Obstacle avoidance using PSO

Carlos López-Franco, Juan Zepeda, Nancy Arana-Daniel, Lilibet López-Franco
Computer Science Department, CUCEI, University of Guadalajara
Guadalajara, Jalisco, México
carlos.lopez@cucei.udg.mx

***Abstract*— In this paper we present an obstacle avoidance algorithm based on Particle Swarm Optimization (PSO), which is a stochastic optimization technique. The PSO algorithm was modified in order to solve the proposed problem, in our case each particle of the PSO represents a new position, during the PSO algorithm each particle is tested to see if it represents a valid position. The best PSO particle represents the new subgoal, this goal is used by a controller to move the robot from its current position to the subgoal. The algorithm is tested with different maps, that show the robot's path avoiding obstacles and reaching the goal.**

Keywords: PSO, Obstacle Avoidance, Projective Geometry, Mobile robot

I. Introduction

In mobile robot navigation refers to move the robot safely towards the goal using the robot's knowledge and sensorial information of the surrounding environment.

Even though there are many different ways to approach navigation, most of them share a set of common components, among which path planning and obstacle avoidance may play a key role.

Path planning is a global procedure, that involves finding a geometric path from the robot start location to the goal.

Obstacle avoidance refers to the methodologies of changing the robot's path to overcome unexpected obstacles. The resulting motion depends on the robot actual location and on the sensors readings.

In the paper the authors propose an obstacle avoidance algorithm based on Particle Swarm Optimization (PSO). PSO is a stochastic optimization algorithm inspired by the social behavior of fish school or bird flock. The algorithm uses PSO particles as new robot's positions, each new position is checked to know if it is a valid position. Then the PSO output is used as the reference of a controller to move the robot to such subgoal, the process is repeated until the robot reaches the goal.

The rest of the paper is organized as follows: The next section will give a brief introduction to Particle Swarm Optimization. In section III we present a brief introduction to the range sensing using a laser range finder. In section IV we present the notation of Projective Geometry, used to define the lines and points of the algorithm. In section V we present the control algorithm that moves the robot from its current position to the desired position. In section VI we present the modified PSO algorithm that is used to avoid obstacles. In section VII we show the results of the proposed obstacle algorithm in different scenarios. Finally, in section VIII we give the conclusions.

II. Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique inspired by the social behavior of fish school or bird flock, as developed by Kennedy and Eberhart [1]. Since its development, PSO has become one of the most promising optimization techniques for solving global optimizations problems.

The PSO algorithm starts with a population of particles whose positions, that represent the possibles solutions. The particles move through the search domain with a specified velocity in search of optimal solution. Each particle maintains a memory which helps it in keeping the track of its previous best position. The positions of the particles are distinguished as global best and personal best.

The PSO particles move through a multidimensional search space looking for a potential solution. Each particle adjusts its position in the search space from time to time according to its moving experience and of its neighbors.

For an N-dimensional search space, the position of the ith particle is represented as

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{in}, \ldots, x_{iN}) \tag{1}$$

Each particle maintains a memory of its previous best position which is represented as

$$P_i = (p_{i1}, p_{i2}, \ldots, p_{in}, \ldots, p_{iN}) \tag{2}$$

The best particle of the population is denoted as

$$P_g = (p_{g1}, p_{g2}, \ldots, p_{gn}, \ldots, p_{gN}) \tag{3}$$

The velocity of each particle is represented as

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{in}, \ldots, v_{iN}) \tag{4}$$

The PSO algorithm starts with a population of particles randomly initialized in the search space. Each particle is moved toward the best position encountered by the particle and the best position found by the population of particles according to

$$V_i(t+1) = V_i(t) + \kappa_1\rho_1(p_i - x_i) + \kappa_2\rho_2(p_g - x_i) \tag{5}$$

and

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{6}$$

The constants κ_1 and κ_2 are the acceleration constants, they represent the weighting of the stochastic acceleration terms that pull each particle towards personal best and global best. The constants ρ_1 and ρ_2 are uniformly generated random numbers in the range of [0,1].

The first term in (5), $v_i(t)$ represents the previous velocity of the particle, this can be considered as a momentum, which prevents the particle from drastically changing its direction. The second term $\kappa_1\rho_1(p_i - x_i)$ is called the cognition part and indicates the personal experience of the particle. The third term $\kappa_2\rho_2(p_g - x_i)$, represents the cooperation among particles and is therefore named as the social component. The effect of this term is that each particle is also drawn towards the best position found by its neighbor.

