Route Optimization of Non-holonomic Leader-follower Control Using Dynamic Particle Swarm Optimization

Bambang Tutuko, Siti Nurmaini, Saparudin, Putri Sahayu

***Abstract*--Mobile robots, when navigating in diverse environments, rely on solutions to trajectory generation problems for achieving the best path. One of those solutions is a heuristic method named Particle Swarm Optimization (PSO). In a previous study, by using such method, the mobile robot could find the best route towards the target without collision; moreover, PSO offers the benefits of simplicity, ease of implementation, and few parameters to regulate. However, the original PSO algorithm cannot guarantee the optimal solution. Local optima still occur, especially in complex and dynamic environments, due to premature convergence. This causes mobile robot collisions with obstacles and generates a long path to the target. In the present study, in order to overcome the problem of premature convergence, dynamic PSO (DPSO) was developed by using a dynamic inertia function to set parameters to accelerate convergence and re-initialize particles. The DPSO was analytically compared with two other algorithms, namely the original PSO (OPSO) and the Gaussian PSO (GPSO). Finally, the proposed DPSO is combined with Fuzzy Logic for obtaining the best control of leader-follower system. In the results, the proposed DPSO algorithm produced the optimum solution faster with convergence of less than 150 iterations for static obstacles and 200 iterations for moving obstacles, 4% shorter traveled lengths, 13% more smoothness, fast processing and guaranteed avoidance of collisions, and stable movement in reaching the target. When the proposed DPSO is combined with Fuzzy Logic, it can improve leader-follower performance in terms of trajectory control, time traveled to the target, and times response in several environmental conditions.**

***Index Terms*--Route Optimization, Non-holonomic, Leader-Follower, Particle Swarm Optimization**

I. Introduction

A distributed robot's coordination and control in a group has attracted many researchers over the past few years. One of many research topics is the problem of coordination between robots in controlling their formation in some applications such as unmanned ground robots, unmanned aerial robots, unmanned underwater robots, flying robots, and satellites [1][2][3][4][5][6]. Various strategies have been proposed with a variety of approaches to control the formation of a group of robots, including behavior-based, virtual structure and leader-follower control [5][6][7][8].

Manuscript received April 21, 2018; revised July 31, 2018.

Bambang Tutuko is currently a researcher and doctoral student in Engineering Faculty, Universitas Sriwijaya, Indonesia; Email beng_tutuko@gmail.com.

Siti Nurmaini (corresponding Author) is currently a Professor and researcher in Intelligent System Research Group, Universitas Sriwijaya, Indonesia; Email sitinurmaini@gmail.com.

Saparudin is currently a researcher in Image Processing Research Group, Universitas Sriwijaya, Indonesia; Email saparudin1204@yahoo.com.

Putri Sahayu is a graduate student in Informatics Engineering, at Universitas Sriwijaya, Indonesia; Email: sahayuputri@gmail.com

In factory or industrial environments for example, the leader-follower approach has become an important application replacing human tasks and, thus, helping people to live better lives. The following task is important to the mobile robot, since the target can be either a static or dynamic object. However, there are many problems that can occur when designing a robot to perform a following task. These problems include the accuracy of tracking the robot when the leader is moving, the distance to avoid collisions between the leader and follower, and the capability to avoid obstacles [6][7][8]. Besides, there are also issues related to the response time of the follower when the leader is moving and the communication between them. In leader-follower applications, one or more robots are appointed as the leaders, and the others are the followers. The leader serves as the reference to the follower robots, who need to position themselves and maintain the desired relative position with respect to the leader [8][9][10]. In such approach, to determine formation maneuvers, it is necessary only to determine the leader's path and the desired relative position and orientation between the leaders and the followers. When the direction of the leader's movement is known, the desired position (distance and angle) of the followers relative to the leader can be achieved by using the local control of each follower. However, if a leader's robot fails, it can lead to the failure of the entire controlling process.

Therefore, controlling the leader-follower robots in terms of position and orientation for achieving targets within a short convergence time and with high accuracy in dynamic environments is desirable. Under such conditions, the optimization route must be implemented for robotic control in a simple algorithm. Several optimization methods have been proposed [11][12], these have excellent convergence characteristics but they face challenges in handling the complex computation. In the leader-follower robotic system, the computational resources are important, due to the swarm characteristics utilize the on-board sensor and processing. Therefore, minimization of computational resources is a very important requirement. Several approaches, with good performance results, have been proposed and reported [8][13][14]. They implement leader-follower robots with global information for sharing. However, due to their complexity, the computational cost is high. When the algorithm is implemented in a simple robot with onboard sensors and processors, a major problem is incurred.

The particle swarm optimization (PSO) algorithm is one of the most efficient optimization strategies for continuous nonlinear optimization problems based on global information about the environment. It can be designed with simple algorithms for derivation of smooth and efficient trajectories [15][16][17]. Unfortunately, the original PSO algorithm is difficult to balance between exploration and exploitation

capabilities. To overcome this limitation, several authors have proposed different methods to achieve better accuracy and convergence [18][19][20]. Finding a proper balance between such two processes is considered a challenging task due to the stochastic nature of meta-heuristics; so, an improved PSO original algorithm is desirable. PSO is used to avoid obstacles in dynamic environments that include navigation and real-time motion planning issues [13][14][16][21][22]. Only a few researchers have proposed methods for multi-robot control systems, especially in the leader-follower configuration based on the kinematic model. Hence, this research is important for the purposes of developing motion control and route optimization for leader-follower robots based on the non-holonomic kinematic model in Cartesian representation.

The structure of this paper is as follows. In section 2, the process of kinematic models using Cartesian coordinates for controlling the formation of two non-holonomic mobile robots is described. In section 3, the route optimization design based on PSO method is explained. Some simulation results are included in section 4 to verify the feasibility of the model and the controller. Finally, conclusions and future work are discussed in section 5.

II. Leader-Follower Kinematic System

In this section, the Cartesian coordinates for leader-based formation controls explains the kinematics model. For simplicity, the configuration of a three-wheeled robot team is considered, with the left and right wheels controlled and one free wheel for balancing. From the illustration of the leader-follower robot movement in Fig. 1, the values of the robot movement parameters are obtained, $X-Y$ being world coordinates, and $x-y$ the fixed Cartesian coordinates of the leader robot. The parameters (X_L, Y_L) and (X_F, Y_F) are the global positions of the leaders and followers, where the subscript 'L' represents the leader and the subscript 'F' represents the follower. Meanwhile, v_L is the linear velocity of the leader and v_F is the linear velocity of the follower, while θ_L is the angle orientation of the leader and θ_F is the angle orientation of the follower. The kinematics model of mobile robot representation can be described by Cartesian representation (x, y) rather than polar coordinates. Due to the representation by polar coordinates, only a single point on the controller will be produced, which might degrade the controller's performance [23].

We can assume that the leader and follower robot follow the kinematics model of a unicycle robot in the inertial frame (see Fig. 1(a)). The kinematics of each robot can be expressed as

$$\begin{pmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\theta}(t)\end{pmatrix} = T_{NH}(q)\, u(t) \tag{1}$$

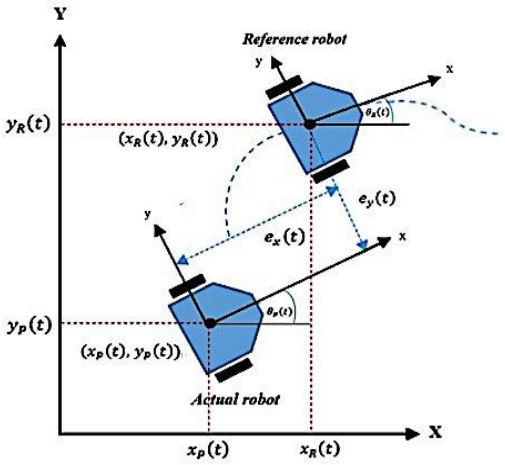
where $q(t)$ is the general variable of the initial position of the robot $q(t) = [x(t), y(t), \theta(t)]^{\mathrm{T}}$, T_{NH} is a non-holonomic transformation matrix, and $u(t)$ is a forward kinematic matrix that is used to estimate position and speed.

