The Open Automation and Control Systems Journal, 2014, *6*, 77-83

Path Planning and Obstacle Avoidance for Mobile Robots in a Dynamic Environment

Liping Sun, Yonglong Luo[*], Xintao Ding and Longlong Wu

College of National Territorial Resources and Tourism, Anhui Normal University, Wuhu, Anhui 241000, China

Abstract: Because traditional obstacle avoidance path planning methods have a lot of problems, such as large amount of calculation, low efficiency, poor optimization capability, and lack of dealing with dynamic obstacles, a new method which implements real-time path planning of mobile robot is presented. The method builds a neural network model for the robot workspace, and then it uses the model to obtain the relationship between the dynamic obstacles and the network output. It can choose the local optimal collision-free path by the path planning in a dynamic environment (PPIDE) algorithm to find the path between two points for dealing with obstacles. The proposed method is suitable for dynamic environment where both linear and planar obstacles exist. Simulation results prove its effectiveness.

Keywords: BP neural networks, dynamic environment, obstacle avoidance, path planning.

1. INTRODUCTION

A path planning method for mobile robots is to generate an optimal or near-optimal collision-free path between an initial location and the desired destination with specific constraint conditions [1-3]. Path planning method can be utilized to system simulation, urban traffic, and urban planning & design [4, 5]. According to mobile robots under the space environment of work proficiency information can be divided into two types: known environmental information for global planning and environmental information completely unknown or partially unknown cases of local planning. There exist a number of specific kinds of methods for global path planning, such as visibility graph methods [6], grid method [7], free-space method [8]. Simultaneously there are methods for local path planning, including neural networks [9], artificial potential field method [10], and fuzzy logic algorithm [11]. Because of the ambiguity of the actual situation of workspace and the practical demand of time and spatial characteristics of algorithm, the researches on local path planning are more than global path planning. Under intensive dynamic obstacle environment, there is a lack of effective methods to consider the control deviation of the moving process of robots for path planning.

The neural network is an arithmetic model, which imitates the behavior characteristic of the animal neural network. As a highly parallel distributed system, it offers the possibility to solve the problem of high real-time robot system. Therefore, artificial neural network model of complex dynamic environment has been widely used [12-18].

*Address correspondence to this author at the Department of Computer Science and Technology, Anhui Normal University, 189 Jiuhua South Rd., Wuhu, Anhui Province 241003, P. R. China; Tel: (+86-553) 5910645; E-mail: ylluo@ustc.edu.cn

In this paper, we present the method of mobile robot path planning in dynamic environment. The static search of algorithm for path planning was firstly proposed to obtain an optimal collision-free path from the beginning to the end point. Artificial neural network is employed to describe the relationship between the dynamic obstacles and the network output. And the searching algorithm for path planning has been revised to solve the local optimal path of the current round. Compared with the traditional path planning algorithm, simulation results show that the path planning in dynamic environment (PPIDE) is more practical and efficient.

2. PATH PLANNING ALGORITHM BASED ON NEURAL NETWORK

2.1. Description of the Workspace

First, this paper proposes the following suppositions to the workspace:

- The robot moves in the two-dimensional bounded space, which will not surpass the boundary.
- The robot itself can be abstracted as a particle, and the particle is the geometric center of the robot.
- Limited obstacles, including linear obstacles and planar obstacles, are assigned in the workspace randomly.
- Obstacles random motion in the workspace, but do not change their shape.
- The robot moves with constant speed, and has good brakes.
- Movement trajectory of the obstacles can be described by the continuous function, which is advantageous for the comparison of the real value and predicted value.

Based on the above suppositious, the workspace of the robot can be set as shown in Fig. (**1**). The robot works in a

