed FLC_R-IDS, FLC_R-DS, and FLC_R-CDS methods in test environment 2. Similar to the previous experiment, Equation (23) is used to assess the performance of each FLC. Table 5 represents the FLC_R-IDS method and the MAE values of other methods for the 30 controllers in the test environment 2. It also shows the comparison of the performance of the FLC_R-IDS method with those of the other methods. From Table 5, several controllers failed in test environment 2 because the environment is more complex than the training environment. When the robot controller collides or stops in the test environment, this represents a failure. After 30 independent training runs, each algorithm got 30 controllers. The best controller performance in the FLC_R-IDS method is better than that in the FLC_R-DS and FLC_R-CDS methods. In the average of 30 controllers, the FLC_R-IDS method performs better than the FLC_R-DS and FLC_R-CDS methods. In the FLC_R-DS and FLC_R-CDS methods, a collision occurred while running test environment 2.

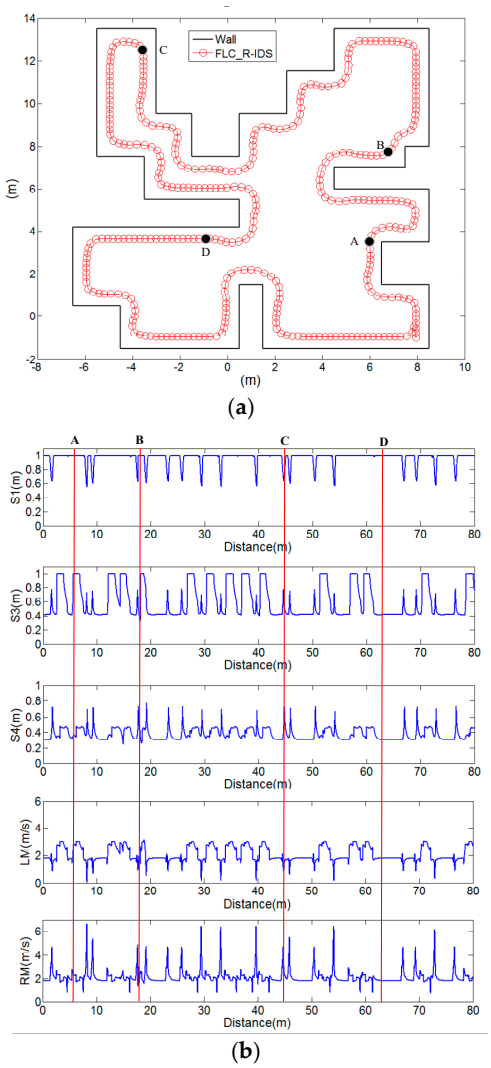

Figure 11. (**a**) Mobile robot path tracking, (**b**) ultrasonic sensors values, and the left-wheel and right-wheel speeds of the robot in the test environment 2 for the FLC_R-IDS method.

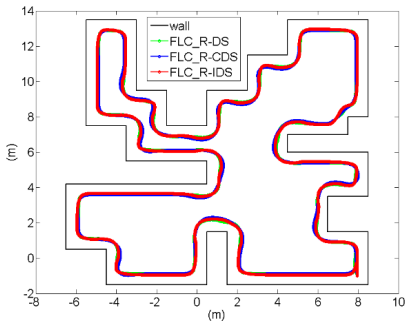

Figure 12. Comparison of the path tracking best performances of the proposed FLC_R-IDS, FLC_R-DS, and FLC_R-CDS methods in test environment 2.

Table 5. Comparison of 30 controllers and their MAE values of various existing models in test environment 2.

Algorithms	FLC_R-IDS	FLC_R-DS	FLC_R-CDS
Minimum MAE	0.0759	0.1024	0.1313
Average MAE	0.1411	0.1632	0.1729
Standard deviation	0.0249	0.0323	0.0269

