IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received January 6, 2019, accepted January 18, 2019, date of publication January 24, 2019, date of current version February 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894626

A Novel GRU-RNN Network Model for Dynamic Path Planning of Mobile Robot

JIANYA YUAN[1], HONGJIAN WANG[1], CHANGJIAN LIN[1], DAWEI LIU[2], AND DAN YU[1]
[1]College of Automation, Harbin Engineering University, Harbin 150001, China
[2]Qingdao Topscomm Communication Co., Ltd., Qingdao 266000, China

Corresponding author: Hongjian Wang (cctime99@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61633008 and Grant 51609046, and in part by the Natural Science Foundation of Heilongjiang Province under Grant F2015035.

ABSTRACT A dynamic path planning method based on a gated recurrent unit-recurrent neural network model is proposed for the problem of path planning of a mobile robot in an unknown space. A deep neural network with sensor input is used to generate a new control strategy output to the physical model to control the movement of the robot and thus achieve collision avoidance behavior. Inputs and tags are derived from sample sets generated by an improved artificial potential field and an improved ant colony optimization algorithm. In order to make the ant colony algorithm converge quickly, the pheromone trail and the state transition probability are improved. The field function of the artificial potential field method is modified. Using the end-to-end network model to learn the mapping between input and output in the sample data, the direction and speed of the mobile robot are obtained. The simulation experiments and realistic simulations show that the network model can plan a reasonable path in an unknown environment. Compared with other traditional path planning algorithms, the proposed method is more robust than the traditional path planning algorithms to differences in the robot structure.

INDEX TERMS Mobile robot, gated recurrent unit-recurrent neural network, dynamic path planning, ant colony optimization, artificial potential field.

I. INTRODUCTION

Obstacle avoidance is one of the most basic problems faced by mobile robots. Over the years, various methods for autonomous mobile robot path planning have been studied, with focusing on the development of collision-free path planning algorithms. The problem can be described as follows: the mobile robot is given a starting point and an ending point in a known or unknown environment, and environmental information is detected by the robot's sensor. Eventually, autonomously avoiding obstacles, the robot finds a movement trajectory from the starting point to the target point, in what constitutes a type of real-time planning. In addition to the security and smoothness of collision avoidance, the real-time performance of the system is also an important aspect of real-time collision avoidance planning. In practical path planning, the environment (specifically the obstacles that might be encountered) in which the mobile robot operates is unknown or partially known. This requires the collision avoidance program to use collected data so that the mobile robot is able to make a series of decisions to avoid obstacles in real time. For this kind of problem, many methods and algorithms are commonly used, such as visibility graphs, artificial potential field (APF) and related algorithms, intelligent optimization algorithms, and fuzzy logic. APF and its variants are often used in mobile robot navigation [1]–[3], [11]. With the traditional APF method, it is easy for the mobile robot to become locally locked, for it to be subject to narrow path vibration, and for neighboring obstacles to be too close to allow planning of the path. In order to overcome the above difficulties, some researchers have made different improvements to the traditional APF method. Weerakoon *et al.* [4] solves the deadlock problem by replacing the traditional function with an exponential function. The APF method is also combined with other intelligent algorithms to improve the parameters of the intelligent algorithm [5]. The traditional ACO method does not deal well with the balance between premature problems and slow convergence speed. Chen *et al.* [6] proposes a combination of ''scent pervasion'' policy and ''one minus search'' strategy to pre-process grid maps to speed up algorithm convergence and quickly complete robot path planning. Cao *et al.* [7] adopt a novel multiagent pheromone-based traffic management framework to reduce traffic congestion. In order to speed up the algorithm, Cekmez *et al.* [8] implement parallel ACO algorithm on CUDA platform to solve the problem of UAV path planning.

On the other hand, for swarm intelligence techniques such as genetic algorithms (GA), the real-time performance is poor, and requires a large data storage space and long computing times. Other authors have proposed hybrid solutions combining the two approaches. Das *et al.* [9] improved the classical gravitational search algorithm (GSA) based on the communication and memory characteristics of particle swarm optimization (PSO). Chaari *et al.* [10] proposed a new efficient hybrid ACO-GA method, using the ACO method to find the suboptimal solution, and then using the GA to search for the optimal solution in the suboptimal solution, which is used to solve the global robot path planning in static environment. These method has been used successfully for navigation of multiple mobile robots. However, in these study, the environment and obstacles were taken to be static. Bodhale *et al.* [11] successfully implemented dynamic path planning by combining the potential field method with a Monte Carlo positioning method. It is difficult to determine the force coefficients influencing the velocity and direction of a mobile in a complex environment. Li *et al.* [12] have designed fuzzy controllers for path planning problems in dynamic environments based on the angles and collision times between dynamic obstacles and the direction of motion of the robot, this method is suitable only for simple and small obstacles. Matveev *et al.* [13] proposed an integrated guidance control strategy belonging to the class of sliding mode control algorithms for autonomous vehicles. This method required the establishment an accurate environmental model, which limits the use of this algorithm in complex environments. The accuracy of the environmental model also has an impact on the algorithm's time-consumption. Rapidly-exploring random tree (RRT) and other methods rarely pay attention to the information on obstacle movement [14], [15]. In the case of difficult motion scenarios such as dense obstacles or frequent movement of obstacles, a long time is required to find the optimal solution.

The need for mathematical modeling of the environment, the limited real-time performance of the algorithms, the occurrence local locking, and other issues arising with the above methods, have all encouraged the search for new approaches. Most previous studies have examined collision avoidance planning strategies within the reinforcement learning (RL) framework [16]–[18]. Deep learning algorithms don't require the construction of an accurate model- and after a large amount of training, the network can map an input to obtain the corresponding output. For a trained network, multiple iterations of the algorithm are necessary. In the application of deep learning to obstacle avoidance problems, the construction of an end-to-end model, allows a network to learn the mapping between input data and output strategies. Carrio *et al.* [18] used a combination of convolutional neural networks (CNN), gated recurrent unit (GRU) networks, and variant Q-learning to solve the problem of unmanned autonomous vehicle (UAV) control when only visual images were input. Inoue *et al.* [19] proposed a novel method combining the rapidly-exploring tree and a long short-term memory (LSTM) network, which overcomes the difficulties involved in the acquisition of a large amount of training data.

The main subject of the present paper is the development of a collision avoidance algorithm for a mobile robot. A teacher system based on an improved ACO algorithm and an improved APF method is established. The pheromone trail and state transition rules of the ACO algorithm are improved to accelerate the convergence. The potential field is built around the robot, and the influence of target point gravitation is removed to avoid local locking. A dynamic planning model based on a Gated Recurrent Unit-Recurrent Neural Network (GRU-RNN) is then designed, with a teacher system based on the above algorithm.

The main contributions of this paper can be summarized as follows:

- We propose a novel pheromone update and state transition rule to speed up the convergence of the algorithm, and we introduce a new potential field to solve the shortcomings of the traditional APF.
- We design a GRU network model to learn the path planning strategy produced by the improved ACO and APF, and we verify the algorithm in both a simulated environment and an outdoor environment.

