Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5524232, 21 pages
https://doi.org/10.1155/2021/5524232

Research Article

A Fusion Method of Local Path Planning for Mobile Robots Based on LSTM Neural Network and Reinforcement Learning

Na Guo, Caihong Li, Tengteng Gao, Guoming Liu, Yongdi Li, and Di Wang

School of Computer Science and Technology, Shandong University of Technology, Zibo 255049, China

Correspondence should be addressed to Caihong Li; lich@sdut.edu.cn

Received 27 January 2021; Revised 1 April 2021; Accepted 1 June 2021; Published 14 June 2021

Academic Editor: Mohammad Amin Hariri-Ardebili

Due to the limitation of mobile robots' understanding of the environment in local path planning tasks, the problems of local deadlock and path redundancy during planning exist in unknown and complex environments. In this paper, a novel algorithm based on the combination of a long short-term memory (LSTM) neural network, fuzzy logic control, and reinforcement learning is proposed, and uses the advantages of each algorithm to overcome the other's shortcomings. First, a neural network model including LSTM units is designed for local path planning. Second, a low-dimensional input fuzzy logic control (FL) algorithm is used to collect training data, and a network model (LSTM_FT) is pretrained by transferring the learned method to learn the basic ability. Then, reinforcement learning is combined to learn new rules from the environments autonomously to better suit different scenarios. Finally, the fusion algorithm LSTM_FTR is simulated in static and dynamic environments, and compared to FL and LSTM_FT algorithms, respectively. Numerical simulations show that, compared to FL, LSTM_FTR can significantly improve decision-making efficiency, improve the success rate of path planning, and optimize the path length. Compared to the LSTM_FT, LSTM_FTR can improve the success rate and learn new rules.

1. Introduction

Robots are widely used in industry, agriculture, medicine, military, and other fields, and can assist or replace human work. Robots can also improve work efficiency and reduce costs. Currently, robot technology is gradually becoming more popular and is used in all aspects of life. Path planning is an important research topic for robot navigation, which is the premise and foundation of mobile robots [1, 2]. Based on an understanding of the running environment, path planning can be divided into global and local path planning [3, 4]. Global path planning calculates an optimal route based on the given start and target points while avoiding obstacles in a known environment [5, 6]. Global path planning primarily includes A$*$ search algorithm, rapidly exploring random tree (RRT) and Voronoi diagram algorithm [7, 8]. However, they typically require long computation times, particularly when the map is larger. The methods may also fail when the robot enters an uncertain environment. Local path planning explores a collision-free optimal path to reach the target point based on environmental information detected by onboard sensors [9, 10]. Commonly used local path planning algorithms include potential field method, fuzzy logic, neural network, heuristic algorithm, and various hybrid algorithms [11–13]. Due to the limitations of the sensor's perception of environmental information, certain problems, such as local deadlock, path redundancy, and unreachable target, exist in the local path planning algorithm. To ensure that the robot can avoid obstacles safely and reach the target point faster, it is necessary to investigate algorithms in more detail [14, 15].

The artificial potential field method is to model the surrounding environment by constructing the power field. The method is simple and convenient for the underlying implementation, and is suitable for simple obstacle environments [16]. So, it has been widely used in robot path planning. But, there also exist some problems in complex environments, such as local minima and target unreachable. Fuzzy control is based on physiological "perception-action" behavior, which does not require accurate environmental information. Fuzzy control does not exhibit many of the shortcomings of traditional algorithms, such as sensitivity to robot positioning accuracy and strong dependence on environmental information, but exhibits advantages in solving path planning in unknown environments [17, 18]. Although the fuzzy control algorithm can yield a better planned path, it must establish complete control rules by artificial experience, and the more the rules, the longer the decision-making time. In addition, fuzzy logic control lacks the ability of learning and generalization and cannot plan a feasible route under special circumstances that the rules are not specifically designed for. Artificial neural network (ANN) is a method to build and train a network model by simulating the behavioral characteristics of biological neural networks. It has intelligent information-processing functions such as learning, association, memory, and pattern recognition. ANN performs online calculation, which is fast and efficient, and is suitable for solving real-time path planning problems [19, 20]. Reinforcement learning is an autonomous learning method for mobile robots to explore state-action pairs. Based on the feedback of interaction with the environment, the optimal action in each state is calculated and continuously strengthened. The algorithm simplifies the artificial operation from the mapping of perception to decision-making as much as possible, avoids fragile human designs, and improves the stability and intelligence of the task. However, reinforcement learning requires long learning time and is difficult to converge [21, 22]. Other intelligent algorithms, such as ant colony algorithm, genetic algorithm, simulated annealing algorithm, and firefly algorithm, mainly search the optimal solution in the defined problem space [23–25].

The current research trend is to combine the above methods to improve the shortcomings of each algorithm, and some researchers have done a lot of research on these methods. Guo et al. proposed a novel step optimal path planning method based on fuzzy control [26]. A fuzzy controller with two inputs and two outputs is designed, which can complete the path planning in the common environment and reach the target point successfully. Patle et al. presented a new path planning algorithm based on probability and fuzzy logic (PFL), which uses distance and speed as combination rules. The method is suitable for static and dynamic environments [27]. Fuzzy control depends on human experience, and its fixed rules are difficult to adapt to complex real environments. Gharajeh and Jond proposed a hybrid GPS-ANFIS based method [28]. It is composed of a GPS-based controller for the global navigation of the robot toward the goal and an ANFIS controller for obstacle avoidance local navigation. ANFIS integrates neural network into a fuzzy system to adapt to the uncertainty environment. The paper verifies the feasibility of the proposed algorithm in discrete environments. However, neural network has the shortcomings of long training time and slow convergence. Liu et al. designed a particle swarm optimization trained fuzzy neural network algorithm to solve this problem [29]. But, the algorithm focused on the efficiency and convergence of the algorithm, and the complexity is not superior. Zhang et al. designed and trained a novel deep convolutional neural network with dual branches (DB-CNN) that improved the convergence speed by extracting global and local features, respectively [30]. It was a global path planning method. Sung et al. used two different offline path planning algorithms to generate different training data sets, trained the neural network path planner, and evaluated the performance of the two network models [31]. However, the network with better performance still exhibited collision phenomena in dense environments. The offline training of neural network depends on a large number of data samples, which makes it difficult to collect data and lacks autonomous learning ability. Chen et al. proposed a knowledge-free path planning approach based on reinforcement learning. Using the classic Q-learning algorithm and adding distance information can plan a shorter path [32]. Because the storage capacity of Q-table is limited, Q-learning exhibits space explosion when there are too many states in the running environment. Fakoor et al. presented a path planning method with a fuzzy Markov decision process, which used a fuzzy controller to select actions through the calculated value function [33]. This process yielded marked advantages when processing noise data. Lin et al. proposed an indoor path planning algorithm based on deep learning to classify obstacles [34]. For static obstacle avoidance, a ray tracing algorithm was proposed to avoid obstacles, and a waiting rule was proposed for dynamic obstacle avoidance. However, training data cannot include all obstacles. Yu et al. proposed a path planning method of mobile robot by neural networks and hierarchical reinforcement learning [35]. The simulation results show that the two were organically combined to improve the performance of mobile robots during path planning in static obstacles environments. Wang et al. proposed a novel navigation model with the combination of supervised learning in cerebellum and reward-based learning in basal ganglia, which used the motivated developmental network (MDN) to mimic the supervised learning of the cerebellum and reinforcement learning based on the radial basis function neural network (RBFNN) to simulate the reward-based learning of the basal ganglia [36]. In unexplored places, the artificial agent used the cerebellum model to choose actions instead of the ε-greedy method to accelerate the learning convergence speed of the basal ganglia. Compared to a single algorithm, the combination of different algorithms can give full play to their respective advantages and make up for the deficiencies of the other algorithms.

In this paper, fuzzy control, neural network, and reinforcement learning are combined to realize the local path planning task of mobile robot and solve the uncertainty in unknown environments. The uncertainty problem comes from the sensor detection error and unknown complex environments that can be resolved by the above algorithms. This paper primarily studies the local path planning problem of robots with unknown environmental information. Through the characteristics of fuzzy mathematics to transform the current robot state from infinite number combination to a fixed number of concepts, the error of decision-making caused by sensor detection error was avoided. But, in the complex environment, such a rough control is not easy to get a feasible path. However, if the states continue to increase, the number of rules will increase with the geometric series, the decision efficiency will decrease, and the probability of decision conflict will increase. In order to optimize the efficiency and length of path planning and improve the accuracy of decision-making, a neural network model can be constructed and trained to calculate the next decision directly through environmental data. There are many neural network models that can realize the prediction task. Compared with Back Propagation (BP) neural network, Adaptive Network-Based Fuzzy Inference System (ANFIS), and Radial Basis Function (RBF) neural network, Recurrent Neural Network (RNN) has a memory function for previous inputs, which has certain advantages in dealing with time series [37, 38]. As a variant of RNN, LSTM has a long-term memory function and is suitable for processing important events with long intervals and delays in time series [39, 40]. According to the common environment and the special obstacle environment with common problems abstracted from reality, representative samples are collected. And, robots can explore the uncollected environment through reinforcement learning, so as to get a path planner with strong adaptability. In addition, a reasonable neural network has generalization ability, and can be successfully implemented in the environment outside the training samples.

Based on the problems with these existing algorithms, this paper proposes a fusion method of local path planning for mobile robots, which combines LSTM neural network, fuzzy control, and reinforcement learning. In the proposed fusion method, various algorithms complement each other to mitigate deadlock, path redundancy, and improve time efficiency. The main contributions of this paper are as follows:

(1) LSTM neural network with memory function is used to fit the samples generated by fuzzy control algorithm, which retains the original function of FL, and improves the time performance compared with FL.

(2) The model training uses offline pretrained methods to avoid the problem of long training time and slow convergence in reinforcement learning [41].

(3) The LSTM_FTR model combined with the three algorithms can learn new rules, optimize the path length, enhance the learning and generalization ability of the network, and be suitable for more work scenes.

The remainder of this paper is organized as follows. Section 2 discusses the problem definition and assumptions in this paper. The design of the path planner based on the LSTM neural network is described in Section 3. Section 4 shows the pretraining process of the transfer learning network using the FL algorithm. In Section 5, the concrete process of strengthening the learning and training network is presented. Section 6 demonstrates and discusses the test results. The last section outlines the conclusions and future work.

2. Problem Definition and Assumptions

The navigation problem defined in this paper simulates the transportation task of robots on indoor level ground. The locations of indoor objects are not fixed, and the robot needs no indoor map information. Based on the task requirements, the start and end points are specified, and the robot is placed at the starting point. The goal of the robot is to reach the target point from the start point while safely avoiding obstacles using the local path planning algorithm. In this paper, a two-wheeled differential-drive robot with a lidar sensor is used. The left and right wheels are independently driven by two DC servo motors, and the front and rear wheels are two universal wheels that can turn freely to support the robot.

Assumption 1. The 180° area in front of the robot is selected as the detection range of obstacles, as shown in Figure 1. The detection distance of the sensor is 5 meters, and any obstacle information outside this range is unknown. One measurement value is returned every 5°, and a total of 37 groups of data are obtained. The obtained data are divided into five groups, and the minimum value of each group represents the distance between the robot left L, left front LM, front M, right front RM, and right R and the obstacle, respectively. The range of the sensor is [0, 5] m.

Assumption 2. During task execution, the robot knows the location coordinates of the target point. At each time step, the angle $Angle\_\theta$ between the current direction of the robot O_{r_head} and the direction of the target point O_{goal} can be obtained, $Angle\_\theta \in [-\pi, \pi]$.

Assumption 3. During task execution, the robot is always aware of its own position information $p = [x_t, y_t, \theta_t]$ at each time step, where t represents the current moment; (x_t, y_t) is the coordinate of the robot's position at time t; and θ_t is the angle between the robot's forward direction and the x-axis. The motion of the robot is shown in Figure 2, and the kinematics model of the robot is

$$\begin{cases} x_{t+1} = x_t + vT\cos\theta_t, \\ y_{t+1} = y_t + vT\sin\theta_t, \\ \theta_{t+1} = \theta_t + T\omega, \end{cases} \tag{1}$$

where T is the sampling period. The pose of the robot (x_{t+1}, y_{t+1}) at time t+1 is calculated based on the pose of the previous time t, the linear velocity ω, and the angular velocity v of the current motion.