The non-holonomic transformation of the mobile robot can be seen through the change of the three initial robotic position variables $q(t)$. By solving (1) for the change in the velocity of the right and left wheels, the single robot kinematic equation can be transformed into (2) below,

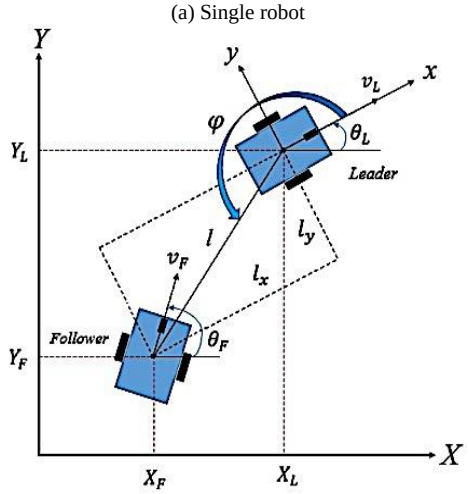
$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\theta}(t)\end{bmatrix} = \begin{bmatrix}cos\theta(t) & 0\\ sin\theta(t) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v(t)\\ \omega(t)\end{bmatrix} \tag{2}$$

To adjust the robot's position and orientation from actual to reference, the error position (3) is used as follows:

$$\begin{bmatrix}e_x(t)\\ e_y(t)\\ e_\theta(t)\end{bmatrix} = \begin{bmatrix}cos\theta_P(t) & sin\theta_P(t) & 0\\ -sin\theta_P(t) & cos\theta_P(t) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_R(t)-x_P(t)\\ y_R(t)-y_P(t)\\ \theta_R(t)-\theta_P(t)\end{bmatrix} \tag{3}$$



(a) Single robot



(b) Leader-Follower robot

Fig.1. Leader-Follower Kinematic System

In the same way, the kinematic system of the leader-follower robots is generated, but the parameters that will be measured are the relative distance between the leader and the follower robot. The modeling of the leader-follower system has been derived directly by kinematic analysis of the robot follower along the *x* and *y* coordinates relative to the robot leader. The leader L has configuration vector $[x_L, y_L, \theta_L]^T$ while the follower F has a vector $[x_F, y_F, \theta_F]^T$. The control inputs of the leader and the follower are the linear and angular velocities $[v_L, \omega_L]^T$ and $[v_F, \omega_F]^T$, respectively. The relative distance between the leader and the follower must be determined so that they can move in the same trajectory. To illustrate the relative position between the robots in Cartesian coordinates, Fig. 1(b) is utilized to projected the relative distance in the *x* and *y* directions. In *x*-*y* Cartesian coordinates, the distance between the robot leader and the follower robot is l. By using the properties of

trigonometric functions i.e., $a.b=|a| \ . \ |b| \ cos\theta$, the rotation matrix equation for the robot follower is obtained as shown in (4) below,

$$\begin{bmatrix} x_F \\ y_F \\ \theta_F \end{bmatrix} = \begin{bmatrix} cos\theta_F & sin\theta_F & 0 \\ -sin\theta_F & cos\theta_F & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Based on Fig. 1, and assuming that the relative distance equation can be derived using the matrix rotation in (3), the robot leader's distance relative to the follower robot is defined in (5), $l_x l_y \theta$

$$\begin{bmatrix} l_x \\ l_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} -cos\theta_L & -sin\theta_L & 0 \\ sin\theta_L & -cos\theta_L & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} X_L - X_F \\ Y_L - Y_F \\ \theta_L - \theta_F \end{bmatrix} \quad (5)$$

where the relative position of the follower robot in the x direction is l_x and in the y direction l_y, with relative orientation θ.

If the position of the leader robot (X_L, Y_L) is determined and (l_x, l_y) are known and fixed to achieve and maintain the desired formation, parameter (l_x, l_y) must be controlled, and then the position with respect to the robot leader can be determined by controlling $l_x \rightarrow l_x^d$ (where l_x^d is the desired relative position in the x direction) and $l_y \rightarrow l_y^d$ (where l_y^d is the desired relative position in the y direction). Under normal conditions, the relative distance between the leader robot and the follower robot is l^d, which needs to be simultaneously projected to control the movement of the follower robot against the leader robot by using (6) to (9), as follows:

$$l_x^d = l^d \cos\varphi^d \text{ or,} \quad (6)$$
$$\dot{l}_x^d = \dot{l}^d \cos\varphi^d - l^d \, \dot{\varphi}^d \sin\varphi^d \quad (7)$$
$$l_y^d = l^d \sin\varphi^d \text{ or,} \quad (8)$$
$$\dot{l}_y^d = \dot{l}^d \sin\varphi^d - l^d \, \dot{\varphi}^d \cos\varphi^d \quad (9)$$

The desired relative distance l^d between the robot leader and the follower robot is required to be constant or $l^d = l_0$, whereas the relative angle φ^d is varied with time. Therefore, (7) and (8) become (10) and (11), as follows:

$$\dot{l}_x^d = -l_0 \dot{\varphi}^d \sin\varphi^d \quad (10)$$
$$\dot{l}_y^d = -l_0 \dot{\varphi}^d \cos\varphi^d \quad (11)$$

From (5) the model of $\dot{l}_x$ is as follows:

$$\begin{aligned} \dot{l}_x &= -\left(\dot{X}_L - \dot{X}_F\right)\cos\theta_L + (X_L - X_F)\dot{\theta}_L sin\,\theta_L - \left(\dot{Y}_L - \right. \\ &\quad YF\sin\theta L + YL - YF\theta L cos\,\theta L \\ &= l_y\dot{\theta}_L + \dot{X}_F cos\theta_L + \dot{Y}_F\, sin\theta_L - \dot{X}_L cos\theta_L - \dot{Y}_L\, sin\theta_L \\ &= l_y\dot{\theta}_L + \dot{X}_F cos\theta_L + \dot{Y}_F\, sin\theta_L - v_L \end{aligned} \quad (12)$$

where v_L represents the linear velocity of the leader robot. The new state variable is defined to represent the orientation angle difference between the robot leader and the follower robot, as

$$e_\theta = \theta_F - \theta_L \text{ or} \theta_L = \theta_F - e_\theta \quad (13)$$

If (13) is substituted into (12), it becomes (14):

$$\begin{aligned} \dot{l}_x &= l_y\dot{\theta}_L + \dot{X}_F\cos(\theta_F - e_\theta) + \dot{Y}_F\, sin(\theta_F - e_\theta) - v_L \\ &= l_y\dot{\theta}_L - v_L + \cos e_\theta\,(\dot{X}_F\, cos\,\theta_F + \dot{Y}_F\, sin\theta_F) - \\ &\quad \sin e_\theta\,(\dot{Y}_F\, cos\,\theta_F - \dot{X}_F\, sin\theta_F) \end{aligned} \quad (14)$$

Due to the holonomic constraint of the mobile robot in (15), (14) is transformed to become (16),

$$\dot{Y}_F\, cos\,\theta_F - \dot{X}_F\, sin\theta_F = 0 \quad (15)$$
$$\dot{l}_x = l_y\omega_L + v_F\, cose_\theta - v_L \quad (16)$$

where $\omega_L = \dot{\theta}_L$ represents the angular velocity of the leader's robot, and v_F represents the linear velocity of follower's robot. In the same way, from (11), the model of $\dot{l}_y$will be obtained as follows:

$$\dot{l}_y = l_x\omega_L + v_F\, sine_\theta \quad (17)$$

The overall equation of the leader-follower kinematic model can be summarized as

$$\begin{bmatrix} \dot{l}_x \\ \dot{l}_y \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} cose_\theta & 0 & -1 & l_y \\ sine_\theta & 0 & 0 & l_x \\ 0 & 1 & 0 & -1 \end{bmatrix} \quad (18)$$

$$u = [v_F, \omega_F, v_L, \omega_L]^T$$

where ω_F is the angular velocity of the follower robot, v_F is the linear velocity of the follower robot, ω_L is angular velocity of the leader robot, and v_L is the linear velocity the leader robot. By using the leader-follower approach, ω_L and v_L are time functions that vary with input controls ω and v.