II. CONSTRUCTION OF THE ENVIROMENT MODEL

In order to achieve accurate collision avoidance behavior, the kinematics equation of the robot is established. Obtaining accurate environmental information is extremely important for robot control. In order to make the input state information more accurate, reasonable coordinate systems must be established. According to the conversion between the coordinate systems, the information of the robot and surrounding obstacles in the global coordinate system can be obtained. The environmental model uses the grid method. Choosing the reasonable grid size can reduce optimization time and improve the quality of the solution.

A. COORDINATE MODEL

To ensure sufficient accuracy of the state information, an appropriate coordinate system must be established, comprising a global coordinate system and two local coordinate systems (Fig. 1): the global coordinate system $X_GO_GY_G$, the robot-centered local coordinate system $X_RO_RY_R$, and the local coordinate system centered on the laser $X_LO_LY_L$. The global coordinates of an obstacle are obtained from the following coordinate transformation:

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} \rho\cos|\theta-\pi/2|+x_e & 0 \\ 0 & \rho\cos|\theta-\pi/2|+x_e \end{bmatrix} \cdot \begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

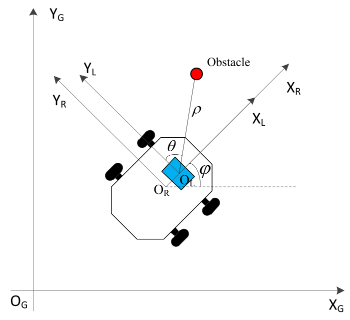


FIGURE 1. Mobile robot coordinate system.

here (x_g, y_g) are the coordinates of the obstacle in the system $X_G O_G Y_G$; (x, y) are the coordinates of the robot in the system $X_L O_L Y_L$; x_e is the distance between the coordinate origins O_L and O_R; ρ and θ are respectively the polar distances and polar angle of the obstacle measured by the laser sensor. φ is the angle between the current direction of motion of the robot and the X_G axis. In the outdoor simulation environment, counter-clock-wise is positive, based on true north.

B. ROBOT MODEL

We make the following assumptions about robot and the environment:

- The working environment of the robot is $X \times Y$.
- The wheels do not slide.

At time t, we denote by (x_t, y_t) the position of the mobile robot in the coordinate system $X_G O_G Y_G$, and by φ the angle between the direction of movement and the global coordinate system X_G axis. After a certain sampling time T, the robot reaches a new position. Assuming that the robot moves with constant speed v within this sampling time, we find that the new position coordinate (x, y) at the next instant is given by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = vT \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\frac{\pi}{2} - \varphi) \\ \sin(\frac{\pi}{2} - \varphi) \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \end{bmatrix} \quad (2)$$

C. ENVIROMENT MODEL

The environment model is constructed as a grid model, with the robot's initial position (x_0, y_0) as its origin point.

We make the following assumptions:

- The robot works in a two-dimensional environment and the number of grids in space is $m \times n$.
- In order to ensure the safety of the robot, the boundary of the obstacle is expanded, by half the length of the robot.
- A black grid indicates that the area is not accessible.

Obstacles in the mobile robot's motion space can be divided into two types: known and unknown. For obstacles in an unknown environment, because there is no positional information about obstacles in advance, they can be detected only by sensors carried by the robot itself, and new effective track points (including deterministic points and uncertain point), must be added to the program as the motion progresses.

If an obstacle is small, then it is easy to determine the visibility of vertices and other points. Such vertices are termed deterministic vertices, examples of which are shown in Fig. 2 by points B and C. If an obstacle is particularly large, the sensor can detect only part of it, as shown in Fig. 3, where points F and G are termed uncertain vertices.

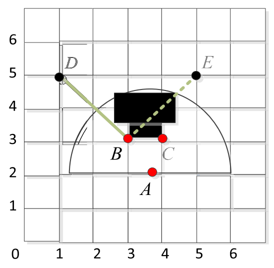


FIGURE 2. Deterministic vertices.

Vertices have the following characteristics:

- A vertex is exactly at the detection edge of the sensor.
- There is an obstacle on one side of the vertex and not on the other side.
- This vertex is visible from any other vertex.

In Fig. 2, there are no obstacles between points B and D, so point D is visible from point B. An obstacle lies between points B and E, so these two points are invisible to each other. In Fig. 3, the two points are uncertain vertices, and it is impossible to judge the specific distribution of obstacles around these vertices. Since the set of visible points $allow_F$ of the current point F cannot be determined, the path cannot be planned when the improved ACO algorithm is applied. If the point F is regarded as a visible point, when moving to the point F, the robot finds a new uncertain vertex, and the set of visible points $allow_k$ of the point K still does not satisfy the condition. So when we have no determinate vertices, we apply an improved APF to avoid collisions until the robot is completely at the determined vertices.

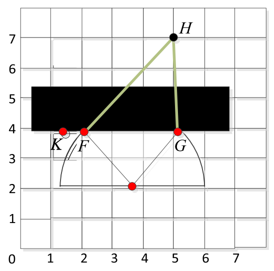


FIGURE 3. Uncertain vertices.

III. AUTONOMOUS COLLISION AVOIDANCE ALGORITHM FOR TEACHER SYSTEM

When the sensor detects new obstacles, the robot is required to respond quickly. This requires that the collision avoidance algorithm has a rapid planning ability. In this paper, an improved ACO and an improved APF method are combined to give the mobile robot the capacity for autonomous collision avoidance. When an obstacle is far from the robot, the ACO algorithm is used as the collision avoidance algorithm. This algorithm uses known environmental information and local information detected by the sensor to perform operations, and provides a long-term, optimized motion path. When the distance between the obstacle and the robot is less than 2/m, an improved APF method is used for emergency collision avoidance. The APF algorithm ignores the influence of the global environment and responds directly to the risk of collision by using the sensor information in a short period of time. To accelerate the convergence of the ACO algorithm, the pheromone trail and state transition rules are improved. An APF is established around the robot, and the gravitational effect of the target point is removed to prevent the occurrence of local extreme points. The pseudo code of the autonomous collision avoidance algorithm of the teacher system can be found in the appendix.

A. PATH PLANNING BASED ON IMPROVED ANT COLONY ALGORITHM

1) PHEROMONE UPDATES

There are given by:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{3}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \textit{if } k \text{ pass } e_{ij} \\ 0 & \textit{else} \end{cases} \tag{4}$$

where, i, j are status point number. τ is the pheromone concentration trails; m is the total number of ants, ρ is the pheromone evaporation rate $(0 < \rho < 1)$; The path length of ant k is L_k ,and Q is a constant. The parameter ρ is introduced to avoid an infinite accumulation of pheromones and make the program ignore previous bad decisions.

To speed up convergence, pheromone enhancement is performed for the path taken by the first quarter of ants in each generation of ants:

$$\tau_{ij}(t+1) = \lambda \times \tau_{ij}(t+1) \tag{5}$$

$$\lambda = 1 + 0.5 \times \frac{D}{L_k} \tag{6}$$

where, D is the Euclidean distance from the start to the end.