(3) Comparison of results of various methods in testing environment 3

This test environment is more complex than the previous environments. The terrain is a straight line, right angle, and acute angle. The test environment is also a circular terrain. Figure 13a shows that the FLC can complete the task in combination with the right angle, acute angle, straight line, and circular terrain environment. Figure 13b shows the distance values according to the ultrasonic sensors S_1, S_3, and S_4 and the left-wheel and right-wheel speeds of the robot at the robot moving distances. When the robot moved along the wall to point A because the terrain of point A was a straight line, it must turn right over the straight terrain to continue the action along the wall. At this moment, the values of ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 1.0, and 0.36 m, respectively. The left-wheel and right-wheel speeds were 2.8 and 2.25 m/s, respectively. When the robot moved along the wall to point B, the robot was along the circular terrain. So, the robot must stay forward to the right front. At this time, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 0.68, and 0.44 m, respectively. The left-wheel and right-wheel speeds were 2.45 and 2.25 m/s, respectively. When the robot moved along the wall to point C, the robot would encounter the acute angle. To avoid collision with the wall, the robot quickly turned left. In this acute angle of the terrain, if the mobile robot turns left quickly, the robot will stay away from the wall. Conversely, if the mobile robot turns left too late, the robot collides with the wall. So, the robot must turn left in the appropriate range to continue the task along the wall. In this case, the ultrasonic sensor values of S_1, S_3, and S_4 were 0.69, 0.58, and 0.33 m, respectively. The left-wheel and right-wheel speeds were 1.7 and 3.5 m/s, respectively. When the robot in the D point of the environment is in a straight line, the robot will continue to go straight ahead. In this case, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 0.42, and 0.3 m, respectively. The left-wheel and right-wheel speeds were 1.81 and 1.82 m/s, respectively. When the robot moves along the wall to point E, the robot has just passed an obtuse angle. The robot is not parallel to the wall, so the robot must turn left to adjust the direction of the body. At this time, the ultrasonic sensor values of S_1, S_3, and S_4 were 1.0, 0.48, and 0.36 m, respectively. The left-wheel and right-wheel speeds were 1.86 and 2.1 m/s, respectively. Figure 14 shows the comparison of the path tracking best performances of the proposed FLC_R-IDS method and other methods in test environment 3. Table 6 represents the FLC_R-IDS method and other methods for the MAE values of 30 controllers in the test environment 3. It also compares the performance of the FLC_R-IDS method with those of the FLC_R-DS and the FLC_R-CDS methods.

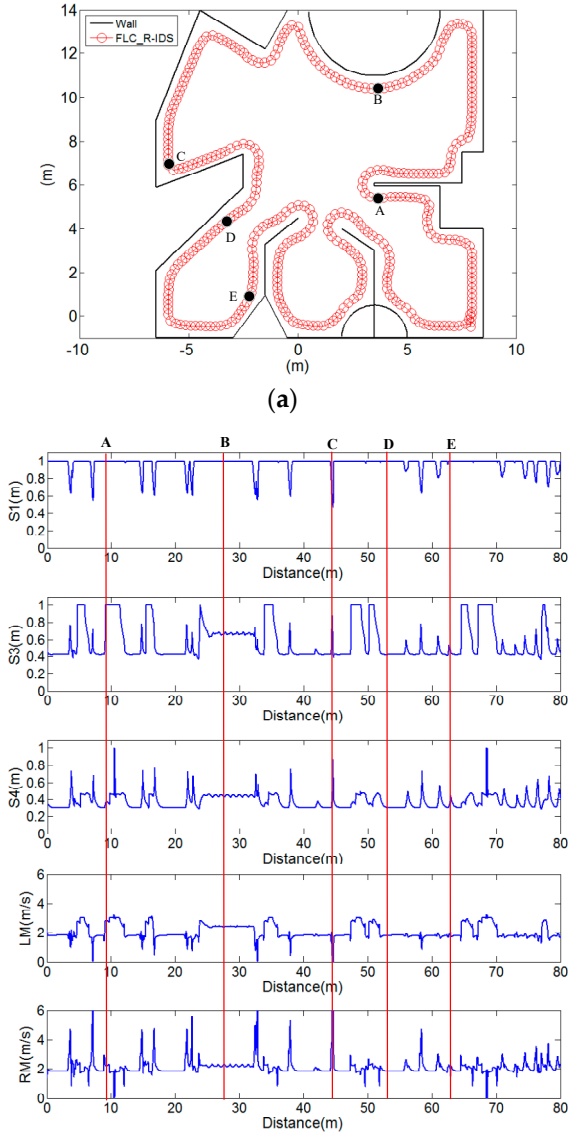

Figure 13. (**a**) Mobile robot path tracking, (**b**) ultrasonic sensor values, and the left-wheel and right-wheel speeds of the robot in test environment 3 for the FLC_R-IDS method.

