2009 Third UKSim European Symposium on Computer Modeling and Simulation

Using Particle Swarm Optimization for Robot Path Planning in Dynamic Environments with Moving Obstacles and Target

Amin Zargar Nasrollahy
Computer Engineering Department
Islamic Azad University Qazvin Branch
Qazvin, Iran
e-mail: amin.zargar@qazviniau.ac.ir

Hamid Haj Seyyed Javadi
Mathematics and Computer Science Department
Shahed University of Tehran
Tehran, Iran
e-mail: h.s.javadi@shahed.ac.ir

***Abstract*— Robot path planning in known and dynamic environments is feasible for mobile robots and its main purpose is to find a collision free path for a robot from an initial position to a goal position in an environment with obstacles. In this paper the goal position is assumed to be moving over the time. Also our environment includes moving obstacles as well as static ones. We present a new approach for path planning mobile robots using Particle Swarm Optimization in order to minimize total path planning time while avoiding the local optimums. Simulation is used to validate and illustrate the approach.**

Keywords-robot path planning; trajectory planning; optimization; Particle Swarm Optimization; dynamic environments; moving obstacles; moving target

I. INTRODUCTION

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. Abundant efforts have been carried out so far to solve the basic problem of path planning. A perfect and precise path planner, which finds the path if one exists, otherwise reports no path existence, is NP-hard [6]. Classic techniques for path planning problem are general methods like Roadmap, Cell Decomposition, Potential Fields and Mathematical Programming. Most of path planning problems can be solved using these classic techniques. However, they are too computationally intensive for real-time response in a large problem space. In order to improve the efficiency of classic methods, probabilistic algorithms like PRM and RRT are proposed. For local optimum problem, many heuristic and meta-heuristic algorithms like Simulated Annealing, Genetic Algorithms and Ant Colony Optimization are used in path planning problem [7, 8, 9].

A PSO based algorithm for path planning mobile robots with mutation operator is presented in [10]. In 2005, a collision prevention approach using multi objective PSO in dynamic environment has discussed [2]. Obstacle avoidance path planning for soccer robots also has been studied in [11]. Finally a smooth path planning for mobile robots is simulated in [12].

In this paper we make use of PSO in robot path planning problem to take its capabilities in dynamic environments and large-scale domains. By using PSO as a multi-agent search technique, we expect to propose an efficient search algorithm as well as creating a small-scale model of a search system, so that it can plan a path while relocation of obstacles and the goal.

Rest of the paper is organized as follows. In section II we give a brief description of Particle Swarm Optimization algorithm. Section III discusses PSO and robot path planning problem. We present our proposed approach in section IV. We discuss how to solve the constraint path planning problem using PSO in section V. Section VI is dedicated to Experimental results and finally we conclude in section VII.

II. PARTICLE SWARM OPTIMIZATION

PSO is a new technique to deal with the problems whose solutions can be represented as a point in a *D*-dimensional solution space. PSO is initialized with a population of random particles $(X_1, X_2, \ldots, X_D)$ which distribute uniformly around search space at first. Assuming that the position and velocity of the i^{th} particle is represented by *D*-dimensional vectors $X_i=(x_{i1}, x_{i2}, \ldots, x_{iD})$ and $V_i=(v_{i1}, v_{i2}, \ldots, v_{iD})$, respectively. The best previous position (*pbest*) of the i^{th} particle is defined as $P_i=(p_{i1}, p_{i2}, \ldots, p_{iD})$, and the best position of the population (*gbest*) is denoted by $P_g=(p_{g1}, p_{g2}, \ldots, p_{gD})$. The new velocity and position are updated according to following equations:

$$\begin{cases} V_i^{k+1} = wV_i^k + c_1 r_1 (P_i - X_i^k) + c_2 r_2 (P_g - X_i^k) & (1) \\ X_i^{k+1} = X_i^k + V_i^{k+1} & (2) \end{cases}$$

where $i=1, 2, \ldots, N$ and N is the size of the population; $k=1, 2, \ldots, K$ and K is the maximum number of iterations; w is the inertia weight; c_1 and c_2 are two positive constants, usually we choose $c_1 = c_2 = 2$; r_1 and r_2 are two random functions in the range from 0 to 1. In PSO, the constraint conditions of velocity and position are:

$$-v_{\max} \le v_{id} \le v_{\max}, \quad x_{\min} \le x_{id} \le x_{\max} \qquad (3)$$

where $v_{\max}$ is the maximum velocity, which allows actually serves as a constraint that controls the maximum global exploration ability PSO can have; $x_{\min}$ and $x_{\max}$ are the lower boundary and upper boundary of the solution space. The performance of each particle is measured according to a pre-defined fitness function which is problem dependent. Each particle observes the "fitness" of itself and its neighbors and emulates successful neighbors by moving towards them. This extremely simple approach has been surprisingly effective across a variety of problem domains.

