Mobile Robot Path Planning in Environments Cluttered with Non-convex Obstacles Using Particle Swarm Optimization

Muhammad Shahab Alam
Department of Mechatronics Engineering
Air University
Islamabad, Pakistan
e-mail: shahaba@hotmail.com

Muhammad Usman Rafique
Department of Mechatronics Engineering
Air University
Islamabad, Pakistan
e-mail: m.usman694@gmail.com

***Abstract*—Generally workspaces of mobile robots are cluttered with obstacles of different sizes and shapes. Majority of the path planning algorithms get stuck in non-convex obstacles pertaining to local minima. Particle Swarm Optimization (PSO) is by comparison simple and readily intelligible yet a very powerful optimization technique which makes it an apt choice for path finding problems in complex environments. This paper presents a particle swarm optimization based path planning algorithm developed for finding a shortest collision-free path for a mobile robot in an environment strewed with non-convex obstacles. The proposed method uses random sampling and finds the optimal path while avoiding non-convex obstacles without exhaustive search. Detailed simulation results show the functionality and effectiveness of the proposed algorithm in different scenarios.**

Keywords-mobile robot; path planning; particle swarm optimization; non-convex obstacles

I. Introduction

Path planning is an elemental component of robotics for the reason that it empowers a robot to maneuver and navigate autonomously. The primary objective of path planning is to compute an optimum collision-free route for a robot from a start point to a goal point among obstacles having different shapes and sizes. Motion planning problem can be classified as PSPACE-hard and hence NP-hard (non-deterministic polynomial time) [1], [2] problem as the memory and computational time required for solving such problems increases at an exponential rate with the increase in size or dimension of the problem.

Research on motion planning started decades ago (e.g. Nilsson's work in late 1960's) [3], [4], however, the idea of configuration space introduced by Lozano-Perez and Wesley [5] in 1979 marked the foundation of modern motion planning problem. Numerous motion planning techniques have been introduced thus far, such as Voronoi Diagrams, Visibility Graphs, Sampling-based algorithms [6], [7], Potential Fields [8] etc. All these techniques are endowed with some merits and demerits. Some common problems that these techniques exhibit are high run time, high computational cost, high memory requirement, local minima problem, poor efficiency in complex or dynamic environments [9], [10].

In the recent years, researchers have focused attention on the use of heuristic techniques for overcoming these limitations and problems in path finding problem. Heuristic techniques offer many advantages over the classical path planning approaches such as easy implementation, and fast generation of acceptable solution if there exists one. Particle swarm optimization is a Swarm Intelligence based heuristic optimization technique which has proved to be very effective in many multi-dimensional optimization problems [11]. As compared to other heuristic techniques Particle Swarm Optimization is simple but very powerful and efficient optimization technique with many advantages such as simplicity, fast convergence and few parameters that are required to be tuned [12]. As mentioned earlier, one of the major problems encountered in path planning is getting trapped in local minima, which stops the robot from reaching its goal. In robot path planning such problems mostly arise due to the presence of non-convex obstacles in the search space.

Researchers have addressed the local minima problem in different ways, such as the concept of virtual obstacle [13] [14] and the use of hybrid algorithms which are the combinations of two or more algorithms [15]-[17]. These methods become cumbersome and difficult to implement correctly in complex environments hence making them error-prone. Some existing methods utilize PSO for path planning e.g. [18] and [19]. But these methods don't address the problem of local minima. Our tests also indicate that these methods get stuck in local minima.

This paper presents a simple path planning approach based on PSO which can efficiently avoid non-convex obstacles and hence escape local minima. Simulation results show the feasibility and effectiveness of the proposed algorithm in avoiding non-convex obstacles.

The remaining paper is organized as follows: Section II briefly explains Particle Swarm Optimization, Section III defines the problem formulation, Section IV describes the proposed path planning algorithm, Section V demonstrates the simulation results and finally Section VI presents the conclusion and future recommendations.