3. Design of LSTM Neural Network Path Planner

This section introduces the working principle of the LSTM neural network and the design process of the path planner. The design of the LSTM neural network path planner includes state input, decision output, and the LSTM neural network model.

3.1. Working Principle of LSTM. Due to its unique structure and working principle, the LSTM neural network has the function of long-term memory that allows the network to predict results based on previous data and current input

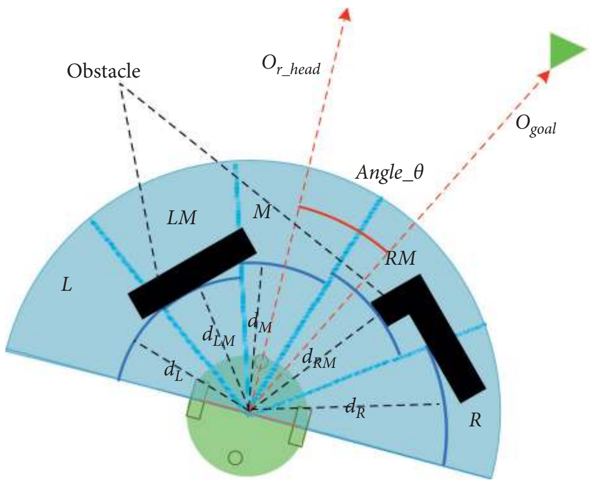


Figure 1: Robot model.

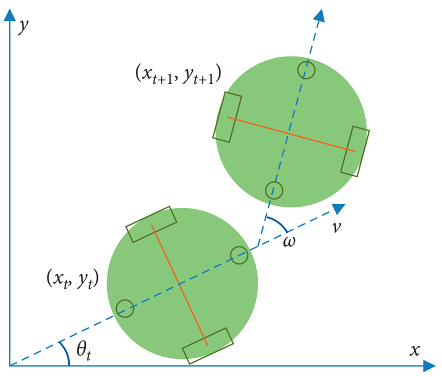


Figure 2: Robot motion diagram.

information, which makes it suitable for processing sequence problems. LSTM can solve the problem that occurs when the input state is the same, but the output action is different in certain environments, allowing the robot to make diverse predictions based on previous path information.

As shown in Figure 3, if we do not consider the previous state data and only judge based on the current state, decision conflicts will occur. In Figure 3(a), the current state of the robot is that there are obstacles located in the *RM* and *R* directions, and the target point is on the left side of the robot. The robot should turn left to avoid obstacles and continue to drive toward the target point. The current state of the robot in Figure 3(b) is exactly the same as that in Figure 3(a). If the same obstacle avoidance strategy is used, a local deadlock will appear. If (a) and (b) are made into two different states based on previous information, the robot in state (b) should move along the obstacle to leave the trap area and then move toward the target point. Therefore, the LSTM neural network structure can be used to design a network model when known information is limited.

The long-term memory ability of LSTM depends on its unit structure, as shown in Figure 4. The inputs of each time sequence pass through these cells, and the concept of the gate is used in the cell structure. The predicted value of the current time is calculated by controlling the state information of the forgetting gate, input gate, and output gate, where t is the current time, x is the input, h is the output, σ is the sigmoid layer, and tanh is the hyperbolic tangent layer. The calculation process of the predicted value is as follows:

Step 1: The sigmoid layer of the forgetting gate determines how much information is forgotten from the previous cell state. The f_t obtained by h_{t-1} and x_t through the sigmoid activation function is a number between 0 and 1, where 1 represents the complete retention of information, and 0 represents the complete discarding of information. f_t is calculated as

$$f_t = \sigma\left(W_{fh}h_{t-1} + W_{fx}x_t + b_f\right). \tag{2}$$

Step 2: The sigmoid layer of the input gate determines which value will be updated. i_t is calculated, then computes the current status candidate $\widetilde{C}_t$ and the status information C_t of the current moment:

$$i_t = \sigma\left(W_{ih}h_{t-1} + W_{ix}x_t + b_i\right), \tag{3}$$

$$\widetilde{C}_t = \tan h\left(W_{Ch}h_{t-1} + W_{Cx}x_t + b_C\right), \tag{4}$$

$$C_t = f_t C_{t-1} + i_t\widetilde{C}_t. \tag{5}$$

Step 3: The sigmoid layer of the output gate determines which cell state to output. o_t is calculated, the cell states are normalized by the tanh function to $[-1, 1]$, and the result of the output gate is multiplied to obtain the value h_t at the current time:

$$o_t = \sigma\left(W_{oh}h_{t-1} + W_{ox}x_t + b_o\right), \tag{6}$$

$$h_t = o_t \tan h\left(C_t\right). \tag{7}$$

Through these processes, the output h_t of LSTM layer at time t is obtained. In formulae (2)–(7), the weight matrices W_{fh}, W_{ih}, W_{Ch}, W_{oh}, W_{fx}, W_{ix}, W_{Cx}, and W_{ox} and bias terms b_f, b_i, b_C, and b_o are 12 sets of parameters for LSTM training.

3.2. Design of Path Planner.

The LSTM neural network path planner is designed as a structure with seven inputs and a single output based on the requirements of local path planning tasks for mobile robots, as well as the kinematics model and sensor configuration of the robot, as shown in Figure 5.

In the input, the nearest obstacle distance detected in five orientations represents the perception of the surrounding obstacles, $Angle\_\theta$ represents the direction relationship between the robot and the target point, and r_{GR} represents the distance relationship between the robot near or away from the target point:

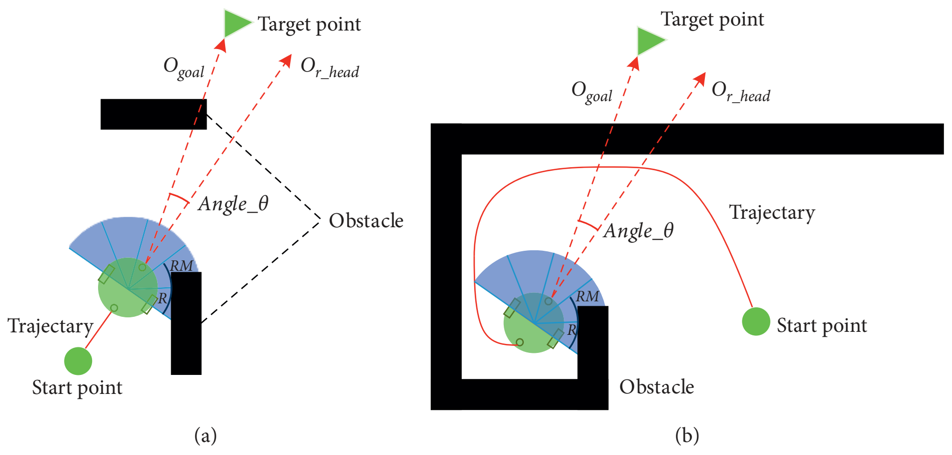


Figure 3: Diagram of decision conflict. (a) Simple environment. (b) Multi-U type environment.

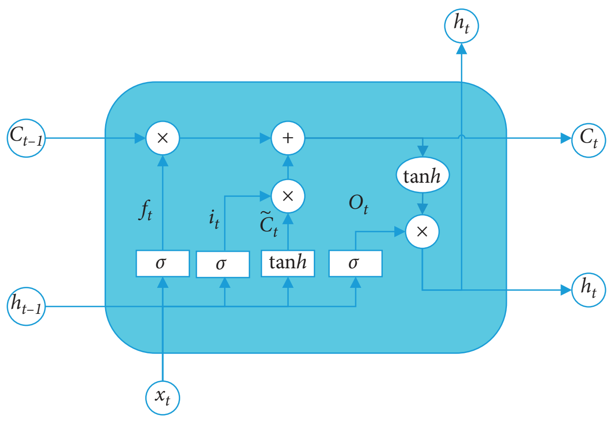


Figure 4: LSTM cell structure.

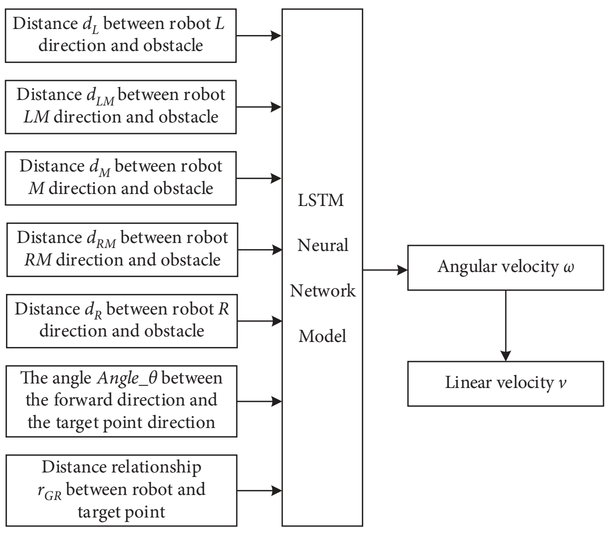

Distance d_L between robot L direction and obstacle
Distance d_{LM} between robot LM direction and obstacle
Distance d_M between robot M direction and obstacle
Distance d_{RM} between robot RM direction and obstacle
Distance d_R between robot R direction and obstacle
The angle $Angle\_\theta$ between the forward direction and the target point direction
Distance relationship r_{GR} between robot and target point
LSTM Neural Network Model
Angular velocity ω
Linear velocity v

Figure 5: Path planner of the LSTM neural network.

$$r_{GR} = \begin{cases} 1, & d_{GNR} < 5 \text{ and} |Angle\_\theta| < \dfrac{\pi}{2}, \\ d - GLRd_{GNR}, & else, \end{cases} \tag{8}$$

where d_{GLR} represents the distance between the position of the robot and the target point at the last moment, and d_{GNR} represents the distance between the position of the robot and the target point at the current moment.

The model output is the angular velocity ω, where $\omega \in [-\pi/2, \pi/2]$. The linear velocity v is calculated based on the angular velocity and the safe distance from the obstacle. The robot travels based on the obtained angular velocity and linear velocity. The safe distance between the robot and the obstacle is not below 1 m, and the maximum linear speed is 0.6 m/T. Based on the safe and feasible distance and the principle of angle deceleration, the linear velocity is calculated:

$$v = \min\left(\max\left(d_{FO} - l - 1, 0.001\right), 0.6\right) \cdot \left(1 - \frac{2|\omega|}{\pi}\right), \tag{9}$$

where d_{FO} is the distance between the robot's next direction and the obstacle, and l is the distance between the robot's left and right wheels.

The defined LSTM model is shown in Figure 6 and consists of five layers. The first layer is the sequence input layer, which contains seven neurons as feature input sequences. The second layer is the LSTM layer, which includes several hidden LSTM units. The determination of the hidden layer nodes is related to the training data sets and the neural network model. The third and fourth layers are fully connected (FC) layers. The fifth layer is the regression layer, which calculates the mean square error loss of the regression problem and follows the final FC layer to obtain the prediction value.

The structure of the LSTM neural network designed in this paper is shown in Figure 7, where seven green nodes of each time sequence correspond to seven inputs in Figure 5. Blue nodes represent LSTM neuron nodes in the first hidden layer, adjacent gray nodes indicate FC layer nodes in the second hidden layer, and the dark blue node in the last layer corresponds to the single output. It is assumed that for a set of path planning data containing n sequences, n steps are required to reach the target point, corresponding to n time series. The input of each time sequence is related to the previous time sequence. Taking moment $t = 2$ as an example, the output value of the current time is determined by the input x_2 of the current time and the state C_1 of the previous time; thus, the neural network model can remember the previous data.