III. Range sensing

Range sensing is a crucial element of our obstacle avoidance algorithm. The sensor used in this algorithm is a 2D laser range finder. A typical laser range finder sensor has a view angle of about 180°, scanned with an angular resolution of $0.25°, 1°$, and a distance range from 10m for indoor devices, to more than 100m for outdoor devices, and an accuracy of the distance measurement from $\pm 1mm$ to $\pm 5cm$.

The laser range finder returns 180 readings in polar coordinates, see Fig. 1. When the laser beam does not hit an obstacle then a high value reading is returned, in Fig. 1 we can see that some of the lectures from reading 123 to reading 138 have very high values and they must be removed, since they do not represent a valid obstacle position.

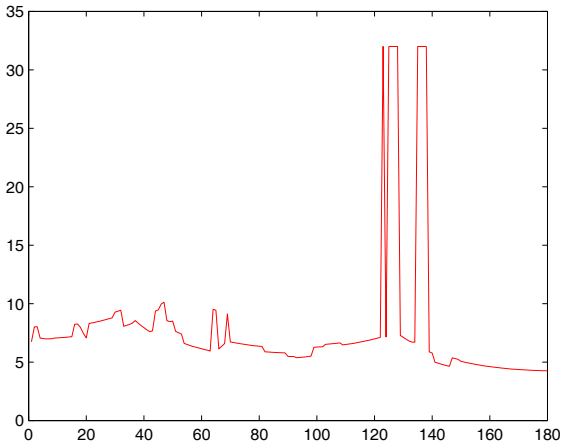

Fig. 1: Laser reading in polar coordinates

The laser readings are generally converted from polar coordinates to Cartesian coordinates. In Fig. 2 the laser readings from Fig. 1 where converted to Cartesian coordinates, and the invalid readings were removed.

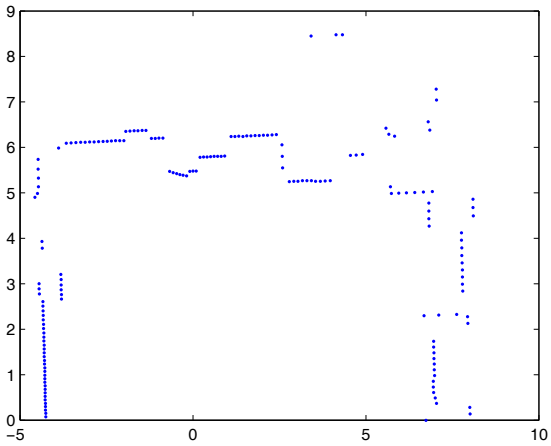

Fig. 2: Laser reading in Cartesian coordinates.

IV. 2D projective plane

A point in the plane can be represented as a pair of coordinates $(x, y) \in \mathbb{R}^2$. In the projective space $\mathbb{P}^2$ the points are represented as $(x, y, 1) \in \mathbb{P}^2$, this representation allows to easily compute lines from points, and to easily compute the distance from a point to a line [2], in the next subesctions we will present some important concepts of projective geometry that are used in this work.

A. Homogeneous representation of lines

A line in the plane is represented by an equation such as $ax + by + c = 0$, this can be represented by the vector $(a, b, c)^T$. The vectors $(a, b, c)^T$ and $k(a, b, c)^T$ represent the same line, for any non-zero constant k.

B. Homogeneous representation of points

A point $\mathbf{x} = (x, y)^T$ lies on the line $\mathbf{l} = (a, b, c)^T$ if and only if $ax + by + c = 0$. This can be written in terms of an inner product of vectors representing the point as $(x, y, 1)(a, b, c)^T)(x, y, 1)\mathbf{l} = 0$, thus the point $\mathbf{x} \in \mathbb{R}^2$ is represented as a $3-vector$ by adding a final coordinate of 1. Similarly to the line, given a point a non-zero constant k and line $\mathbf{l}$, the equation $(kx, ky, k)\mathbf{l} = 0$ if and only if $(x, y, 1)\mathbf{l} = 0$. Therefore, a point $(x, y)^T$ can be represented by the set of vectors $(kx, ky, k)^T$ for varying values of k different of zero.

