ICMIEE-PI-140282

Autonomous Robot Path Planning Using Particle Swarm Optimization in Dynamic Environment with Mobile Obstacles & Multiple Target

Md. Rakibul Islam[1*], *Md. Tajmiruzzaman*[2], *Md. Mahfuzul Haque Muftee*[3], *Md. Sanowar Hossain*[4]
[1*, 2, 4] Department of Industrial & Production Engineering,
Rajshahi University of Engineering & Technology, Rajshahi- 6240, BANGLADESH
[3] Department of Computer Science & Engineering,
Rajshahi University of Engineering & Technology, Rajshahi- 6240, BANGLADESH

ABSTRACT
Now a day, there are even demands for application of robots in homes and hospitals. The goal of this research is to plan a trajectory and minimizing the path lengths with collisions avoidance for a mobile robot in dynamic environment. In this paper, an intelligent approach for navigation of a mobile robot in dynamic environment with multiple targets is proposed. Particle Swarm Optimization (PSO) method is used for finding proper solutions of optimization problems. PSO has been demonstrated to be a useful technique in robot path planning in dynamic environment with mobile obstacles and multiple goals, as a feasible approach for self organized control of robot to avoid obstacle throughout the trajectory. The authors here has been used a grid based search approach for robot. The positions of the obstacles will be changed randomly. Finally, simulation results confirm the effectiveness of our algorithm**.**

Key words: Robot Path Planning, Particle Swarm Optimization, Trajectory Planning, Moving Obstacles

1. Introduction

Autonomous mobile robots used in the environment where many human beings are working, cooperating with robots. In these environments, the collision-free path planning is one of the major problems to realize autonomous mobile robots. Since there are many stationary or moving obstacles in these environments, autonomous mobile robots should plan their own path that can avoid not only stationary obstacles but also moving ones such as human workers and other robots. The main problem in robot path planning is to find a motion trajectory from a starting position to a goal position regarding to some optimization criteria. Path Planning is one of the most vital issues in the navigation of mobile robot, which means to find out an optimized collision free path from the start state to the goal state according to some performance merits. It can be classified into two categories global path planning with all the information of the robot's known environment and local path planning in a partly or totally unknown environment [1]. In global navigation methods cost of environmental change, especially in dynamic environment is very high, because supply a new map is difficult. Therefore, research on the local navigation is necessary. These methods could be able to detect the unknown environment, and it does not need to environment model. In this paper, a new method of local navigation based on particle swarm optimization technique is proposed.

There have been many algorithms for global path planning, such as artificial potential field, visibility graph, and cell decomposition etc. PSO is a heuristic search technique that is inspired by the behavior of bird flocks. Although PSO is relatively new, the relative simplicity, the fast convergence and the population-based feature [2] have made it a considerable viable alternative for solving the robot path planning problem. PSO has been used for hazardous target search applications, such as landmine detection, fire fighting, and military surveillance, and are an effective technique for collective robotic search problems.

Swarm intelligence is an emerging research area with similar population and evolution characteristics to those of genetic algorithms. Swarm intelligence is used to solve optimization and cooperative problems among intelligent agents, mainly in computer's networks, mobile robotics [3] and cooperative and/or decentralized control [4]. Swarm intelligence is inspired in nature, in the fact that contribution among living animals of a group contribute with their own experiences to the group, making it stronger in face of others. In This method, an optimization problem based on position of obstacles, and goal is designed and then PSO is used to solve the optimization problem. Every step of the algorithm, the global best position of particle is selected and the robot moves on the points in order to reach the goal. Whenever sensors detect changes in their environment or whenever the robot reaches to a local goal the local, processor of robot updates its data.