To ensure that the algorithm retains its ability to explore in the later stages of the search, after completion the pheromone update and reinforcement, processing of the pheromone trail limit is performed:

$$\tau_{ij}(t+1) = \begin{cases} \tau_{\min} & \text{if } \tau_{ij}(t+1) < \tau_{\min} \\ \tau_{\max} & \text{if } \tau_{ij}(t+1) > \tau_{\max} \\ \tau_{ij}(t+1) & \text{otherwise} \end{cases} \tag{7}$$

where, $\tau_{\min}$, $\tau_{\max}$ are the artificially set upper and lower limits of pheromone trail.

2) STATE TRASITION RULES

The state transition rule is the selection rule for the next state when the ant moves to that state from the current state. In path planning is the selection of the visible points of the current location, because normally, multiple points are visible from any one location, and the next step is to select which of these visible points leads to progress of the planning process toward the optimal solution. This paper implements probability transfer in the form of probability selection. The state transition probability is formulated as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_j)^{\beta}}{\sum\limits_{r \in allow_i} (\tau_{ir})^{\alpha}(\eta_r)^{\beta}}, & \text{if } r \in Allow_i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$p_{ij}^{k} = \gamma p_{ij}^{k} + (1-\gamma)\frac{L_{ij}}{N-1} \tag{9}$$

according to this formula, if ant k currently at point i and probability of its next transfer to visible point j is p_{ij}^{k}. $allow_i$ is the set of visible points to which ant k can perform state transition. η is a heuristic function, taken as the reciprocal of the distance from the current point to the target point. α is the importance of the pheromone. β is the importance of the heuristic function. N is the number of visible points in the range measured by the current point i. L_{ij} is the distance between points i and j. γ is a weight. (in this experiment, γ takes the value 0.95).

B. IMPROVEMENT OF ARTIFICIAL POTENTIAL FIELD

In the traditional APF method, only distance information about an obstacle is considered, and directional information is ignored. When the obstacle is in the emergency collision avoidance zone of the robot, the APF is used only for emergency collision avoidance, and it is not necessary to consider the influence of the target point. Therefore, in the improved APF, the gravitational effect of the target point is removed to prevent the creation of local extreme points. The improved potential field takes the form:

$$V(x_f, x_r, x_p, y_{xp}) \tag{10}$$

with its shape being determined by the parameters x_f, x_r, x_p, y_{xp}. In the simulation, the detection range of the sensor is taken as 12/m, $x_f = 10.5$/m, $x_r = 4$/m, $y_{xp} = 6.8$/m, $x_p = 4$/m. The parameter definition is shown in Fig. 4.

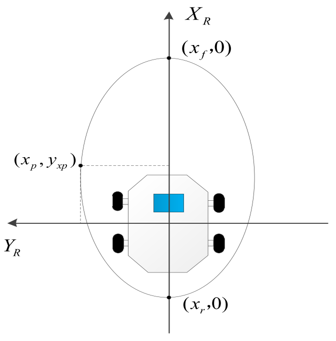


FIGURE 4. Parameter definition of improved APF.

In this method, the smaller the potential, the closer is the obstacle to the robot. The value of potential v_p is given by

$$v_p = (y^2 + (ay + c)x^2)/(py + q) \tag{11}$$

where, $p = x_f - x_r$, $q = x_f \cdot x_r$, $a = (-2y_{xp} + p)/x_f^2$, $c = (q + y_{xp}^2)/x_f^2$, and $py + q \neq 0$. The potential value v_p is positively related to $|y|$.

When there is an obstacle in the emergency collision avoidance area of the robot, the linear velocity and angular velocity of the robot are given by:

$$v_l = (v_{pre} + v_{\max}) \cdot e^{-(\alpha \cdot \lambda)^2} - v_{\max} \tag{12}$$
$$\omega_r = (\omega_{pre} + \omega_{\max}) \cdot e^{(\beta \cdot \lambda)^2} - \omega_{\max} \tag{13}$$
$$\lambda = \max(1 - v_p, 0) \tag{14}$$

where, v_l, ω_r represent the linear velocity and angular velocity obtained after planning, respectively; Current cornering speed $\omega_{\max}$ and maximum cornering speed ω_{pre}. α, β are intensity factors, with values of 1 and 2, respectively. If v_p approaches 0, this means that the robot is about to touch the obstacle. At such a time, v_l takes its minimum value and ω_r takes its maximum value.

IV. DESIGN OF GRU-RNN NETWORK MODEL

GRU, as a variant of LSTM, combines a forgotten gate and input gate into a single update gate. In addition, there are some other changes to the mixed cell state and the hidden state. The final model is simpler than the standard LSTM model and is a very popular variant. Given a set of observations, the learned model can provide the corresponding control output vector. The learning process terminates when the differences between the actual and model output converges to a very small value. Once the GRU-RNN network has been trained, it can be used as a path planning model for mobile robots, enabling them to move in an autonomous manner while avoiding collision.

A. GRU-RNN DYNAMIC PATH PLANNING MODEL

The GRU-RNN model designed in this paper is shown in Fig. 5. The input layer has a total of 61 dimensions, of which the first 60 comprise the distance information detected by the laser rangefinder. The final one-dimensional piece of data is φ. There are two hidden layers in the model: hidden layer 1 consists of 40 GRU modules and hidden layer 2 consists of 30 neurons. These hidden layers are fully connected. There are two neurons in the output layer, the velocity and angle of the mobile robot.

The most important structure in the GRU-RNN network model is the GRU module unit, which receives the data in chronological order: input data from time $t-9$ to t, and the output at time t. The output at the final instant constitutes the true output of the module. The structure of the module is shown in Fig. 6, where x^t is the input at the current instant, y^t is the output layer, h^t is the output of the module at the current instant, and h^{t-1} is the output of the module at the final instant.

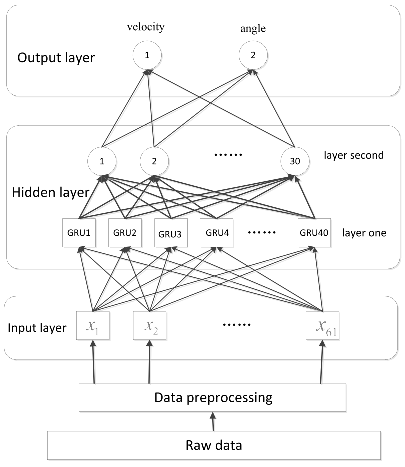


FIGURE 5. GRU-RNN model for dynamic path planning.

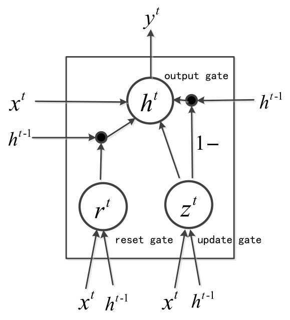


FIGURE 6. GRU unit structure.

the update gate is given by

$$z^t = \sigma\left(W_{hz}h^{t-1} + W_{xz}x^t + b_z\right) \tag{15}$$

the reset gate by

$$r^t = \sigma\left(W_{hr}h^{t-1} + W_{xr}x^t + b_r\right) \tag{16}$$

and the output layer by

$$h^t = z^t \cdot h^{t-1} + (1 - z^t) \cdot \tilde{h}^t \tag{17}$$

and the output layer by

$$y^t = \sigma(net_y^t) \tag{18}$$

$$E = \sum_{t=1}^{T} E_t = \sum_{t=1}^{T} \frac{1}{2}(y^{dt} - y^t)^2 \tag{19}$$

here $\tilde{h}^t = \tanh(r^t W_{hh} h^{t-1} + W_{xh}x^t + b_h)$; z^t, r^t and h^t are the output of the update gate, reset gate and memory module, respectively, at time t; x^t is the input vector of the memory module at time t; h^{t-1} is the output vector of the memory module at time $t-1$; W_{xz}, W_{xr}, W_{xh} are the weight matrices between the module input and the update gate, reset gate, and $\tilde{h}^t$ respectively; W_{hz}, W_{hr} are the weight matrices between the output of the memory module and the update gates and reset gates, respectively, at time $t-1$; b_z, b_r, b_h are the biases of the update gate, reset gate, and $\tilde{h}^t$, respectively.

