Intelligent Route Planning for Multiple Robots using Particle Swarm Optimization

Sai Krishna Mulpuru
Dept. of Electronics and Communication Engg.
National Institute of Technology
Warangal, India
saimulpuru@ieee.org

Krishna Chaitanya Kollu
Dept. of Electronics and Communication Engg.
National Institute of Technology
Warangal, India
krishna.kollu@gmail.com

***Abstract*- In this paper, we propose a new method of route planning for multiple robots in an unknown space with randomly placed obstacles. First, we use the basic principles of Multi Agent Systems and Particle Swarm Optimization algorithm. At the core, we implement our proposed method of circularly drifting the robot around the corners of the obstacle at a desired angle which gives us the freedom in designing the robot's angular mobility. Also, communication between robots about the obstacles and the traversed space helps in obtaining their next positions. Then the optimal route planned is shown till the robots safely reach the target without any collisions. Simulations are performed by avoiding obstacles using circular drift method.**

I. INTRODUCTION

Current trend in the fully autonomous world is the use of communicating robots with incorporated intelligence. They are being used in controlling nanobots within the body for the purpose of killing cancer tumors, self assembly, interferometry, and mine detection in military warfare [1], planetary mapping and other various applications in day to day life. The robot manufacture is also an expensive task. Hence, for a sensitive and precise control of these robots, there is a need of more sophisticated methods of control.

Intelligent route planning deals with generating an optimized route in an unknown environment by drifting away from the obstacles in the search space. Many algorithms in Multi Agent Systems like artificial potential field [2], particle swarm optimization [3], genetic algorithm [4], and flocking algorithm [5] have been proposed for solving this problem. But most of these algorithms work well for the robots traversing a known environment. The actual implementation of an efficient algorithm like Particle Swarm Optimization is required when robots need to avoid the randomly placed obstacles in space and reach the target point [6]. Here we only have basic information about the environment like the entire area size and the final target. Hence real time planning using coordination among the robots about their surrounding environment information is necessary. However, ordinary methods of obstacle avoidance have not proven good results on route planning.

. The problem deals with a number of robots deployed in an unknown environment reaching their target by avoiding obstacles encountered on their way. Here, we deploy a set of robots at a corner of the space from where they start moving towards the target. In this process, they broadcast the information about their surroundings continuously to other robots. A circular drift function is used here to effectively avoid collisions of robots with the obstacles. The overview of implementation shown in Fig.1 contains robots deployment, obstacle detection, obstacle avoidance and route planning.

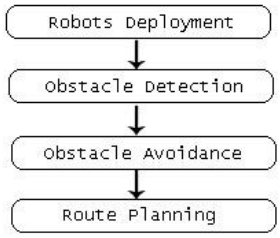

Fig. 1 Implementation Overview

The remaining paper is organized in the following manner: Section II gives an overview of Particle Swarm Optimization algorithm. Section III describes the proposed circular drift method for avoiding obstacles. In Sections IV and V, we compare the simulations with and without using circular drift technique. Lastly, Section VI consists of conclusions and future work.

II. AN OVERVIEW OF PARTICLE SWARM OPTIMIZATION

Particle swarm algorithm imitates human or insects' social behavior. The approach is derived in part from the interesting way flocks of birds and swarms in nature search for food [7]. It uses a number of particles that constitute a swarm traversing in space searching for their target. The particles continuously interact with one another while learning from their own experience. Gradually, all the particles, using a good technique move into better regions of the problem space. The basic concept of PSO lies in guiding each particle towards the resultant of its best position and global best position [6]. Global best position is the position of the particle having the best fitness value among all the particles.

Here fitness is calculated using the Euclidian distance function [8]-[9]. As the particle gets closer to the target, the distance to the target location decreases, thereby decreasing the value of fitness. Among the swarm, the particle with the least fitness is considered as the global best particle as it is closest to the target. The swarm is said to have accomplished the task if all the particles in it have acquired fitness less than or equal to half the range of sensors incorporated in the robot.

978-0-7695-3892-1/09 $26.00 © 2009 IEEE
DOI 10.1109/ICCTD.2009.32

In the particle swarm optimization algorithm, we perform the following actions:

(i) Particles are initialized with random position and velocity vectors.