2. A Review of Previous Research

To solve the navigation problem for the robot, researchers have proposed various methods. In conventional navigation methods such as cell decomposition (Latombe, 1990) [5] and road map (Wang. 2000) [6], due to the high volume of calculations, we are not able to solve problems in complex environments. Artificial potential field method (Shi, 2009) [7], because of simplification frequently is used for local navigation. But due to stop at local minima, this method will fail. In recent years a series of intelligent ideas, such as genetic algorithms and particle swarm optimization because of the robust and ability to the Simultaneous calculations to solve the navigation

* Corresponding author. Tel.: +88-01774686877
E-mail addresses: rbn_khan@yahoo.com

problems are used. Ghorbani and colleagues (Ghorbani, 2009) [8], use the genetic algorithm for solving the problem of mobile robot navigation. Sugiwara and colleagues, (*Sugawara,* 2004) [9], used ants colony algorithm to solve the problem of navigation in a dynamic virtual environment. Qu and colleagues, (Qu, 2009) [10] used neural networks for navigation and obstacles avoid in dynamic environments. PSO, by Kennedy in 1995, based on observation of the collective behavior of certain species of animals such as birds and fish have been proposed (Eberhart, 1995& Kennedy, 1995) [11]. Due to simplicity, this method is used in robot navigation. Doctor and colleagues (*Doctor,* 2004) [12], using the PSO method for navigation an unmanned vehicle that can converge well. Chen and colleagues, (Chen, 2006) [13], suggests a soft and efficient navigation method for mobile robot using the Stochastic PSO. Qin and colleagues (Qin, 2004) [14] used the Chaotic PSO with Mutation operator for navigation and moving the robot meets the immediate needs. Hao and colleagues, (Hao, 2007) [15] proposed a method of obstacles avoiding using the PSO and polar coordinate system in a dynamic environment.

3. Particle Swarm Optimization

The proposal of such algorithm appeared from some scientists that developed computational simulations of the movement of organisms such as flocks of birds and fish schooling. Such simulations were heavily based in manipulating the distances between individuals, that is, the synchrony of the behavior of the swarm was thought as an effort to keep an optimal distance between them.

In theory, at least, individuals of a swarm may benefit from the prior discoveries and experiences of all member of the swarm when foraging. The fundamentals of developing particle swarm optimization (PSO) are a hypothesis in which the exchange of information among beings of a same species offers some sort of evolutionary advantage.

Similarly to genetic algorithms (GAs), PSO is an optimization tool based in a population, where each member is called a particle, that is, each particle is a potential solution to the analyzed problem. However, unlike GAs, PSO does not have operators, like crossover and mutation. PSO does not implement the survival of the fittest individuals; instead, it implements the simulation of social behavior.

The PSO algorithm works as follows, initially, a random position population exists, each of these particles has a speed and the particles start to “fly around” the search space. Each particle has a memory, allowing it to remember the best position it has visited in history (*pbest*), and also the fitness in that position.

The best position ever achieved by the whole swarm is denominated the global best (*gbest*). The basic concept of PSO algorithm is to accelerate the particles towards *pbest* and *gbest*, considering a random weight at each time step. Mathematically, the particles move following the equations:

$$V_{id}^{t+1} = W \times V_{idt} + c_1 \times rand_1 \times (P_{id} - X_{id}^t) + c_2 \times rand_2 \times (Pgd_{id} - X_{id}^t) \qquad (1)$$

$$X_{id}^{t+1} = X_{id}^t + V_{id}^{t+1}\Delta t \qquad (2)$$

Where $\Delta t = 1$, t represents the actual iteration and $t+1$ represents the next iteration V_{id} and X_{id} represent the particle speed and position respectively, $rand_1$ and $rand_2$ are two random numbers with uniform distribution in [0,1], used to maintain populations' diversity.

Eq. (1) is used to update each particle's speed, and Eq. (2) represents the position update, according to its previous position and its speed, considering $\Delta t = 1$.

Positive constants c_1 and c_2 are denominated cognitive and social components, respectively. These are the acceleration constants, responsible for varying the particle speed towards *pbest* and *gbest.* Constants c_1 and c_2 are not critical factors for determining the algorithm convergence; however, a correct tuning may cause the algorithm convergence to occur faster.