The training methods of the neural network model include unsupervised and supervised learning methods. Unsupervised learning continuously explores and tries different solutions in various environments, rewards correct behavior, punishes incorrect behavior, and fully learns the best behavior in various states to obtain the final model. However, inexperienced autonomous learning requires long

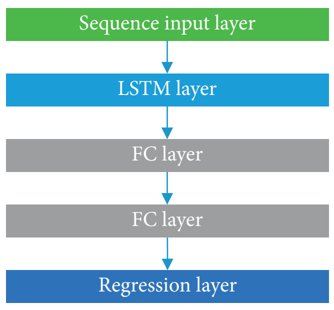


Figure 6: LSTM neural network model.

computation time and energy, and requires redesigning and debugging many functions and parameters that are prone to improper settings and typically lead to nonconvergence. Supervised learning uses many path data sets to supervise model training so that the model can fit the data set, learn the rules from the data, and be somewhat generalizable. However, obtaining the training data set is critical. In previous work [42], we proposed a path planning algorithm (FL) based on fuzzy control, which can be used for data collection. This paper uses both unsupervised and supervised learning with collected data to pretrain the model and then optimizes the model through reinforcement learning.

4. Pretraining of Fuzzy Logic Control Transfer Learning

FL is a four-input model with a small detection range. In some complex environments, FL will produce unnecessary redundant paths due to the limitation of known environmental information. If the fuzzy controller with the same structure is redesigned, the number of fuzzy rules to be designed will increase with increasing input; thus, it is difficult to set effective conflict-free rules. In addition, the more the rules, the longer the time each decision takes, which reduces the efficiency of the system. Therefore, a low-dimensional input fuzzy algorithm is used to collect the data required by the neural network model to pretrain the network model via transfer learning. The FL algorithm includes the design of the behavior fusion fuzzy controller of the robot in the general environment and the solution strategy of multiple U-traps in the special environment.

4.1. Fuzzy Logic Control. Common obstacle avoidance behavior and walking along the wall behavior are combined into a fuzzy controller through the design of the fuzzy control rules [43].

The fuzzy controller is designed as a structure of four inputs and two outputs, as shown in Figure 8, where the inputs are d_{LM}, d_M, d_{RM}, and $Angle\_\theta$, and the outputs are v_l, and v_r. Set up the membership function of the inputs and outputs, and establish the fuzzy control rules. The inputs and outputs are fuzzy, and fuzzy language variables are specified. Table 1 shows the domain and semantics of fuzzy variables.

Figure 9 shows the membership function of input $Angle\_\theta$. When the value of $Angle\_\theta$ falls into L, M, or R, the

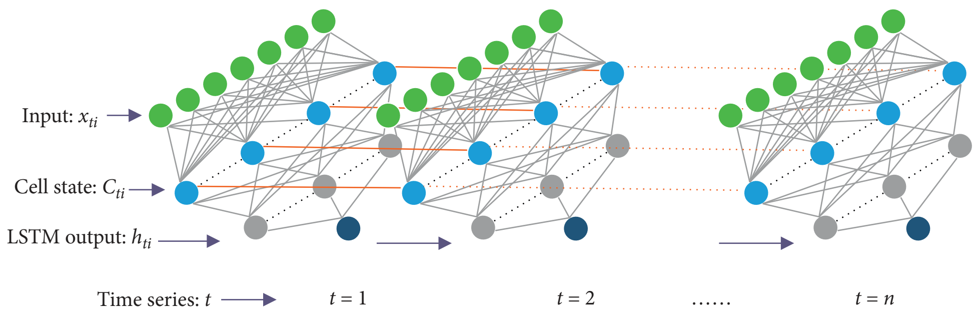


FIGURE 7: LSTM neural network structure.

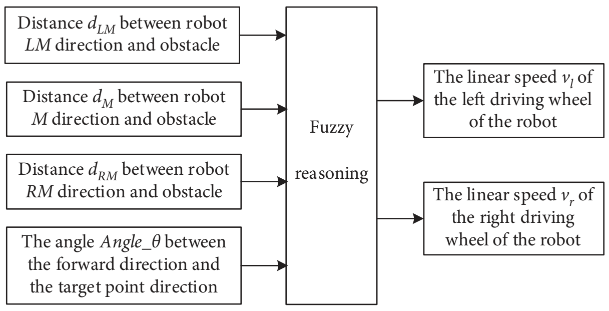

Distance d_{LM} between robot LM direction and obstacle
Distance d_M between robot M direction and obstacle
Distance d_{RM} between robot RM direction and obstacle
The angle $Angle\_\theta$ between the forward direction and the target point direction
Fuzzy reasoning
The linear speed v_l of the left driving wheel of the robot
The linear speed v_r of the right driving wheel of the robot

FIGURE 8: Fuzzy controller structure.

TABLE 1: Domain and semantics of fuzzy variables.

Fuzzy Variable	$d_{LM}d_Md_{RM}$	$Angle\_\theta$		$v_l \cdot v_r$
Discourse domain	[0,5] m	[$-3\pi/2$, $3\pi/2$]		[0,0.8] m/T
Fuzzy linguistic variable/Corresponding semantics	*NF* NearFar	*LB*	The robot is at the left rear of the target point	*SMF* SlowMediumFast
		L	The robot is at the left front of the target point	
		M	The robot is at the front of the target point	
		R	The robot is at the right front of the target point	
		RB	The robot is at the right rear of the target point	

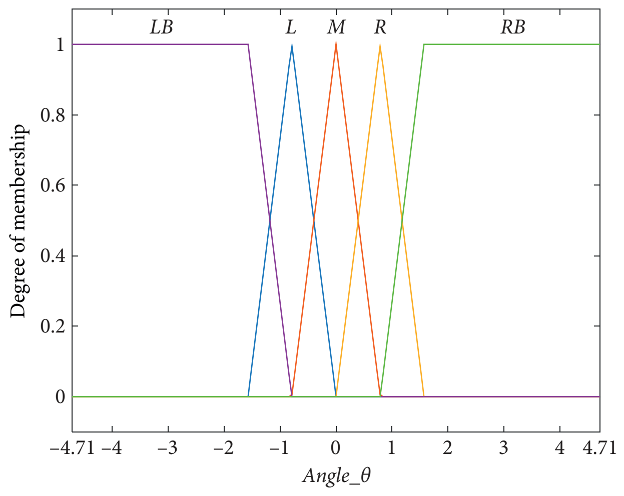


FIGURE 9: Membership function of input *Angle_θ*.

fuzzy rules are set as ordinary obstacle avoidance. When the value of *Angle_θ* falls into *LB* or *RB*, the fuzzy rules are set as the behavior walking along the wall. The designed fuzzy control rules are shown in Table 2, and the Mamdani method is used for fuzzy reasoning [44].

As shown in Table 2, there are two actions to be chosen from when $Angle\_\theta = M$ $[d_{LM}d_{M}d_{RM}] = FNF$ or NNN. The two optional actions are divided into two fuzzy controllers, and the other rules remain unchanged. In these cases, the fuzzy controller that chooses the left turn (*SM*, *SF*) is called the left turn fuzzy controller, and the fuzzy controller that chooses the right turn (*MS*, *FS*) is called the right turn fuzzy controller.

4.2. Multiple U-Shaped Complex Obstacle Solution Strategy. To solve the deadlock problem of the fuzzy controller running in a multiple U-shaped complex obstacle environment, a solution strategy of multiple U-shaped obstacles is proposed. A variable *wc* is set as the cumulative rotation angle sum, and its initial value is zero. Its left turn *Angle_θ* is a positive value, and its right turn *Angle_θ* is a negative value. Therefore, the absolute value of *wc* will not exceed 180° when the robot walks normally. If it is more than 180°, it means the robot is trapped in the trap area. Setting *Angle_θ* based on the following formula, the robot can continue to walk along the wall until it escapes the trap:

$$Angle\_\theta = \begin{cases} Angle\_\theta + \pi, Angle\_\theta \in \left(0, \dfrac{\pi}{2}\right], \\ \\ Angle\_\theta - \pi, Angle\_\theta \in \left[-\dfrac{\pi}{2}, 0\right]. \end{cases} \tag{10}$$

When using a fuzzy controller to generate a data set, the next decision is made based on the accumulated rotation angle *wc* and input parameters obtained from sensors. Although *wc* is not included in the dataset, the dataset generated contains relevant information; thus, the LSTM neural network can learn this function via training with its own memory properties.

4.3. Data Set. The input and output of fuzzy control are different from the required training data. The information is converted into the required data through calculation while collecting them. The detection range of the sensor is expanded to obtain obstacle information in five directions; calculate the distance relationship r_{GR} between the robot and the target in each step; and convert the output linear velocity of the left and right wheels into linear velocity and angular velocity record data, respectively. l is the distance between the two wheels:

$$\begin{cases} v = \dfrac{v_l + v_r}{2}, \\ \\ \omega = \dfrac{v_l - vr}{l}. \end{cases} \tag{11}$$

In barrier-free environments, common obstacle environments, and complex obstacle environments, the FL algorithm is used for 150 simulation experiments, and 150 groups of data are collected. The frequencies of left and right turns cannot be guaranteed to be the same, and it is easy to walk in the same direction after model training. Therefore, the 150 sets of paths are mirrored, and 150 new sets of data are obtained. Therefore, the training set consists of 300 sets of data, each of which contains several sequences. Table 3 lists one set of 52 sequences, each containing 8 data, of which 7 are inputs and 1 is output. Another 30 experiments are conducted to obtain 30 groups of data as the test set.

4.4. Pretraining of the LSTM Neural Network. The back propagation algorithm is used to train the LSTM neural network model. The weight matrices W_{fh}, W_{ih}, W_{Ch}, W_{oh}, W_{fx}, W_{ix}, W_{Cx}, and W_{ox} and bias terms b_f, b_i, b_C, and b_o are 12 sets of parameters that require training. First, the network parameters are initialized based on the training data, the prediction output value of each time is calculated forward, and the error term of each neuron is calculated backward. Then, the gradient of each weight is calculated based on the error term. Finally, the training parameters are determined by the gradient descent method. Because the original gradient descent method is prone to gradient explosion and disappearance, the adaptive moment estimation (Adam) optimizer is selected to update the weights, which is suitable for solving the optimization problem with large-scale data and parameters, and avoiding gradient explosion.

The selection of network parameters is determined via experiments, and the orthogonal experimental design (OED) method is used to design the experiment [45]. The number of levels for four factors are set as follows: five levels for the learning rate $LR \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$, five levels for the batch size $BS \in \{10, 20, 50, 100, 150\}$, five levels for the training time $TT \in \{200, 300, 400, 500, 600\}$, and five levels for the hidden layer $HL \in$ {L128, L150, L100F50, L128F60, L100L50}. L100 represents the LSTM layer that contains 100 LSTM neurons, and F50 represents the fully connected layer that contains 50 neurons. A complete factor analysis requires $5^4 = 625$ experiments. Compared to total factor analysis, the OED method uses an orthogonal array, which markedly reduces the number of required experiments, time cost, and labor cost. Therefore, we used an orthogonal array $L_{25}(5^4)$ with only 25 experiments.

Table 4 shows the experimental design and results: the closer the root mean square error (RMSE) and loss values are to zero, the better the prediction results will be. The mean square error (MSE) is used as the loss function, and R^2 is the determinant of the fit degree of the model, where the closer R^2 is to 1, the better the model fits the data:

$$R^2 = 1 - \frac{\sum (y_i - h_i)^2}{\sum (y_i - \overline{y})^2}. \tag{12}$$

The success rate is a performance index of whether the goal can be reached by each model based on 100 groups of randomly determined start and end points. Figure 10 shows

Table 2: Fuzzy control rules.

$v_l \cdot v_r$		$d_{LM}d_Md_{RM}$							
		FFF	*FFN*	*FNF*	*FNN*	*NFF*	*NFN*	*NNF*	*NNN*
	L	*MS*	*FF*	*MS*	*SM*	*MS*	*MS*	*FS*	*SF*
	M	*FF*	*FF*	*SM or MS*	*SM*	*FF*	*FF*	*MS*	*SF or FS*
Angle_θ	*R*	*SM*	*SM*	*SM*	*SM*	*FF*	*SM*	*MS*	*FS*
	LB	*MS*	*FF*	*MS*	*SM*	*MS*	*FF*	*FS*	*SF*
	RB	*SM*	*SM*	*SM*	*MS*	*FF*	*FF*	*FS*	*FS*

Table 3: Data set.