II. Particle Swarm Optimization

Particle Swarm Optimization was developed by James Kennedy and Russell Eberhart in 1995 [20]. PSO is a population based stochastic optimization technique which basically mimics the social behavior of colonies of ants & bees, flocks of birds and schools of fish. The word particle

978-1-4673-7523-8/15/$31.00 ©2015 IEEE

in PSO actually denotes an ant, a bird or a fish searching for food. When a bird or an ant looking for food finds a good path to the food, it instantly transmits the information to the whole swarm and hence rest of the swarm slowly and gradually gravitates towards the food (target).

In PSO, all particles possess two characteristics i.e. position and velocity. A swarm of particles is initialized by giving each particle a random position and velocity. These particles are placed in the search space of some problem or function. The fitness function is evaluated with these particles and personal best of each particle is stored in Pbest and global best of the whole swarm is stored in Gbest. In the next iteration these particles are then moved to new positions by using PSO's two basic update equations i.e. velocity update and position update equation. The particles gradually reach the global best positions by communicating the personal best and global best positions to each other. This process is repeated until maximum number of iterations is achieved or until all the particles converge to the same point. Positions and velocities of the particles are initialized with uniform random numbers from $[X_{min}, X_{max}]$ and $[V_{min}, V_{max}]$ respectively.

PSO uses the following two equations for updating the velocity and position of each particle:

$$V_i = \omega V_{i-1} + c_1 r_1 (P_{best} - X_{i-1}) + c_2 r_2 (G_{best} - X_{i-1}) \quad (1)$$

$$X_i = X_{i-1} + V_i \quad (2)$$

Where c_1 and c_2 represent the cognitive and social learning rates, respectively, and their values are usually taken to be equal to 2 [21]. r_1 and r_2 indicate random numbers uniformly distributed between [0-1]. The parameters c_1 and c_2 denote the relative importance of the particle's own best position to its neighbor's best position. ω is the inertia weight factor which is used for improving the search stability. For promoting global exploration of the swarm a larger value of ω is used and for promoting local exploration a smaller value of ω is preferred. A balance between local and global exploration can be achieved by a commonly used linearly decreasing inertia weight strategy with values of $\omega_{max} = 0.9$ and $\omega_{min} = 0.4$ [21].

PSO can be implemented using the following procedure:

Algorithm 1: PSO Process

for each particle *i (i=1 to n)* **do**
    Initialize position x_i
    Initialize velocity v_i
end for
repeat
    for each particle *i (i=1 to n)* **do**
        Evaluate fitness $f(x_i)$
        Update position x_i using (1)
        Update velocity v_i using (2)
        Update *Pbest* & *Gbest*
    end for
until termination criterion met

III. Problem Formulation

Robot path planning is a very important constituent of robotics as robots need to maneuver in workspaces for performing different tasks.

Suppose $\mathcal{A}$ is a rigid robot that moves in a 2D or 3D Euclidean workspace $\mathcal{W}$ ($\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$). Let the workspace $\mathcal{W}$ be populated with rigid obstacles $\mathcal{O}$, where $\mathcal{O}$ denotes the set of all points in $\mathcal{W}$ that lie inside the obstacles i.e. $\mathcal{O} \subseteq \mathcal{W}$. $\mathcal{A}$ can move in the workspace $\mathcal{W}$, whereas $\mathcal{O}$ remains fixed. Given a start and goal position of $\mathcal{A}$ in $\mathcal{W}$, the objective of path planning is to generate a shortest collision-free path between the start and goal position while avoiding any contact with $\mathcal{O}$. Report failure if desired path does not exist.

A. Workspace Modelling

The robot and obstacles from the real world are explicitly transformed into the configuration space (C-Space). A circular mobile robot with radius r is considered such that the robot can translate in the search space without any rotation. The robot is allowed to move in a two-dimensional Euclidean space with static convex and non-convex obstacles. The shaded polygonal shapes represent obstacles and the white area represents free space for robot movement as shown in Fig. 1.