The use of W, called inertia weight was proposed by Shi and Eberhart (1998) [16]. This parameter is responsible for dynamically adjust the speed of the particles, so, it's responsible for balancing between local and global search, consequently, needing less iterations for the algorithm to converge. A small value of inertia weight implies in a local search, by the other side, a high value leads to a global search.

Applying a high inertia weight at the start of the algorithm and making it decay to a low value through the PSO algorithm execution, makes the algorithm globally search in the start of the search, and search locally at the end of the execution. Eq. (3) shows how the inertia weight is updated, considering $iter_{max}$ the maximum number of iterations of the algorithm and $iter$ the actual iteration.

$$W = W_{max} - \frac{W_{max} - W_{min}}{iter_{max}} \times iter \qquad (3)$$

The first step of the PSO algorithm is to start each particle with random numbers, considering that the random number must belong to the search space. Next a loop starts being executed, and it remains until the stopping criteria is met, the stopping criteria may be the convergence of the algorithm, a maximum number of iterations, or anything else. Inside the loop the value of the fitness and the *pbest* of each particle are determined. Once all particles have been analyzed, it's calculated the *gbest*, and with this value, the velocity and position of all particles is achieved.

4. Trajectory Planning

The position of the robot is represented by Cartesian coordinates such as x- and y-coordinate positions and its velocity is modified by PSO.

As the particles initiated moves through the search space for finding the next optimum position for the robot to move, the positions lying inside any obstacle should be discarded and the path, generated after choosing the next position, should not collide with any obstacle.

4.1 Objective function

The path-planning, in this study, is nothing but a constrained optimization where the obstacles represent constraints and the length of path has to be optimized (minimized) i.e. the Euclidean distance from current position to the goal is the objective function.
The positions which give the minimum value of the objective function i.e. the positions nearest to the goal point must be selected for the next move of the robot in every iteration of the algorithm.
In this study, we have used a simple objective function

$$F = \sqrt{(x_i - x_g)^2 + (y_i - y_g)^2};\ i = 1, 2, 3, \dots\dots, N \quad (4)$$

Where, N is number of particles. (x_i, y_i) is current position of i-th particle and (x_g, y_g) is co-ordinate of goal point.

4.2 Proposed Approach

The overall steps of the algorithm are as follows:

Step 1: A preset number of particles are generated around the robot's initial position and within its sensing range.

Step 2: Each particle takes a new velocity and position based on the constantly updated PSO equations.

Step 3: All the particles are checked if the lines connecting the new positions of the particles to the robot's current position intersect any obstacle; if any particle intersects, then the particle is relocated to another new position.

Step 4: A candidate for the robot's next position is determined by the position of the best particle (i.e., the one nearest to the goal). Set it as the robot's next position and go to Step 2.

Step 5: Execute Steps 2–4 until the goal is within the robot's sensing range and can be accessed via a straight line.

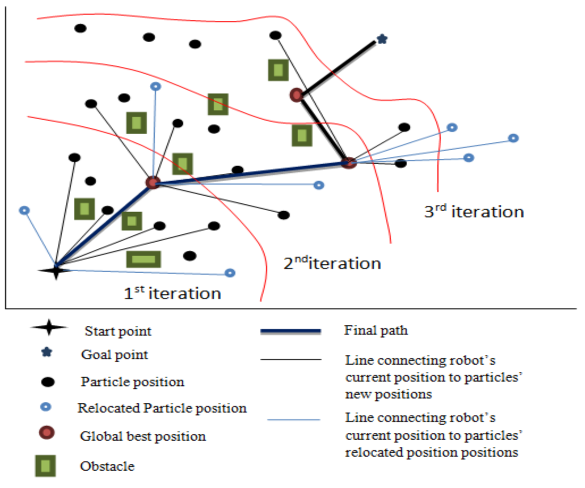
1st iteration
2nd iteration
3rd iteration
Start point
Goal point
Particle position
Relocated Particle position
Global best position
Obstacle
Final path
Line connecting robot's current position to particles' new positions
Line connecting robot's current position to particles' relocated position positions