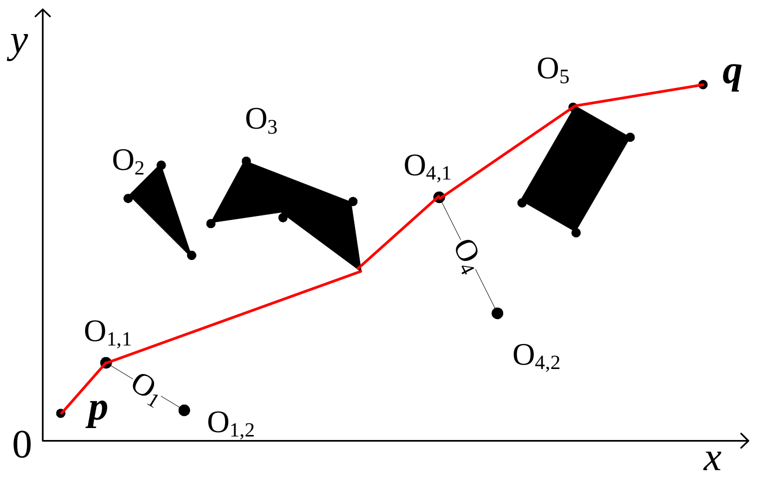

Fig. (1). Schematic diagram of workspace with obstacle constraint.

finite two-dimensional space, which is indicated with the O_{xy} coordinate system. Point p represents the starting point of the robot, and point q represents the end point of the robot. In the figure, black shaded area o_i represents the i-th obstacle to the positive direction of the x-axis. $o_{i,j}$ expresses the j-th vertex of the i-th obstacle.

2.2. Static Search Algorithm for Path Planning

In Fig. (**1**) as described in the robot workspace, all the obstacles have been vectorized line or polygons. Each obstacle keeps still in the static space. As a result of the robot to the overall situation environment understanding, the best collision-free path from beginning p to end point q can be achieved. For the convenient expression on the searching algorithm, the relevant symbols are defined as follows: Assume that abscissa of p is not greater than q. L denotes the set of linear obstacles, and S denotes the set of planar obstacles. *path*1 and *path*2 respectively are the obstacle paths of the left and right side of $\overrightarrow{pq}$; p_1 and p_2 denote the starting points of the paths that will be added to the *path*1 and *path*2 separately; Similarly, q_1 and q_2 denote the end points of the paths that will be added to the *path*1 and *path*2 separately. Function *reach*(p, q) is to judge whether point p and q are directly reachable, and *dis*(p, q) is the Euclidean distance between point p and q.

The search algorithm shows as follows:

```
Algorithm 1: Search algorithm for path planning
1: $path1=\phi$; $path2=\phi$; $p_1 = p_2 = p$ ; $q_1 = q_2 = q$ ; $i = 0$.
2: **If** ($reach(p, q)$ = TRUE)
3: $D_{pq} = dis(p, q)$; $path1 = path1 \cup \overrightarrow{p_1q_1}$ ; $path2 = path2 \cup \overrightarrow{p_2q_2}$ ; return;
4: **End if**
5: Find the obstacles intersect with obstacle group. The obstacles expressed in turn are $O_1, O_2, \ldots, O_m$, in which $m$ is the number of the obstacles. Sorting the vertices of $O_i$ ($1 \le i \le m$) in descending order according to its y-coordinate values, denoted as $o_{i,j}$ ($j = 1, 2, \ldots, k$, $k$ is the number of vertices of $O_i$).
6: **If** ($O_i \in L$)
7: $q_1 = O_{i,1}$, $q_2 = O_{i,2}$. The point of intersection of $\overrightarrow{pq}$ and $\overrightarrow{O_{i,1}O_{i,2}}$ , denoted as $h$.
8: **If** ($reach(p_1, q_1)$ = TRUE)
9: $path1 = path1 \cup \overrightarrow{p_1q_1}$ ;
10: **Else**
11: $path1 = path1 \cup \overrightarrow{p_1h} \cup \overrightarrow{hq_1}$ .
12: **End if**
13: **If** ($reach(p_2, q_2)$ = TRUE)
14: $path1 = path1 \cup \overrightarrow{p_2q_2}$ ;
15: **Else**
16: $path2 = path2 \cup \overrightarrow{p_2h} \cup \overrightarrow{hq_2}$ .
17: **End if**
18: Set $p_1 = q_1$; $p_2 = q_2$; $i$++; Go 43.
19: **End if**
20: **If** ($O_i \in S$)
21: The point with the smallest abscissa of intersection points of $\overrightarrow{pq}$ and $O_i$, denoted as $h$; $V_1$ and $V_2$ respectively are the sets of vertices of $O_i$ on the left and right side of $\overrightarrow{pq}$ . Sorting the vertices of $V_1$ in descending order according to its ordinate, denoted as $o_{i,j}$ ($j = 1, 2, \ldots, k$, $k$ is the number of vertices of $V_1$). Sorting the vertices of $V_2$ in ascending order according to its ordinate, denoted as $o_{i,j'}$ ($j' = 1, 2, \ldots, k'$, $k'$ is the number of vertices of $V_2$). Let $o_{i,k+1} = h$; $o_{i,k'+1} = h$.
22: **If** ($reach(p_1, q)$ = TRUE)
23: $path1 = path1 \cup \overrightarrow{p_1q}$ ; Go 43.
24: **Else**
```