In PSO, the inertia weight w plays a considered important role, because the balance between the global and local exploration abilities is mainly controlled by the inertia weight. Therefore, the parameter w will influence the PSO's convergence behavior and choose a suitable w will help algorithm find the optimum solution accurately and rapidly. Large inertia weight at the beginning helps to find good seeds and the later small inertia weight facilitates fine search. So, a linearly decreasing inertia weight technique is developed, which linearly vary from 0.95 at the beginning of the search to 0.4 at the end. This technique has proven to be very efficient for balancing between the global and local exploration abilities. For this reason, this technique is used in our research and the inertia weight is determined by following equation:

$$w = w_{start} - \frac{w_{start} - w_{end}}{K} k \tag{4}$$

where w_{start} and w_{end} denote the start and end value of inertia weight, respectively.

The procedure of standard PSO can be summarized as follows (Algorithm 1):

```
Step 1: Initialize the size of the population $N$
        Initialize the dimension of the solution space $D$
        Initialize the maximum number of iterations $K$
        Initialize the inertia weight $w_{start}$ and $w_{end}$
Step 2: For each particle
        Initialize the particle position $X_i$ randomly
        Initialize the particle velocity $V_i$ randomly
        Initialize the current position as $P_i$
        Evaluate the fitness value
        Initialize $P_g$ according to the fitness value
Step 3: Calculate new inertia weight according to (4).
Step 4: Update velocity of each particle according to (1),
        If $v_{id} > v_{\max}$, then $v_{id} = v_{\max}$
        If $v_{id} < -v_{\max}$, then $v_{id} = -v_{\max}$
Step 5: Update position of each particle according to (2),
        If $x_{id} > x_{\max}$, then $x_{id} = x_{\max}$
        If $x_{id} < x_{\min}$, then $x_{id} = x_{\min}$
Step 6: Evaluate the fitness values of all particles. For each
        particle, compare its current fitness value with the
        fitness of its pbest. If current value is better, then
        update pbest and its fitness value. Furthermore,
        determine the best particle of current population with
        the best fitness value. If the fitness value is better
        than the fitness of gbest, then update gbest and its
        fitness value with the position and objective value of
        the current best particle.
Step 7: If the maximum number of iterations or any other
        predefined criterion is met, then stop; otherwise go
        back to Step 3.
```

III. PSO and Robot Path Planning Problem

Problem Formulation

Problem formulation for robot path planning is to determine next position of the robot according to its current position in a dynamic and known environment. Taking some set of predefined principles into consideration, here we give a common and general definition for path planning problem:

A. Pre-assumptions

1) Current position of robot is known with respect to given reference coordinate system.

2) Robot has a limited maximum footstep, V_{robot}, for which the robot cannot move further in each step.

3) There is also a maximum footstep for goal, V_{goal}, that should be smaller than that of the robot's. Otherwise, although the goal is not ever moving and moves according to a probability P_{goal} in each step, *in the worst case it can escape from the robot all the time.*

4) Every obstacle in the environment has a relocation probability of P_{obs}. Regarding to this probability, all the obstacles change their position by V_{obs}, before the planning process starts in each step. For static and immobile obstacles $P_{obs}=0$.

5) Obstacles are assumed to be circular. No two obstacles in the environment have overlaps. However, they can be adjacent.

6) Path planning algorithm runs until the robot reaches a predefined distance (threshold) from the goal.

B. Principles

The following principles have been used in the present context, satisfying the given pre-assumptions.

1) The robot always selects the shortest path for moving towards the goal.

2) If a direct path from the robot to the goal results in a collision with static and moving obstacles, robot *investigates its environment for a radius of V_{robot} in order to find the next position with a path of minimum collisions.*

3) Robot rotates in order to reduce the possible collisions with obstacles. This rotation can be any degree of the motion radius, $0 \leq \alpha < 2\pi$.