III. Dynamic Particle Swarm Optimization

Particle swarm optimization (PSO) is based on the behavior of a swarm of insects (e.g., ants, termites, bees) or birds [18][20]. The algorithm mimics the social behavior of such organisms. Social behavior consists of individual actions and the influence of other individuals in the group. The word "particle" denotes the individual. Each individual or particle behaves interconnected by using its own intelligence and also by being influenced by the behavior of its collective group [18]. Thus, if one particle finds the right or a short way to the target, the rest of the other group will also be able to follow the path immediately, even though they are located far away in the group. There are two kinds of PSO algorithm: the original PSO and the improved PSO [16][17][18][19]. In the original PSO (OPSO) algorithm, the inertial weight (w) is set at 1; thus, the convergence speed of particles is fast, and the adjustments of cognition and social components make particles search around one point. This can produce a local minimum condition, and the whole swarm will converge to this position. However, if the inertial weight value as selected is about $0 < w < 1$, the whole swarm has difficulty jumping out of the local optimum. This characteristic produces a fatal weakness, because no global optimum (g_{best}) is achieved. Hence, the dynamic inertial weight is desirable to regulate.

In this paper, the PSO algorithm optimizes the leader-follower robots' path to the target without collision. Therefore, it works not only for the optimization process, but

also to control the leader's movement. The dynamic inertial weight is needed, due to the fact that the leader-follower robots move in an unstructured and dynamic environment. The dynamic PSO (DPSO) is created by using (19) and (20) below,

$$v_{i,j}^{n+1} = w * v_{i,j}^{n} + c_1 * r_1 * \left(P_{i,j}^{n} - x_{i,j}^{n}\right) + c_2 * r_2 * \left(P_{g,j}^{n} - x_{i,j}^{n}\right) \quad (19)$$

$$x_{i,j}^{n+1} = x_{i,j}^{n} + v_{i,j}^{n+1}, \quad j = 1,2,\dots\dots,d \quad (20)$$

Vector $P_i = (p_{i1}, p_{i2}, \dots\dots, p_{i,d})$ is the best previous position of the *i*th particle that gives the best fitness value, named the personal best position p_{best} . Vector $P_g = (p_{g1}, p_{g2}, \dots\dots, p_{g,d})$ is the best particle among all particles in the population, named the global best (g_{best}). The inertial weight *w* is used to balance the global exploration as well as the local exploitation. r_1 and r_2 are random numbers uniformly distributed between [0,1]. The velocity $v_{i,j}$ is restricted to the range $[-v_{max}, v_{max}]$ in order to prevent the particles from flying out of the solution space. The acceleration coefficients c_1 and c_2 provide for a better balance of search space between the local exploitation and the global exploration.

In the leader-follower case, the PSO must ensure that the leader reaches the target and that the follower robot can follow the leader and maintain its formation without collision. In this paper, the dynamic inertial weight *w* and the learning factors c_1 and c_2 are improved in (21) and (22).

$$w = w_i - \left(\left(\frac{i_{max} - i}{i_{max}}\right) * (w_{max} - w_{min})\right) \quad (21)$$

The DPSO algorithm serves to control the PSO capabilities in local searches efficiently and to achieve convergence to global optimum solutions. The inertial weight *w* is updated to obtain an adaptive *w* value for each iteration; therefore, the value can be dynamic and capable of improving the expected optimization result. The greater the value of iteration, the *w* value will be smaller, and preferably, if the iteration is still early, the value *w* will tend to be larger. If the *w* value gets larger, the particle is more focused towards exploration, but as it gets smaller, it is more focused towards exploitation [16]. *c*1 and *c*2 are the acceleration coefficients or learning rates of a single current particle for obtainment of a better balance between global exploration by all particles in neighboring topology and local exploitation to achieve the best fitness. In this paper, *c*1 and *c*2 are used to find the target and to avoid obstacles, due to the necessity of changing the vector of velocity and vector position. When obstacles move, the robot environment changes dynamically. However, targets need to be found in a short period of time. In this paper, a dynamic linear adjustment strategy for learning factors is proposed. The expression is given as in (18),

$$c_1 = \frac{\Delta c_1 t}{t_{max}} + c_{1i} \text{ and } c_2 = \frac{\Delta c_2 t}{t_{max}} + c_{2i} \quad (22)$$

where $\Delta c_1 = \left(c_f - c_i\right)$ and $\Delta c_2 = \left(c_f - c_i\right)$ c_f= final value, and c_i = c initial value

If the number of iterations is increased, the cognitive ability of the individual is gradually reduced by improvement of the learning factors and improvement of the global searching ability of the particles. This strategy can improve particles' global search ability in the whole search space at early times, thus helping them to converge to the global optimum in the end.

IV. Results and Discussion

In this work, the OPSO and the Gaussian PSO (GPSO) are compared with the proposed DPSO for the above-noted function minimization problems of leader-follower kinematic control. For such purpose, about 50 particles and 1000 maximum iterations are taken. However, the number of particles is not very influential to the optimum solution generated PSO, but it affects the speed of the process. If the number of particles is too small, the process can get stuck on the local optimum even though the processing time is very fast. In contrast, large numbers of particles are rarely trapped in a local optimum, but the process takes longer. Respective inertial weights and acceleration coefficients are selected for balancing between exploration and exploitation capabilities.

A. Static Obstacles

To verify the proposed algorithm, a new leader-follower model based on a non-holonomic system for target seeking is simulated. Static and dynamic obstacles are utilized in the testing environment. In Fig. 2(a) and (b), the leader-follower robots move in relation to a static obstacle to reach the target. The leader robot is a blue line and the follower robot is a red line. The two types of PSO are used to view the movement performance. Obstacle avoidance in passing near and reaching the goal on a short trajectory is performed while avoiding the obstacle coming from the right and deviating from the straight line. The leader-follower robot using the OPSO is able to reach the target, but the resulting trajectory is not smooth; thus, a long processing time is taken and large amounts of data are generated in reaching the target. Using the proposed DPSO approach, the leader-follower robot movement performs better, using smoother movements, shorter travel times, and generating less data.

The movement of the leader robot by OPSO requires a fairly long route in reaching the target; therefore, the execution time is longer by about 29.14 sec. Conversely, if the movement of the robot leader by DPSO is optimized for a shorter route, the execution time is faster by about 13.55 sec while the follower robot follows the movement of the robot leader in a relatively equal time. Moreover, if DPSO is used, the movement of the leader robot is smooth, and the robots can choose the simplest route to the target. In addition to the other test environments, a rectangular environment also is created. The robot must move from the initial position to the target position, as seen in Fig. 2 (c) and (d). The robot moves not smoothly and with a long trajectory using the original PSO. The performance is not satisfactory using OPSO: the processing times necessary to finish the route are about 29.15 sec for the leader and about 29.18 sec for the follower, and at some points of the trajectory, the leader-follower robots crash into a wall. Meanwhile, by using DPSO, the processing time is about 13.50 sec for the leader and about 13.65 sec for the follower, with a short and smooth trajectory. Furthermore, the leader and the follower robots have the ability to maintain their positions relative to the wall without collision.

From the robots' trajectories in Fig. 2 (c) and (d), it is seen that from the initial position, the robot leader moves in search

of the target and manages to find it, and the success of reaching the target is seen from the result of the route leading to a point. However, the route taken by the leader-follower robot using the DPSO algorithm in achieving the target is more efficient than using the OPSO algorithm. This is due to the DPSO algorithm's use of parameter control of inertial function and coefficient acceleration to accelerate convergence and produce a global solution.

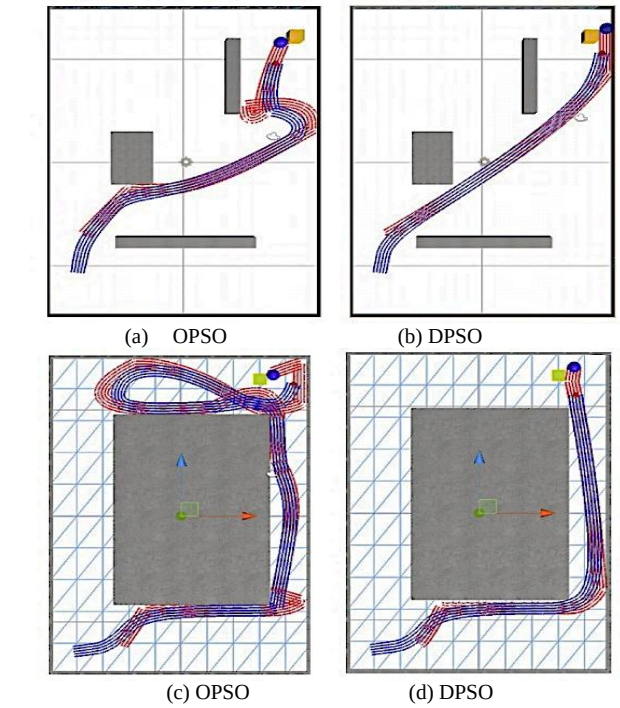


Fig. 2. Trajectory control in simple environment

Using the OPSO algorithm, premature convergence always happens; this condition occurs when particles converge and particle velocity is close to zero, but no global solution has been found. The time taken by each robot from the starting point to the target using the OPSO algorithm is about 66 seconds, while that taken using the DPSO algorithm is only 31 seconds. DPSO algorithm uses inertial parameters that are adaptable to the environmental dynamics that are encountered in the course of the target search.