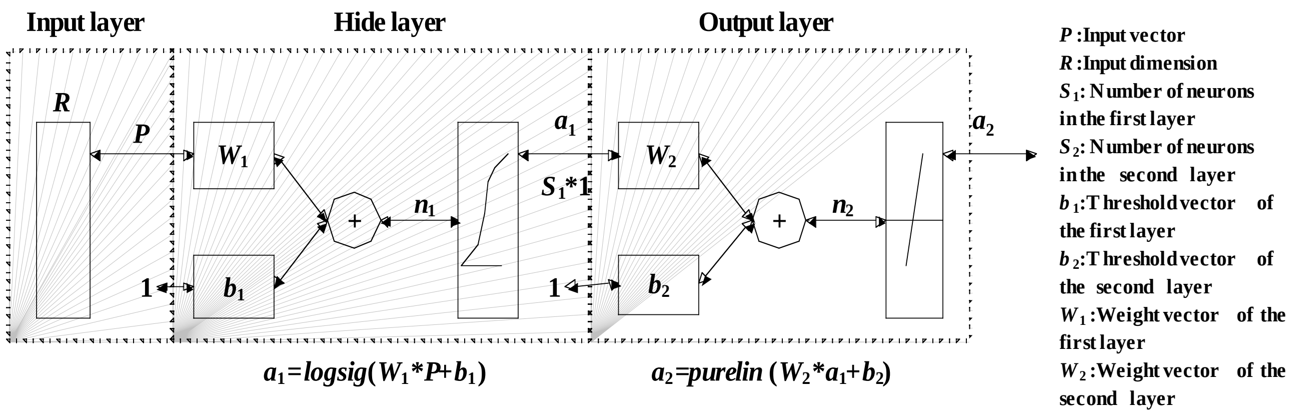
Input layer
Hide layer
Output layer
R
P
1
W_1
b_1
n_1
a_1
S_1*1
1
W_2
b_2
n_2
a_2
a_1=logsig(W_1*P+b_1)
a_2=purelin (W_2*a_1+b_2)
P :Input vector
R :Input dimension
S_1: Number of neurons in the first layer
S_2: Number of neurons in the second layer
b_1:Threshold vector of the first layer
b_2:Threshold vector of the second layer
W_1 :Weight vector of the first layer
W_2 :Weight vector of the second layer

Fig. (2). Prediction model based on BP neural network.

```
25: Set $j = 1$, $q_1 = o_{i,j}$;
26: **If** ($reach(p_1, q_1)$ = TRUE)
27: $path1 = path1!\mathrm{E}\,\overrightarrow{p_1q_1}\bigcup_{k=2}^{j}\overrightarrow{O_{i,k}O_{i,k-1}}$ ; Go 32.
28: **Else**
29: Let $j = j + 1$; $q_1 = o_{i,j}$; Go 26.
30: **End if**
31: **End if**
32: **If** ($reach(p_2, q)$ = TRUE)
33: $path2 = path2 \cup \overrightarrow{p_2q}$ ; Go 43.
34: **Else**
35: Set $j' = 1$, $q_2 = o_{i,j'}$;
36: **If** ($reach(p_2, q_2)$ = TRUE)
37: $path2 = path2!\mathrm{E}\,\overrightarrow{p_2q_2}\bigcup_{k=2}^{j'}\overrightarrow{O_{i,k'}O_{i,k'-1}}$ ; Go 43.
38: **Else**
39: Let $j' = j' + 1$; $q_2 = o_{i,j'}$; Go 36.
40: **End if**
41: **End if**
42; **End if**
43: **If** ($i < m$)
44: go 6;
45: **Else**
46: $path1 = path1 \cup \overrightarrow{p_1q}$ , $path2 = path2 \cup \overrightarrow{p_2q}$ .
47: Compare the length of *path1* and *path2*, and
    choose the shorter one.
48: **End if**
```

2.3. Prediction Model Based on Back-propagation (BP) Neural Network

Since the movements of obstacles are mutually independent in a dynamic environment, the movement trend of obstacle should be predicted in path planning. The position data of the previous moments and the next moment to be predicted are taken as input and the position data of the next moment by mathematical dynamic models as output. Prediction model only emphasizes input and output, which has no structural limits on specific mathematical models. For the whole work space and obstacles are studied as a macroscopic system in the process of path planning, it is difficult to get accurate mathematical model for each obstacle for multi-step prediction.