(ii) Fitness is evaluated for every particle at its current position using Euclidian distance as the fitness function [6].

$$fitness(p)= \sqrt{(Loc_x - P_x)^2 + (Loc_y - P_y)^2} \quad (1)$$

Where (Loc_x, Loc_y) is the target location and (P_x, P_y) is the robot's current position.

(iii) If the fitness of the particle is greater than that of the best particle, then the particle would be the best particle for the next move, and the fitness of this particle is taken as best fitness.

(iv) Each particle is made to modify its current position, current velocity, the distance between current position and pbest, the distance between current position and gbest [6].

$$v(k+1)= inertia \times v(k) + cf \times rand(\,) \times (P_{lb}(k)-P(k)) + cf \times rand(\,) \times (P_{gb}(k)-P(k)) \quad (2)$$

$$P(k+1) = P(k) + v(k+1) \quad (3)$$

Where v(k+1) is next velocity, v(k) is current velocity, *cf* is the correction factor, $P_{lb}(k)$ is the particle's best position, $P_{gb}(k)$ is the global best position, $P(k)$ is particle's current position, $P(k+1)$ the particle's next position.

(v) If the next position of the particle is inside the obstacle, then the particle chooses a new position, otherwise, the same algorithm is continued.

(vi) This process is repeated in iterations, until all the particles communicate with each other and generate a route to the target location.

III. Circular Drift Method

A. Advantage

First, at every position, the robot checks whether its next position falls within the boundary area of any random obstacle. If this happens, the next position of the robot is calculated using a circular drift function to drift the robot around a corner of the obstacle at a specified angular value. This gives a clear picture on the route that is planned near the obstacles.

B. Algorithm

The input arguments of the circular drift function are the coordinates of the corners of the obstacle that fall in the sensor range of the robot, current position of the robot and the angle of drift. The maximum velocity of robot is selected in such a way that it is less than or equal to half of the sensor range as in Equation 4.

$$V(k) = 0.5 * sensor\ range \quad (4)$$

Now, the corner of the obstacle, which is nearer to the robot, is taken as the centre. The distance between the robot and the nearest corner is taken as the radius. Using these parameters a virtual circle is drawn.

Two points are possible on the circle, each on either side of the robot such that the arc between one of these points and the robot subtends an angle θ at the centre of the circle. One of these points becomes the next position of the robot depending on its relative distances to the other corners of the obstacle. Very acute values of θ may result in a situation where all the robots get accumulated on one side of the obstacle. In contrast, very obtuse values may lead the next position of the robot to again fall inside the boundaries of the obstacle. In this paper, we implemented this function using 60^o and 90^o for θ. 60^o and 90^o are the angles at which the drift is noticeable.

As shown in Fig. 2, if P1 is the robot's current position, circular drift function with angular drifts of 60^o and 90^o decides the robot's next position at P2 and P3 respectively.

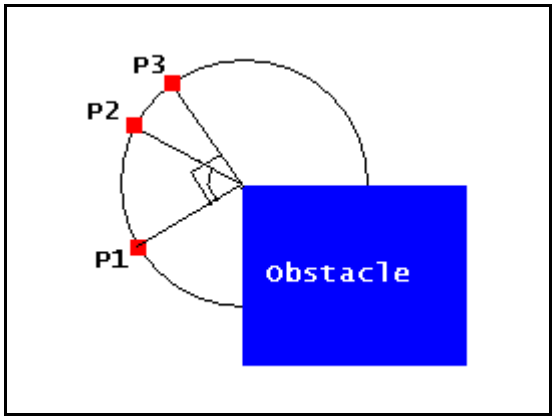

Fig. 2. Obstacle Avoidance using Circular Drift

The next position of a robot is calculated by using circular drift function as follows:

$$(x, y)_{next} = circular\_drift((x, y), obstacle\_corners, \theta) \quad (5)$$

In equation (5), (x, y) is the current position of the robot, $(x, y)_{next}$ is the next position of the robot, *obstacle_corners* represents the four corners of the obstacle.

IV. Simulations

The search space in each simulation is set to 20 by 20 units and the maximum velocity is limited to 1 unit. The number of robots used for planning the route is 9. The initial positions of the robots are randomly generated at a corner of the search space. The inertial weight is chosen as 1.0 and the correction factor is taken as 2.0. Four obstacles of sizes 2 by 2 units are randomly placed over the search space. The PSO algorithm specified in section 2 is simulated by setting the parameters.