Fig.1 Obstacles avoidance and path generation

We have used relocation method instead of penalty method for obstacle avoidance, because in penalty method, some particles will be lost but in relocation method, particles' positions are relocated, so no particles are lost.
The algorithm part for relocation of particles' positions is:

```
for i =1 to M, M is no. of obstacles do
    for j=1 to N, N is no. of particles do
        Find I, I is matrix of intersection points of
        obstacle and line connecting the new position
        of the particle to the robot's current position.
          while ( I ~= empty) do
              Find new position of the particle
                for k=1 to M do
                    Find I
                    if(I==empty )
                      Break
                    end if
                end for k
          end while
    end for j
end for i
```

5. Simulation

5.1 Settings and environment

All the experiments are executed in a Fujitsu LH531 computer and the configuration of PC was Intel(R) Pentium(R) CPU B960 @ 2.20 GHz and 2 GB RAM. The environment used for the trajectory planning is a 20x20 meters field. And as we have mentioned earlier, we have used a grid based environment, where robots can choose only integer-valued co-ordinates for the candidate points of trajectory. As for creating a dynamic environment, once we have assigned some positions to the obstacles initially before starting iterations, the obstacles move randomly in a range of (-1,1) for both x and y co-ordinates in each iteration. The PSO parameters are: maximum number of iterations 100, maximum inertia weight 0.9 and minimum inertia weight 0.4 and $c1 = c2 = 2$.

5.2 Simulation result

The MATLAB simulation results of our research work for autonomous robot path planning in dynamic and static environment is summarized in table 1 for different combinations.

Obstacle No.	Static Obstacles	Dynamic Obstacles	Swarm No.	Target No.	Path Length	Time (S)
4	4	0	30	1	26.24	2.57
8	8	0	35	1	28.64	4.71
12	12	0	40	1	27.69	4.82
12	8	4	50	1	30.56	6.45
12	0	12	50	1	32.35	6.70
12	0	12	50	2	43.26	6.9

Table 1 Simulation results for the proposed algorithm

These results for different combinations are also shown in different figures below. The figure 1 through figure 3 has been shown for static environment. The figure 4a and 4b is for partly dynamic and partly static environment and the figure 5a through figure 5d is used to show the fully dynamic environment. These upper-mentioned cases are for single goal. Finally, the figure 6 is used to show for the case of multiple goals as two goals.

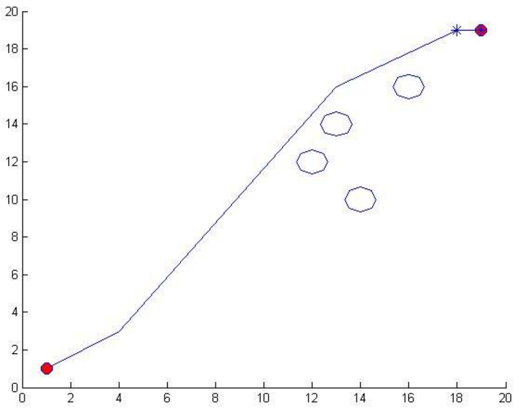

Fig.2 Path generated for the working environment using PSO (Static obstacles = 4; Dynamic obstacles = 0; Swarm size = 30; Goal = 1)