The neural network can carry on study and training to the system data approaching the real system, without knowing the internal structure of the real system. BP neural network can learn and store large amounts of input-output model mapping, which does not need to know mathematical equation to describe this kind of mapping relations in advance. Due to the following advantages, such as massively parallel processing, self-organizing, self-learning, BP neural network model is used to predict the location of obstacles in this paper. The prediction model based on BP neural network is represented in Fig. (**2**).

2.4. Path Planning in Dynamic Environment (PPIDE) Algorithm

This paper presents a dynamic environment of path planning algorithm. The algorithm first uses neural network to predict the position of obstacles in the next turn, followed by the searching algorithm for path planning to determine the local optimal path of the current round, and finally update the actual location of obstacles through the robot's sensors. An optimal collision-free path from the beginning to the end in a dynamic environment can be achieved by the iterations. The diagram of the algorithm, denoted PPIDE algorithm, is shown in Fig. (**3**).

Search algorithm for path planning needs to be revised to solve the local optimal path of the current round. $U = (x_U, y_U)$

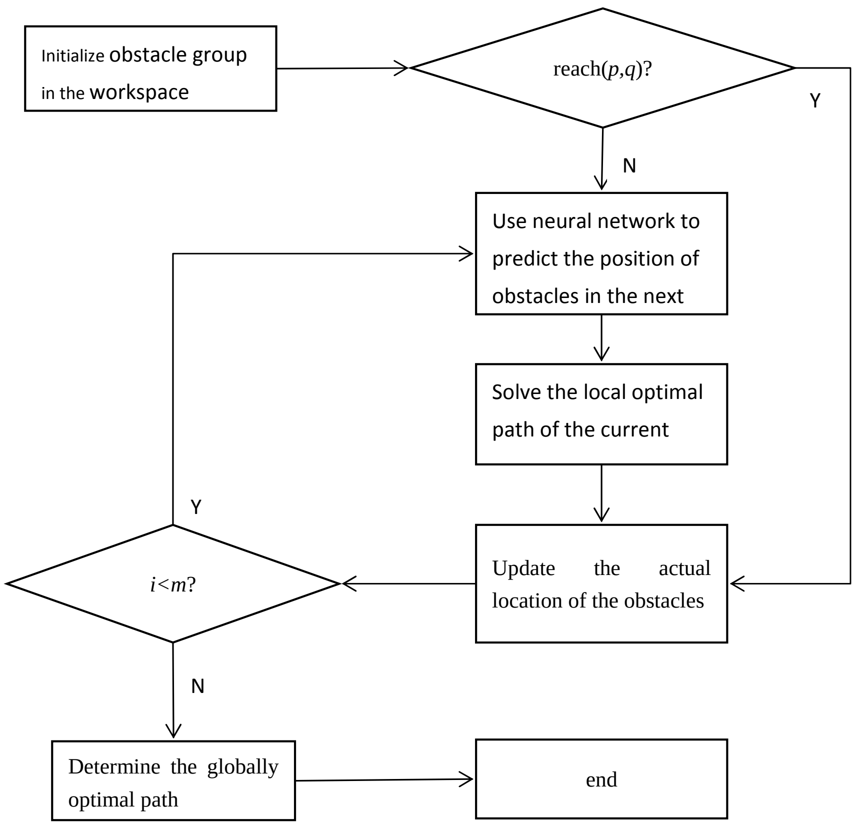
Initialize obstacle group in the workspace
reach(p,q)?
Y
N
Use neural network to predict the position of obstacles in the next
Solve the local optimal path of the current
Update the actual location of the obstacles
Y
i<m?
N
Determine the globally optimal path
end

Fig. (3). Flow diagram of the PPIDE algorithm.