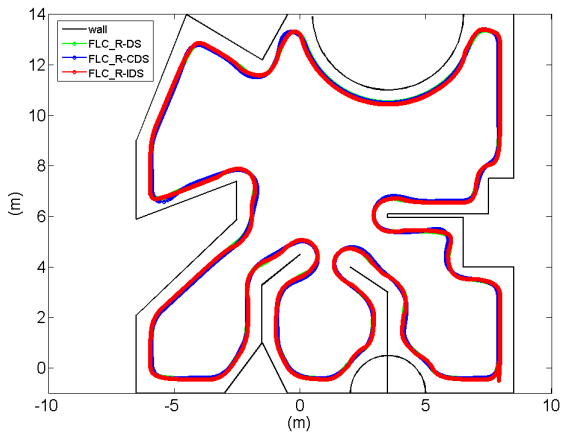

Figure 14. Comparison of the path tracking best performances of the proposed FLC_R-IDS, FLC_R-DS, and FLC_R-CDS methods in test environment 3.

Table 6. Comparison of 30 controllers and their MAE values of various existing models in test environment 3.

Algorithms	FLC_R-IDS	FLC_R-DS	FLC_R-CDS
Minimum MAE	0.0791	0.1001	0.1271
Average MAE	0.1424	0.1682	0.1794
Standard deviation	0.0283	0.0399	0.0280

4.3. Real Mobile Robot Wall-Following Control

This study performed the experiment to show the execution of an actual mobile robot wall-following control task using the FLC_R_IDS method and a PIONEER 3-DX robot. To illustrate the feasibility of the FLC_R_IDS method, a real environment was created to test the mobile robot's performance in an actual wall-following task. In the simulations, the inputs of the FLC were the ultrasonic sensor values. The outputs of the FLC were the robot's left-wheel and right-wheel speeds. The maximum value of each ultrasonic sensor was 0.8 m. Each wheel reached a maximum of translation speed of 10 m/s. The PIONEER 3-DX robot ultrasonic sensor was 5 m. The PIONEER 3-DX robot reached a maximum of translation speed of 1.4 m/s. In the experiments, the inputs and outputs of the FLC were linear conversion. Figure 15 shows the images of the wall-following control results of the proposed approach. The PIONEER 3-DX robot could move along the wall and maintain a user-defined distance from the wall.

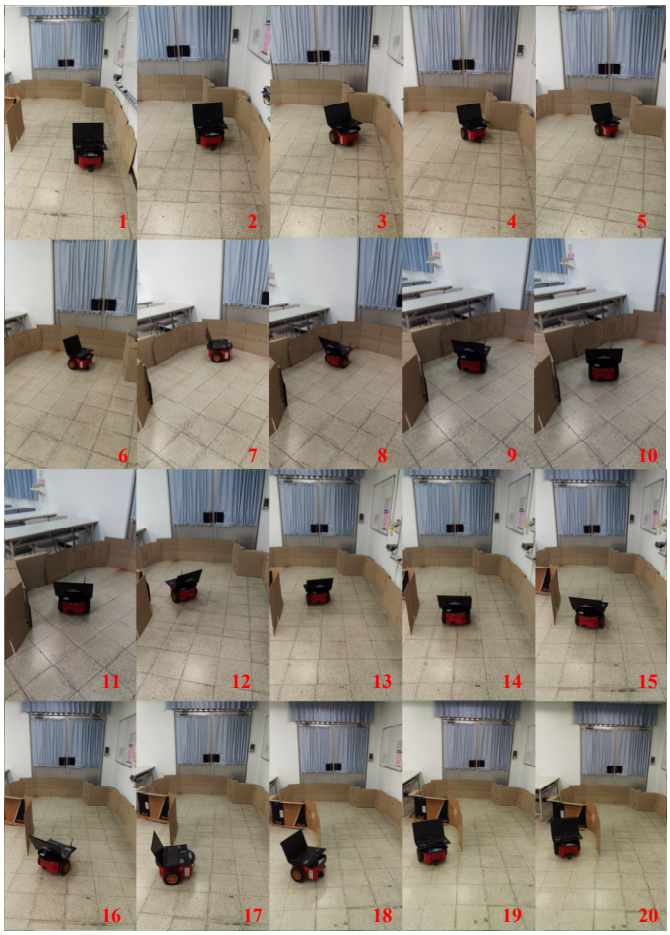

Figure 15. Wall-following control results of the PIONEER 3-DX in an actual environment using the FLC_R-IDS method.