The gradient of the parameter matrix is calculated using the chain rule for derivatives, and the weight updates are given by the following expressions: the weight update between the output of the module and the output layer is

$$\frac{\partial E}{\partial W^y} = h^t \frac{\partial E}{\partial y^t} \frac{\partial y^t}{\partial net_y^t} \tag{20}$$

that between the input and the update gate z is

$$\frac{\partial E}{\partial W^z} = (x^t)^T \frac{\partial E}{\partial h^t} \frac{\partial h^t}{\partial z^t} \frac{\partial z^t}{\partial net_z^t} \tag{21}$$

that from the module output to the update gate z at the previous instant is

$$\frac{\partial E}{\partial U^z} = (h^{t-1})^T \frac{\partial E}{\partial h^t} \frac{\partial h^t}{\partial z^t} \frac{\partial z^t}{\partial net_z^t} \tag{22}$$

that from the input to $\tilde{h}_t$:

$$\frac{\partial E}{\partial W} = (x^t)^T \cdot \delta_h^t \cdot z^t \cdot g'(net_{\tilde{h}}^t) \tag{23}$$

that from the module output to the update gate z at the previous instant is

$$\frac{\partial E}{\partial U} = (r^t \cdot h^{t-1})^T \cdot \delta_h^t \cdot z^t \cdot g'(net_{\tilde{h}}^t) \tag{24}$$

that from the input to the reset gate r is

$$\frac{\partial E}{\partial W^r} = (x^t)^T \cdot h^{t-1} \cdot [(\delta_h^t \cdot z^t \cdot g'(net_{\tilde{h}}^t))U^T] \cdot f'(net_r^t) \tag{25}$$

and that from the module output to the update gate r at the previous instant is:

$$\frac{\partial E}{\partial U^r} = (h^{t-1})^T \cdot h^{t-1} \cdot [(\delta_h^t \cdot z^t \cdot g'(net_{\tilde{h}}^t))U^T] \cdot f'(net_r^t) \tag{26}$$

where, $\delta_h^t = \delta_y^t W^{yT} + \delta_z^{t+1} U^{zT} + \delta^{t+1} U^T r^{t+1} + \delta_r^{t+1} U^{rT} + \delta_h^{t+1}(1 - z^{t+1})$, $net_z^t = x^t W^z + h^{t-1} U^z$, $net_r^t = x^t W^r + h^{t-1} U^r$, $net_y^t = h^t W^y$. For the GRU-RNN network defined by these expressions, the weights are denoted by W_{hc}, W_{xz}, W_{hr}, W_{xr}, W_{hh}, W_{xh}, W^z, W^r, W^y etc.

B. COLLECTING SAMPLES AND PREPROCESSING

Before the start of training, the first thing to do is to collect and preprocess the sample data. In the simulation training phase, the mobile robot performs dynamic collision avoidance and path planning under the improved ACO algorithm and the APF algorithm, and collects sample data. Each set of sample data has a total of 184 dimensions. The first 181-dimensions of data are obtained by the laser sensor collecting surrounding environmental information, φ is the 182th dimension of data, and the final two dimensions of data comprise the robot's angle and velocity. The first 182 dimensions of data are used as the input value of the network, and the final two dimensions of data are used as the training label. The network input laser data are measured in real time. The output labels are taken as the angle and speed of the robot obtained from the improved ACO algorithm and APF. In order to speed up the training of the model and enhance the robustness of the model (small fluctuations in the input cause dramatic fluctuations in the output), the data must be processed before entering the input layer. First, regularize the input data and merge the first 181 data dimensions. The 182th dimension data remains unchanged, and after processing, there are 61 dimensions of input data in total.

For an accurate training result of the network model, the input data are normalized to give a pure value.

$$c_{ij}^* = \frac{c_{ij} - \min(c_j)}{\max(c_j) - \min(c_j)} \tag{27}$$

The training data here are stored in the form of row vectors: c_{ij} denotes the data before normalized while $\max(c_j)$ and $\min(c_j)$ are the maximum and minimum values of column j.

V. SIMULATION RESULTS AND ANALYSIS

In this section, the network model is trained by samples generated by the teacher system, which is implemented using the improved ACO algorithm and APF. In the test experiment, the planning capabilities of the two methods are compared, and the ability of the GRU network to avoid obstacles is verified.

The improved ACO algorithm and APF are applied, and the mobile robot performs collision avoidance training in the simulation training field. The generated training set contains 60312 sets of data, which are used for network model training. The test set contains 1000 sets of data for testing the trained network model. Using the data in the training set, the network is trained 8 million times, with each training procedure being performed 2,000 times. A certain amount of data is selected from the test set, and the error is tested using the final network weight. To eliminate correlation from the sample, the position and shape of the obstacles, their density are randomly generated. In our example, the training set is used as GRU-RNN network input to learn weights in a given observation sequence that will reproduce the corresponding control output φ.

The input of the GRU-RNN network at time step t is a 61-dimensional vector consisting of a 60-dimensional input distance vector $x_t = \left[x_t^0, x_t^1, \cdots, x_t^{60}\right]$ and φ_t. In order to improve the generalize ability of the model, Gaussian noise $N(\mu, \sigma^2)$ is added to x_t and φ_t, with $\mu = 0.2, \sigma = 0.3$ in this experiment.

A total of 50 datasets of obstacle size and position are randomly selected within a certain range. At each time, the initial heading angle is randomly chosen from a uniform distribution $\{-\frac{\pi}{2}, \frac{\pi}{2}\}$. Of these datasets, 45 are used for training the model, and the remaining five datasets are used for testing. Stochastic Gradient Descent (SGD) is used for parameter optimization. To examine the robustness of the model, it is necessary to test the final model on a different workspace that is unlike the training phase. The results show that the

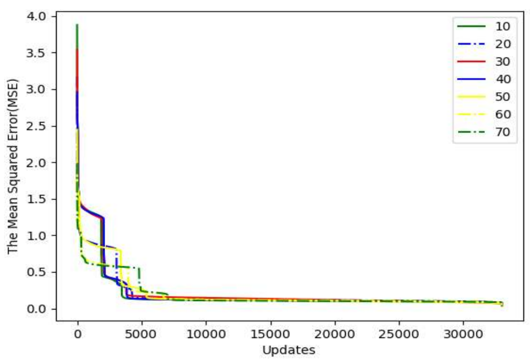


FIGURE 7. GRUs loss function convergence curve.