No.	Input							Output
	d_L	d_{LM}	d_M	d_{RM}	d_R	*Angel_θ*	r_{GR}	ω
1	3.8487	5	3.3419	5	5	0	0	0.0441
2	3.6353	2.8085	2.8600	5	5	−0.0450	0.4806	−0.0734
3	3.4723	2.3934	2.6536	5	5	0.0289	0.4458	−0.0871
4	3.3491	2.0446	5	5	5	0.1181	0.4295	0.0124
5	3.2897	1.6592	5	5	5	0.1082	0.5362	0.0099
6	5	12964	5	5	5	0.1007	0.5525	0.0083
7	0.9444	0.9462	5	4.4307	4.4307	0.0947	0.5648	0.0072
8	0.6002	0.6010	5	4.0658	4.0658	0.0898	0.5743	0.0064
9	0.2599	5	5	3.7345	3.7345	0.0857	0.5815	0.0060
10	5	5	5	3.5361	3.5361	0.0820	0.5852	0.0055
52	5	5	5	3.0623	5	0.2163	1	0.0306

Table 4: Orthogonal experimental design and results.

No.	*LR*	*BS*	*TT*	*HL*	Training set			Test set			Success rate (%)
					R^2	RMSE	Loss	R^2	RMSE	Loss	
1	0.1	10	200	L128	0.6246	0.0944	0.0089	0.66	0.0941	0.0089	31
2	0.1	20	300	L150	0.1955	0.1383	0.0191	0.198	0.1445	0.0209	25
3	0.1	50	400	L100 F50	0.0447	0.1507	0.0227	0.0739	0.1553	0.0241	21
4	0.1	100	500	L128 F64	−0.2102	0.1696	0.0288	−0.2188	0.1782	0.0317	9
5	0.1	150	600	L100 L50	0.8379	0.0621	0.0039	0.8082	0.0707	0.005	42
6	0.05	10	300	L100 F50	0.5611	0.1021	0.0104	0.5769	0.105	0.011	33
7	0.05	20	400	L128 F64	0.7094	0.0831	0.0069	0.7613	0.0789	0.0062	35
8	0.05	50	500	L100 L50	0.8929	0.0504	0.0025	0.84	0.0646	0.0042	47
9	0.05	100	600	L128	0.907	0.047	0.0022	0.8528	0.0619	0.0038	60
10	0.05	150	200	L150	0.7686	0.0742	0.0055	0.7701	0.0774	0.006	28
11	0.01	10	400	L100 L50	0.9327	0.04	0.0016	0.8741	0.0573	0.0033	75
12	0.01	20	500	L128	0.9177	0.0442	0.002	0.8757	0.0569	0.0032	63
13	0.01	50	600	L150	0.9381	0.0384	0.0015	0.8841	0.055	0.003	72
14	0.01	100	200	L100 F50	0.9128	0.0455	0.0021	0.8714	0.0579	0.0033	75
15	0.01	150	300	L128 F64	0.9359	0.039	0.0015	0.8748	0.0571	0.0033	92
16	0.005	10	500	L150	0.9351	0.0393	0.0015	0.8918	0.0531	0.0028	80
17	0.005	20	600	L100 F50	0.9336	0.0397	0.0016	0.8836	0.0551	0.003	91
18	0.005	50	200	L128 F64	0.8797	0.0535	0.0029	0.8629	0.0598	0.0036	76
19	0.005	100	300	L100 L50	0.945	0.0362	0.0013	0.8875	0.0541	0.0029	86
20	0.005	150	400	L128	0.9463	0.0357	0.0013	0.8778	0.0564	0.0032	97
21	0.001	10	600	L128 F64	0.9593	0.0311	9.67E-04	0.8887	0.0538	0.0029	88
22	0.001	20	200	L100 L50	0.9213	0.0432	0.0019	0.8676	0.0587	0.0034	78
23	0.001	50	300	L128	0.9107	0.0461	0.0021	0.8703	0.0581	0.0034	91
24	0.001	100	400	L150	0.9053	0.0474	0.0022	0.8428	0.064	0.0041	82
25	0.001	150	500	L100 F50	0.8798	0.0534	0.0029	0.8481	0.0629	0.004	74

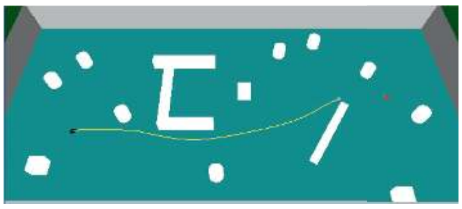


Figure 10: Success rate test environment.

Table 5: Variance analysis.

Source	SS	df	F	$F_{0.05}$
LR	1.481	4	3.650	3.010
BS	0.017	4	0.042	3.010
TT	0.080	4	0.197	3.010
HL	0.045	4	0.111	3.010
Total	1.62	16		

a mixed obstacle environment for testing the success rate. The blue color indicates the ground, the surrounding gray fence indicates a boundary in the indoor environment, white cuboids and cylinders indicate obstacles, the black dot indicates the start point, the red dot indicates the end point, and the yellow line indicates the path of the robot. Based on the designed space range, if the robot can reach the target point within 150 steps, then the path is successful.

Table 5 shows the analysis of variance (ANOVA) of the experimental results. The learning rate is shown to have a significant impact on the results among the four factors. If the learning rate is too low, the training time must be extended; if the learning rate is too high, there may be a local optimum. Figure 11 shows a comparison of the four factors with the training set fitting degree, test set fitting degree, and path planning success rate. The best learning rate is shown to be 0.005, the best batch size 150, and the lowest training times 600.

The results of the hidden layer structure in the training and test sets are not consistent; thus, they must be analyzed separately. The determination of the number of hidden layers and the number of neuron nodes has an important influence on the training and performance of the neural network, which is related to the amount of training data. L128 yields the best prediction effect in the training set because it has fewer nodes and is easy to train, and the amount of data in the training set is not large. However, with more training data, the model fitting ability will decline. L100L50 yields the best prediction effect in the test set because more layers and nodes yield better data fitting and stronger modelling ability, although the training speed decreases. However, the weight to be calculated increases accordingly, which may make the neural network difficult to train; thus, the success rate of path planning in the experiments is low. Based on the other determined parameters, the experiment is performed again, and the settings of the hidden layer are compared separately.

The experimental results are shown in Table 6. Although L128F64 yields a mediocre prediction with the training set, this model yields a better prediction with the test set, and the success rate of path planning is the highest in the 3D simulation obstacle environment. Therefore, the hidden layer setting of No. 4 is selected as the final training model. The model that is pretrained using fuzzy control transfer learning is called LSTM_FT, and the training process and results of the LSTM_FT model are shown in Figure 12 and Table 7, respectively. To facilitate observation, Figure 12 shows the training data of the first 150 times. Because each training divides the data into 2 batches, a total of 300 iterations are performed. Table 7 shows the details of the training process. With increased training time, RMSE and Loss gradually decrease and finally converge to more stable values.

Four groups of data are randomly selected from the test set, and the predicted values are obtained by the LSTM_FT model. The comparison of the predicted and real values are shown in Figure 13, in which red "O" represents the predicted values of the test data, and blue "+" represents the true values. The figures show that there are more data with output values near zero, and the prediction accuracy is also high. The number of other output values is small and scattered, and the predictive accuracy also decreases; however, the overall predictive ability is strong.

5. Combined with Reinforcement Learning Training Network

In the last section, the pretrained LSTM_FT model is developed. Combined with reinforcement learning, the purpose of the training network is to retain the original function of the model through the design of the reward function to learn autonomously in various environments. By fully learning the best mapping of all environmental states and prediction actions, the final trained model can plan a better path, improve planning efficiency, and be generalizable.

5.1. Principle of Reinforcement Learning. Reinforcement learning refers to the agent learning the best action corresponding to each state by interacting with the environment, that is, learning how to make the best decision of the agent. Reinforcement learning algorithms generally follow the Markov decision process. The basic elements of reinforcement learning mainly include state, action, policy, and reward. The purpose of reinforcement learning is to get the maximum cumulative reward and make the whole sequence decision optimal.

The principle of reinforcement learning is shown in Figure 14, where the agent represents the robot, the environment represents the running environment of the robot, state $s_t \in s$, s is a finite state set, s_t represents a state determined by the environment at the current moment, action $a_t \in A$, a is a finite action set, a_t represents an action made by the robot according to the environment at the current moment, and reward r is the immediate reward obtained by taking action a at state s. The agent selects the action through policy (which means the probability of generating action a according to the current state s).

The purpose of reinforcement learning is to maximize the long-term future reward. It does not need specific

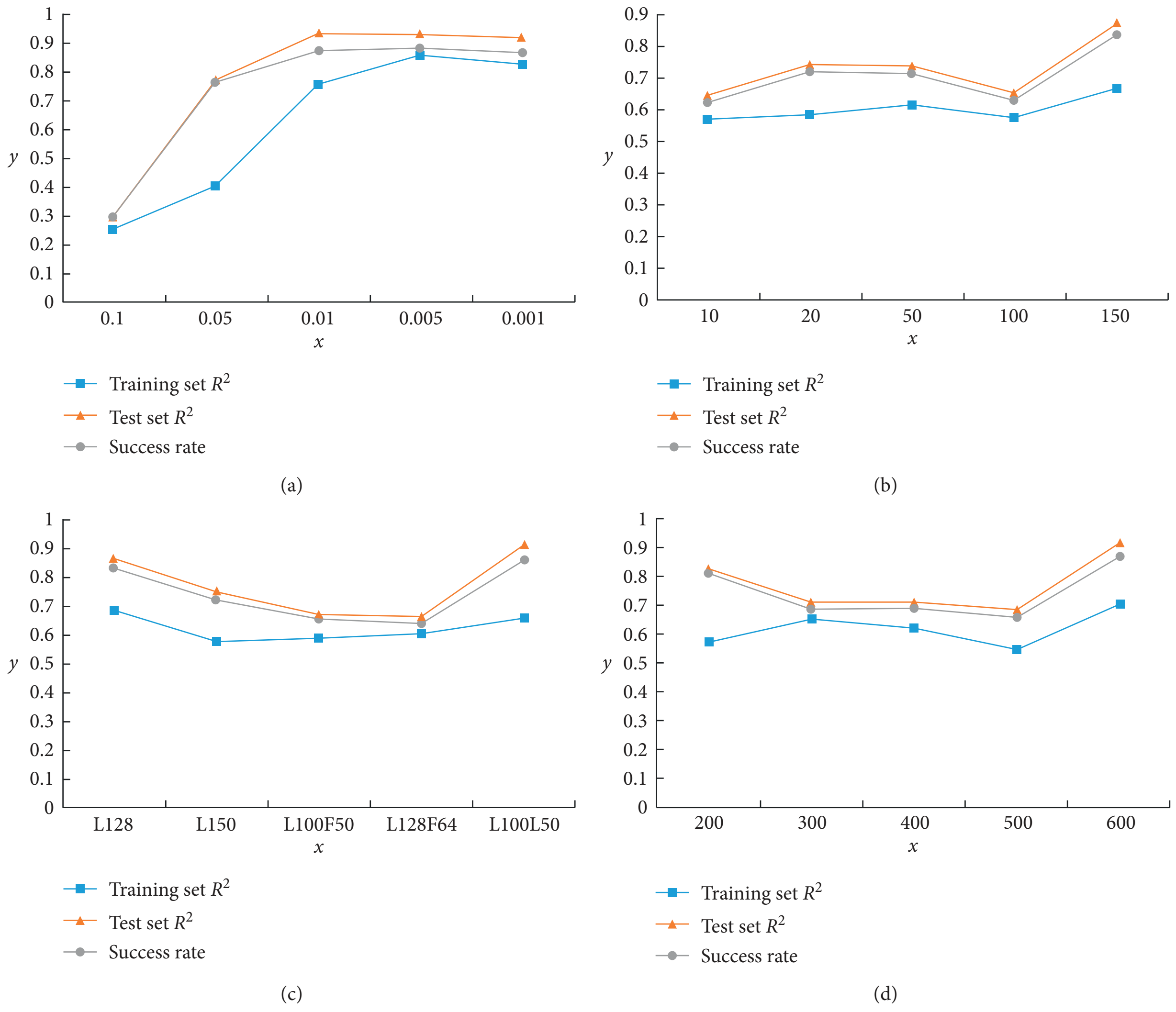


FIGURE 11: Effect graph. (a) Learning rate. (b) Batch size. (c) Hidden layer. (d) Training times.