Let (x_r, y_r) be the current position of the robot at time *t*, (x'_r, y'_r) be the next position of the robot at time (*t+1*) and (x_g, y_g) be the goal position of the robot.

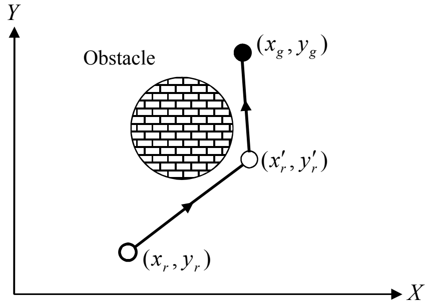


Figure 1. Selection of (x'_r, y'_r) from (x_r, y_r) to avoid collision with obstacle.

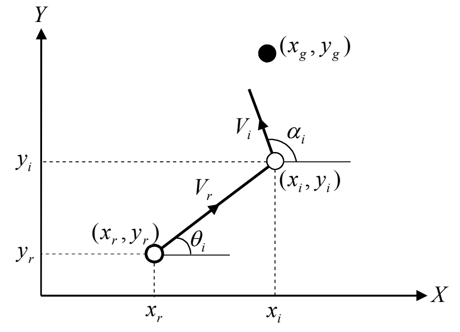
To find the intermediate position, (x'_r, y'_r) , we need to formulate the PSO algorithm properly. Hence, here we give a mapping of the particle to some element of the problem. Particle position as the solution for the robot path planning problem represents a point in a *2*-dimensional solution space. PSO is initialized with a population of random particles which distribute uniformly around (x_r, y_r) at first with random positions and velocities. Velocity vector for a particle is calculated in the same way as its position according to (6) and (7), but with x_r and y_r set to 0 and rotation angle of α_i (See Fig. 2). At the end of the algorithm's execution, we choose the particle with the best position to be the next position of the robot.

OBJECTIVE FUNCTION

Consider the robot is located at (x_r, y_r) . We need to select point (x'_r, y'_r) , i.e. next position of the robot, such that line joining $\{(x_r, y_r)(x'_r, y'_r)\}$ and $\{(x'_r, y'_r)(x_g, y_g)\}$ do not touch the obstacle in Fig. 1. This in fact is realized by PSO algorithm. Since our problem space is assumed to be continuous, we use Euclidean distance function to calculate the distance between two points in the space.

We now form a constraint that minimizes the total path length without touching the obstacle.

Let F be an objective function that determines the length of the trajectory.



$*\ V_r, V_i = rand(V_{robot});\quad \theta_i, \alpha_i = rand(2\pi)$

Figure 2. Position and velocity of the i^{th} particle populated around (x_r, y_r) .

For the particle in Fig. 2 with the position (x_i, y_i) that represents a potential solution for next position of the robot,

$$F = \sqrt{((x_r - x_i)^2 + (y_r - y_i)^2} + \sqrt{((x_i - x_g)^2 + (y_i - y_g)^2} \tag{5}$$

It can be seen from Fig. 2 that

$$x_i = x_r + V_r \cos\theta_i \tag{6}$$

$$y_i = y_r + V_r \sin\theta_i \tag{7}$$

Thus,

$$F = V_r + \sqrt{((x_r + V_r \cos\theta_i - x_g)^2 + (y_r + V_r \sin\theta_i - y_g)^2)} \tag{8}$$

To take case of static and dynamic obstacles in the environment, we add a penalty function to the objective function (8). Thus the present constraint optimization problem is transformed to

$$F = V_r + \sqrt{((x_r + V_r \cos\theta_i - x_g)^2 + (y_r + V_r \sin\theta_i - y_g)^2)} + Penalty\ (i) \tag{9}$$

Former works used static penalty functions as a positive constant that is applied when an obstacle is present on the planned trajectory regardless of the obstacle size and its position toward the so called trajectory [4]. However, in an environment where mobility of the obstacles and goal are affective in choosing a proper trajectory, not only the presence of an obstacle on a path is important, but also its size and position toward the path are to be taken into consideration while evaluating the trajectory. (Fig. 3)

In the next section we present a dynamic penalty function based on the environment state that overcomes the drawback of a static version.