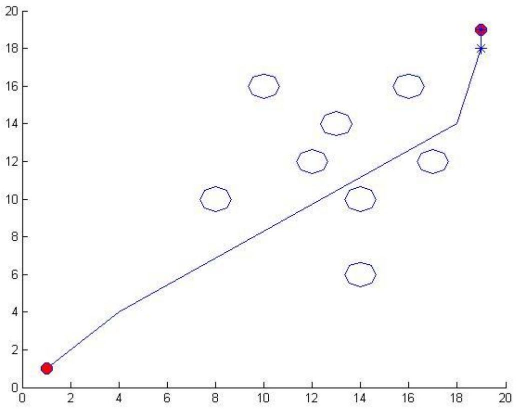

Fig.3 Path generated for the working environment using PSO (Static obstacles = 8; Dynamic obstacles = 0; Swarm size = 35; Goal = 1)

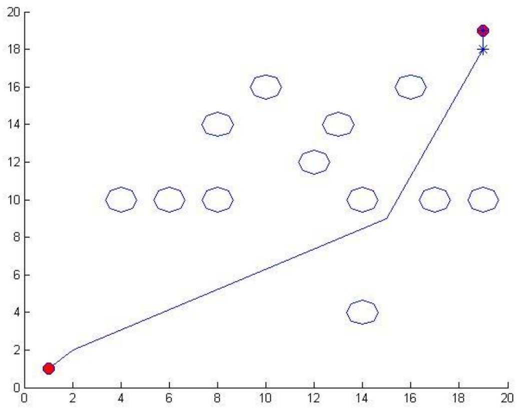

Fig.4 Path generated for the working environment using PSO (Static obstacles = 12; Dynamic obstacles = 0; Swarm size = 40; Goal = 1)

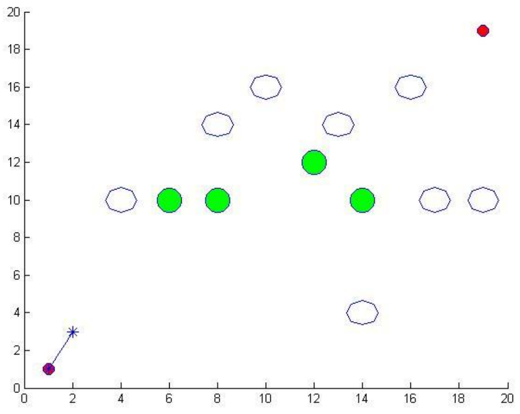

Fig.5a (Starting) Path generated for the working environment using PSO (Static obstacles = 8; Dynamic obstacles = 4; Swarm size = 50; Goal = 1)

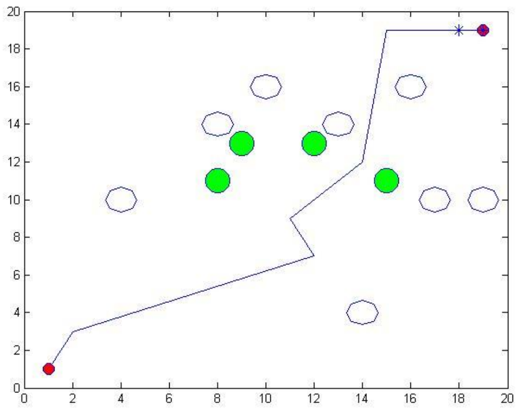

Fig.5b (End) Path generated for the working environment using PSO (Static obstacles = 8; Dynamic obstacles = 4; Swarm size = 50; Goal = 1)

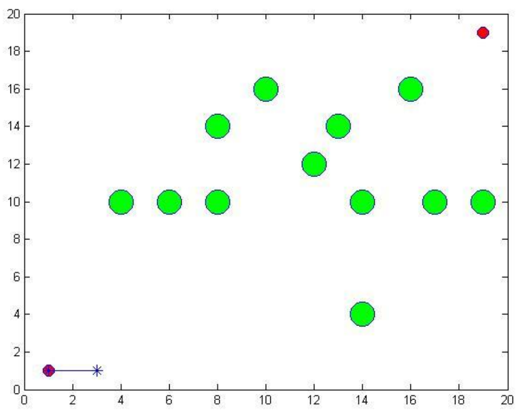

Fig.6a (Starting**)** Path generated for the working environment using PSO (Static obstacles = 12; Dynamic obstacles = 0; Swarm size = 50; Goal = 1)