TABLE 6: Experiments with different numbers of hidden layers and neuron nodes.

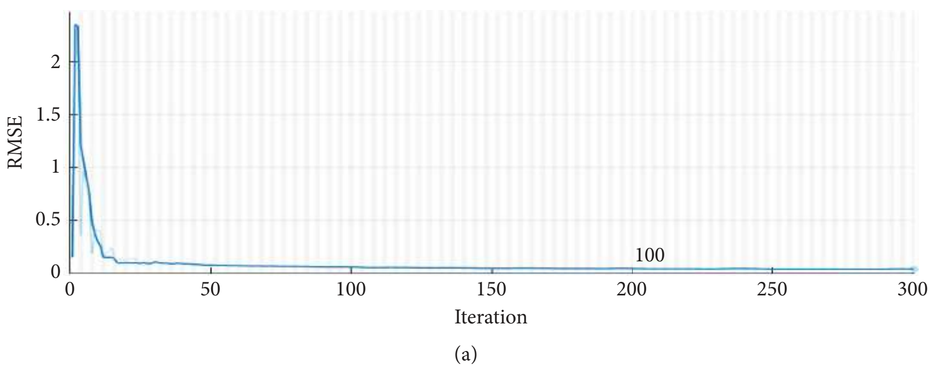
No.	Hidden layer	Training set			Test set			Success rate
		R^2	RMSE	Loss	R^2	RMSE	Loss	
1	L128	0.9558	0.0324	0.0011	0.8749	0.0571	0.0033	86%
2	L150	0.9537	0.0332	$1.10E-03$	0.8772	0.0566	0.0032	91%
3	L100 F50	0.9417	0.0372	$1.40E-03$	0.8727	0.0576	0.0033	88%
4	L128 F64	0.9468	0.0356	0.0013	0.8796	0.056	0.0031	97%
5	L100 L50	0.978	0.0229	$5.23E-04$	0.8721	0.0577	0.0033	89%

training data, only reward signals. However, the reward signal may not be given in real time, and in most cases it lags behind. Therefore, based on the prior experience, this paper uses reinforcement learning method to design, state, action, reward, and policy, so that the robot can learn autonomously in various environments to obtain better strategies.

5.2. Design of State and Action. Based on the LSTM neural network model, the "state-action" pairs <*s*, *a*> of reinforcement learning are designed. Because the input of the neural network is the detection value of the robot sensor, which is the value in the continuous interval, the state s is the return value of the sensor in the current environment. The state set S is expressed as:

$$S = \{s_t | s_t = (d_L, d_{LM}, d_M, d_{RM}, d_R, Angle\_\theta, r_{GR})\}. \quad (13)$$

The current environmental state information is the input to the pretrained network model to obtain the output of angular velocity ω. To make the model learn a more accurate mapping relationship and expand the range of action



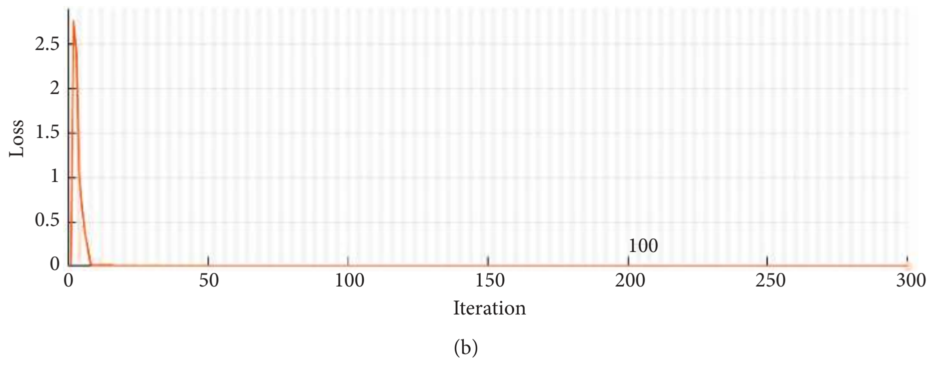


Figure 12: Training process of the model. (a) RMSE. (b) Loss.

Table 7: Detailed information about model training.

Epoch	Iteration	Time	Mini-batch RMSE	Mini-batch Loss	Learning rate
1	1	00 : 01	0.59	0.2	0.005
100	200	02 : 36	0.05	$1.0e-03$	0.005
200	400	05 : 16	0.04	$6.7e-04$	0.005
300	600	07 : 50	0.03	$5.5e-04$	0.005
400	800	10 : 23	0.03	$4.3e-04$	0.005
500	1000	13 : 13	0.03	$4.4e-04$	0.005
600	1200	15 : 50	0.03	$3.5e-04$	0.005

selection, a disturbance value ε is added to obtain three optional actions: $a_1 = \omega - \varepsilon$, $a_2 = \omega$, and $a_3 = \omega + \varepsilon$. The action set is $A = \{a_1, a_2, a_3\}$. Each time based on the state obtained by the sensor corresponds to three optional actions, and then the linear velocity is calculated based on the action selected by the strategy.

The strategy of action selection is to allocate the probability based on the exploration factor e. ω is the original action obtained by the LSTM neural network model based on the state input, and the exploration probability of selecting the other two actions is $e/2$, respectively. Thus, the selection probability of action $[a_1, a_2, a_3]$ is $[e/2, 1-e, e/2]$. The roulette algorithm is used to select the action and generate a random number $rand_p \in [0, 1]$. The action selection strategy is as follows:

$$a = \begin{cases} a_1, & 0 \le rand_p \le \dfrac{e}{2}, \\ a_2, & \dfrac{e}{2} < rand_p \le 1 - \dfrac{e}{2}, \\ a_3, & 1 - \dfrac{e}{2} < rand_p \le 1. \end{cases} \tag{14}$$

5.3. Design of Reward Function. Based on the size of the training environment, when the robot reaches the target point within 150 steps, the length and steps of the path are recorded and stored in the experience pool. If the target

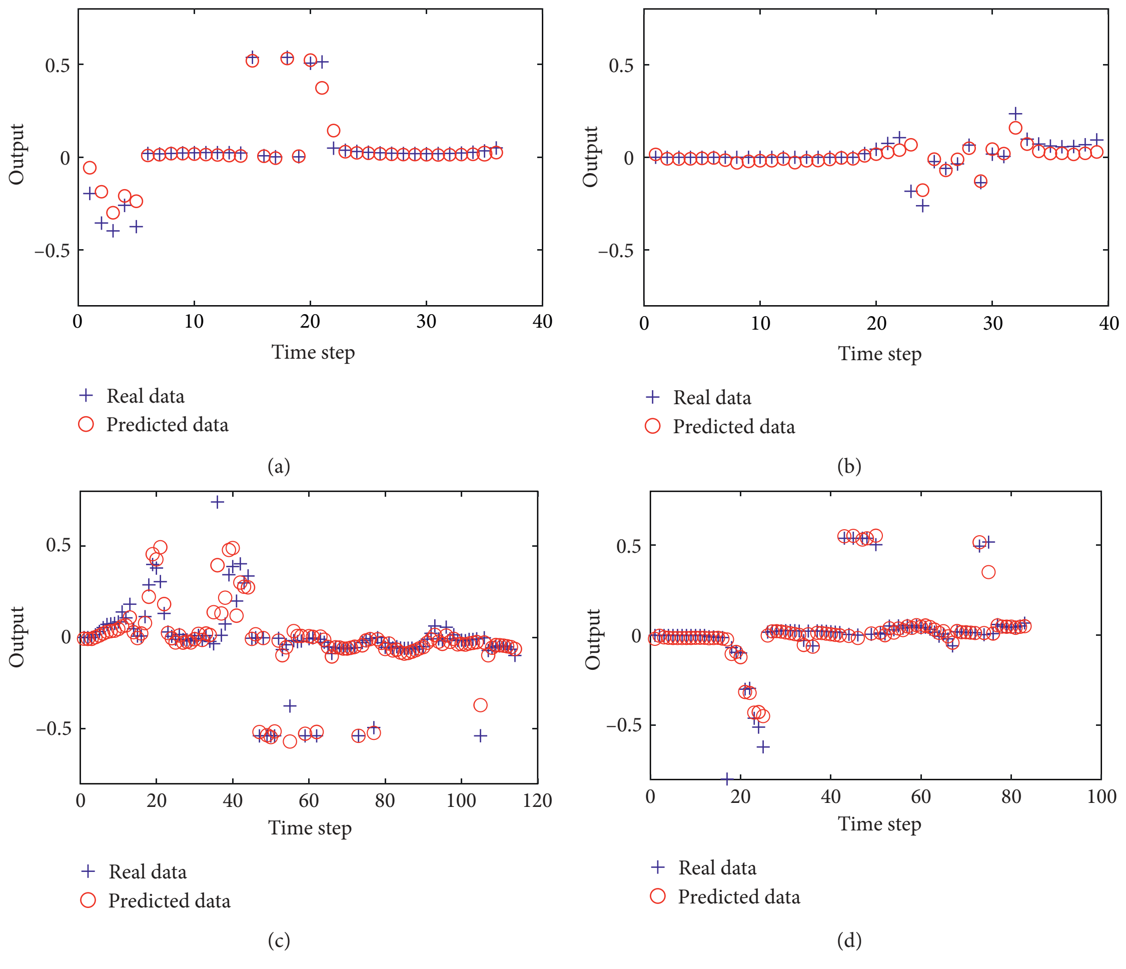


Figure 13: Comparisons between real and predicted values of the test set. (a) Test observation 8. (b) Test observation 10. (c) Test observation 29. (d) Test observation 28.

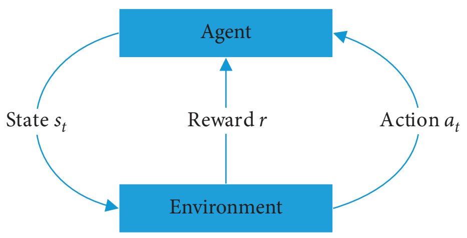


Figure 14: Schematic diagram of reinforcement learning.

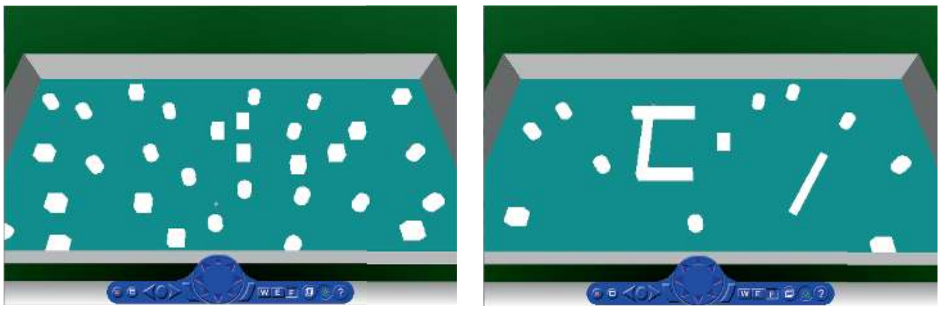
point is not reached within 150 steps, the path is discarded. The same group of start and end points yields 20 reachable paths and then resets the start and end points. Based on the reward function, the path with the minimum reward value r is selected as the optimal path and recorded as the strengthened data set. Reward function is calculated as:

$$r = 0.5r_{path} + 0.5r_{step}, \tag{15}$$

where r_{path} is the normalized path length of 20 paths, and r_{step} is the normalization steps.