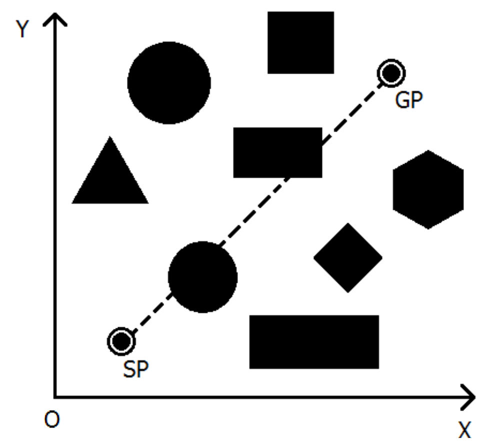

Figure 1. Mobile Robot Workspace

Let SP and GP be the start and goal point of the robot respectively in the global co-ordinate system XOY. For converting the two-dimensional data of the cost function to one-dimensional data, and thereby accelerating the search speed of the algorithm, a new coordinate system $X'OY'$ is built by performing a coordinate transformation. The X' axis of the new coordinate system $X'OY'$ is made to coincide with the line joining the start point SP and goal point GP by performing a rotation by θ, followed by a translation by x_t, y_t using:

$$\begin{pmatrix} cos\theta & sin\theta & -x_t \\ -sin\theta & cos\theta & -y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} xcos\theta + ysin\theta - x_t \\ -xsin\theta + ycos\theta - y_t \\ 1 \end{pmatrix}$$

In order to reach the goal point GP, the robot will need to go through a series of waypoints wp_n. By using the idea proposed in [22] first the straight line joining the start and goal point $\overline{SP - GP}$ is equally divided by $n + 1$, where n represents the desired number of waypoints for the robot.

$$Np = d(SP, GP)/n + 1 \tag{3}$$

Then n numbers of perpendicular lines are drawn on X'-axis, each placed at a distance Np, to get a set of parallel lines $\lambda_1, \lambda_2, \ldots, \lambda_n$ as shown in Fig. 2.

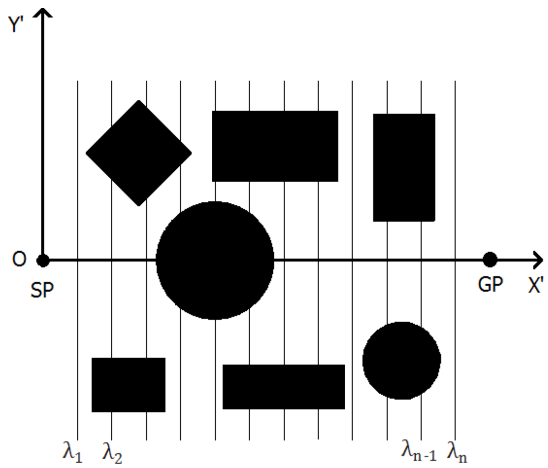

Figure 2. Workspace Modeling

Then random sampling is performed on each perpendicular line λ_i in order to get an optimal set of waypoints which are required for navigating the robot from SP to GP while satisfying the desired performance criteria. The set of waypoints giving the robot path RP is given as:

$$RP = \{SP, wp_1, wp_2, \ldots, wp_n, GP\} \tag{4}$$

The waypoints representing the coordinates of the robot are transformed back to the global coordinate system XOY by using the following transformation matrix:

$$\begin{pmatrix} cos\theta & -sin\theta & x_t \\ sin\theta & cos\theta & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} xcos\theta - ysin\theta + x_t \\ xsin\theta + ycos\theta + y_t \\ 1 \end{pmatrix}$$

B. *Objective Function*

The objective of path planning is to generate a path or a set of waypoints for a robot from an initial position to a goal position in an environment populated with obstacles while satisfying certain optimization criterion such as shortest distance, minimum time, minimum energy consumption and maximum safety etc.