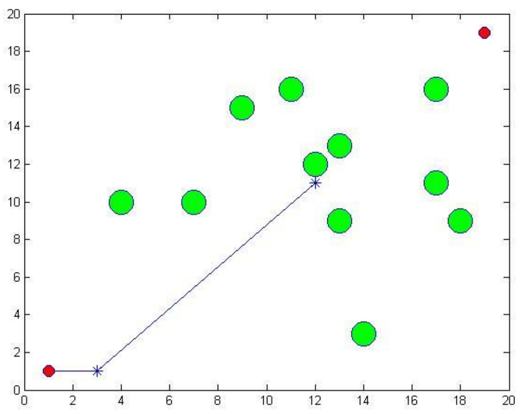

Fig.6b (Middle**)** Path generated for the working environment using PSO (Static obstacles = 12; Dynamic obstacles = 0; Swarm size = 50; Goal = 1)

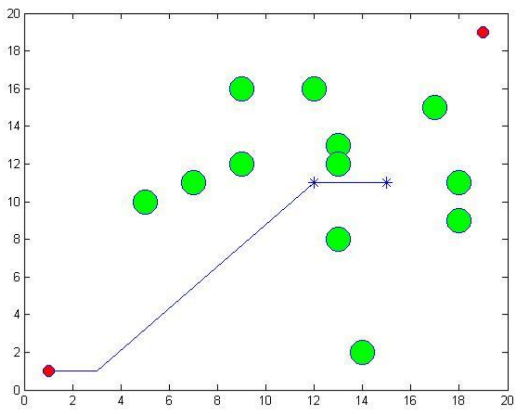

Fig.6c (Middle**)** Path generated for the working environment using PSO (Static obstacles = 12; Dynamic obstacles = 0; Swarm size = 50; Goal = 1)

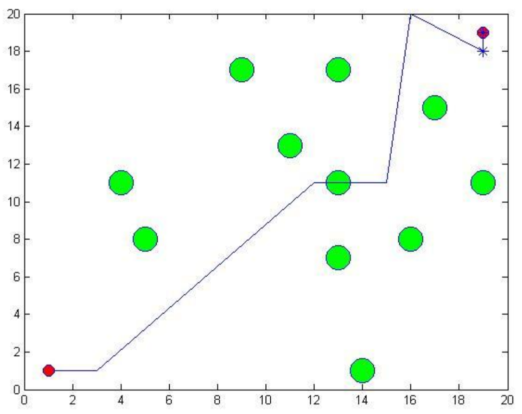

Fig.6d (End**)** Path generated for the working environment using PSO (Static obstacles = 12; Dynamic obstacles = 0; Swarm size = 50; Goal = 1)

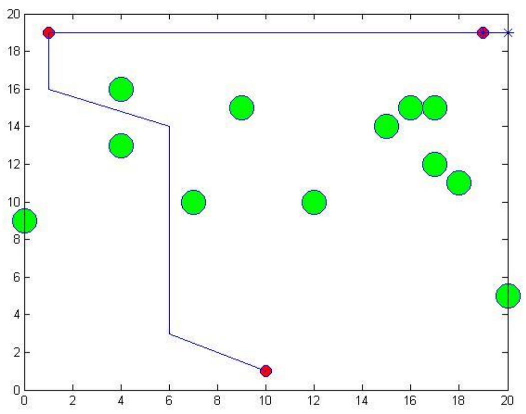

Fig.7 Path generated for the working environment using PSO (Static obstacles = 12; Dynamic obstacles = 0; Swarm size = 50; Goal = 2)

6. Conclusion & Future Works

A sensor-based path planner is presented in this paper. The proposed method is able to deal simultaneously with both global and local planning requirements. The advantages of the approach can be summarized by the fact that the trajectories obtained are smooth and safe, and at the same time, free of local traps due to the integration of the real-time sensor information in the recalculation of the path.