There is no reward function for each decision. In certain cases, the decision-making based on reward setting in different environments produces conflicts. The rules set by humans cannot be considered comprehensively, and what the model learns is the process. The reward function is determined by the length of the path and the number of steps. The purpose of learning is to learn the shortest path. In addition, collisions will not occur if the sensor does not have a large detection error, which is suitable for vulnerable environments where collisions are not allowed. It may take a long time to adapt to various environments; however, previous experience with pretraining will markedly shorten training times.

5.4. Training Process of the LSTM_FTR Model. Each training randomly determines 20 sets of start and end points in Figure 15(a) to select 20 optimal routes and 15 sets of start and end points in Figure 15(b) to select 15 optimal routes. To reduce learning time, improve efficiency, and retain the ability to escape traps, 15 more pieces of data collected by the FL algorithm in obstacle-free and multi-U environments are added, and 100 pieces of data are obtained as training data each time after mirror processing of 50 pieces of data. The learning parameters are set as shown in Table 8. The network model has a priori knowledge and it does not need to select actions randomly; thus, the initial exploration factor e_i is set



(a) (b)

Figure 15: Learning environment. (a) Discrete and dense environment. (b) Mixed obstacle environment.

Table 8: Parameter settings.

Parameter name	Parameter value
Initial exploration factor e_i	0.5
Difference of each training factor Δe	0.1
Terminate exploration factor e_t	0.1
Learning rate	0.005
Training times	20

to 0.5. Due to the randomness of action selection, the data do not necessarily represent the best path; thus, the number of training iterations is set to 20.

Each round of learning should be trained five times based on the setting of exploration factors. The relationship between success rate and training times is shown in Figure 16. After each training, the new model is used to test the success rate in the test environment. The line chart shows that the success rate of the early training process shows a fluctuating upward trend. The success rate reaches 100% after 36 training iterations.

Figure 17 shows the path length of 100 paths in the test success rate after the model converges, and Figures 17(a) and 17(b) contain the change information of 50 paths, respectively. When training occurs 44 times, each path reaches a better value, before some path lengths increase with increasing training times. These results may be due to the randomness of action selection or the overfitting of the model, which weakens network generalization and stability. Therefore, 44 training results are considered to be the final network model LSTM_FTR.

The process of reinforcement learning is shown in Figure 18, which shows the test effect in a U-shaped environment after every two rounds of learning, where *Round* represents the number of learning rounds. As shown in Figures 18(a) and 18(b), the first four rounds did not develop a path that reaches the target point. In the sixth round, as shown in Figure 18(c), the model produces too many redundant paths, even though the path does reach the target point within the specified number of steps. Finally, in the eighth round, as shown in Figure 18(d), the model learned how to move along a shorter path to the target point.

6. Simulation Results and Discussion

In this section, experimental results are presented, and the characteristics of the proposed method are discussed. On the MATLAB R2020a platform, the algorithms proposed in this paper are simulated and verified in various environments. The computer used to run the algorithms is configured as follows: Windows 7 operating system, Intel (R) Pentium (R) CPU G3260 @ 3.30 GHz processor, and 6.00 GB running memory. In the simulation, "●" represents the start point, "▶" represents the target point, the black areas represent obstacles, and the green dotted line represents the straight line path from the start point to the target point. The red dotted line represents the planned path calculated by LSTM_FTR, the purple red dotted line indicates the planned path obtained by LSTM_FT, and the blue dotted line indicates the planned path acquired by FL.

6.1. Simulation Tests in Static Environments

6.1. Simulation Tests in Static Environments. The fusion method LSTM_FTR was tested in various barrier environments that were defined by a 30×30 two-dimensional rectangular coordinate system. Figure 19 shows the test results of FL, LSTM_FT, and LSTM_FTR in various environments and records the path length and average decision time, and the results are recorded in Table 9.

Based on the data in Table 9 and the paths shown in Figure 19, a comprehensive analysis is performed. The graphical comparison between the three algorithms based on path length and average decision time is shown in Figure 20. Based on the average decision-making time in Figure 20(b), the real-time decision-making performances of LSTM_FT and LSTM_FTR with the same neural network structure are similar in time performance; however, they are markedly better than the fuzzy logic control method. The neural network model can accelerate the calculation speed and improve the operating efficiency of the system. From Figure 19(a), the fully trained LSTM_FTR can fit the training data well, and the robot can travel straight to the target point in a barrier-free environment. Figure 19(b) is a discrete obstacle environment, and the network model that expands the detection range can judge the accessible area in time through the perceived information and make a more reasonable plan to obtain a shorter path. Figures 19(c) and 19(d) show that after intensive training, new rules are learned for the new states. After leaving the trap, the robot can judge the feasible direction and make decisions in time, so as to reduce the phenomenon of path redundancy and target unreachable. From Figure 19(e), the LSTM neural

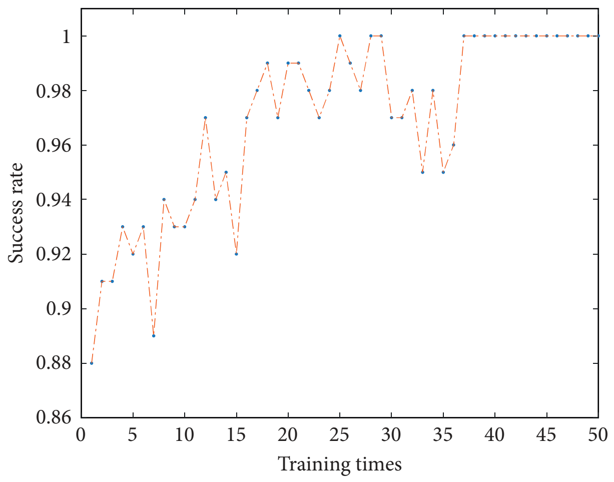


Figure 16: Relationship between training times and success rate.

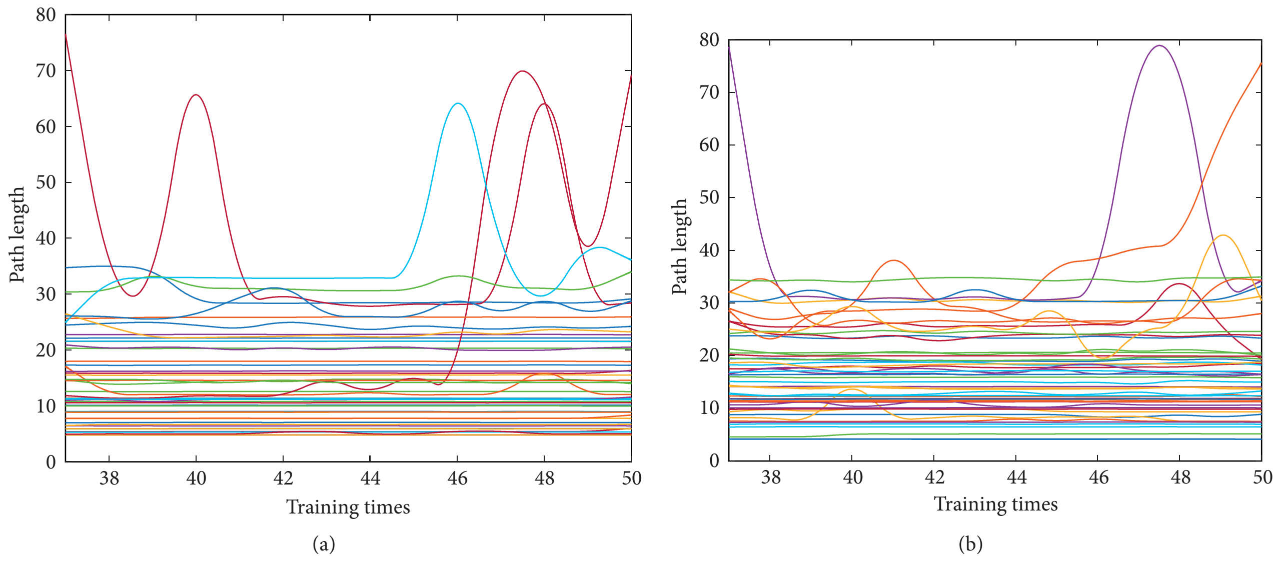


Figure 17: Relationship between training times and path length.

network learns the rule of data in the training set, which can avoid decision conflict by the memory ability, and the robot can escape multiple U-traps along the wall, avoiding local deadlock. In a complex environment, such as Figure 19(f), LSTM_FTR can also explore an accessible path. Simulation results show the feasibility and effectiveness of the proposed fusion algorithm.

In order to further verify the improvement of time performance and path planning success rate of the fusion algorithm, a 3D simulation of home environment is designed, as shown in Figure 21. In the figure, the wall and furniture are obstacles, the red dot represents the starting point, the black dot represents the target point, and the yellow curve represents the path that the robot has passed. In the simulation scene, 100 groups of start and end points are initialized randomly. Based on the designed space range, if the robot can reach the target point within 150 steps, then the path is successful. FL, LSTM_FT, and LSTM_FTR models are used for testing, and the results are shown in Table 10. The success rate of the LSTM_FTR method in unknown complex environment is 93%. The success rate of FL is only 64%. LSTM_FT fits FL rules and has generalization ability, so the success rate is slightly higher than that of FL, which is 68%. Compared with FL, LSTM_FTR saves 81.6768% decision time in time performance. The 3D simulation environment is close to the actual environment,

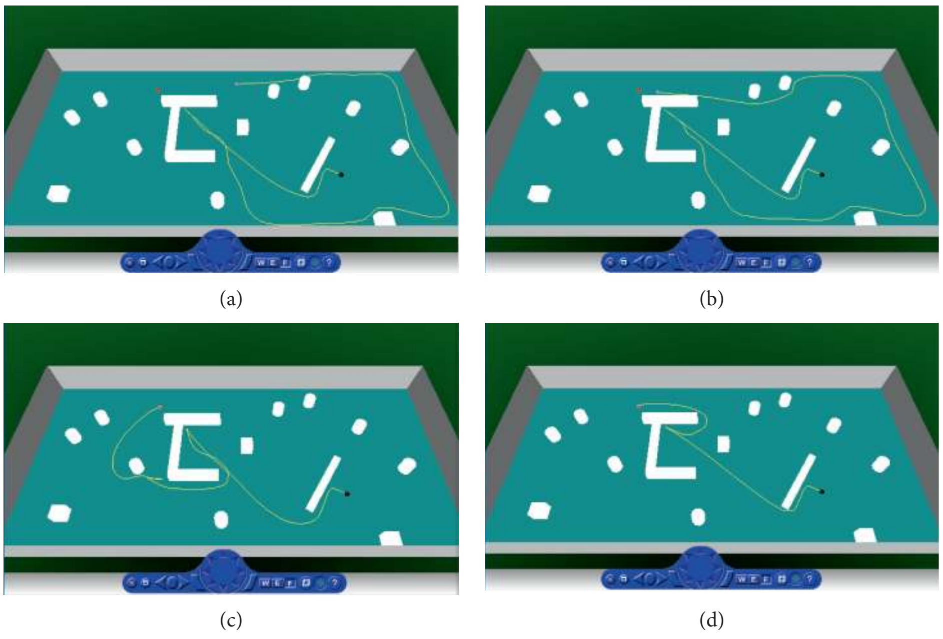


FIGURE 18: Test results during learning. (a) Round = 2. (b) Round = 4. (c) Round = 6. (d) Round = 8.