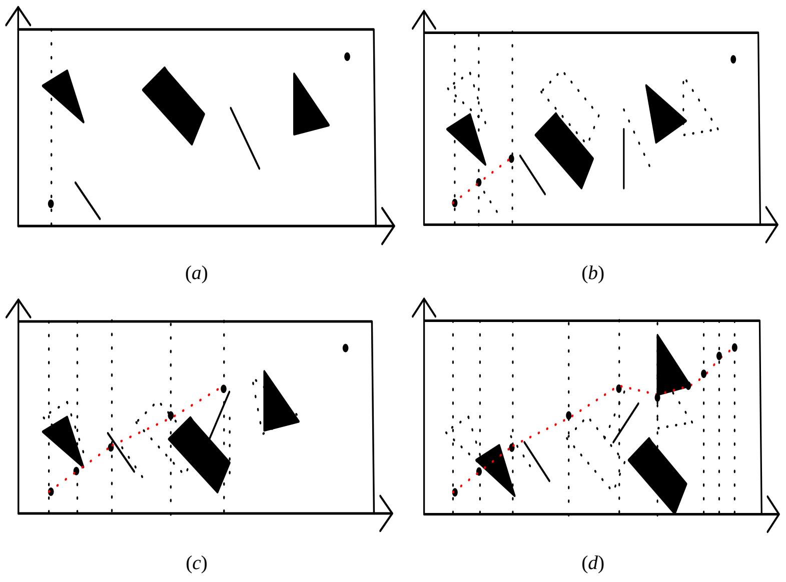

Fig. (4). The movement of obstacles and path planning in dynamic environment.

denotes point U in the workspace, where x_U and y_U respectively are the abscissa and ordinate. v_0 represents the moving speed of a robot. Let t_0 be the time interval for each round. Let U, V represent the position which the robot reaches with the movement in the direction of $\overrightarrow{P_1Q_1}$ and $\overrightarrow{P_2Q_2}$ after one round. Let the angle between the positive x-axis and $\overrightarrow{P_1Q_1}$ be θ_1, the angle between the positive x-axis and $\overrightarrow{P_2Q_2}$ be θ_2.

The coordinates of point U and point V can be expressed as follows: $U = Q_1 + L_{P_1Q_1}$, where $U = (x_U, y_U)$, $Q_1 = (x_{Q_1}, y_{Q_1})$, and $L_{P_1Q_1} = (v_0 t_0 \cos\theta_1 \quad v_0 t_0 \sin\theta_1)$. Similarly, $V = Q_2 + L_{P_2Q_2}$, where $V = (x_V, y_V)$, $Q_2 = (x_{Q_2}, y_{Q_2})$, and $L_{P_2Q_2} = (v_0 t_0 \cos\theta_2$, $v_0 t_0 \sin\theta_2)$. Subsequently, the paths can be replaced by $path1 = path1 \cup \overrightarrow{P_1U}$ and $path2 = path2 \cup \overrightarrow{P_2V}$.

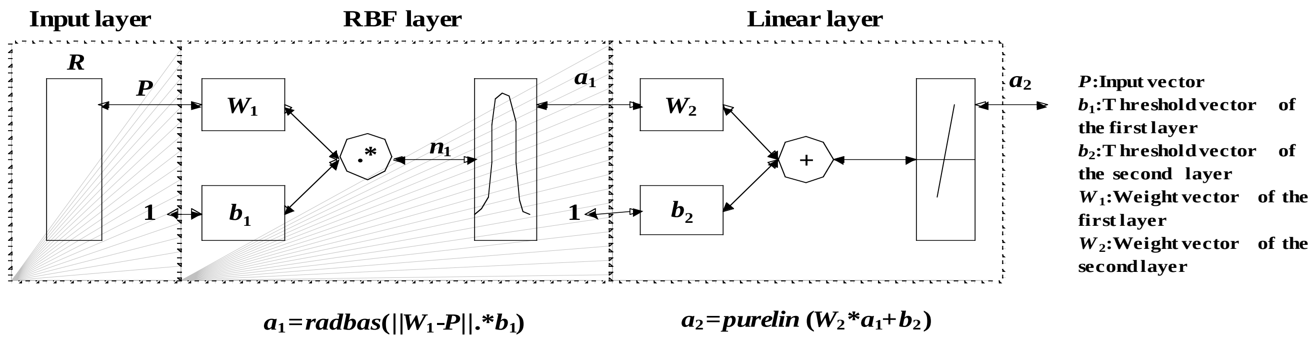
Input layer
RBF layer
Linear layer
R
P
W1
b1
1
.*
n1
a1
W2
b2
1
+
a2
P:Input vector
b1:Threshold vector of the first layer
b2:Threshold vector of the second layer
W1:Weight vector of the first layer
W2:Weight vector of the second layer
a1=radbas(||W1-P||.*b1)
a2=purelin (W2*a1+b2)

Fig. (5). Prediction model based on RBF neural network.