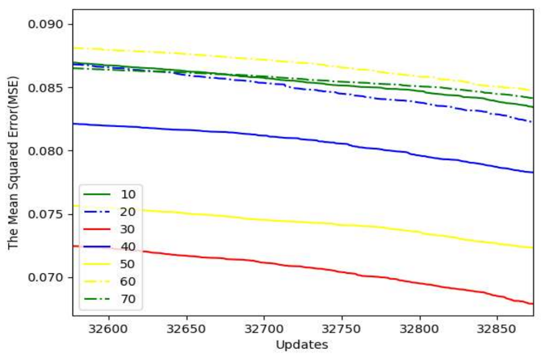


FIGURE 8. GRUs loss function curve between updates 32600-32850.

GRU-RNN model gives good output values even in test environments. Hence, the learned model can provide very good path planning capabilities in an unknown environment. For a particular test case, a comparison it with the path of the teacher system, can be seen in Fig. 11 and Fig. 14 below.

In the verification environment, the learning model can be used to predict the control output angle at the next instant. The average value of the loss function is calculated and represents the generalization ability of the network: the smaller this value, the greater is the generalize ability. Fig. 7 shows the convergence curve of the loss function for different GRU-RNN network models. The largest value of the loss function at the beginning of training is 3.8. After 35, 000 iterations, the value is close to zero, indicating that convergence has been achieved. The horizontal axis represents the number of trainings, and the vertical axis is the Mean Squared Error. Fig. 8 shows the loss function curves for each model after learning is complete. From Table. 1, it is

TABLE 1. Performance of different numbers of GRUs during training.

Number of GRUs	10	20	30
Minimum iteration error	0.083	0.086	**0.067**
Convergence steps	**3962**	4573	4762

40	50	60	70
0.076	0.072	0.088	0.083
4258	5819	7332	7514

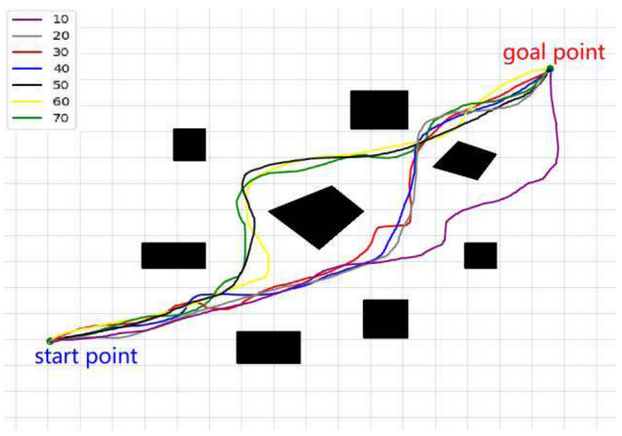


FIGURE 9. Compare the planned routes of different GRU units.

concluded that GRU30 achieves the smallest iteration error, and GRU10 has the fastest convergence rate. Evaluate the planning path capabilities (such as path length, planning time) of different models in the same environment. Fig. 9 shows the planned route for different GRU models. It shows that GRU10 has the shortest planning time and GRU40 has the shortest path length in Table. 2. The path length and planning time are the average of 50 experiments.

TABLE 2. Performance of different GRUs in statistical experiments.

Number of GRUs	10	20	30
Path length(m)	1201.4	1296.7	1163.8
Planning time(ms)	**30.25**	33.68	37.29

40	50	60	70
1153.7	1172.9	1238.1	1185.4
40.36	43.27	46.19	48.61

The success rate is given by:

$$Success\ rate = \frac{\sum_N 1(s, g)}{N} \tag{28}$$

where, $1(s, g)$ is equal to 1 if the robot can safely reach the goal point from the starting point on the test dataset(100 different environments). Fig. 10 curve shows the success rate achieved by the learned GRU-RNN network model for different test environments. It can be seen that the learned model can provide very good predictions of the robot head-ing angle in unknown dynamic environments from Fig. 12, 15. Thus, the GRU-RNN model can be used as an efficient method for offline path planning.

From the above comparison, it can be concluded that GRU40 has little difference between the convergence speed and the minimum iteration error and the optimal value. However, it is clearly dominant in the length of the planned path and the ability to successfully avoid obstacles. So we choose GRU40 to compare with the algorithm of the teacher system.

Fig. 12 shows the direction of movement of the mobile robot, and it can be seen that the trend of the orientation

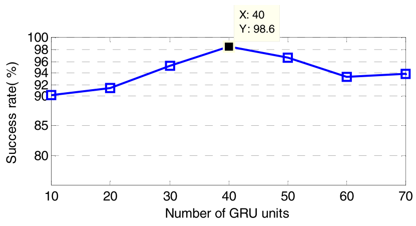


FIGURE 10. Success rate achieved by the learned model for different test environments.

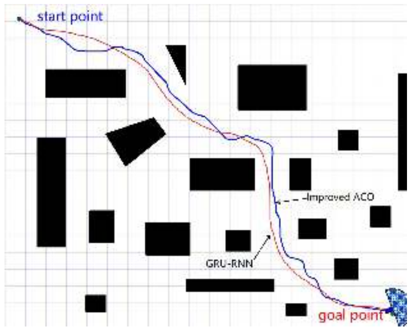


FIGURE 11. Comparison of planning routes in an unknown environment with simple obstacles.

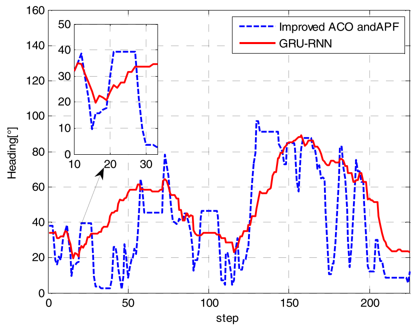


FIGURE 12. Curves of the heading process.

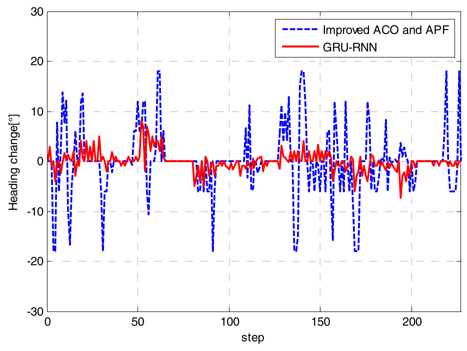


FIGURE 13. Curve of the heading change process.

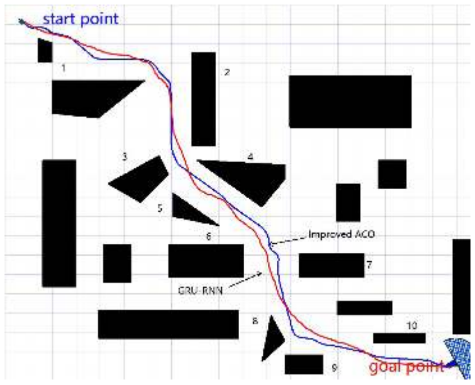


FIGURE 14. Comparison of algorithm planning routes with more complicate obstacles.

angle of the robot is generally the same for the two planning methods. It can also be seen that the direction given by the improved ACO algorithm changes very steeply at times 12-22, 27-30, 73-78, 104-112, and 121-131. The direction angle of the GRU-RNN network also changes frequently, but, in the same time range, it changes slowly, which is beneficial for the robot's motor when executing instructions in real-world applications.