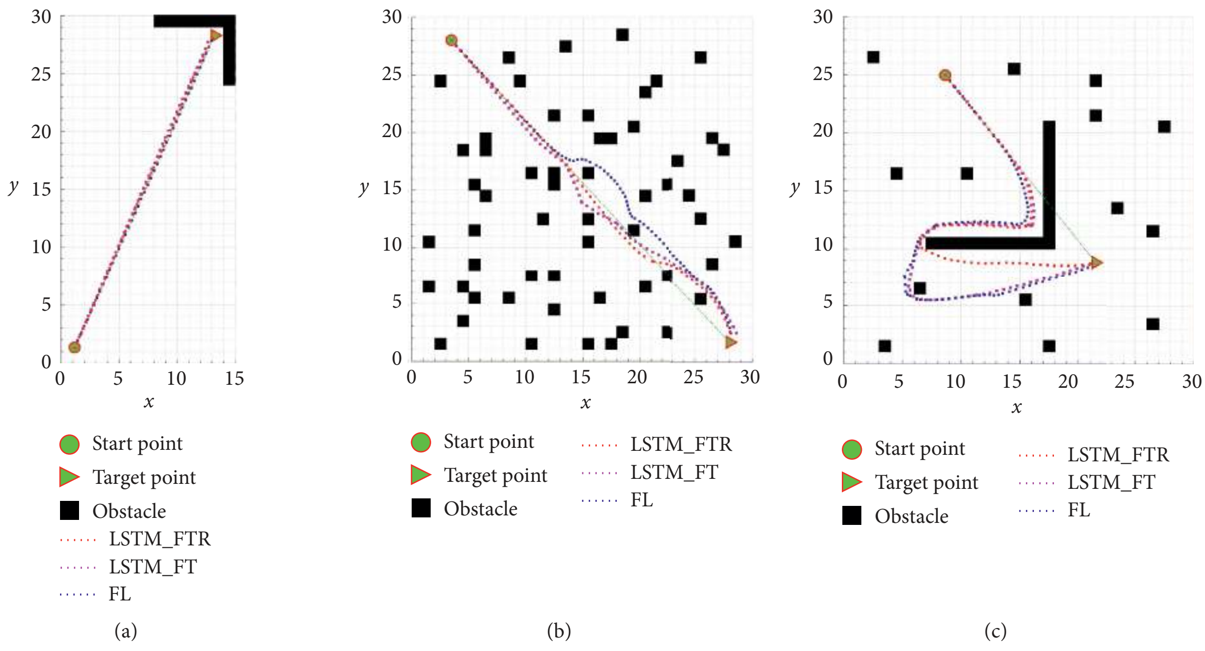


FIGURE 19: Continued.

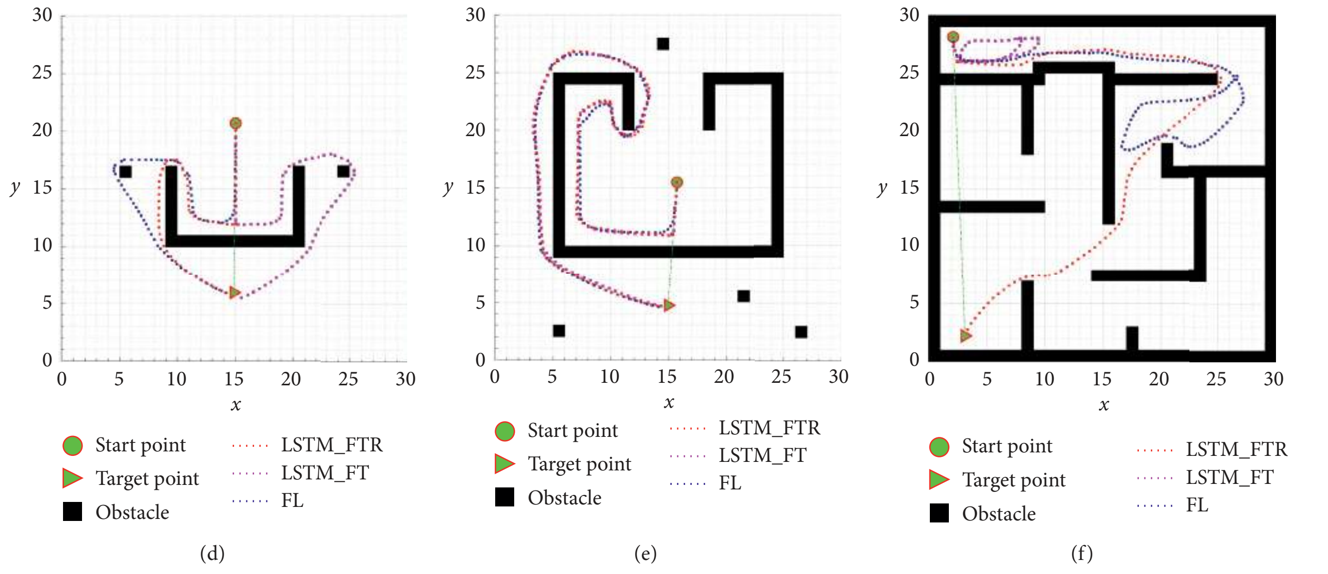


Figure 19: Comparison of the FL, LSTM_FT, and LSTM_FTR models. (a) Obstacles near the target point. (b) Discrete obstacles. (c) L-shaped obstacles. (d) U-shaped obstacles. (e) Multiple U-shaped obstacles. (f) Complex obstacle environment.

Table 9: Results of the FL, LSTM_FT, and LSTM_FTR models in Figure 19.

Figure 19	FL		LSTM_FT		LSTM_FTR	
	Path length	Average decision time (s)	Path length	Average decision time (s)	Path length	Average decision time (s)
a	29.5418	0.4906	29.5878	0.3619	29.5469	0.3747
b	37.4485	0.6677	37.5232	0.4496	36.8804	0.4528
c	46.3133	0.5446	46.7485	0.4197	40.9222	0.4131
d	38.2498	0.5212	39.8980	0.3928	33.7467	0.3813
e	70.0993	0.6475	71.1376	0.5141	72.0711	0.5174
f					59.1128	0.7891

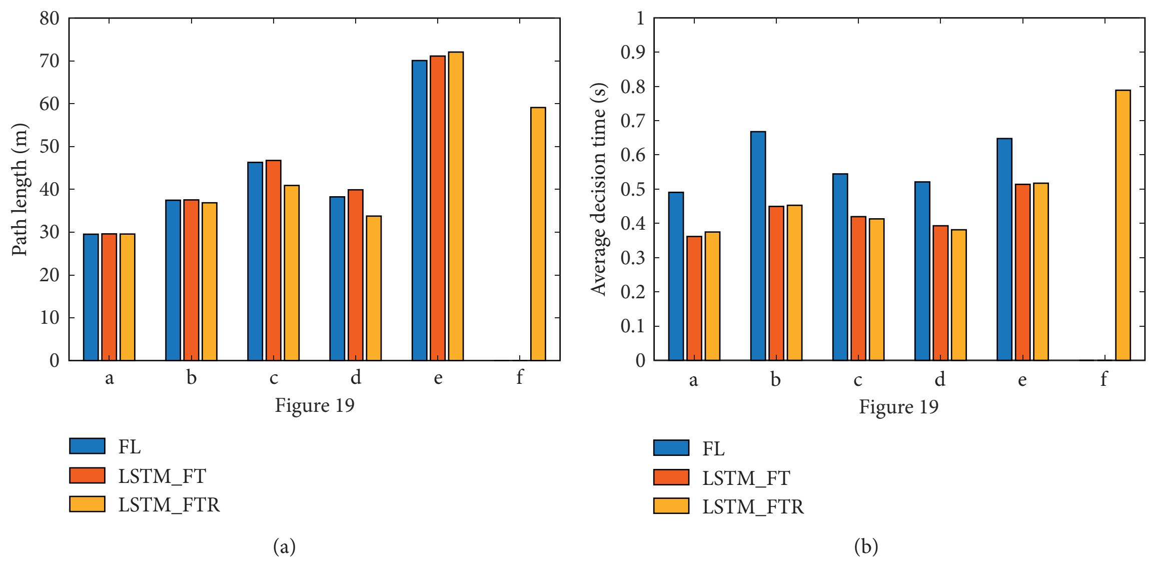


Figure 20: Graphical comparison of path length and average decision time of three algorithms. (a) Path length. (b) Average decision time.

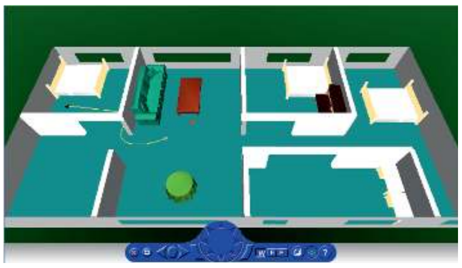


Figure 21: 3D Simulation of home environment.

Table 10: Results of the FL, LSTM_FT, and LSTM_FTR models in Figure 21.

Figure 21	Success rate	Average decision time (s)	% of time saving than FL	Max turning angle (rad)	Avg turning angle (rad)
FL	64%	0.1396		2.2825	0.2616
LSTM_FT	68%	0.0256	81.6619	1.0297	0.1771
LSTM_FTR	93%	0.0253	81.8768	0.8500	0.1202

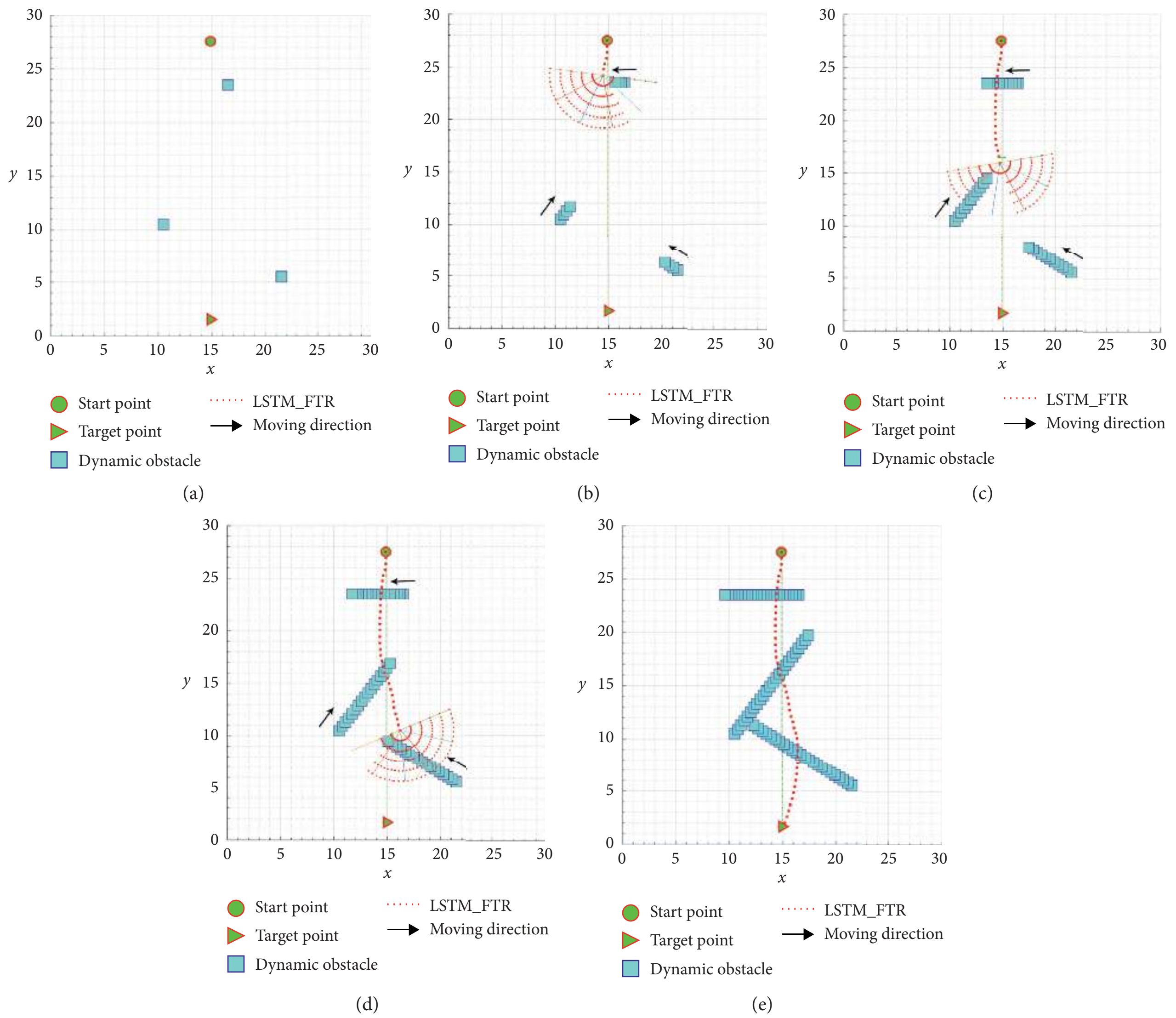


Figure 22: LSTM_FTR in a dynamic environment.

and the decision-making time saves the process of calculating the distance from the obstacle, so it is far less than the time in the two-dimensional simulation environment. In the aspect of path smoothness, the turning angle of the robot is used as the evaluation value, and the average turning angle and the maximum turning angle during the experiment are counted. It can be seen from the results in Table 10 that the minimum turning angle of LSTM_FTR indicates that the planned path is smooth and there is no sharp turning.