5. Conclusions

This study proposed the IDS with the reinforcement learning designed FLC (FLC_R-IDS) to achieve a mobile robot wall-following control task. In the proposed approach, the IDS algorithm uses the self-adaptive parameter to adjust the control parameters. The IDS algorithm adjusts the number that random scheme times for stopover site. The above two methods are used to optimize the FLC parameters. Using reinforcement learning to train mobile robots along the wall, reward conditions will affect the overall learning situation in reinforcement learning. If the reward conditions are set to be too difficult, it will be challenging for the robot to learn. On the contrary, if the reward conditions are set to be too simple, the robot will learn easily, but the performance may be worse. In this study, three conditions are proposed. If the three conditions are reached at the same time, the robot controller will get the reward value. Finally, the accumulated reward value can assist in evaluating the standard FLC performance. In this method, the mobile robot is not using training data during the learning process. The experimental results show that the proposed method in three experimental environments reduced the MAE values by 12.44%, 22.54%, and 25.98%, respectively, compared with the various existing models. This study proved that the FLC_R-IDS method performs better than the FLC_R-DS method and the FLC_R-CDS method, in terms of relative fewer number of evaluations of achieved "success". Moreover, the FLC_R-IDS method can be applied successfully to a wall-following control task. The advantages of the proposed FLC_R-IDS are summarized as follows: (1) it does not use training data during the learning process; (2) it uses a self-adaptive method to adjust the control parameters; (3) it designs three conditions to achieve a mobile robot wall-following control task as follows: to maintain a user-defined robot-wall distance, to avoid robot-wall collision and to ensure that the robot can successfully move along the wall to go round the stadium.

In the simulations, the three reward conditions can help robots learn along the wall. The reward conditions proposed in this study can be effectively learned along the wall. However, this method may fail in the dead zone in a more complex experimental environment. Thus, setting appropriate reward conditions is important for learning and designing reward conditions in the future. In recent years, several heuristic algorithms have been proposed, such as the flower pollination algorithm and Jaya algorithm. These algorithms have a good ability to search the solution space and control the robot to learn more quickly along the wall for escaping the dead zone.

Author Contributions: Conceptualization, C.-H.C. and C.-J.L.; methodology, C.-H.C. and C.-J.L.; software, C.-H.C. and S.-Y.J.; data curation, C.-H.C. and S.-Y.J.; writing—original draft preparation, C.-H.C., S.-Y.J. and C.-J.L.; writing—review and editing, C.-H.C. and C.-J.L.; funding acquisition, C.-H.C. and C.-J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology of the Republic of China, grant number MOST 108-2221-E-167-026 and MOST 108-2221-E-150-021-MY2.

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

References

1. Jhang, J.Y.; Lee, C.L.; Lin, C.J.; Young, K.Y. Using a self-clustering algorithm and type-2 fuzzy controller for multi-robot deployment and navigation in dynamic environments. *Asian J. Control* **2020**. [CrossRef]
2. Jhang, J.Y.; Lin, C.J.; Young, K.Y. Cooperative carrying control for multi-evolutionary mobile robots in unknown environments. *Electronics* **2019**, *8*, 298. [CrossRef]
3. Parker, L.E. Current research in multi-robot systems. *Artif. Life Robot.* **2003**, *7*, 1–5. [CrossRef]
4. Ordonez, C.; Collins, E.G.; Selekwa, M.F.; Dunlap, D.D. The virtual wall approach to limit cycle avoidance for unmanned ground vehicles. *Robot. Auton. Syst.* **2008**, *56*, 645–657. [CrossRef]
5. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]
6. Lin, C.T.; Lee, C.S.G. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1996.
7. Hagras, H.A. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Trans. Fuzzy Syst.* **2004**, *12*, 524–539. [CrossRef]