In complex environments (see Fig. 3 (a) – (f)), the proposed DPSO produces small processing times compared with the OPSO and GPSO, due to its ability to change the positions and orientations of the leader-follower robots in an adaptive manner. The leader-follower robots have the ability to achieve the target, but especially with OPSO, they cannot finish the task; rather, they stop at one point and stack in the local minima (see Fig. 3 (a) and 3 (d)). However, when the GPSO and DPSO algorithms are applied in the robot, the task can be completed. Unfortunately, with GPSO, the leader-follower robots crash into the wall (see Fig. 3 (e) and 3 (b)), but they still move to the target. But when the leader movement is controlled based on DPSO, the robot is able to reach the target without a collision, via a short route and within a short processing time. Such also is the case with the follower robot.

The leader-follower performance in X and Y coordinates when moving in a complex environment can be depicted as in Fig. 4 (a) – (c). By OPSO, the robots can reach the target, but they stop moving after 40 sec. When GPSO is applied, they finish the task in about 47 secs, which is a poor response, due to their having moved in gradual steps. Their performance is improved when they use DPSO: the task is accomplished in about 22 secs, with smooth movement and shortest route.

B. Moving Obstacles

In this section, the moving obstacle is considered to be another robot that crosses the path of leader robot. Such obstacles are used to test the robot leader's ability in a dynamic environment. The presence of a moving obstacle causes the issue of path search to be a matter of dynamic optimization. Path search with the dynamic optimization approach has a higher difficulty level, because it demands an optimization algorithm to have adaptability to changing search-space conditions.

For testing of the proposed DPSO algorithm with moving obstacles, the simulations are performed by creating two environmental scenarios to ensure the leader's performance. The first scenario consists of static square obstacles in the form of walls that cut the path on which the mobile robot passes. The second scenario considers only moving obstacles. The performance of the robot's movement as it passes through a moving obstacle can be seen in Fig. 5. From Fig. 5 (a) to (i), it appears that the leader-follower robots by OPSO and GPSO are able to reach the target, but when they encounter a moving obstacle, they crash. Especially in Fig. 5 (j), the robots by OPSO cannot finish the task, as the U-Shape condition produces a local optima condition. By GPSO and DPSO, this does not happen: the robots have the ability to get out of the U-shape situation and move to the target.

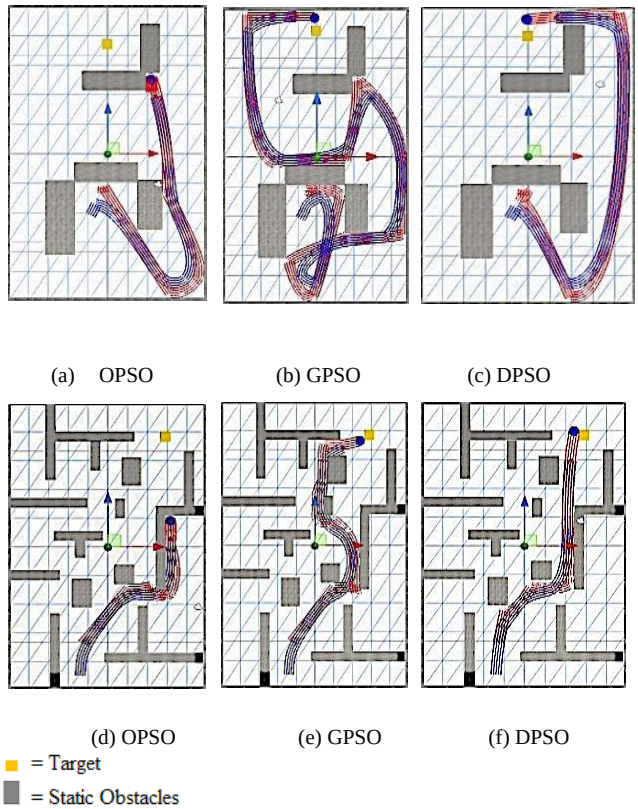
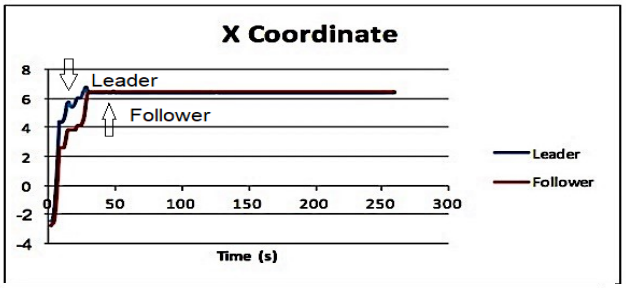
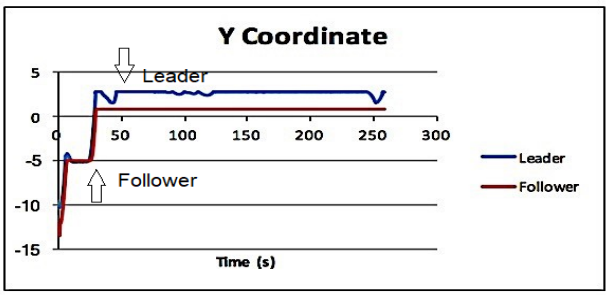
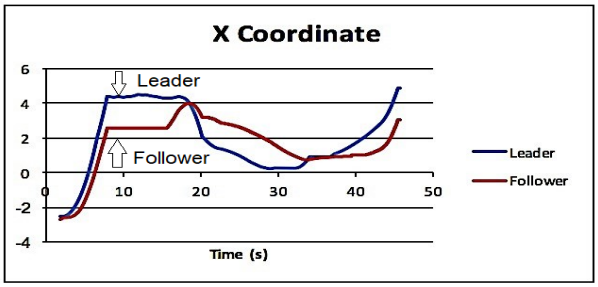


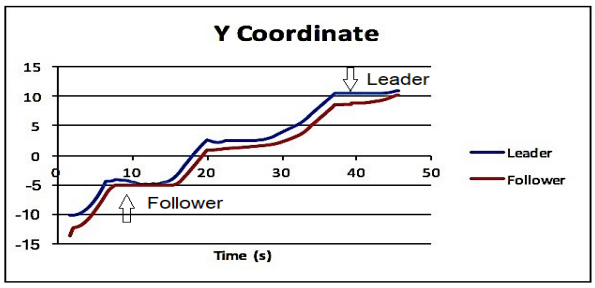
Fig. 3. Trajectory control with static obstacle in cluttered environment



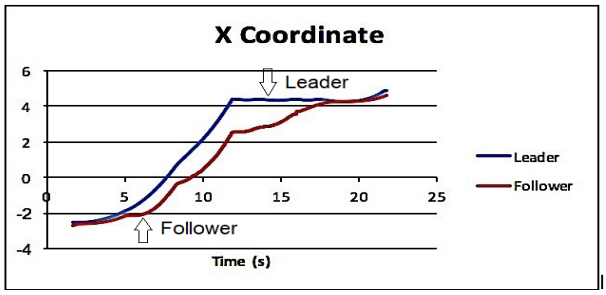


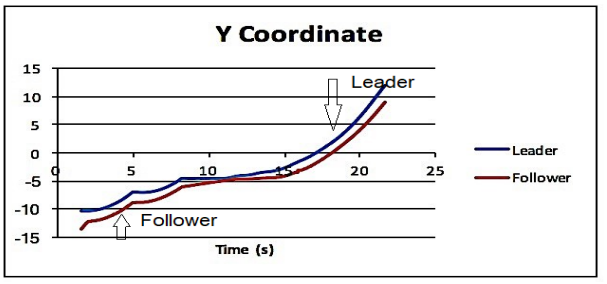
(a) OPSO





(b) GPSO





(c) DPSO

Fig. 4. The Respon of Robot in X-Y Coordinate on Complex Environment (Fig. 3 (d), (e), (f))