6.2. Simulation Tests in Dynamic Environments. The fusion algorithm LSTM_FTR proposed in this paper can be applied to the unknown and uncertain environment, not only in the static unknown environment but also in the environment with dynamic obstacles. The simulation test of robot in dynamic environment is shown in Figure 22. The blue rectangles represent the dynamic obstacles and their running trajectories, the black arrow represents the moving direction of the obstacle, and the red sector represents the detection range of the sensor mounted on the robot. There are three dynamic obstacles in the environment. The obstacles move along a straight line with different speeds, but they are less than or equal to the average speed of the robot. It can be seen from Figure 22 that the robot can avoid dynamic obstacles in different directions and reach the target point successfully. If the speed of the dynamic obstacle is higher than that of the robot, the robot may not be able to avoid collision, so the corresponding control strategy should be added. This will be carried out in the future research work.

7. Conclusion

In this paper, a local path planning fusion method for mobile robots based on LSTM neural network, fuzzy logic control, and reinforcement learning is proposed. The method combines the advantages of the three algorithms and finally obtains a network model LSTM_FTR that can be used in complex environments. The trained model is tested and compared to FL and LSTM_FT algorithms. Experimental results show that the LSTM_FTR network model has the following advantages:

(1) Compared to the traditional fuzzy control algorithm, the path planning efficiency of the fusion model is improved by 81.8768%, which reduces the calculation cost and is more suitable for real-time decision-making system.

(2) Through the independent exploration of reinforcement learning, new rules are acquired, which can avoid obstacles in time and plan a shorter path in complex environments.

(3) The proposed model is adaptable, can self-learn, and can quickly plan a better path based on real-time sensor information in unknown and complex environments. The success rate of path planning is improved.

This paper studies the local path planning of mobile robots based on fusion of three algorithms, and has achieved some research results, but there also exist the following problems and shortcomings:

(1) The research is still in the simulation stage, without considering various problems in practical application. If it is applied in real environment, the parameter setting needs to be modified and debugged, and add prejudgment for the movement direction and position of dynamic obstacles, so as to make accurate judgment according to the real environments.

(2) The environment input only depends on the lidar ranging sensor, which cannot accurately understand the real environment information. Future research will obtain more abundant environmental information as state input, and improve and perfect the reward and punishment function of reinforcement learning. Furthermore, it will join the research of computer vision, through the analysis of different types of obstacles to make different processing decisions, to train a more adaptive model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant nos. 61473179, 61973184, and 61573213, and SDUT and Zibo City Integration Development of China under grant no. 2018ZBXC295.

References

[1] H. Zhang, W. Lin, and A. Chen, “Path planning for the mobile robot: a review,” *Symmetry*, vol. 10, no. 10, Article ID 450, 2018.

[2] B. K. Patle, G. Babu L, A. Parhi, and A. Jagadeesh, “A review: on path planning strategies for navigation of mobile robot,” *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[3] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, “Heuristic approaches in robot path planning: a survey,” *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[4] M. N. A. Wahab, S. Nefti-Meziani, and A. Atyabi, “A comparative review on mobile robot path planning: classical or meta-heuristic methods?” *Annual Reviews In Control*, vol. 50, pp. 233–252, 2020.

[5] J. Han and Y. Seo, “Mobile robot path planning with surrounding point set and path improvement,” *Applied Soft Computing*, vol. 57, pp. 35–47, 2017.

[6] C. Lamini, S. Benhlima, and A. Elbekri, “Genetic algorithm based approach for autonomous mobile robot path planning,” *Procedia Computer Science*, vol. 127, pp. 180–189, 2018.

[7] C. Zhang, L. Zhou, Y. Li et al., "A dynamic path planning method for social robots in the home environment," *Electronics*, vol. 9, no. 7, Article ID 1173, 2020.

[8] B. B. K. Ayawli, X. Mei, M. Shen, A. Y. Appiah, and F. Kyeremeh, "Mobile robot path planning in dynamic environment using Voronoi diagram and computation geometry technique," *IEEE Access*, vol. 7, pp. 86026–86040, 2019.

[9] M. A. Ali and M. Mailah, "A simulation and experimental study on wheeled mobile robot path control in road roundabout environment," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, 2019.

[10] H. Shin and J. Chae, "A performance review of collision-free path planning algorithms," *Electronics*, vol. 9, no. 2, Article ID 316, 2020.

[11] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: electrostatic potential field approach," *Expert Systems with Applications*, vol. 100, pp. 68–78, 2018.

[12] D. Wang, S. Chen, Y. Zhang, and L. Liu, "Path planning of mobile robot in dynamic environment: fuzzy artificial potential field and extensible neural network," *Artificial Life and Robotics*, vol. 26, pp. 1–11, 2021.

[13] S. B. C. Lamini, Y. Fathi, and S. Behlima, "A fuzzy path planning system based on a collaborative reinforcement learning," *International Review Of Automatic Control*, vol. 102 pages, 2017.

[14] M. N. Zafar and J. C. Mohanta, "Methodology for path planning and optimization of mobile robots: a review," *Procedia Computer Science*, vol. 133, pp. 141–152, 2018.

[15] L. Chang, L. Shan, C. Jiang et al., "Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment," *Autonomous Robots*, vol. 45, pp. 1–26, 2021.

[16] S. M. H. Rostami, A. K. Sangaiah, J. Wang et al., "Obstacle avoidance of mobile robots using modified artificial potential field algorithm," *EURASIP Journal on Wireless Communications and Networking*, vol. 1, pp. 1–19, 2019.

[17] A. Aouf, L. Boussaid, and A. Sakly, "Same fuzzy logic controller for two-wheeled mobile robot navigation in strange environments," *Journal of Robotics*, vol. 2019, Article ID 2465219, 11 pages, 2019.

[18] A. M. Alshorman, O. Alshorman, M. Irfan et al., "Fuzzy-based fault-tolerant control for omnidirectional mobile robot," *Machines*, vol. 8, no. 3, p. 55, 2020.

[19] F. Nicola, Y. Fujimoto, and R. Oboe, "A LSTM neural network applied to mobile robots path planning," in *Proceedings of the 2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pp. 349–354, IEEE, Porto, Portugal, February 2018.

[20] X. Song, Y. Liu, L. Xue et al., "Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model," *Journal of Petroleum Science and Engineering*, vol. 186, Article ID 106682, 2019.

[21] P. Mirowski, R. Pascanu, F. Viola et al., "Learning to navigate in complex environments," 2016, https://arxiv.org/abs/1611.03673.

[22] M. Hausknecht and P. Stone, "Deep recurrent Q-LEARNING for partially observable MDPs," 2015, https://arxiv.org/abs/1507.06527.

[23] Q. Luo, H. Wang, Y. Zheng, and J. He, "Research on path planning of mobile robot based on improved ant colony algorithm," *Neural Computing and Applications*, vol. 32, no. 6, pp. 1555–1566, 2020.

[24] V. Jamshidi, V. Nekoukar, and M. H. Refan, "Analysis of parallel genetic algorithm and parallel particle swarm optimization algorithm UAV path planning on controller area network," *Journal Of Control, Automation and Electrical Systems*, vol. 31, no. 1, pp. 129–140, 2020.

[25] B. K. Patle, A. Pandey, A. Jagadeesh, and D. R. Parhi, "Path planning in uncertain environment by using firefly algorithm," *Defence Technology*, vol. 14, no. 6, pp. 691–701, 2018.

[26] 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, 2019.

[27] B. K. Patle, D. R. K. Parhi, A. Jagadeesh, and S. K. Kashyap, "Application of probability to enhance the performance of fuzzy based mobile robot navigation," *Applied Soft Computing*, vol. 75, pp. 265–283, 2019.

[28] M. S. Gharajeh and H. B. Jond, "Hybrid global positioning system-adaptive neuro-fuzzy inference system based autonomous mobile robot navigation," *Robotics and Autonomous Systems*, vol. 134, Article ID 103669, 2020.

[29] X. Liu, D. Zhang, J. Zhang et al., "A path planning method based on the particle swarm optimization trained fuzzy neural network algorithm," *Cluster Computing*, vol. 2021, pp. 1–15, 2021.

[30] Z. Jiang, Y. Xia, and G. Shen, "A novel learning-based global path planning algorithm for planetary rovers," *Neurocomputing*, vol. 361, pp. 69–76, 2019.

[31] I. Sung, B. Choi, and P. Nielsen, "On the training of a neural network for online path planning with offline path planning algorithms," *International Journal of Information Management*, vol. 57, Article ID 102142, 2021.

[32] C. Chen, X. Q. Chen, F. Ma et al., "A knowledge-free path planning approach for smart ships based on reinforcement learning," *Ocean Engineering*, vol. 189, Article ID 106299, 2019.

[33] M. Fakoor, A. Kosari, and M. Jafarzadeh, "Humanoid robot path planning with fuzzy Markov decision processes," *Journal of Applied Research and Technology*, vol. 14, no. 5, pp. 300–310, 2016.

[34] Z. Lin, Y. Zhang, and Y. Li, "Path planning for indoor mobile robot based on deep learning," *Optik*, vol. 219, Article ID 165096, 2020.

[35] J. Yu, Y. Su, and Y. Liao, "The path planning of mobile robot by neural networks and hierarchical reinforcement learning," *Frontiers in Neurorobotics*, vol. 14, p. 63, 2020.

[36] D. Wang, Y. Hu, and T. Ma, "Mobile robot navigation with the combination of supervised learning in cerebellum and reward-based learning in basal ganglia," *Cognitive Systems Research*, vol. 59, pp. 1–14, 2020.

[37] S. S. Das, D. R. Parhi, and S. Mohanty, "Insight of a six layered neural network along with other AI techniques for path planning strategy of a robot," in *Proceedings of the Emerging Trends in Engineering, Science and Manufacturing (ETESM-2018)*, Dhenkanal, India, March 2018.

[38] R. J. C. T. Ai and E. P. Dadios, "Neuro-fuzzy mobile robot navigation," in *Proceedings of the 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, IEEE, Baguio City, Philippines, December 2018.

[39] F. Martínez, E. Jacinto, and H. Montiel, "Neuronal environmental pattern recognizer: optical-by-Distance LSTM model for recognition of navigation patterns in unknown environments," *International Conference on Data Mining and Big Data*, Springer, Berlin, Germany, pp. 220–227, 2019.

[40] C. Yu, X. Qi, H. Ma, X. He, C. Wang, and Y. Zhao, "LLR: learning learning rates by LSTM for training neural networks," *Neurocomputing*, vol. 394, pp. 41–50, 2020.
[41] E. Chalmers, E. B. Contreras, B. Robertson, A. Luczak, and A. Gruber, "Learning to predict consequences as a method of knowledge transfer in reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2259–2270, 2018.
[42] N. Guo, C. H. Li, D. Wang, N. Zhang, and G. M. Liu, "Path planning of mobile robot based on prediction and fuzzy control," *Computer Engineering and Applications*, vol. 56, no. 8, pp. 104–109, 2020.
[43] L. X. Wei, S. K. Wu, H. Sun, and J. Zheng, "Mobile robot path planning based on multi-behaviours," *Control and Decision*, vol. 34, no. 12, pp. 2721–2726, 2019.
[44] I. Iancu, "A mamdani type fuzzy logic controller," *Fuzzy Logic: Controls, Concepts, Theories and Applications*, pp. 325–350, IntechOpen, London, UK, 2012.
[45] G. Taguchi, R. Jugulum, and S. Taguchi, *Computer-Based Robust Engineering: Essentials for DFSS*, ASQ Quality Press, Milwaukee, WI, USA, 2004.