8. Farooq, U.; Khalid, A.; Amar, M.; Habiba, A.; Shafique, S.; Noor, R. Design and Low Cost Implementation of a Fuzzy Logic Controller for Wall Following Behavior of a Mobile Robot. In Proceedings of the 2010 2nd International Conference on Signal Processing Systems, Dalian, China, 5–7 July 2010; pp. 740–746.
9. Wang, Y.; Zhu, X. A Supervised Adaptive Learning-Based Fuzzy Controller for a Non-Linear Vehicle System Using Neural Network Identification. In Proceedings of the 2016 American Control Conference, Boston, MA, USA, 6–8 July 2016; pp. 3946–3951.
10. Jiang, J.; Zeng, X.; Guzzetti, D.; You, Y. Path Planning for Asteroid Hopping Rovers with Pre-Trained Deep Reinforcement Learning Architectures. *Acta Astronaut.* **2020**, *171*, 265–279. [CrossRef]
11. Cherroun, L.; Boumehraz, M. Intelligent Systems Based on Reinforcement Learning and Fuzzy Logic Approaches, "Application to Mobile Robotic". In Proceedings of the 2012 International Conference on Information Technology and e-Services, Sousse, Tunisia, 24–26 March 2012.
12. Chung, H.Y.; Hou, C.C.; Liu, S.C. Automatic Navigation of a Wheeled Mobile Robot Using Particle Swarm Optimization and Fuzzy Control. In Proceedings of the 2013 IEEE International Symposium on Industrial Electronics, Taipei, Taiwan, 28–31 May 2013.
13. Civicioglu, P. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. *Comput. Geosci.* **2012**, *46*, 229–247. [CrossRef]
14. Liu, B. Composite differential search algorithm. *J. Appl. Math.* **2014**, *2014*, 294703. [CrossRef]
15. Kumar, V.; Chhabra, J.K.; Kumar, D. Differential search algorithm for multiobjective problems. *Procedia Comput. Sci.* **2015**, *48*, 22–28. [CrossRef]
16. Chen, G.Z.; Wang, J.Q.; Li, R.Z. Parameter identification of the 2-chlorophenol oxidation model using improved differential search algorithm. *J. Chem.* **2015**, *2015*, 313105. [CrossRef]
17. Wang, Z.; Gao, D.; Liu, J. Multi-objective sidetracking horizontal well trajectory optimization in cluster wells based on DS algorithm. *J. Pet. Sci. Eng.* **2016**, *147*, 771–778. [CrossRef]
18. Brest, J.; Greiner, S.; Boskovic, B.; Mernik, M.; Zumer, V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 646–657. [CrossRef]
19. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
20. Frommberger, L.; Wolter, D. Structural knowledge transfer by spatial abstraction for reinforcement learning agents. *Adapt. Behav.* **2010**, *18*, 507–525. [CrossRef]
21. Sendari, S.; Mabu, S.; Hirasawa, K. Fuzzy Genetic Network Programming with Reinforcement Learning for Mobile Robot Navigation. In Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics, Anchorage, AK, USA, 9–12 October 2011; pp. 2243–2248.
22. Jaradat, M.A.K.; AI-Rousan, M.; Quadan, L. Reinforcement based mobile robot navigation in dynamic environment. *Robot. Comput.-Integr. Manuf.* **2011**, *27*, 135–149. [CrossRef]
23. Dong, D.; Chen, C.; Chu, J.; Tarn, T.J. Robust quantum-inspired reinforcement learning for robot navigation. *IEEE/ASME Trans. Mechatron.* **2010**, *17*, 86–97. [CrossRef]
24. Turennout, P.V.; Honderd, G.; Schelven, L.J.V. Wall-Following Control of a Mobile Robot. In Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, 12–14 May 1992; pp. 280–285.
25. Wang, X.; Hou, Z.G.; Tan, M.; Wang, Y.; Hu, L. The Wall-Following Controller for the Mobile Robot Using Spiking Neurons. In Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence, Shanghai, China, 7–8 November 2009; pp. 194–199.
26. Haidegger, T.; Kovács, L.; Preitl, S.; Precup, R.E.; Benyó, B.; Benyó, Z. Controller design solutions for long distance telesurgical applications. *Int. J. Artif. Intell.* **2011**, *6*, 48–71.
27. Farooq, U.; Hasan, K.M.; Asad, M.U.; Saleh, S.O. Fuzzy Logic Based Wall Tracking Controller for Mobile Robot Navigation. In Proceedings of the 2012 7th IEEE Conference on Industrial Electronics and Applications, Singapore, 18–20 July 2012; pp. 2102–2105.
28. Castillo, O.; Martínez-Marroquín, R.; Melin, P.; Valdez, F.; Soria, J. Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. *Inf. Sci.* **2012**, *192*, 19–38. [CrossRef]

29. Juang, C.F.; Jeng, T.L.; Chang, Y.C. An interpretable fuzzy system learned through online rule generation and multiobjective ACO with a mobile robot control application. *IEEE Trans. Cybern.* **2015**, *46*, 2706–2718. [CrossRef]
30. Precup, R.E.; Voisan, R.E.; Petriu, E.M.; Tomescu, M.L.; David, R.C.; Szedlak-Stinean, A.I.; Roman, R.C. Grey wolf optimizer-based approaches to path planning and fuzzy logic-based tracking control for mobile robots. *Int. J. Comput. Commun. Control* **2020**, *15*, 1–17. [CrossRef]