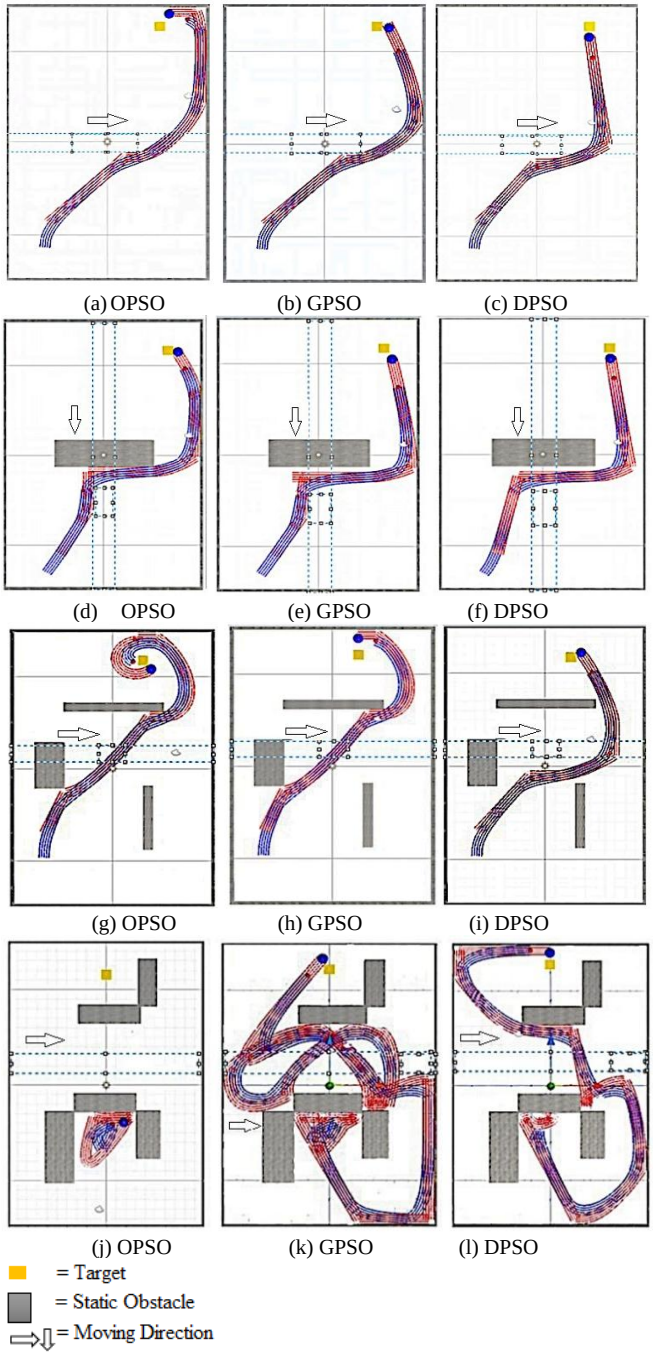


Fig. 5. Trajectory control with moving obstacles with 4 environments

Using the OPSO algorithm, the robots cannot quickly change direction or velocity when they encounter a moving obstacle. Therefore, movement is processed only in accordance with the parameter value that was determined in the initial condition, which situation produces collisions and system instability. The inertial functions are able to adjust the cognition and social components by using the GPSO algorithm, making the particles search around in several points. Even though the leader-follower robots still would crash, they are more stable in their movement. Also, due to shorter time to reach the target, the GPSO algorithm is able to change the values of the particles' position and velocity to appropriate ones.

However, when DPSO is applied for moving obstacles, the problems can be overcome. The leader-follower robots move quickly to the target via a short route, are able to avoid collisions with moving obstacles and walls, and show stable

movement. This is due to the dynamic inertial weight and the learning factor, which can improve robot performance and enable the global search ability of particles in the whole search space. The leader-follower robots can find the particles' position and velocity for the safest value in the shortest time. This allows the leader to move on a safe route with a smooth trajectory without collision. Improvement of learning factors and inertial weights can reduce the individual particles' cognitive ability and improve their global search ability when the number of iterations is increased. All robot performances using the three algorithms' optimizations can be seen in Table I.

TABLE I
PERFORMANCE OF LEADER-FOLLOWER ROBOT

Moving obstacles	Algorithm					
	PSO		GPSO		DPSO	
	Leader	Follower	Leader	Follower	Leader	Follower
Resource (kb)	28.6	28.8	21.7	21.8	16.8	17
Time (sec)	23.32	23.32	16.84	16.84	14.94	14.94
Data (iteration)	1032	1032	771	771	599	599
Static and moving Obstacle	Algorithm					
	PSO		GPSO		DPSO	
	Leader	Follower	Leader	Follower	Leader	Follower
Resource (kb)	28.7	28.6	22	21.2	20.1	21.1
Time (sec)	81.42	81.42	74	74	20.6	20.6
Data (iteration)	1035	1035	760	771	718	718

In all of the experiments performed using OPSO and GPSO, the robot leader managed to reach the target only via a very long route and after colliding with obstacles. This happens because the robots cannot adapt to environmental changes. At time *t* the robots are able to move towards the target, but in the period (t + 1), there is a moving obstacle that necessitates difficult changes of robot position and direction of movement, which changes result in a collision. With the PSO algorithm, convergent characteristics incur difficulty for the leader-follower robots in adapting to a dynamic environment; or, in other words, the particles tend to go to a certain point. Basically, the PSO algorithm was not developed to handle environmental changes, but this does not imply impossibility. By means of DPSO, divergence properties are enhanced to overcome the natural convergence characteristics of the PSO by moving the positions of particles randomly. This particle transfer is expected to enable change of robot position and orientation quickly for avoidance of collisions with obstacles.

C. PSO Performances

In the PSO algorithm, population size correlates with convergence. Increasing the population size can decrease the number of iterations required to find the optimum. However, the convergence rate and the optimal solution depend not only on the number of swarm but also on the objective function to be minimized/maximized. In this paper, the swarm population is selected to be about 40, without regard to dimensionality. A small population size will cause premature convergence; a large population size will allow for accurate results, but an extremely large population size will increase the computational time and memory requirement. Therefore, the number of iterations and swarm size should be selected for optimal convergence results for certain conditions. In this paper, the swarm population is changed from 10 to 30 to determine the PSO performance. In all of the experiments with the OPSO, GPSO and DPSO algorithms using swarm populations of 10 and 30, there was a long travel time to the target (see. Fig. 6(a) to (f)), with large iteration numbers up to 7000 and, in some environments, robot failure to reach the target (see Fig. 6 (a), 6(b), and 6(d)).

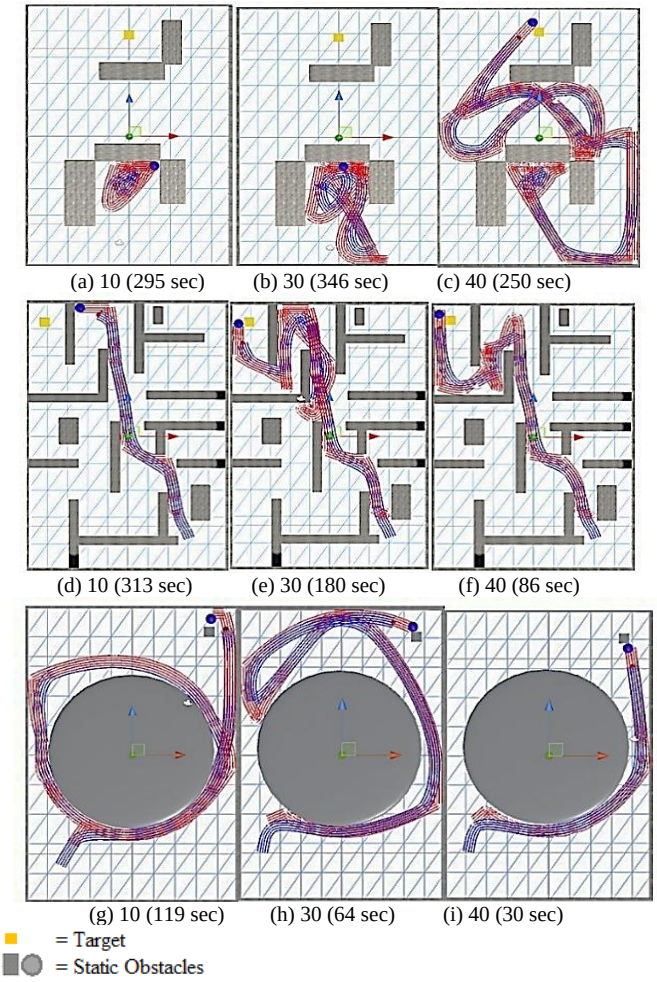


Fig. 6. Swarm population and leader-follower performance

When the number of swarm population is increased to 40, it produces a short route, a faster processing time (64 secs from 30 secs, and 180 secs from 86 secs), and a smaller iteration number (see Fig. 6 (i) and (f)). Even under the condition of leader-follower failure, performance was improved and the task could be achieved (see Fig. 6 (c), (e), and (f)).