Table 1. Neural network predictive value and the true value of BP neural network.

Round	X_1	X_1^*	$\|x_1 - x_1^*\|$	X_2	X_2^*	$\|x_2 - x_2^*\|$	X	Y	D
1	0.3			-0.1			1	0	
2	0.9			-0.4			0.809	0.5878	
3	1.5			-0.9			0.309	0.9511	
4	2.1			-1.6			-0.309	0.9511	
5	2.7			-2.5			-0.809	0.5878	
6	3.3	3.3640	0.0640	-3.6	-3.6559	0.0559	-1	0	0.4104
7	3.9	3.9000	0	-4.9	-4.9000	0	-0.809	-0.5878	0.0001
8	4.5	4.5000	0	-6.4	-6.4000	0	-0.309	-0.9511	0.0001
9	5.1	5.1000	0	-8.1	-8.1000	0	0.309	-0.9511	0.0001
10	5.7	5.7000	0	-10	-10.0000	0	0.809	-0.5878	0
11	6.3	6.3000	0	-12.1	-12.1000	0	1	0	0

3. EXPERIMENTAL RESULTS

The actual movement of the state transition and path planning process is shown in Fig. (**4**), where only four representative intermediate states are selected. Solid line with nodes represents the currently selected path. Dotted lines represent different moments, in which the last one is the current time, and the interval between the two dotted lines represents the 5s time-gap. The dotted line polygon and dotted represents the position of the line linear obstacles and planar obstacles in the previous state time respectively, and the solid shadow part is the current position of obstacle.

Radial Basis Functions (RBF) neural network a class of widely used neural networks (Fig. **5**). To compare the performance of BP neural network and RBF neural network, the continuous function is used to describe the obstacle path in movement. Uniform rectilinear motion $X_1 = x_0 + vt$ ($x_0 = 0.3$, $v = 0.6$), uniformly decelerated linear motion $X_2 = \int_0^t atdt(a = -0.2)$, $\begin{cases} X = r\cos(\theta) \\ Y = r\sin(\theta) \end{cases}$ ($\theta = at, r{=}1$, $a = \pi / 6$) are selected as the reference examples separately. X_1, X_2 , X, Y are taken separately as input vector P. The setting of BP neural network parameters is as follows: S_1=6, S_2=6, R=6; W_1, W_2, b_1 and b_2 are initialized by function *initnw* of neural network toolbox. Parameters of RBF neural network are set as follows: $W1$, $W2$, $b1$ and $b2$ are initialized by function *initnw* of neural network toolbox. RBF neural network probability density *SPREAD* = 1. The experiments have been conducted based on the proposal conjectured in 2.1.

Since there is a stable relationship between the horizontal and vertical coordinates, the simulation to two kinds of movement x-coordinates should be conducted to obtain the predicted value of abscissa and ordinate as shown in Table **1** and Table **2**. As can be seen by comparing Table **1** and Table **2**, the error between neural network predictive value and the true value of RBF neural network is bigger than BP neural network.

X_1^*, X_2^* denote the predicted values for X_1, X_2 respectively. Let $D = \sqrt{(X^* - X)^2 + (Y^* - Y)^2}$, where X^*, Y^* denote the predicted values for X, Y respectively. According to

Table 2. Neural network predictive value and the true value of RBF neural network.