The performance criterion considered in this research paper is minimizing the path length between start and goal points. The distance between two points in Euclidean n-space is given by Pythagorean Theorem. Hence for seeking a shortest path from $SP(x_{SP}, y_{SP})$ to $GP(x_{GP}, y_{GP})$ via n waypoints (wp) for a robot in a 2-dimensional plane can be achieved by the following objective function δ:

$$\delta(SP, GP) = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{5}$$

Where

$$\Delta x = \textstyle\sum_{i=1}^{n}(x_i - x_{GP}) \qquad \Delta y = \sum_{i=1}^{n}(y_i - y_{GP})$$

But since the abscissae of all the waypoints (wp) have been computed as explained in Section II, the objective function can be reformulated as:

$$\delta(SP, GP) = \sqrt{(\textstyle\sum_{i=1}^{n}(d(SP, GP)/n + 1))^2 + (\Delta y)^2} \tag{6}$$

Another term β is added to the objective function δ in order to avoid the discontinuity in the path which arises when at any point the distance between GP and a waypoint wp_i turns out to be equal on either side of an obstacle

$$\beta = \sqrt{\textstyle\sum_{i=1}^{n}(wp_{i-1} - wp_i)} \tag{7}$$

Hence the objective function to be minimized becomes:

$$\delta(SP, GP) = \beta + \sqrt{(\textstyle\sum_{i=1}^{n}(d(SP, GP)/n + 1))^2 + (\Delta y)^2} \tag{8}$$

IV. PATH PLANNER

To solve the problem encountered when a non-convex obstacle obstructs a mobile robot, a novel method is proposed in this paper. After defining the start and goal point of the robot, a translation followed by a rotation of the system is performed using the transformation matrix. Then the line segment joining the start and goal point is equally divided by the desired number of waypoints for generating a set of lines $\lambda_1, \lambda_2, \ldots, \lambda_n$. The proposed planner then performs random sampling on each λ and selects the optimal waypoint as per optimization criteria using Algorithm 1, as shown by green circles in Fig. 3. As the distance between two points is the length of the line segment that connects them, so an *allowable distance* Ω is defined between two successive waypoints. After the selection of every optimal waypoint, the planner performs a check by comparing Ω with the distance d between the selected waypoints wp_i and wp_{i-1}.

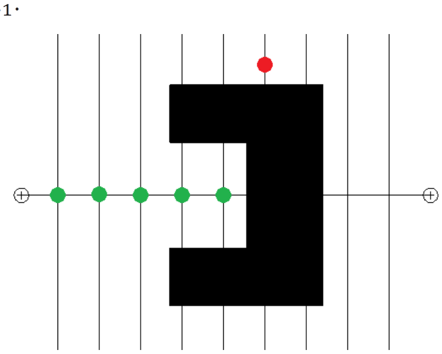

Figure 3. Waypoints Selection in U-Shaped Obstacle

The algorithm proceeds and keeps on computing next optimal waypoints until the goal point is reached. If at any iteration i the value of d turns out to be greater than Ω, this discontinuity implies there is a local minima. A recursive approach is used to cater this problem. The algorithm stores the actual goal point in $aGP$and declares wp_{i-1} and wp_i as temporary start point tSP and temporary goal point tGP respectively. The algorithm repeats the same steps i.e. performs transformation and creates segments of equal length between the temporary start and goal points and then computes the optimal waypoints between tSP and tGP. On reaching tGP the algorithm restores original goal position and starts moving towards the actual goal position aGP. The algorithm runs recursively until the robot reaches the actual goal point.

The steps involved in implementing the proposed planner are illustrated as follows:

Algorithm 2: Path Planner

Input: SP, GP, n, Ω, Coordinates of Obstacles
Output: Path (Waypoints) from SP to GP

01: Set $flag = 0$
02: Perform coordinate transformation
03: Divide the line $\overline{SP - GP}$ into $n + 1$ segments
04: Generate n vertical lines λ_n on $\overline{SP - GP}$
05: **repeat**
06: | **for** $i = 1$ to n do
07: | | Compute optimal wp_i using Algorithm 1
08: | | Compute distance d between wp_{i-1} and wp_i
09: | | **if** $d > \Omega$
10: | | | Set $aGP = GP$
11: | | | Set $tSP = wp_{i-1}$ and $tGP = wp_i$
12: | | | Set $SP = tSP$ and $GP = tGP$
13: | | | Set $flag = 1$
14: | | | break for & go to 02
15: | | **end if**
16: | **end for**
17: | **if** $flag == 1$
18: | | **if** $wp_i == GP$
19: | | | Set $SP = tGP$ and $GP = aGP$
20: | | | Set $flag = 0$ & go to 02
21: | | **end if**
22: | **end if**
23: **until** GP is reached