D. Proposed DPSO with Fuzzy Logic control

In this section, the proposed DPSO algorithm is combined with Fuzzy Logic to improve the leader-follower performance in terms of trajectory control and time traveled to the target. In this experiment, the selected DPSO parameters are used for tuning the fuzzy membership functions (MFs). This is an important stage of the fuzzy logic algorithm. The fuzzy MFs will produce the rule base and use the fuzzy inference mechanism. If the MFs can be dynamically changed for every environmental condition, the output of the fuzzy logic control can be adapted every time. It is necessary for the leader robot to control its movement so as to reach the target with good performance. The leader-follower control process using Fuzzy-DPSO can be seen in the block diagram of the control system in the following Fig. 7.

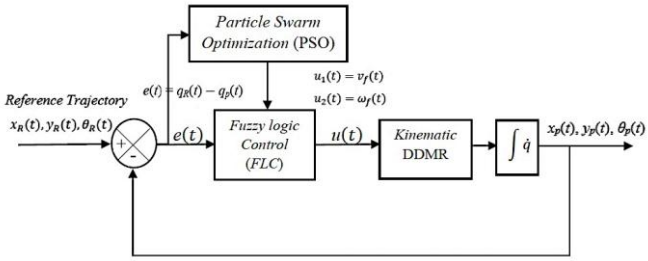


Fig. 7. The block diagram of Fuzzy-DPSO algorithm to improve the leader-follower movement in route optimization

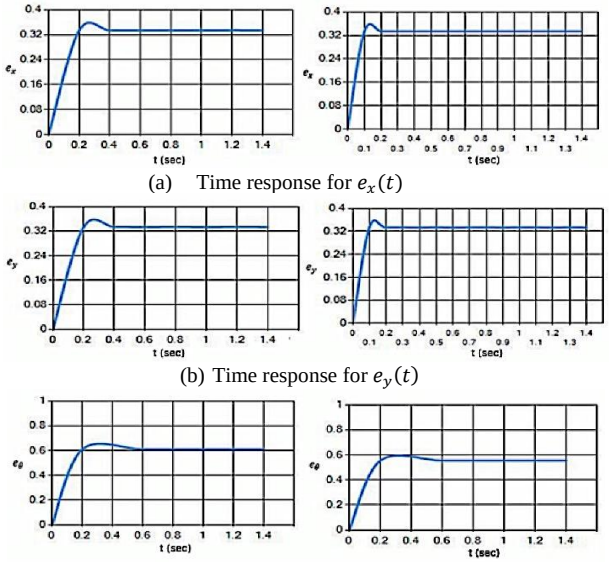
The four stages are fuzzification, rule base, inference, and defuzzification. Fuzzification is a process that can change the input crisply, which includes numbers or crisp values. It is converted to a fuzzy system in the form of linguistic values based on certain MFs. The fuzzy membership function consists of 9 parameters and 3 linguistic values that are Negative, Zero, and Positive. The fuzzy rule base identifies the values of position and orientation error $(e_x(t), e_y(t), e_\theta(t))$ as the fuzzy inputs and the linear velocity $v_f(t)$ and angular velocity $\omega_f(t)$ as the fuzzy outputs. The inference method used is Mamdani (Min-Max). Defuzzification, the final stage in a fuzzy logic system, is the process of mapping a quantity of the fuzzy set in the form of a crisp value.

In this section, DPSO is used to set a more optimal route by dynamically changing the fuzzy input MFs and to better control the movement of the leader robot to reach the target. Without fuzzy logic, the leader route to be taken would not be optimal, and the resulting movement would be unsmooth. The results would be different if the proposed DPSO were utilized to dynamically adjust the fuzzy MFs and, thereby, produce some rule bases corresponding to the environmental situation. By using such rules, the leader robot produces good position and orientation to achieve the target, and the follower moves in the same direction with good orientation. Fig. 8 shows the plot of fuzzy MFs after tuning, with 10, 20, 30, 40 and 50 swarm populations respectively.

TABLE II
INITIALIZATION INPUT ERROR MFS

	Particles	Fitness
S1	-1.5, -1, -1, 1.1, 1.3, 1.2, 1.5	2
S2	-1.9, -1.2, -1.6, 0.1, 1.5, 1.2, 1.1	2
S3	-1.4, -1, -1.7, 1.2, 1.9, 0.5, 1.5	1.89
S4	-1.8, -0.5, -1.2, 0, 1.8, 1.4, 1.8	1.76
S13	-1.3, -0.2, -0.6, 0.1, 1.7,1.1, 1.7	1.61
S25	0.4, 0.6, 0.4, 0.8, 1.6, 0.2, 1.6	1.44
S37	0.4, 0.7, 0.4, 0.8, 1.59,0.2, 1.59	1.42
S45	0.4, 1, 0.4, 0.7, 1.58, 0.3, 1.58	1.4
S47	0.4,0.8, 0.4, 0.5, 1.59, 0.8, 1.59	1.42
S49	0.4, 0.9, 0.4, 0.9, 1.6, 0.3, 1.6	1.44
S50	-1.3, -0.3, -1.2, 1.2, 1.4, 1.7,1.4	1.61

According to the setting of the swarm population, the Fuzzy MFs also change, as shown in Fig. 8 and Table II. The control performances based on the selected parameters can be seen in the transient response (see Fig. 9). DPSO with a swarm population of about 50 results in the best performances in terms of the rise time, maximum overshot and settling time to a stable condition (see Table. III). By using Fuzzy-PSO, the values of the three parameters above are small compared with DPSO. It can be seen that the proposed DPSO effectively improves leader-follower performance.



(a) Time response for $e_x(t)$

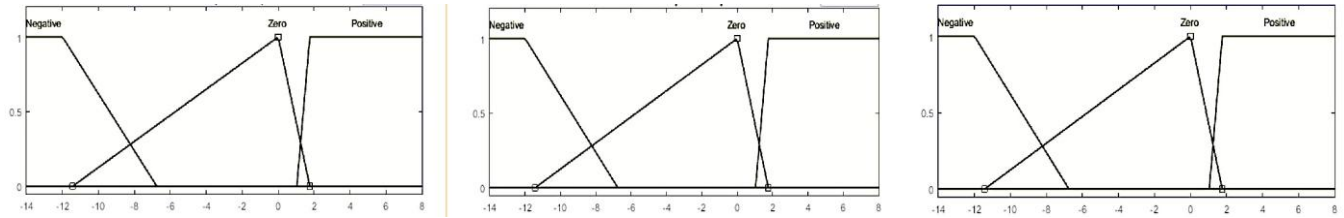
(b) Time response for $e_y(t)$

(c) Time response for $e_\theta(t)$

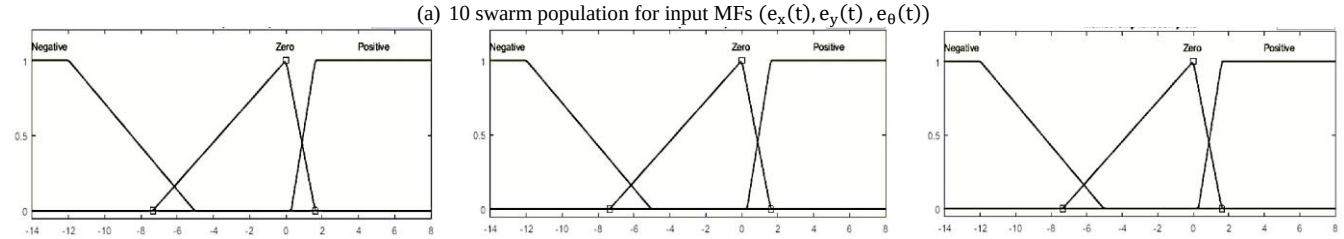
Fig. 9. Comparison of controller response using DPSO and Fuzzy-DPSO (left: DPSO; right: Fuzzy-DPSO)