A. *Route Planning without using Circular Drift Function*

Here, we apply Particle Swarm Optimization without using any special function for avoiding the obstacles. This results in some robots passing through the obstacles, which is not practically possible. The simulation is shown in Fig. 3

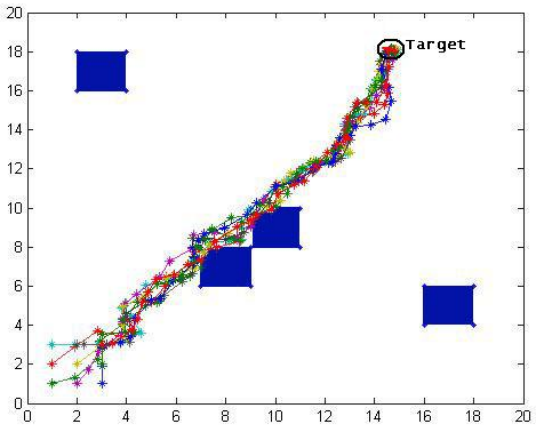

Fig. 3. Route Planning without using Circular Drift Function

As observed in Fig. 3, some robots have their path through the obstacles encountered on their way to the target location. To avoid this problem, we use circular drift function.

B. *Route Planning using Circular Drift Function*

As a modification to the above algorithm, we implement Circular Drift Function. Here, we check in every iteration if the robot's next position lies inside the obstacle area. If this happens, circular drift function is initiated. Fig. 4 shows effective routes planned between the target and the initial positions of the robots avoiding the obstacles by drifting away from them. In this simulation, we have used an angular drift of 60^{o}.

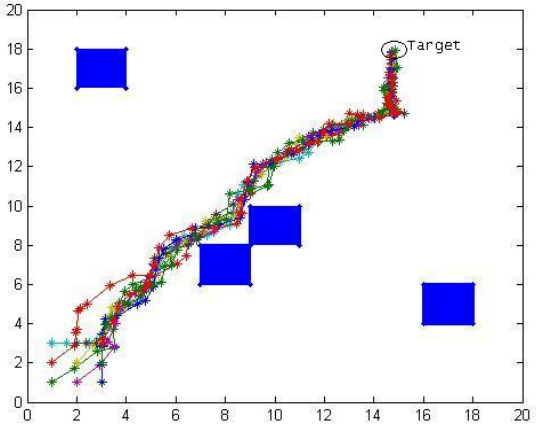

Fig. 4. Route Planning using Circular Drift Function

In Fig. 5, we can observe the angular drift of 90^{o} at a closer view. Here the robot uses the obstacle corner as its virtual circle's centre and drifts to a position on the circle which makes an arc with the current position that subtends 90^{o} at the centre.

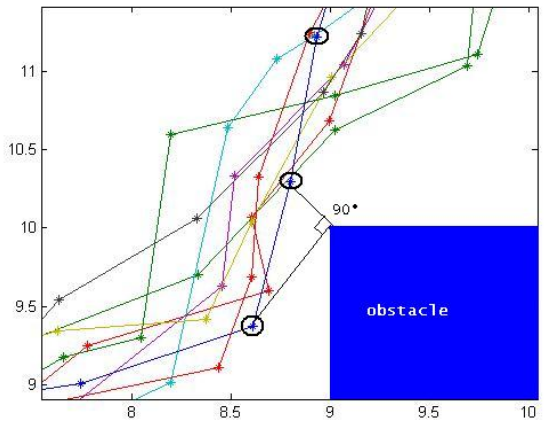

Fig. 5. Simulation showing 90^{o} Circular Drift

A similar implementation is shown in Fig. 6. Here we used an angular drift of 60^{o}. The robot uses the obstacle corner as its virtual circle's centre and drifts to a position on the circle which makes an arc with the current position that subtends 60^{o} at the centre.