The method is easy to implement, the algorithm is very fast and can work online. It works in cluttered and changing environments with moving obstacles. As demonstrated along this work, the method can perform in all types of environments without restrictions in the form of the obstacles.
Obstacles in the environment by the robot sensors, in a limited radius around it are detected. The robot will come. It cannot be said with certainty that the path travelled by the robot to the global is optimum because the environ-

ment is dynamic and unknown. The proposed method is flexible, that way you can change any parameters, or control the degree of importance of avoiding or moving toward the goal. As future work we have the intention to apply other types of nature inspired algorithms to the path planning problem. We can also improve the performance by the hybridization of various nature inspired algorithms.

7. References

[1] L. Li, T. Ye, M. Tan, and X. Chen, Present state and future development of mobile robot technology research, *Robot,* vol. 24(5), pp.475-480, (2002).

[2] M. Reyes-Sierra, and C.A. Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, *Int. J. Comput. Intell.* Res. 2 (3), 287–308, (2006).

[3] Y. Liu, and K.M. Passino, Stable social foraging swarms in a noisy environment, *IEEE Transactions on Automatic Control*, vol. 49, no. 1, pp. 30-44, (2004).

[4] J.S. Baras, X. Tan, and P. Hovareshti, Decentralized control of autonomous vehicles, *Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii, USA,* pp. 1532-1537, (2003).

[5] J. C. Latombe, Robot motion planning, *Springer Verlag,* (1990).

[6] Y. Wang, Two novel approaches for unmanned underwater vehicle path planning: constrained optimization and semi-infinite constrained optimization, Robotica, vol. 18, pp. 123-14, (2000).

[7] P. Shi and Y. Zhao, An efficient path planning algorithm for mobile robot using improved potential field, *IEEE International Conference on Robotics and Biomimetics,* pp. 1704-1708, (2009).

[8] A. Ghorbani, Using Genetic Algorithm for a Mobile Robot Path Planning, *International Conference on Future Computer and Communication,* pp. 164-166, (2009).

[9] K. Sugawara, Foraging behavior of interacting robots with virtual pheromone, *Proceedings of the International Conference on Intelligent Robots and Systems,* pp. 3074-3079 vol. 3, (2004).

[10] H. Qu, Real-time robot path planning based on a modified pulse-coupled neural network model, *Neural Networks, IEEE Transactions on,* vol. 20, pp. 1724-1739, (2009).

[11] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, *in Proceedings of the Sixth International Symposium on Micro Machine and Human Science,* pp. 39-43, (1995).

[12] S. Doctor and G. Venayagamoorthy, Unmanned vehicle navigation using swarm intelligence, *in Proceedings of International Conference on Intelligent Sensing and Information Processing,* pp. 249-253, (2004).

[13] X. Chen and Y. Li, Smooth path planning of a mobile robot using stochastic particle swarm optimization, *in Proceedings of the IEEE International Conference on Mechatronics and Automation,* pp. 1722-1727, (2006).

[14] Q.Y. Qin, Path planning for mobile robot using the particle swarm optimization with mutation operator, *in Proceedings of International Conference on Machine Learning and Cybernetics, pp.* 2473-2478 vol. 4, (2004).

[15] Y. Hao, Real-Time Obstacle Avoidance Method based on Polar Coordination Particle Swarm Optimization in Dynamic Environment, *in IEEE Conference on Industrial Electronics and Applications,* pp. 1612-1617, (2007).

[16] Y. Shi, and R. C. Eberhart, Parameter selection in particle swarm optimizer, *Proceedings Seventh Annual Conference on Evolutionary Programming, V.W. Porto, N. Saravan, D. Waagen, and A.E. Eiben (eds.). Berlin: Springer-Verlag,* pp. 591-601, (1998).