V. SIMULATION RESULTS

Simulations of the proposed algorithm are carried out on 1.9 GHz (4 CPUs) Intel® CPU, 2048MB RAM using MATLAB 7.0 R2009b. The following parameter settings are selected: Swarm Size $N = 500$, Maximum Iterations $it_{max} = 100$, Maximum Inertia Weight $\omega_{max} = 0.9$ and Minimum Inertia Weight $\omega_{min} = 0.4$, Maximum Velocity $V_{max} = 200$ and Minimum Velocity $V_{min} = 0$, Social Learning Factor $c_1 = 2$ and Cognitive Learning Factor $c_2 = 2$, Number of Waypoints $n = 100$, Maximum Allowable Distance $\Omega = 1.5$.

A. *Enviornment 1*

Environment 1 is populated with nine static convex obstacles. $SP(0,0)$ and $GP(9,9.5)$ are taken to be the robot's start point and goal point respectively. The green line shown in Fig. 4 represents the optimum path generated by the proposed algorithm.

B. *Enviornment 2*

Environment 2 is populated with static non-convex obstacle. $SP(0,0)$ and $GP(5.7,5.1)$ are assumed to be the robot's initial point and goal point respectively. The green line shown in Fig. 5 represents the optimum path generated by the proposed algorithm.

C. *Enviornment 3*

Environment 3 is populated with a U-shaped non-convex obstacle. The robot's initial point and goal point are assumed to be $SP(0,0)$ and $GP(11,0)$ respectively. The optimum path generated by the proposed algorithm is shown in Fig. 6 by the green line.