Round	X_1	X_1^*	$\|x_1 - x_1^*\|$	X_2	X_2^*	$\|x_2 - x_2^*\|$	X	Y	D
1	0.3			-0.1			1	0	
2	0.9			-0.4			0.809	0.5878	
3	1.5			-0.9			0.309	0.9511	
4	2.1			-1.6			-0.309	0.9511	
5	2.7			-2.5			-0.809	0.5878	
6	3.3	2.9032	0.3968	-3.6	-3.3943	0.2057	-1	0	1.4024
7	3.9	3.6950	0.2050	-4.9	-4.3712	0.5288	-0.809	-0.5878	0.6668
8	4.5	4.3002	0.1998	-6.4	-6.8722	0.4722	-0.309	-0.9511	0.7011
9	5.1	5.2771	0.1771	-8.1	-8.4720	0.372	0.309	-0.9511	0.4412
10	5.7	5.5893	0.1107	-10	-9.8923	0.1077	0.809	-0.5878	0.1558
11	6.3	6.2844	0.0156	-12.1	-12.2251	0.1251	1	0	0.0785

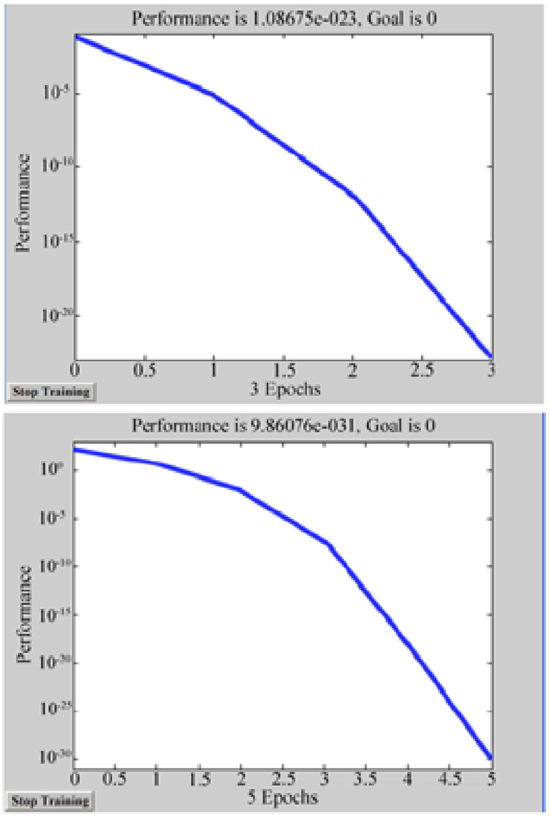

Fig. (6). Two groups of experimental training relationship between generations and errors.

Table **1**, through the true values and the predicted values of the error, the two movements using neural network simulation can get the predicted values which are close to the true values, and in which better forecast results has been achieved separately in 6th and 7th round. As it can be seen from (Fig. **6**) that the error between the output value of the neural network in the 3rd, 4th generation and the true values achieve 10^{-20} order. Therefore it can be recognized that this model converges rapidly. The simulation results show that the performance of the model is good. But it simultaneously indicates that achieving good results requires training sample as much as possible. For workspace it reduces the time scale for each round, which makes robotic perception greatly increase the speed of samples collection.

A* algorithm is an efficient heuristic search method for path planning, and will therefore be compared with the PPIDE algorithm. To verify the correctness and validity of the algorithm, this paper conducts simulation experiments

Table 3. Comparison of PPIDE algorithm and A* algorithm.

	10 Obstacles (50 Vertices)		**50 Obstacles (250 Vertices)**	
	Average Time Cost/s	**Average Value of R**	**Average Time Cost/s**	**Average Value of R**
PPIDE algorithm	44.6	1.136	123.8	1.345
A* algorithm	89.7	1.432	201.5	1.896

using Matlab 2012b and Visual C++ 6.0. R represents the ratio of the length of an optimized collision-free path and the length of the straight distance between *p* and *q*. Assume that the robot makes uniform motion with the 3M/s speed. And obstacle set including linear obstacles and planar obstacles is generated randomly. With a mean value of 500 random simulations to compare their performances, experimental results are shown in Table **3**. Experimental results show the efficiency of the path planning algorithm based on BP neural network in a dynamic environment.

CONCLUSION

This paper presents a mobile robot path planning method based on artificial neural networks and a search algorithm for path planning in a dynamic environment. Compared with the traditional path planning algorithm, it solves the problems of traditional obstacle avoidance path planning method for computer-intensive, low efficiency and optimizing capacity and unable to cope with dynamic issues such as obstacles. By comparative analysis of BP neural network and RBF neural network, simulation results show that neural network model accurately predicts the movements of the dynamic obstacles, and it converges rapidly. Local optimization strategy based on searching algorithm also achieves good results that the optimum collision-free path can be obtained. The next step will be to consider the volume of the robot, so that the robot denoted by the particle would be extended to a convex polygon case. Meanwhile, further research about the mode of motion of obstacles would be conducted.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (No. 61370050) and Natural Science Foundation of Anhui Province (No. 1308085QF118).