Fig. 13 shows the variation of the heading change angle at each instant for the mobile robot according to the two planning algorithms. It can be seen that the results of the two methods are very different. The improved ACO algorithm leads to much sharper cornering by the robot with the greatest change in direction angle being more than 36°. Under GRU-RNN planning, the rotation angle of the robot changes more frequently, but the maximum variation is only 15°. Frequent changes occur because the network model must maintain a certain degree of generalization so that it is valid for all input data. For a given location, the GRU-RNN network is unable to reduce the cornering to zero as in contrast to the ACO algorithm, but it can reduce the cornering changing by continuous training. In the final curve of Fig. 13, there is no obstacle within the detection range of the mobile robot, so the change in rotation angle given by both methods is close to zero.

The situation shown in Fig. 15, the GRU-RNN algorithm gives a smoother curve. It can be seen that the emergency collision avoidance algorithm is called from obstacle number 3 to 6, and from number 8 to 9. The path planned using the improved APF is clear close to the obstacle, and the path of the GRU-RNN is safer.

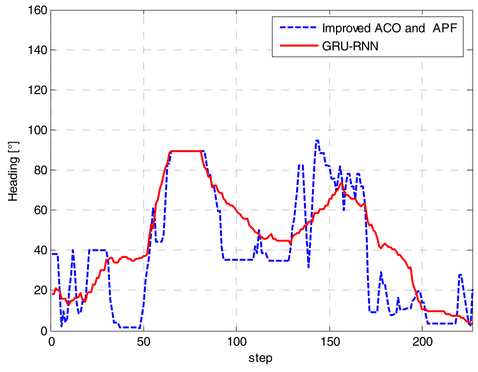


FIGURE 15. Curves of the heading process.

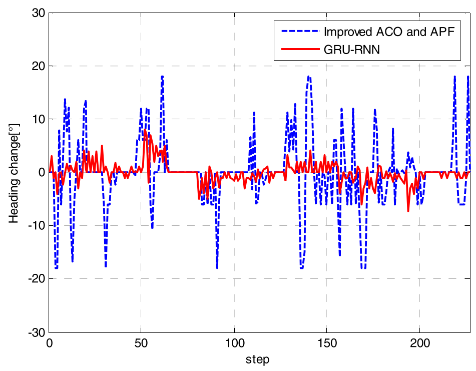


FIGURE 16. Curves of the heading change process.

Fig. 15 shows the improved ACO algorithm and APF with multiple stages maintains the same direction angle, while the GRU-RNN has few, but when the rotation is needed, the former is more intense and the latter is smoother. It can be seen from Fig. 16 that according to the GRU-RNN algorithm the change in rotation angle is within 10°, although there are frequent changes between steps 50-65 because the robot will then encounter obstacles 1,2 and will continually be adjusting its direction. However, when the heading angle is adjusted according to the improved ACO and APF algorithm, the change can reach 34°, and in practical application, it would then be difficult for the implementing agency to complete the action within the specified time.

From the comparison of the simulation results of the two algorithms, it can be seen that the GRU-RNN network model proposed in this paper has a strong learning ability. The results show that the learned GRU-RNN network model almost replicates the teacher system trajectory in test cases. After training, the network model can output the appropriate velocity and direction of movement. To test the robustness of the algorithm, a network trained in a complex environment is applied to a simple environment. It can be seen from Fig. 17 that the proposed algorithm in this paper can also implement path planning ability in a simple environment. Its dynamic programming effect is better than that obtained with the improved ACO and APF algorithm.

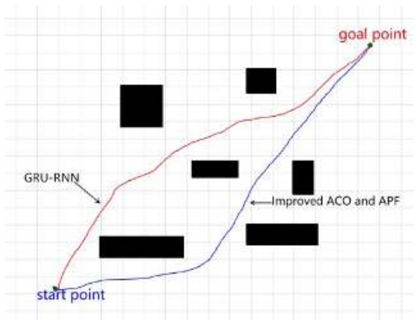


FIGURE 17. Comparison of the algorithm planning routes in a simple environment.

Figure 18 shows the planning routes for all algorithm. The data in Table. 3 shows that the planned path of GRU40 is the shortest. The time required for each step is almost half of the improved ACO and APF algorithm. The planning time of all GRU models is less than the time required to improve ACO and APF, indicating that the GRU models has high real-time performance. The path length of GRU40 is also smaller than the improved ACO and APF algorithm.

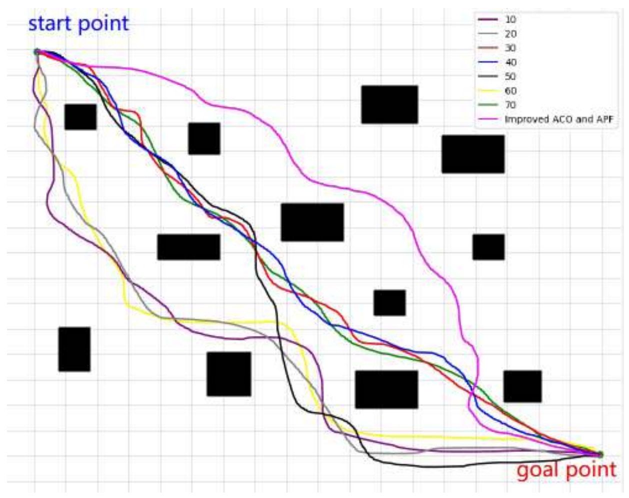


FIGURE 18. Comparison of all algorithm planning routes.

TABLE 3. Performance of different algorithm.

Number of GRUs	10	20	30	40	50	60	70	Improed ACO and APF
Path length(m)	1328.3	1302.4	1187.9	**1142.6**	1358.7	1287.6	1267.5	1245.7
Planning time(ms)	32.06	**31.25**	35.86	**37.21**	42.39	45.61	49.52	**73.36**

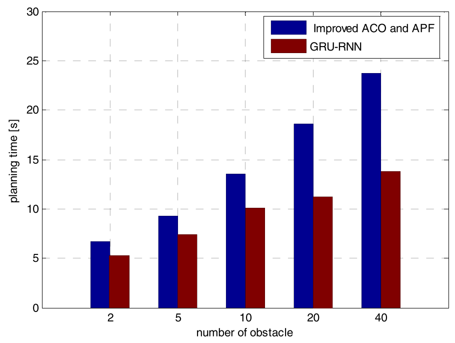


FIGURE 19. Comparison of planning time and number of obstacles between the two algorithms.

Fig.19 provides statistics on the relationship between planning time and the number of obstacles for the two algorithms. It can be seen that with an increase number of obstacles, the planning time advantage of the GRU-RNN algorithm also gradually increases. In terms of computational time required, the GRU-RNN network model is superior to the improved ACO algorithm. This is because traditional intelligent algorithms must iterate multiple times, which increases the runtime of the algorithm. After the data is input, the trained network can output the result after a series of mathematical operations, which undoubtedly shortens the running time of the program. For the GRU-RNN network model, if the training set can be increased in size, the network model will give a more stable and better performance.