TABLE III
CONTROLLER PERFORMANCES

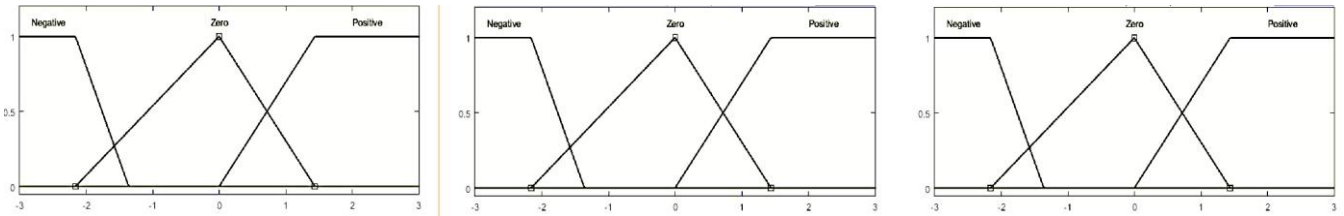
Controller	Rise Time (sec)			Max. overshoot (%)			Settling time (sec)		
	$e_x(t)$	$e_y(t)$	$e_\theta(t)$	$e_x(t)$	$e_y(t)$	$e_\theta(t)$	$e_x(t)$	$e_y(t)$	$e_\theta(t)$
No-Controller	0.37	0.37	0.37	18	10	28	Oscillation	Oscillation	Oscillation
DPSO	0.27	0.27	0.27	2	2	4	0.4	0.4	0.6
Fuzzy-DPSO	0.12	0.12	0.2	2	2	3	0.18	0.18	0.54



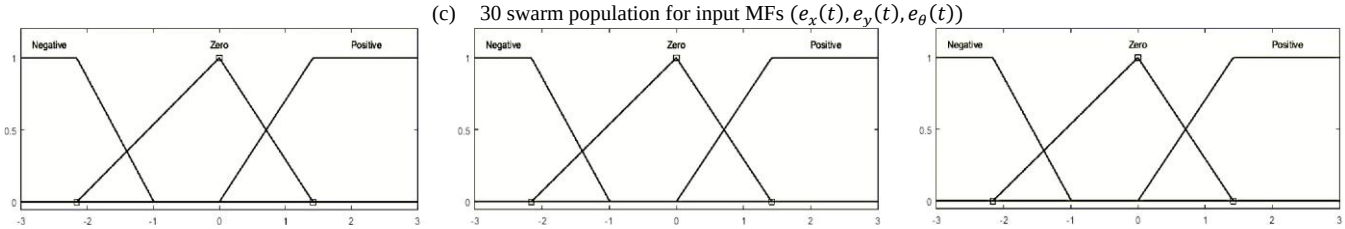
(a) 10 swarm population for input MFs ($e_x(t), e_y(t), e_\theta(t)$)



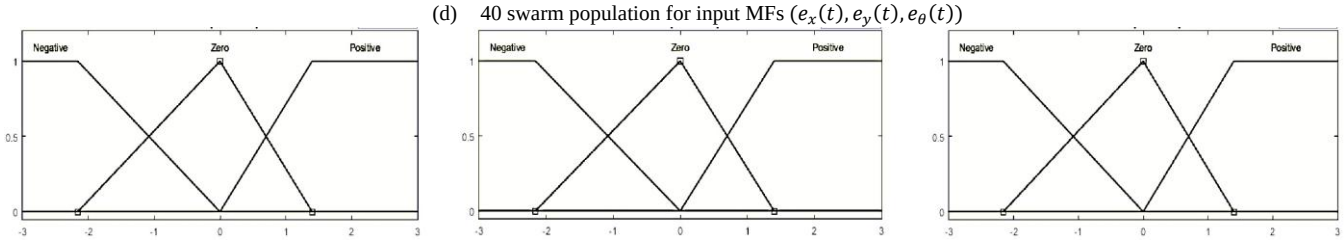
(b) 20 swarm population for input MFs ($e_x(t), e_y(t), e_\theta(t)$)



(c) 30 swarm population for input MFs ($e_x(t), e_y(t), e_\theta(t)$)



(d) 40 swarm population for input MFs ($e_x(t), e_y(t), e_\theta(t)$)



(e) 50 swarm population for input MFs ($e_x(t), e_y(t), e_\theta(t)$)

Fig. 8. Fuzzy membership functions tuning with DPSO based on 10 to 50 swarm population ($e_x(t)$ right curve, $e_y(t)$ middle curve, and $e_\theta(t)$ left curve)

The implementation of the DPSO algorithm on fuzzy MFs for controlling the leader-follower robots produces better performance. Due to the fact that the DPSO algorithm cannot control the leader robot's movement, that robot only moves to reach the target position. The leader robot will control the actuator only based on the sensor rule; therefore, it is unable to maintain distance from obstacles. But if the DPSO algorithm is combined with Fuzzy logic, the leader robot has the ability to control the actuator. Thereby, it can move more precisely to reach the target, with a smooth trajectory and the ability to maintain distance from obstacles (see Fig. 10). The result obtained in Fig. 10 (b) shows that the leader-follower robots with Fuzzy-DPSO are able to maintain distance from the oval obstacles and move more smoothly. As can be seen in Fig. 10 (a), the robots with only DPSO takes a longer route to search the target and incurs three collisions due to its inability to control its movement or maintain distance from obstacles. In Fig. 10 (c), it is clearly apparent that with Fuzzy-DPSO, the leader-follower robot can go directly to the target, move according to the rules to avoid obstacles, and reach the target in a shorter time and with a smoother trajectory. Whereas, if only the DPSO algorithm is used (see Fig 10 (d)), the robots move according to the sensor, and the travel time is longer, but they both reach the target. Fig. 10 (f) and (h) show better performance in a complex environment by use of Fuzzy-DPSO. The robots will follow the fuzzy rule and move directly to the target, faster and without collision. However, without the fuzzy algorithm, the robots crash and take a longer time to reach the target (see Fig. 10 (e) and (g)).

V. Conclusion

The PSO algorithm applies social adaptation to perform tasks, considering all individuals to be of the same generation. It has many advantages, such as simplicity, good convergence performance, and fewer control parameters. However, it does not provide a mechanism for escaping from local optima solutions, and local extremum values are easily fallen into. In the case of leader-follower control, by using original PSO, the leader always moves around to find the

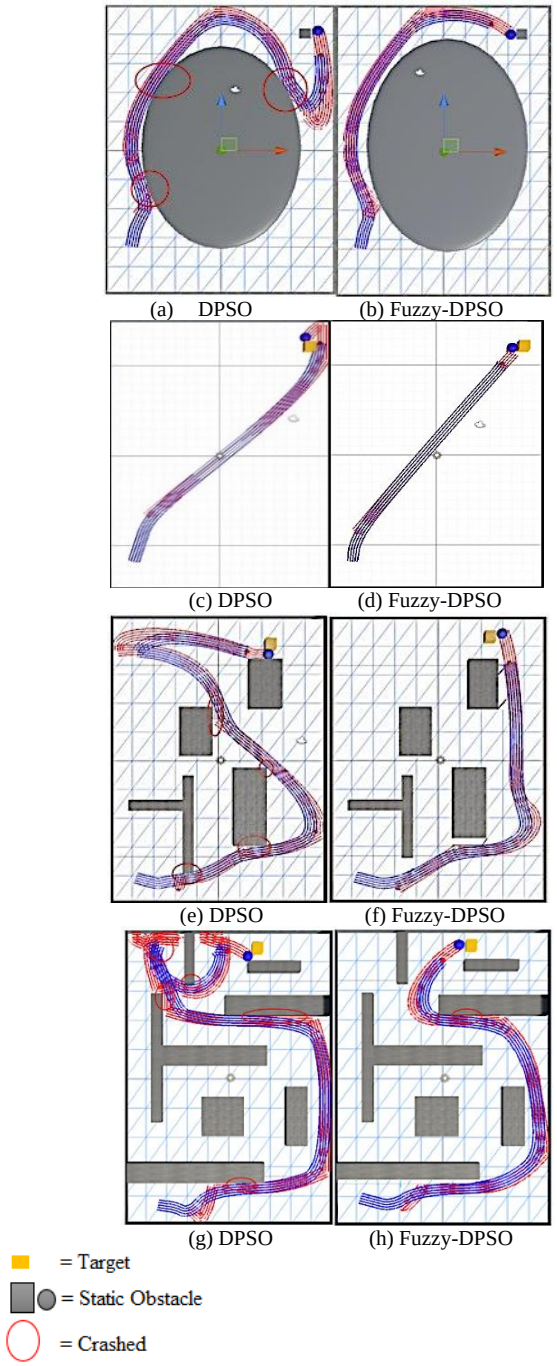


Fig. 10. Leader-follower trajectories based on DPSO and Fuzzy-DPSO (with 50 swarm population)

target thus generating a lot of data to complete the task. Therefore, the large search space for finding the possible solution space of the optimal solution by using inertial weight adjustment strategy into the original PSO.