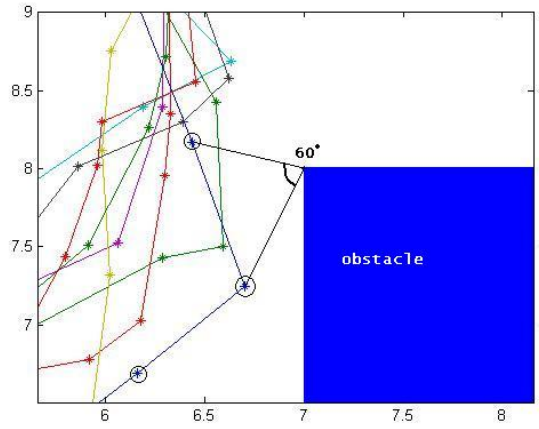

Fig. 6. Simulation showing 60^{o} Circular Drift

V. RESULTS

We performed the experiment using different number of robots to compute the number of iterations and time taken for route planning with circular drift function. We have computed results for angular drifts of 60^{o} and 90^{o}. The number of iterations is computed by taking the best of 20 simulations for the same number of robots. We have used AMD Athlon 3200+ 64-bit Processor. The time taken may vary with the use of a different processor.

Table I shows the number of robots against the iterations the time taken to complete the task. Here the angular drift used is 90^o.

TABLE I
ITERATIONS AND TIME TAKEN BY THE ROBOTS WITH 90^o ANGULAR DRIFT

No. of Robots	Iterations	Time Taken(sec)
6	44	0.610527
8	38	0.647956
9	37	0.551106
10	33	0.453307
12	31	0.436187

Table II also shows the number of robots, the iterations taken to complete and the time taken by the robots to reach the target. Here the angular drift used is 60^o.

TABLE II
ITERATIONS AND TIME TAKEN BY THE ROBOTS WITH 60^o ANGULAR DRIFT

No. of Robots	Iterations	Time Taken(sec)
6	60	0.792262
8	39	0.523014
9	35	0.471768
10	32	0.448400
12	30	0.426324

The results in Table I and Table II show that the iterations taken by the algorithm decrease with the increase in the robot number no matter what the angular drift value is.

VI. CONCLUSIONS

The paper has presented a novel circular drifting function for the robots to efficiently avoid the obstacles and plan an optimal route to the target. Circular drift function is a successful solution for the problem of the obstacle avoidance in the route planning. The simulations and results also prove that this function can be applied to real time robotics. Future work includes programming multiple robots and testing them in different environments with dynamic obstacles.

VII. ACKNOWLEDGMENT

This work was supported by Electronic Design and Automation Laboratory, National Institute of Technology, Warangal, India.

VIII. REFERENCES

[1] Kashif Zafar, Shahzad Badar Qazi, A. Rauf Baig, " Mine Detection and Route Planning in Military Warfare using Multi Agent Systems" Proceedings of the 30th Annual International Computer Software and Applications Conference, volume 2, pp. 327-332, September 2006.

[2] Min Gyu Park, Jae Hyun Jeon, Min Cheol Lee, " Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing" ISIE 2001, vol. 3, pp. 1530-1535, 2001.

[3] Fan Chunxia, WanYouhong, "An adaptive simple particle swarm optimization algorithm" CCDC 2008, pp. 3067-3072, July 2008.

[4] Ming Chen, Zhengwei Yao, "Classification Techniques of Neural Networks Using Improved Genetic Algorithms" WGEC 2008, pp. 115-119, September 2008.

[5] Bin Lei, Wenfeng Li, Fan Zhang, "Flocking algorithm for multi-robots formation control with a target steering agent" SMC 2008, pp. 3536-3541, October 2008.

[6] Li Lu, Dunwei Gong, "Robot Path Planning in Unknown Environments Using Particle Swarm Optimization" ICNC apos 08, vol. 4, pp. 422-426, October 2008.

[7] James Lane, Andries Engelbrecht, James Gain, " Particle Swarm Optimization with Spatially Meaningful Neighbours" SIS 2008, pp. 1-8, September 2008.

[8] Lisa L. Smith, Ganesh K Venayagamoorthy, Phllip G. Holloway, "Obstacle Avoidance in Collective Robotic Search Using Particle Swarm Optimization" IEEE Swarm Instelligence Symposium, May 2006.

[9] X. Li, "A multimodal particle swarm optimizer based on fitness euclidean-distance ratio" Proceedings of the 9th annual conference on Genetic and evolutionary computation 2007, pp. 78-85.