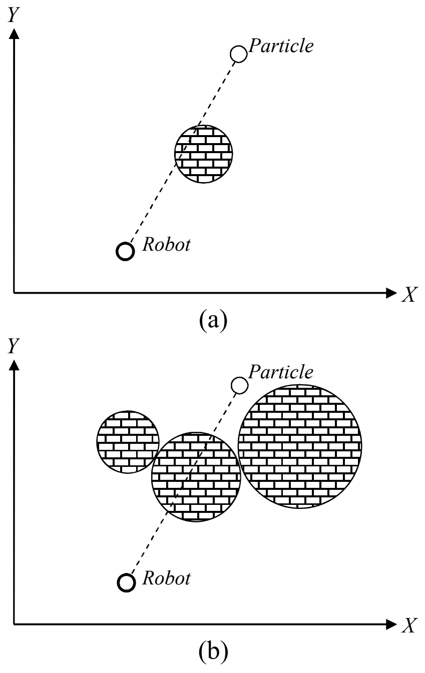


Figure 3. Obstacles with different size and placement on the direct path from robot position to particle position. (a) Single obstacle on the path. (b) Adjacent obstacles of different size.

IV. PROPOSED APPROACH

In this section we first present newly proposed penalty function that best fits the dynamic environments regarding to constrained optimization. Proposed penalty function observes the size and position of the obstacle which has blocked the trajectory and tries to find the shortest path to the goal regarding to these parameters, thus ensuring no trap in local optimum. Unlike the approach discussed in [4], this new approach does not trap in a local optimum and always finds the path if there exists.

To calculate the penalty value for a particle, first we find two intersect points of direct path from the robot to particle with the obstacle. This results in dividing the obstacle into two portions as it can be seen in Fig. 4. This implies that the robot can turn in either direction on the first intersection point to round the obstacle and reach the second one. But it is rational for the robot to select the shorter path. So the cost – length of the arc – of the both portions are calculated and the smaller one is chosen to be the penalty value. All the obstacles in the environment are examined for possible collisions with the direct path from the robot to the particle and if there is any, corresponding penalty value is calculated and added to the total penalty.

Although two obstacles can not have overlaps as discussed in pre-assumptions in section III, they can be adjacent; thus preventing the robot from breaking throught them toward the goal. For this kind of obstacles, when calculating the arc length of two portion of the blocking obstacle, circumference of the adjacent obstacle(s) is added to length of the arc for that portion. An example of this situation is shown in Fig. 5.

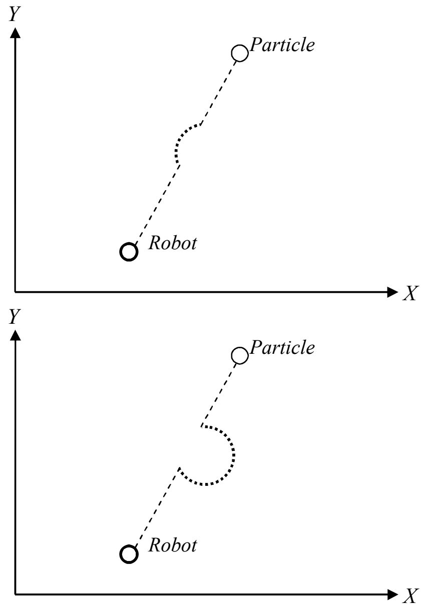


Figure 4. Two possible routes to round the obstacle in Fig. 3 (a). The route with a shorter length is selected as the penalty value for the particle.

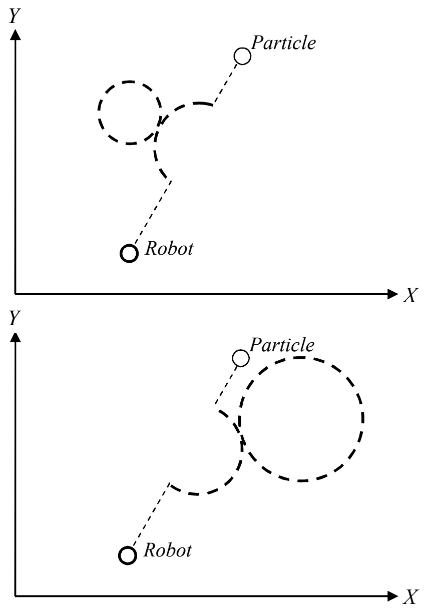


Figure 5. An example of penalty evaluation for adjacent obstacles in Fig. 3 (b). Shorter total length is selected as the penalty value for the particle.