In this paper, a dynamic PSO is proposed in order to improve leader-follower performance when they move in several environments containing static and dynamic obstacles. Using DPSO, the leader-follower robots perform better, showing smoother movements, shorter travel times and producing less data. The comparison results show that the proposed DPSO algorithm is more capable of obtaining the global optimization solution and overcoming the problem of local minima when the leader-follower robots move in complex and dynamic environments. Moreover, to improve leader-follower control and performance, the proposed DPSO is combined with Fuzzy Logic, which enhances performance in terms of trajectory control, time traveled to the target, and response times for several environmental conditions. In the future work, leader-follower robot motion control in response to dynamic environmental changes will be further studied.

ACKNOWLEDGMENT

The authors would like to thank the editors and the reviewers for their constructive comments and suggestions. This work is supported by Universitas Sriwijaya and the Ministry of Research Technology and Higher Education of Indonesia under Grant Penelitian Unggulan Perguruan Tinggi 2018 and Hibah Bersaing 2018.

REFERENCES

[1] G. Lee and D. Chwa, “Decentralized behavior-based formation control of multiple robots considering obstacle avoidance,” *Intell. Serv. Robot.*, vol. 11, no. 1, pp. 127–138, 2018.

[2] L. He, P. Bai, X. Liang, J. Zhang, and W. Wang, “Feedback formation control of UAV swarm with multiple implicit leaders,” *Aerosp. Sci. Technol.*, vol. 72, pp. 327–334, 2018.

[3] F. Berlinger, J. Dusek, M. Gauci, and R. Nagpal, “Robust Maneuverability of a Miniature, Low-Cost Underwater Robot Using Multiple Fin Actuation,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 140–147, 2018.

[4] M. A. Lewis and K.-H. Tan, “High precision formation control of mobile robots using virtual structures,” *Auton. Robots*, vol. 4, no. 4, pp. 387–403, 1997.

[5] W. Ren and R. Beard, “Decentralized scheme for spacecraft formation flying via the virtual structure approach,” *J. Guid. Control. Dyn.*, vol. 27, no. 1, pp. 73–82, 2004.

[6] Y. Abbasi, S. A. A. Moosavian, and A. B. Novinzadeh, “Formation control of aerial robots using virtual structure and new fuzzy-based self-tuning synchronization,” *Trans. Inst. Meas. Control*, vol. 39, no. 12, pp. 1906–1919, 2017.

[7] T. Balch and M. Hybinette, “Behavior-based coordination of large-scale robot formations,” in *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, 2000, pp. 363–364.

[8] A. Loria, J. Dasdemir, and N. A. Jarquin, “Leader--follower formation and tracking control of mobile robots along straight paths,” *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 2, pp. 727–732, 2016.

[9] S. Nurmaini and B. Tutuko, “Intelligent Robotics Navigation System: Problems, Methods, and Algorithm,” *Int. J. Electr. Comput. Eng.*, vol. 7, no. 6, pp. 3711–3726, 2017.

[10] S. Nurmaini, and A. Zarkasi, "Simple Pyramid RAM-Based Network Architecture for Localization of Swarm Robots, " *Journal of Information Processing Systems*, vol. 11, no. 3, pp. 370-388, 2015.

[11] W. Chunfeng, W. Song, and L. Liu, "An Adaptive Bat Algorithm with Memory for Global Optimization," *IAENG International Journal of Computer Science*, vol. 45, no. 2, pp. 320-327, 2018.

[12] G. Chen, Z. Lu, Z. Zhang, and Z. Sun, "Research on Hybrid Modified Cuckoo Search Algorithm for Optimal Reactive Power Dispatch Problem," *IAENG International Journal of Computer Science*, vol. 45, no. 2, pp. 328-339, 2018.

[13] H. Wang, D. Guo, X. Liang, W. Chen, G. Hu, and K. K. Leang, “Adaptive vision-based leader--follower formation control of mobile robots,” *IEEE Trans. Ind. Electron.*, vol. 64, no. 4, pp. 2893–2902, 2017.

[14] A. N. Asl, M. B. Menhaj, and A. Sajedin, “Control of leader--follower formation and path planning of mobile robots using Asexual Reproduction Optimization (ARO),” *Appl. Soft Comput.*, vol. 14, pp. 563–576, 2014.

[15] C.-J. Lin, T.-H. S. Li, P.-H. Kuo, and Y.-H. Wang, “Integrated particle swarm optimization algorithm based obstacle avoidance control design for home service robot,” *Comput. Electr. Eng.*, vol. 56, pp. 748–762, 2016.

[16] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, “A novel particle swarm optimization algorithm with adaptive inertia weight,” *Appl. Soft Comput.*, vol. 11, no. 4, pp. 3658–3670, 2011.

[17] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, “Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients,” *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, 2004.

[18] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in *Evolutionary Computation Proceedings, 1998. IEEE World Congress*

on Computational Intelligence., The 1998 IEEE International Conference on, 1998, pp. 69–73.
[19] B. Tang, Z. Zhu, and J. Luo, "Hybridizing particle swarm optimization and differential evolution for the mobile robot global path planning," *Int. J. Adv. Robot. Syst.*, vol. 13, no. 3, p. 86, 2016.
[20] F. den Bergh and A. P. Engelbrecht, "A convergence proof for the particle swarm optimiser," *Fundam. Informaticae*, vol. 105, no. 4, pp. 341–374, 2010.
[21] H. Mo and L. Xu, "Research of biogeography particle swarm optimization for robot path planning," *Neurocomputing*, vol. 148, pp. 91–99, 2015.
[22] M. Senanayake, I. Senthooran, J. C. Barca, H. Chung, J. Kamruzzaman, and M. Murshed, "Search and tracking algorithms for swarms of robots: A survey," *Rob. Auton. Syst.*, vol. 75, pp. 422–434, 2016.
[23] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 813–825, 2002.

Bambang Tutuko was born in Pekalongan, Indonesia, in 1960. He received the bachelor degree in Electrical Engineering from Universitas Sriwijaya, Indonesia in 1987, and master degree in Control System from Institut Teknologi Bandung, Indonesia, in 1998. Currently, he is a Ph.D student in Informatic Engineering, Universitas Sriwijaya. In 1995, he joined the Department of Computer Engineering, Faculty of Computer Science, Universitas Sriwijaya as a Lecture. His current research interest includes; robotic, control system, and artificial intelligence.

Siti Nurmaini was born in Palembang, Indonesia, in 1969. This author became a Member (M) of IAENG in 2011. He received the bachelor degree in Electrical Engineering from Universitas Sriwijaya 1992, and master degree in Control System from Institut Teknologi Bandung, Indonesia, in 1998 and Ph.D. degree in Computer Science from Universiti Teknologi Malaysia in 2011. In 1995, he joined the Department of Informatics, Faculty of Computer Science, Universitas Sriwijaya as an Associate Professor. Her current research interest includes; deep learning, machine learning, robotic, and medical.

Saparudin was born in Pangkalpinang, Indonesia, in 1969. He received the bachelor degree in Mathematics Education from Universitas Sriwijaya, Indonesia in 1993, and master degree in Informatics Engineering from Institut Teknologi Bandung, Indonesia, in 2000 and Ph.D. degree in Computer Science from Universiti Teknologi Malaysia in 2012. In 1995, he joined the Department of Informatics, Faculty of Computer Science, Universitas Sriwijaya as an Associate Professor. His current research interest includes; image processing, computer vision, and pattern recognition.

Putri Sahayu was graduate student in Informatic Engineering from Universitas Sriwijaya, Indonesia in 2017. He received the bachelor degree from Informatic Engineering from Universitas Sriwijaya, Indonesia in 2014. Her current research interest includes; pattern recognition and machine learning.