REFERENCES

[1] S. Shital, N. Chiddarwar, and B. Ramesh, “Conflict free coordinated path planning for multiple robots using a dynamic path modification sequence,” *Robotics and Autonomous Systems*, vol. 59, pp. 508-518, 2011.

[2] G. Tan, H. HE, and A. Sloman, “Ant colony system algorithm for real-time globally optimal path planning of mobile robots, ”, *Acta Automatica Sinica*, vol. 33, pp.279-285, 2007.

[3] R. Kala, “Multi-robot path planning using co-evolutionary genetic programming,” *Expert Systems with Applications*, vol. 39, pp. 3817-3831, 2012.

[4] D. Mansoor, P. Fatemeh, M. Ali, and N. H. Seyed, “Multi-objective path planning in discrete space, ”*Applied Soft Computing*, vol. 13, pp. 709-720, 2013.

[5] T. Adem, Y. Mehmet, “Dynamic path planning of mobile robots with improved genetic algorithm,” *Computers & Electrical Engineering*, vol. 38, pp. 1564-1572, 2012.

[6] E. Welzl, “Constructing the visibility graph for *n*-line segments in O(n^2) time,” *Information Processing Letters*, vol. 20, pp. 167-171, 1985.

[7] C. E. Thorpe, “Path relaxation: Path planning for a mobile robot,” In: *Proceedings of the 4th National Conference on Artificial Intelligence*, pp. 576-581, 1984.

[8] P. Nagabhushan, and M. M. Manohara Pai, “Cognition of free space for planning the shortest path: A framed free space approach,” *Pattern Recognition Letters*, vol. 22, pp. 971-982, 2001.

[9] R. Glasius, A. Komoda, S. C, and A. M. Gielen, “Neural network dynamics for path planning and obstacle avoidance,” *Neural Networks*, vol. 8, pp. 125-133, 1995.

[10] E. S. Conkur, “Path planning using potential fields for highly redundant manipulators,” *Robotics and Autonomous Systems*, vol. 52, pp. 209-228, 2005.

[11] G. T. Zoumponos, and N. A. Aspragathos, “Fuzzy logic path planning for the robotic placement of fabrics on a work table,” *Robotics and Computer-Integrated Manufacturing*, vol. 24 pp. 174-186, 2008.

[12] N. Noboru, T. Hideo, “Path planning of an agricultural mobile robot by neural network and genetic algorithm,” *Computers and Electronics in Agriculture*, vol. 18, pp. 187-204, 1997.

[13] Y. Naganuma, and A. Igarashi, “A packet routing strategy using neural networks on scale-free networks,” *Physica A: Statistical Mechanics and its Applications*, vol. 389, pp. 623-628, 2010.

[14] K. G. Jolly, R. Sreerama Kumar, and R. Vijayakumar, “Intelligent task planning and action selection of a mobile robot in a multi-agent system through a fuzzy neural network approach,” *Engineering Applications of Artificial Intelligence*, vol. 23, pp. 923-933, 2010.

[15] G. Mehdi, and M. Ali, “Motion planning in order to optimize the length and clearance applying a Hopfield neural network,” *Expert Systems with Applications*, vol. 36, pp. 4688-4695, 2009.

[16] Y. Zhang, D. Gong, and J. Zhang, “Robot path planning in uncertain environment using multi-objective particle swarm optimization,” *Neurocomputing*, vol. 103, pp. 172-185, 2013.

[17] R. Taormina, K. W. Chau, and R. Sethi, “Artificial Neural Network simulation of hourly groundwater levels in a coastal aquifer system of the Venice lagoon”, *Engineering Applications of Artificial Intelligence*, vol. 25, pp. 1670-1676, 2012.

[18] Z. K. Huang, and K. W. Chau, “A New Image Thresholding Method Based on Gaussian Mixture Model,” *Applied Mathematics and Computation*, vol. 205, pp. 899-907, 2008.

Received: December 21, 2013 Revised: February 17, 2014 Accepted: March 10, 2014