Given a vector $\mathbf{x} = (x_!, x_2, x_3)^T \in \mathbb{P}^2$, the point in $\mathbb{R}^2$ can be recovered as $(\frac{x_1}{x_3}, \frac{x_2}{x_3})^T$.

C. Point line distance

The first two terms of the line $\mathbf{l} = (a, b, c)^T$ represent the line normal of Euclidean geometry, and the third element represents the distance of the line to the origin. In 2-vector notation this can be written as $\mathbf{n} \cdot \mathbf{x} + c = 0$, where $\mathbf{n} =$

$(a,b)^T$, $\mathbf{x} = (x,y)^T$. If the vector $\mathbf{n}$ is not a unit vector then $c/\|\mathbf{n}\|$ is the distance of the line from the origin.

The distance between a point $\mathbf{x} \in \mathbb{P}^2$ and a line $\mathbf{l} = (a,b,c)^T$ is given by

$$d = \mathbf{x} \cdot \mathbf{l} \tag{7}$$

If the vector $(a,b)^T$ of the line is not a unit vector then the distance between a point and a line can be computed as

$$d = \mathbf{x} \cdot \frac{\mathbf{l}}{\|\mathbf{n}\|} \tag{8}$$

where $\mathbf{n} = (a,b)^T$.

D. Line joining points

A line through two points $\mathbf{x}$ and $\mathbf{x}'$ can be defined as the cross product between the two points, that is

$$\mathbf{l} = \mathbf{x} \times \mathbf{x}' \tag{9}$$

V. Mobile robot navigation

The objective of the obstacle avoidance algorithm is to navigate locally following a set of subgoals lying on a path generated by a path-planner. Therefore, the path-planer must compute a path from the initial position to the goal position. The path generated by the path-planner cannot take into account new obstacles or moving obstacles, therefore the robot requires a local planer to take into account environment's changes.

Once we have the subgoals from the path-planner we must have a controller to move the robot from its current position to the next subgoal. One useful solution for a stabilizing feedback control of differential-drive mobile robot is presented in the next subsection. Its very similar to the controller presented in [3], [4], [5], other can be found in [6].

A. Feedback control

Consider a mobile robot trying to reach a goal, see Fig. 3. Without loss of generality let us assume that the goal is at the origin of the inertial frame, and let us represent the robot pose with respect to inertial frame as $(x,y,\theta)^T$. Therefore the error can be defined as $\mathbf{e} = (x,y,\theta)^T$.

The kinematics of a differential drive mobile robot is defined as

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{10}$$

The objective of the controller can be defined as

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = K\mathbf{e} \tag{11}$$

where

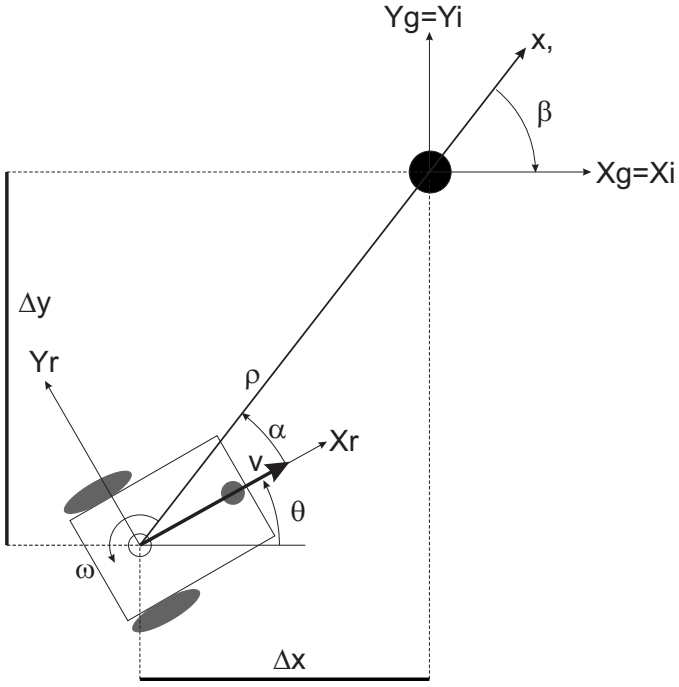

Fig. 3: Robot kinematics and its coordinates with respect to the goal coordinates.