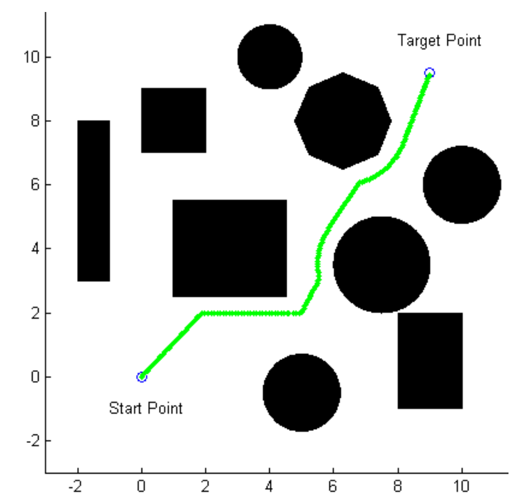

Figure 4. Environment 1

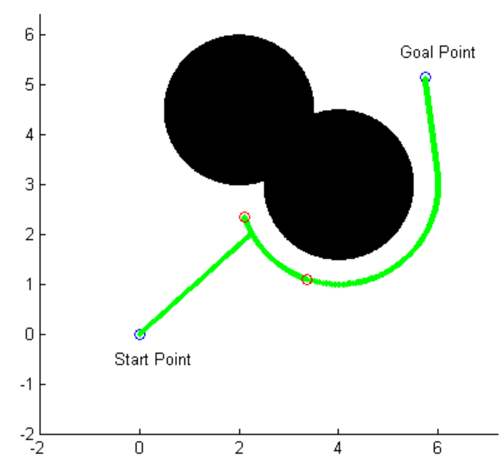

Figure 5. Environment 2

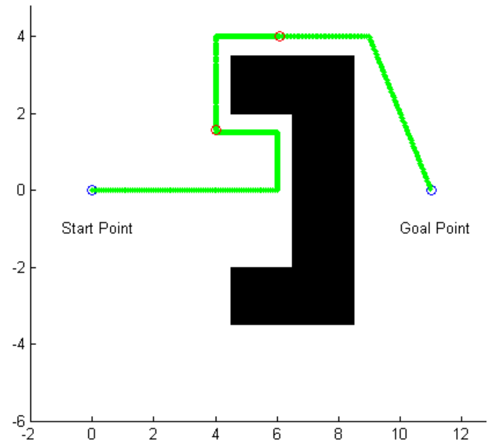

Figure 6. Environment 3

VI. CONCLUSION

In this research study, robot path planning is studied as an optimization problem and a simple algorithm based on PSO is employed for path planning and avoiding non-convex obstacles. The algorithm performs random sampling and finds the optimal path to goal point without exhaustive search. When a robot gets stuck in non-convex obstacle and hence falls into local minima, a random point is selected based on a predefined allowable distance between two successive waypoints. This randomly selected point acts as virtual or temporary sub-goal and the robot starts moving towards it. When the robot reaches this temporary sub-goal it is replaced with the actual goal point and hence the robot again starts moving towards the actual goal point. The simulation results show that the proposed method efficiently avoids non-convex obstacles i.e. evades local minima and finds an optimum path from an initial point to a given goal point without exhaustive search and huge computation.

As dynamic obstacles were not considered in this research study, so we intend to extend our research work to dynamic environments with non-convex obstacles in future. Another interesting study can be extension of this method in higher degrees of freedom.

REFERENCES

[1] J. H. Reif, "Complexity of the mover's problem and generalizations extended abstract," In *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, pp. 421-427. 1979.

[2] J. F. Canny, "*The complexity of robot motion planning*," The MIT press, Cambridge, Mass., 1988.

[3] N. J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," In *1st International Conference on Artificial Intelligence*, pages 509–520, 1969.

[4] R. E. Fikes, and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial intelligence* 2, no. 3 (1972): 189-208.

[5] T. Lozano-Pérez, and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*22, no. 10 (1979): 560-570.

[6] L. E. Kavraki, P. Svestka, J-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on* 12, no. 4 (1996): 566-580.

[7] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," (1998).

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research* 5, no. 1 (1986): 90-98.

[9] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[10] H. Choset, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[11] J. Kennedy, "Particle swarm optimization," In *Encyclopedia of Machine Learning*, pp. 760-766. Springer US, 2010.

[12] A. Abraham, and L. Jain, *Evolutionary multiobjective optimization*. Springer London, 2005.

[13] L. Chengqing, M. H. Ang, H. Krishnan, and L. S. Yong, "Virtual obstacle concept for local-minimum-recovery in potential-field based navigation," In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, pp. 983-988. IEEE, 2000.

[14] M. C. Lee, and M. G. Park, "Artificial potential field based path planning for mobile robots using a virtual obstacle concept," In *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on*, vol. 2, pp. 735-740. IEEE, 2003.

[15] Y-Q. Qin, D-B. Sun, N. Li, and Y-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 4, pp. 2473-2478. IEEE, 2004.

[16] G. Masuyama, A. Yamashita, and H. Asama, "Robot motion planning utilizing local propagation of information based on Particle Swarm and its internal parameters," In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pp. 1053-1058. IEEE, 2011.

[17] H-C. Huang, and C-C. Tsai, "Global path planning for autonomous robot navigation using hybrid metaheuristic GA-PSO algorithm," In *SICE Annual Conference (SICE), 2011 Proceedings of*, pp. 1338-1343. IEEE, 2011.

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

[19] M. Yang, and C-Z. Li, "Path planing and tracking for multi-robot system based on improved PSO algorithm," In *Mechatronic Science,Electric Engineering and Computer(MEC), 2011 International Conference on,* pp. 1667-1670. IEEE, 2011.

[20] R. C. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory," In *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1, pp. 39-43. 1995.

[21] R. C. Eberhart, and Y. Shi, "Particle swarm optimization: developments, applications and resources," In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1, pp. 81-86. IEEE, 2001.

[22] W-G. Han, S-M. Baek, and T-Y. Kuc, "Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots," In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 3, pp. 2747-2751. IEEE, 1997.