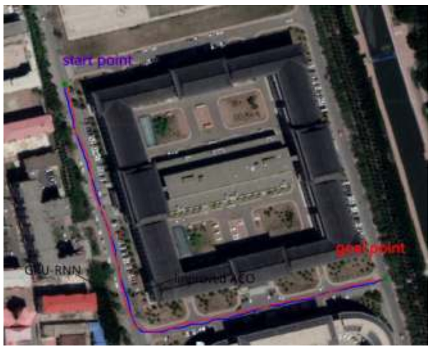


FIGURE 20. Path map for outdoor collision avoidance planning.

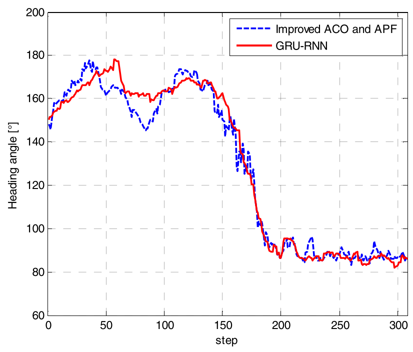


FIGURE 21. Curves of the heading process.

Fig. 20 shows the application of the best network model for final training to real case of robot collision avoidance planning. The outdoor collision avoidance planning experiment is performed and the results are compared with those of the improved ACO and APF algorithm. It can be seen from Fig. 21 that the network model proposed in this paper makes the robot angle smoother and also significantly reduces the changes in angle, especially at the corner at steps 150-200. The angle is based on true North.

VI. CONCLUSION

This paper introduces the development of a mobile robot collision avoidance algorithm based on improved ACO and APF and designs a new GRU-RNN network model for dynamic path planning of mobile robots in an unknown environment. From the results of simulation, it can be concluded that the GRU-RNN network has learned the planning policy of the teacher system and that there is an overall performance due to the use of the improved ACO and APF algorithm. The network model can learn the input data, learn the corresponding input-output mapping relationship, estimate the output and make the correct decision. Strengths: The algorithm in this paper has a good performance in terms of real-time and smoothness of the planning path. The time required for each step of planning is much smaller than that of the teacher system. The training of the completed network is directly transplanted to the real robot, and the results are also satisfactory to show that the algorithm is robust. In addition, the algorithm can be applied to the new environment without the need to change parameters, and can also achieve collision avoidance planning. Weakness: The disadvantage is that the training network needs samples generated by the teacher system, sometimes it is impossible to reach the target point accurately, although the probability is less than 2%.

In the future, we will focus on the use of deep reinforcement learning methods (such as deep Q-learning (DQN), deterministic policy gradient algorithm (DDPG), policy search, asynchronous advantage actor-critic(A3C)) to deal with robot navigation tasks by learning from its own success or failure experience. This approach eliminates the need for additional teacher systems to generate training samples.

APPENDIX

Algorithm Teacher System Collision Avoidance Algorithm

```
Initialize the environment, and parameters.
**if** there is no effective visible point **or** robot is located in the emergency collision avoidance zone, the improved APF method is used to avoid collision.
**else:** #Using improved ACO for collision avoidance.
    **Step 1:** each visible point $i$ is defined with a correspond-ing set of visual points $allow_i$.
    **Step 2:** give pheromone trails $\tau_{ij}$ a smaller positive number, historically optimal retained algebraic counter $count = 0$, evolution algebraic counter $G_count = 0$ set historical optimal maximum retained algebra $Max$ and maximum evolution algebra $Generation$, set the initial historical optimal ant path cost $history_best = \infty$,for each ant population $m = 30$.
    **Step 3:** $k = 1$
    **Step 4:** if $k > m$, go to Step7; otherwise, place the ant $k$ in the starting position $g_s$ and go to Step 5.
    **Step 5:** set current position of the ant $k$ is $g_i$, if the visible point list $allow_i$ of ant $k$ is empty, then the ant $k$ die, and go to Step 4.
    **Step 6:** if the target point $g_e \in allow_i$, the ant $k$ finds the complete path, $k = k + 1$ and goes to Step 4; otherwise, according to formula (9), the probability is selected that the ant $k$ selects any point $g_j \in allow_i$ from the $allow_i$ of the current position $g_i$, and delete $g_i$ from $allow_i$, set $g_j$ is the current position of ant $k$, and go to Step 5.
    **Step 7:** $G_count+ = 1$, if the path of the ant $k$ in this iteration is the best, update $history_best$ and $count = 0$,otherwise $count+ = 1$.
    **Step 8:** **if** $count > Max$ **or** $G_count \geq Generation$, end. Otherwise, the pheromone trail is processed according to the formulas (3) (5) (7), then go to Step 3.
```

REFERENCES

[1] P. B. Kumar, H. Rawat, and D. R. Parhi, "Path planning of humanoids based on artificial potential field method in unknown environments," *Expert Syst.*, vol. 43, no. 12, pp. 7655–7678, Dec. 2018, doi: 10.1007/s13369-018-3157-7.

[2] C.-C. Kao, C.-M. Lin, and J.-G. Juang, "Application of potential field method and optimal path planning to mobile robot control," in *Proc. IEEE Int. Conf. Automat. Sci. Eng. (CASE)*, Gothenburg, Sweden, Aug. 2015, pp. 1552–1554.

[3] R. Osorio-Comparán, I. López-Juárez, A. Reyes-Acosta, M. Pena-Cabrera, M. Bustamante, and G. Lefranc, "Mobile robot navigation using potential fields and LMA," in *Proc. IEEE Int. Conf. Automatica (ICA-ACCA)*, Curico, Chile, Oct. 2016, pp. 1–7.

[4] T. Weerakoon, K. Ishii, and A. A. F. Nassiraei, "An artificial potential field based mobile robot navigation method to prevent from deadlock," *J. Artif. Intell. Soft Comput. Res.*, vol. 5, no. 3, pp. 189–203, Sep. 2015, doi: 10.1515/jaiscr-2015-0028.

[5] D. A. Beloglazov, V. I. Finaev, A. E. Titov, I. O. Shapovalov, and V. V. Soloviev, "Group robot control in non-deterministic environments using the potential field method," in *Proc. 16th Int. Conf. Control, Automat. Syst. (ICCAS)*, Gyeongju, South Korea, Oct. 2016, pp. 371–376.

[6] X. Chen, Y. Kong, X. Fang, and Q. Wu, "A fast two-stage ACO algorithm for robotic path planning," *Neural Comput. Appl.*, vol. 22, no. 2, pp. 313–319, Feb. 2013, doi: 10.1007/s00521-011-0682-7.

[7] Z. Cao, S. Jiang, J. Zhang, and H. Guo, "A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1958–1973, Jul. 2017.

[8] U. Cekmez, M. Ozsiginan, and O. K. Sahingoz, "A UAV path planning with parallel ACO algorithm on CUDA platform," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Orlando, FL, USA, May 2014, pp. 347–354.