$$K = \begin{pmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \end{pmatrix} \tag{12}$$

drives the error $\mathbf{e}$ toward to zero, that is

$$\lim_{t\to\infty} e(t) = 0 \tag{13}$$

Now, let us consider a coordinate transformation into polar coordinates with its origin at the goal position

$$\rho = \sqrt{\delta x^2 + \delta y^2} \tag{14}$$

$$\alpha = \arctan 2(\delta y, \delta x) - \theta \tag{15}$$

$$\beta = -(\alpha + \theta) \tag{16}$$

B. Control law

The velocities v and ω must be designed to drive the robot from its current pose $(\rho_0, \alpha_0, \beta_0)$ to the goal position. The linear control can be defined as

$$v = \kappa_\rho \rho \tag{17}$$

$$\omega = \kappa_\alpha + \kappa_\beta \tag{18}$$

The system has a unique equilibrium point at $(\rho, \alpha, \beta) = (0,0,0)$, therefore the controller will drive the robot to this point, which is the goal position.

VI. Obstacle avoidance using PSO

In this section we describe the modifications that we make to the PSO algorithm in order to allow the obstacle avoidance for a mobile robot.

In our case we represent a new position with a PSO particle, thus at each iteration the particles are moving to minimize an objective function. The objective function is defined as the Euclidean distance from the new positions to the goal.

Before the robot moves from its current position to a new position the algorithm must check that the new position is valid. A valid position is a position that does not collide with a laser scanning, that is not behind an obstacle, and that is not behind the robot. These three test must be applied to all the new position in order to discard a collision. In the next subsection we describe with more detail the tests.

In Fig. 4 we present the modified PSO algorithm. Note that the algorithm first tries to use the global best, but if the global best is an invalid position the it tries with the current iteration local best, and if it is also an invalid position, then the PSO particles are randomly changed to obtain a valid position, this process repeats until a valid position is obtained.

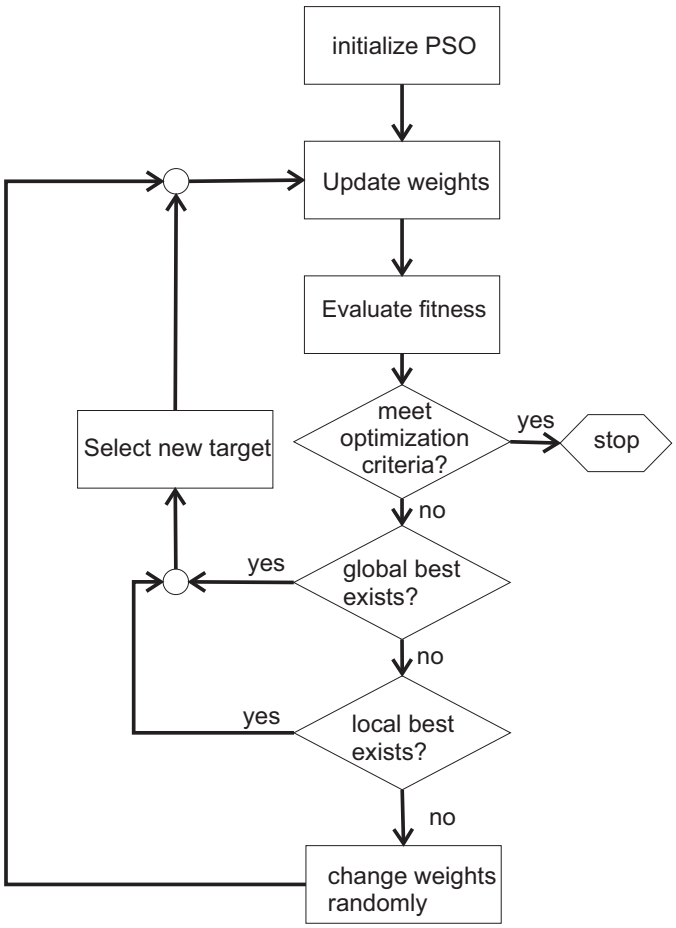

Fig. 4:

A. First test

In this test we check that the new position does not collide with an obstacle. To do this, we use the particle position to define a circle of radius ρ which is a constant that represents the robot's radius plus a δ value for safety reasons. Thus, if a new position has laser readings inside or very close to it, then this position is considered as an invalid position.

In Fig. 5 a situation with a set of new positions, after this test has been applied the new valid positions are draw in blue and the invalid positions with red.