Finally the same process is done for the path between particle and the goal and if the direct path collided with any obstacle, penalty value is calculated in the same way and added to the total penalty value of that particle.

This process can be summarized in (10), as the penalty function for the i^{th} particle,

$$Penalty\ (i) = \min\left\{ \overline{arc}_j + \sum_{s=1}^{S} Circumfere\ nce(s) \middle| j \in \{1,2\} \right\} \quad (10)$$

where S is the number of adjacent obstacles with arc_j and *Circumference* is a recursive function that calculates the circumference of the adjacent obstacle.

V. Solving The Constraint Path Planning Problem Using PSO

In this section we propose a solution to the constraint path planning problem introduced in section III by using PSO.

The proposed scheme presumes current position of robot, obstacles and the goal as well as the problem dependant parameters and determines next position of the robot by optimizing the given constraint objective function.

The procedure of path planning algorithm outlining the scheme can be summarized as follows (Algorithm 2):

```
Step 1:  Add current position of robot to the trajectory.
Step 2:  If the distance between the current position of the
         robot and the goal is less than or equal to a
         predefined threshold, go to step 6.
Step 3:  Move the obstacles and the goal according to their
         corresponding relocation probability and velocity.
Step 4:  Initialize PSO swarm around current position of the
         robot.
         Evolve PSO swarm according to Algorithm 1.
Step 5:  Choose gbest position of the swarm to be the
         current position of the robot and go back to step 1.
Step 6:  Add the goal position to the trajectory and stop.
```

VI. Simulation Results

The proposed robot path planning algorithm is implemented in MATLAB and has been tested in different environments with varying parameters. Motion trajectory for the robot has shown in black and multiple blue dots on the trajectory represent robot's position in each step of the algorithm execution. Assuming the goal to be a point in the reference coordinate system and thus modeling it in form of a point, its relocation trajectory is shown in red in the simulations.

Capturing the movements of mobile obstacles is possible by taking a snapshot of the environment in every step of the algorithm. Because of the great number of the snapshots for a simple execution of the algorithm, here we only show the last step and relocation of the obstacles are not sensible in presented experiments.

Simulation environment has considered being 100x100. The number, size and location of the obstacles toward each other have been varied in Fig. 6 and 7, whereas maximum velocity (step size) and relocation probability of the goal and obstacles have been varied in Fig. 8 and 9.

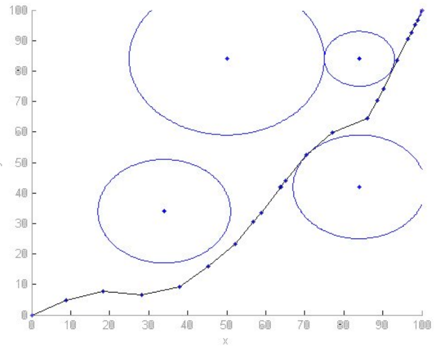


Figure 6. For the robot, V_{robot}=10. For every obstacle, P_{obs}=0. For the goal, P_{goal}=0.

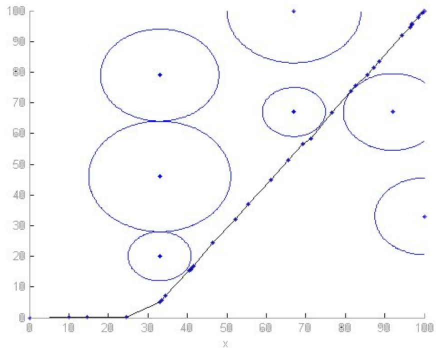


Figure 7. Three adjacent obstacles forming a single obstacle. This obstacle is passed by the proposed penalty function. All the parameters are the same as in Fig. 6.

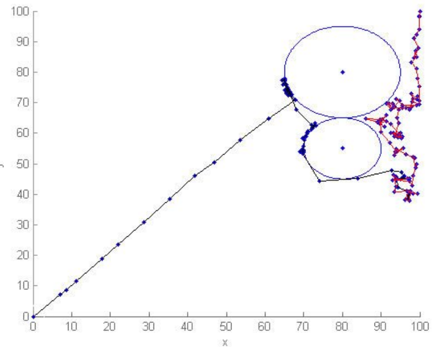


Figure 8. For the robot, V_{robot}=10. For every obstacle, P_{obs}=0. For the goal, P_{goal}=1 and V_{goal}=5.