[9] P. K. Das, H. S. Behera, and B. K. Panigrahi, "A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning," *Swarm Evol. Comput.*, vol. 28, pp. 14–28, Jun. 2016.

[10] I. Châari, A. Koubâa, and S. Trigui, "SmartPATH: An efficient hybrid ACO-GA algorithm for solving the global path planning problem of mobile robots," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 7, p. 94, Jan. 2014, doi: 10.5772/58543.

[11] D. Bodhale, N. Afzulpurkar, and N. T. Thanh, "Path planning for a mobile robot in a dynamic environment," in *Proc. IEEE ROBIO*, Bangkok, Thailand, 2009, pp. 2115–2120.

[12] Q. Li, C. Zhang, and C. Han, "Path planning based on fuzzy logic algorithm for mobile robots in dynamic environments," *J. Central South Univ.*, vol. S2, 2013.

[13] A. S. Matveev, H. Teimoori, and A. V. Savkin, "A method for guidance and control of an autonomous vehicle in problems of border patrolling and obstacle avoidance," *Automatica*, vol. 47, no. 3, pp. 515–524, Mar. 2011, doi: 10.1016/j.automatica.2011.01.024.

[14] M. Kothari and I. Postlethwaite, "A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees," *J. Intell. Robot. Syst.*, vol. 71, no. 2, pp. 231–253, Aug. 2013, doi: 10.1007/s10846-012-9776-4.

[15] O. Adiyatov and H. A. Varol, "Rapidly-exploring random tree based memory efficient motion planning," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, Takamatsu, Japan, Aug. 2013, pp. 354–359.

[16] H. Shen and C. Guo, "Path-following control of underactuated ships using actor-critic reinforcement learning with MLP neural networks,"in *Proc. 6th Int. Conf. Inf. Sci. Technol. (ICIST)*, Dalian, China, May 2016, pp. 317–321.

[17] S. Shalev-Shwartz, S. Shammah, and A. Shashua. (2016). "Safe, multi-agent, reinforcement learning for autonomous driving." [Online]. Available: https://arxiv.org/abs/1610.03295

[18] A. Carrio, C. Sampedro, and A. R. Ramos, "A review of deep learning methods and applications for unmanned aerial vehicles," *J. Sensors*, vol. 2017, pp. 1–13, Aug. 2017, doi: 10.1155/2017/3296874.

[19] M. Inoue, T. Yamashita and T. Nishida, "Robot path planning by LSTM network under changing environment," in *Proc. ICCS*, Thailand, Oct. 2017, pp. 317–329.

[20] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *J. Mach. Learn. Res.*, vol. 3, pp. 115–143, Aug. 2002, doi: 10.1162/153244303768966139.

[21] M. Dorigo, M. Birattari, C. Blum, and M. Clerc, "Ant colony optimization and swarm intelligence," in *Proc. 6th Int. Conf. ANT*, Berlin, Germany, Sep. 2008, pp. 13–24.

[22] A. Graves, "Long short-term memory," in *Supervised Sequence Labelling With Recurrent Neural Networks*. Berlin, Germany: Springer, Feb. 2012, pp. 37–45.

[23] W. Hong-jian, Z. Jie, B. Xin-qian, and S. Xiao-cheng, "An improved path planner based on adaptive genetic algorithm for autonomous underwater vehicle," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, Niagara Falls, ON, Canada, Jul./Aug. 2005, pp. 857–861.

[24] L. Tai, P. Giuseppe, and L. Ming, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 31–36.

[25] L. Tai, L. Shaohua, and L. Ming, "A deep-network solution towards model-less obstacle avoidance," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Daejeon, South Korea, Oct. 2016, pp. 2759–2764.

[26] L. Joonwoo, "Heterogeneous-ants-based path planner for global path planning of mobile robot applications," *Int. J. Control, Automat. Syst.*, vol. 15, no. 4, pp. 1754–1768, Aug. 2017, doi: 10.1007/s12555-016-0443-6.

[27] E. Masehian and D. Sedighizadeh, "A multi-objective PSO-based algorithm for robot path planning," in *Proc. IEEE Int. Conf. Ind. Technol.*, Vina del Mar, Chile, Mar. 2010, pp. 465–470.

[28] A. Pal, R. Tiwari, and A. Shukla, "Modified a algorithm for mobile robot path planning," in *Soft Computing Techniques in Vision Science*, vol. 395. Berlin, Germany, Springer, 2012, pp. 183–193.

[29] Z. Cao *et al.*, "Maximizing the probability of arriving on time: A practical Q-learning method," in *Proc. AAAI*, San Francisco, CA, USA, 2017, pp. 4481–4487.

[30] A. Khan and F. Zhang, "Using recurrent neural networks (RNNs) as planners for bio-inspired robotic motion," in *Proc. IEEE Conf. Control Technol. Appl. (CCTA)*, Mauna Lani, HI, USA, Aug. 2017, pp. 1025–1030.

[31] D. Silver *et al.*, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, Jun. 2014, pp. 378–395.

[32] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[33] V. Mnih *et al.* (2013). "Playing Atari with deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1312.5602

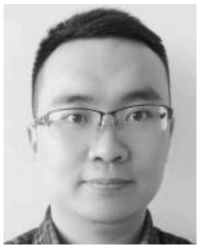
[34] T. P. Timothy *et al.* (2016). "Continuous control with deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1509.02971

JIANYA YUAN received the B.S. degree in electrical engineering and automation from Mudanjiang Normal University, China, in 2015. He is currently pursuing the Ph.D. degree in control science and engineering with Harbin Engineering University, China. His current research interests include multi-agent systems, path planning, and deep reinforcement learning.

HONGJIAN WANG was born in Harbin, Heilongjiang, China, in 1971. She received the B.S. degree in computer science from Heilongjiang University, in 1993, and the M.S. degree in computer science and the Ph.D. degree in control theory and control engineering from Harbin Engineering University, in 1998 and 2004, respectively.

She was with Harbin Engineering University as a Research Assistant, from 1993 to 1998, as a Lecturer, from 1998 to 2003, and as an Associate Professor, from 2003 to 2005, where she has been a Professor with the College of Automation, since 2005, and has been a Doctoral Tutor, since 2006. From 2004 to 2006, she was with the Postdoctoral Mobile Station, Harbin Institute of Technology. Her research interests include autonomy control and swarm intelligence for unmanned systems, intelligent optimization theory and methods, machine learning, deep learning, deep reinforcement learning, target tracking, and simultaneous localization and mapping.

CHANGJIAN LIN received the B.S. degree in electrical engineering and automation from the Changchun University of Science and Technology, China, in 2015. She is currently pursuing the Ph.D. degree in control science and engineering with Harbin Engineering University, China. Her current research interests include navigation and control, intelligent control methods, and machine learning.



DAWEI LIU received the B.S. degree in electrical engineering and automation from Qingdao University, China, and the master's degree in control science and engineering from Harbin Engineering University, China, in 2018. His research interests include path planning and machine learning.

DAN YU received the B.S. degree in measurement and control technology and instruments from the Harbin University of Science and Technology, China. She is currently pursuing the M.S. degree in control science and engineering, Harbin Engineering University, China. Her main research interest includes deep-learning-based cluster collision avoidance.