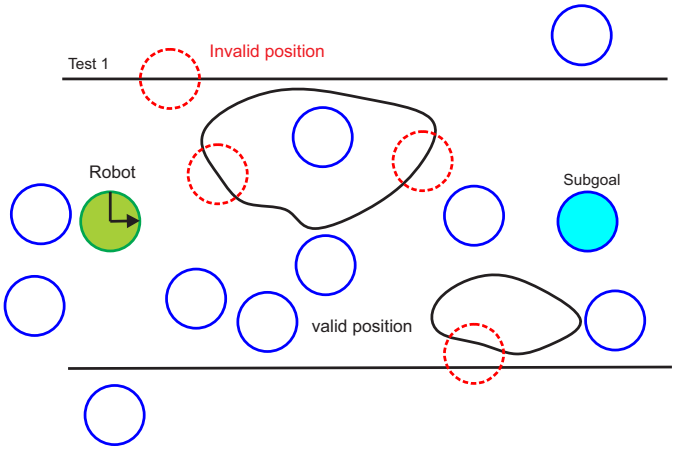

Fig. 5: First test, the new positions do not have to collide with obstacles. The valid positions obtained with this test are represented with solid circles, whereas invalid position are represented with dashed circles.

B. Second test

In this test we use the results of the test1 as an input. Let us assume that the robot is at the origin of the inertial frame, Fig. 6. Then for each position we trace a line $\mathbf{l}_1 = (a, b, c)$ passing through the robot center $(0, 0, 1)^T$ and the new position center $(c_x, c_y, 1)^T$, this line can be found using (9), that is

$$\mathbf{l}_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} c_x \\ c_y \\ 1 \end{pmatrix} \tag{19}$$

With the line $\mathbf{l}_1 = (a, b, c)$ we define two new lines

$$\mathbf{l}_2 = (a, b, \rho) \tag{20}$$

and

$$\mathbf{l}_3 = -(a, b, -\rho) \tag{21}$$

Then we define a line passing through the origin, and orthogonal to l_2 that is

$$\mathbf{l}_4 = (b, -a, 0) \tag{22}$$

Finally we define a line passing through the new position center (c_x, c_y) at a distance $d = \sqrt{c_x^2 + c_y^2}$, that is

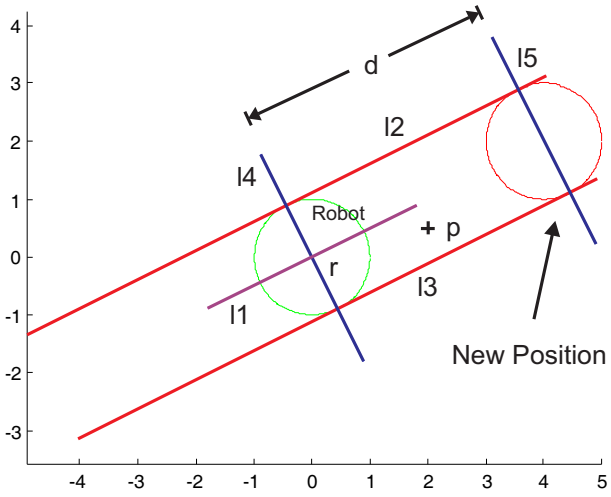

Fig. 6: Second test, if some laser readings are inside the rectangle then the position is invalid.

$$\mathbf{l}_5 = (-b, a, d) \tag{23}$$

To test if the path is a collision free path, we must check if there exist laser readings inside the rectangle defined with lines $l_2 \dots l_5$, if some laser readings are inside the rectangle then the position ins invalid. To test if the laser readings $p = (x, y, 1)^T$ is inside the rectangle we use the dot product between the point and the lines, this can be defined as

$$inside(p) = step(p \cdot l_2) + step(p \cdot l_3) + step(p \cdot l_4) + step(p \cdot l_5) \tag{24}$$

if the value of inside(p) is equal to 4 then the laser reading is inside the rectangle.

The function step is defined as

$$step(\alpha) = \begin{cases} 1 & \text{if } \alpha \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

An example of the second test applied to the results of test1 is shown in Fig. 7. The figure shows a scenario with new positions, after the second test the positions that are behind an obstacle are discarded.

C. Third test

In this test we check if the positions are behind the robot. This can be checked easily using a line defined as

$$\mathbf{l}_6 = (1, 0, 0) \tag{26}$$

and if $\mathbf{l}_6 \cdot \mathbf{x}$ is greater than zero then the new position is in front of the robot.