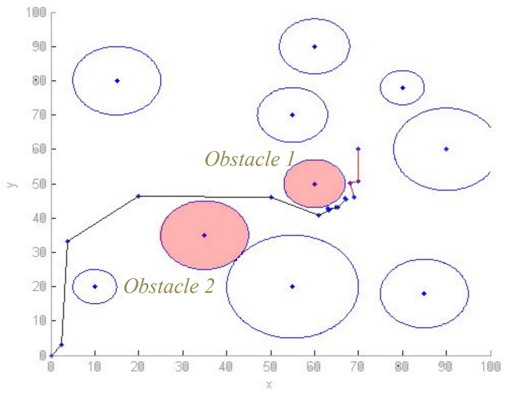


Figure 9. For the robot, V_{robot}=30. For the obstacles, V_{obs}=5; P^1_{obs}=0.4 and P^2_{obs}=0.7. For the goal, P_{goal}=0.3 and V_{goal}=10.

VII. Conclusion

In this paper we proposed a novel PSO based approach for robot path planning problem in dynamic and known environments with moving obstacles and target in order to minimize the total cost. This approach is simple and at the same time efficient in both static and dynamic environments. Although in this work we assume the obstacles to be circular shape, the proposed method can be applied to any form of obstacles regardless of their shape and size.

Also, by extending the algorithm, we can apply it to a multi-robot scenario with multiple goals in the environment. Experimental results show that our approach outperforms the similar works in escaping from local optimal solutions.

References

[1] Ellipse Masehian; Davoud Sedighizadeh, "Classic and heuristic approaches in robot motion planning-A chronological review", *Proceedings of world academy of Science, Engineering and Technology*, August 2007, pp. 1307-6884.

[2] Hua-Qing M.; Jin-Hui Z.; Xi-Jing Z.; "Obstacle avoidance with multiobjective optimization by PSO in dynamic environment", *Proc. Int. Conf. Machine Learning and Cybernetics*, Vol. 5, (2005) pp. 2950-2956.

[3] Guo Yi-nan; Yang Mei, "A novel knowledge included path planning strategy for the mobile robots", *International Journal of Computer Science and Network Security*, May 2008.

[4] Jayasree Chakraborty; Saswati Saha, "Co-operative multi robot path planning using particle swarm optimization", 2005.

[5] Da-Qing Guo; Yong-Jin Zhao; Hui Xiong; and Xiao Li, "A new class of hybrid particle swarm optimization algorithm", *Journal of Electronic Science and Technology of CHINA*, VOL. 5, NO. 2, June 2007, pp. 149-152.

[6] J. Canny, "The complexity of robot motion planning", MIT Press, Cambridge, MA, 1988.

[7] Qidan Z.; Yongjie Y. and Zhuoyi X. "Robot path planning based on artificial potential field approach with simulated annealing" *Proc. ISDA'06*, (2006) pp. 622-627.

[8] Qing L.; Xinhai T.; Sijiang X. and Yingchun Z.; "Optimum path planning for mobile robots based on a hybrid genetic algorithm" *In Proc. HIS'06*. (2006) pp. 53-58

[9] Shirong Liu; Linbo Mao and Jinshou Yu; "Path planning based on Ant Colony algorithm and distributed local navigation for multi-robot systems" *Proc. IEEE Int. Conf. on Mechatronics and Automation*,(2006) pp. 1733 – 1738.

[10] Yuan-Qing Q.; De-Bao S.; Ning L. and Yi-Gang C.; "Path planning for mobile robot using the particle swarm optimization with mutation operator", *Proc. Int. Conf. on Machine Learning and Cybernetics*, (2004) pp. 2473 – 2478.

[11] Li W.; Yushu L.; Hongbin D. and Yuanqing X.; "Obstacle-avoidance path planning for soccer robots using particle swarm optimization", *Proc. IEEE Int. Conf. on Rob. and Biomimetics* (ROBIO '06). (2006) pp. 1233- 1238.

[12] Xin C. and Yangmin L.; "Smooth path planning of a mobile robot using stochastic particle swarm optimization", *Proc. IEEE on Mechatronics and Aut.*, (2006) pp. 1722-1727.