Figure 8 shows a scenario using the third test to the results of test 2, the red circles are invalid positions and the blue circles are valid positions.

An schematic diagram of the PSO and the control loop is shown in Fig. 9. As we can see in the figure the PSO

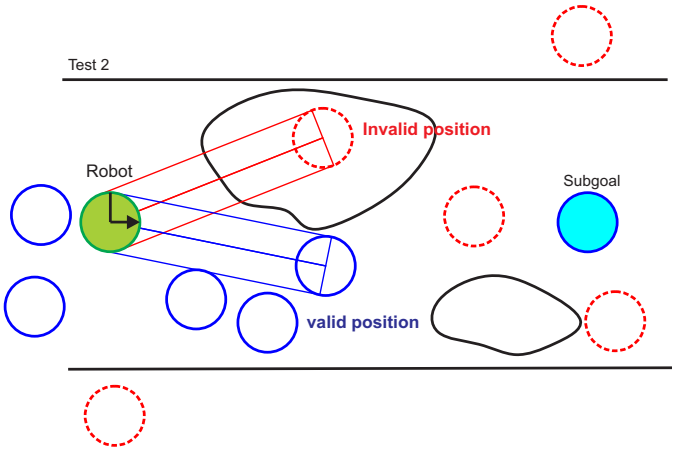

Fig. 7: Second test, the valid positions obtained with this test are represented with solid circles, whereas invalid position are represented with dashed circles.

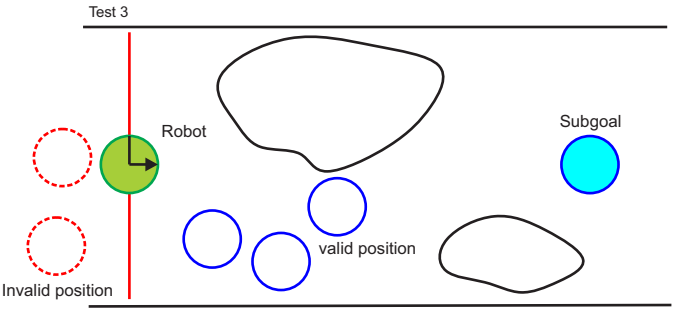

Fig. 8: Third test, in this scenario the invalid positions are the positions behind the robot. The valid positions obtained with this test are represented with solid circles, whereas invalid position are represented with dashed circles.

algorithm provides the reference to the control algorithm. This reference is changing through time in order to avoid obstacles and to reach the goal.

VII. Simulations results

In this section, we present our simulations results.

In Fig. 10 we show the simulation results of the proposed algorithm with a single obstacle. The green circles represent the particles best positions founded by the PSO algorithm. The red line represent the robot's trajectory which has been generated using the control algorithm of section V. As we can see the robot is able to avoid the obstacle and to reach the goal.

In Fig. 11 we show the simulation results of the proposed algorithm with two obstacles, and again the robot is able to avoid the obstacles and to reach the goal.

In Fig. 12 we show the simulation results of the proposed algorithm with five obstacles, and again the robot is able to avoid the obstacles and to reach the goal.

VIII. Conclusions

In this work we have presented a new obstacle avoidance algorithm for mobile robots based on Particle Swarm Optimization (PSO). The PSO algorithm was modified in order

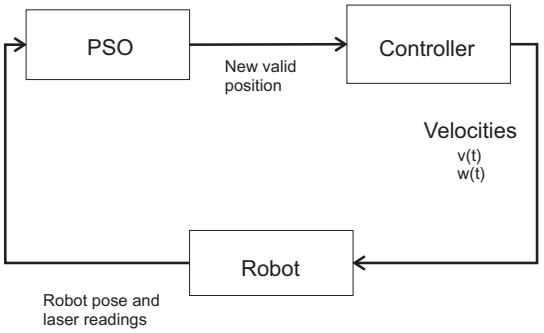

Fig. 9: Diagram of the PSO and control loop algorithms. We can observe that the output of the PSO is the reference for the controller.

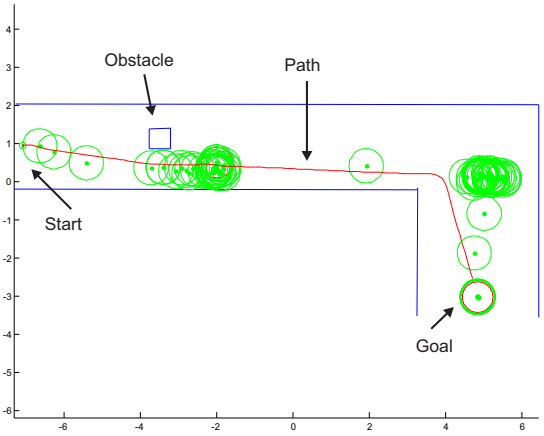

Fig. 10: Simulation scenario for the obstacle avoidance.

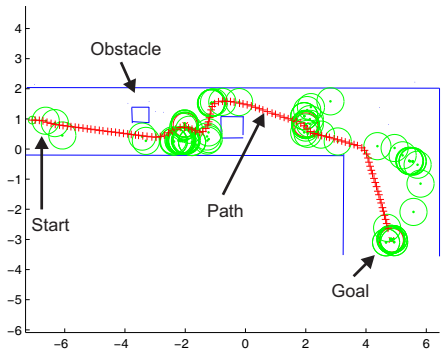

Fig. 11: Simulation scenario for the obstacle avoidance.

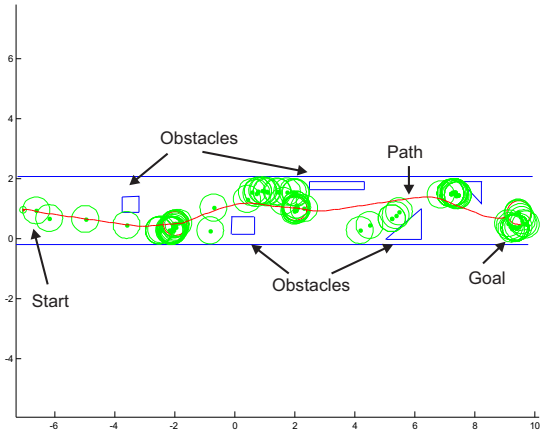

Fig. 12: Simulation scenario for the obstacle avoidance.

to accomplish the proposed task, where each particle of the PSO algorithm represents a new position. The objective of the PSO algorithm was to minimize the distance from the current robot position to the goal. The output of the PSO where new positions that minimize such distance, these new positions were validated to check if they can be reached by the robot and that the robot does not collide with an obstacle when moving to the new position. The PSO best particle was used as a new subgoal, and this position was passed to the controller in order to move the robot from its current position to the new subgoal. The process is repeated until the robot reaches the goal. The algorithm was tested with different scenarios, were we can observe that the robot is able to avoid obstacles and reach the goal. Future works will include using the proposed algorithm in conjunction with neural controllers, like recurrent high order neural network (RHONN) trained with an extended Kalman filter (EKF)-based algorithm [7], [8], and the use of sliding mode controllers [9], [10].

References

[1] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[2] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[3] A. Astolfi. Exponential stabilization of a mobile robot. In *European Control Conference*, pages 3092–3097, Rome, Italy, 1995.

[4] Sung-On Lee, Young-Jo Cho, Myung Hwang-Bo, Bum-Jae You, and Sang-Rok Oh. A stable target-tracking control for unicycle mobile robots. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 1822 –1827 vol.3, 2000.

[5] R. Siegwart and I. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. MIT Press, ISBN: 978-0-262-19502-7, 2004.

[6] E. Canudas de Wit, B. Siciliano., and G. Bastian. *Theory of Robot Control*. Springer-Verlag, 1997.

[7] A. Y. Alanis, M. Lopez-Franco, N. Arana-Daniel, and C. Lopez-Franco. Discrete-time neural identifier for electrically driven nonholonomic mobile robots. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1067 –1073, 31 2011-aug. 5 2011.

[8] A. Y. Alanis, M. Lopez-Franco, N. Arana-Daniel, and C. Lopez-Franco. Discrete-time neural control for electrically driven nonholonomic mobile robots. In press, DOI: 10.1002/acs.2289, 2012.

[9] A. Salome, A. Y. Alanis, and E. N. Sanchez. Discrete-time sliding mode controllers for nonholonomic mobile robots trajectory tracking problem. In *Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on*, pages 1 –6, oct. 2011.

[10] M. Lopez-Franco, A. Salome-Baylon, A. Y. Alanis, and N. Arana-Daniel. Discrete super twisting control algorithm for the nonholonomic mobile robots tracking problem. In *Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on*, pages 1 –5, oct. 2011.