2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)
Barcelona, Spain, Sept. 7-9, 2016

Particle Swarm Optimization-Based Method for Navigation of Mobile Robot in Unstructured Static and Time-Varying Environments

Irfan Jabandžić[1] and Jasmin Velagić[2]

Abstract— **The paper considers a problem of the mobile robot navigation in unstructured environments based on an improved Particle Swarm Optimization (PSO) method. It introduces a new fitness function divided into sub-functions, where each sub-function is responsible for accomplishing given objectives and imposed constraints. The main objectives of proposed sub-functions are to maximize the distance between the robot and a previously detected nearest obstacle, minimize distance from the robot to a global best position of the particles and the local minima problem avoidance. Also, proper constraints are included to prevent the robot from penetrating into restricted (prohibited) areas around detected obstacles or to avoid that particle moves too far away from the mobile robot's current position. The proposed PSO based planner generates a smooth collision free path with a low computational cost and approaches the goal position employing an adaptive control strategy. The adjustment and selection of suitable parameters values of PSO method are based on extensive simulations performed in various environments. The effectiveness of the proposed method has been verified with a series of simulations.**

I. INTRODUCTION

Mobile robots are increasingly used in industry, medical applications, search and rescue missions as well as in many other areas. To carry out their activities it is essential to solve a navigation problem in which the mobile robot should detect obstacles and avoid them during the goal position seeking. A trajectory should be generated in real-time on the basis of current sensor's measurements.

Many methods have been developed to solve this problem based on the potential fields [1], genetic algorithms [2], fuzzy logic [3] and neural networks [4]. In static environments, a large number of methods that provide the optimal path of the mobile robot from the start to the goal position are implemented, such as A*, Rapidly-exploring Random Trees (RRT), Probabilistic Roadmap Method (PRM) and potential field method [5]-[7]. Often, the mobile robots perform tasks in unstructured environments with moving obstacles. Due to unlimited possibilities of the mobile robot environment configurations, a universal method for navigation of robots in unstructured dynamic environments has not yet been implemented. For overcoming this problem, different path planning methods are employed, especially D* and dynamic window approach (DWA) [8], [9]. Also, there are already many researches for the path planning in a dynamic environment with stochastic optimization [10], [11].

[1]Irfan Jabandžić is with Faculty of Electrical Engineering, Department of Automatic Control and Electronics, University of Sarajevo, 71000 Sarajevo, Bosnia and Herzegovina ijabandzic1@etf.unsa.ba

[2]Jasmin Velagić with Faculty of Electrical Engineering, Department of Automatic Control and Electronics, University of Sarajevo, 71000 Sarajevo, Bosnia and Herzegovina jasmin.velagic@etf.unsa.ba

Due to complexity of the motion planning problem, many heuristic and metaheuristic methods have been developed or applied recently. They generally exhibited better performance than the classic methods in terms of computational burden. However, heuristic methods do not guarantee to find a solution, but if they find the solution, it is done in much shorter time than exact methods [12]. One of widely used heuristic methods to solving the problem of mobile robot motion planning is the PSO method. This method was successfully applied to solve optimization problems in many fields such as: optimization of functions, training artificial neural networks, developing fuzzy systems and other fields [13], [14]. There are more studies of using the PSO method for solving the problem of mobile robot navigation in static and time-varying environments. The motion of the robot in such environments is still an unresolved and topical issue. The standard PSO method does not produce successful results. Thus, a large number of modifications that attempts to accomplish a successful motion of the mobile robot in such environments [15]-[17] was introduced. The main drawback of the PSO method is falling into a local minima. This deficiency was tried to be corrected by certain modifications of the standard PSO method [18], [19].

The problem of mobile robot navigation has multiple objectives, such as reaching the goal position, avoiding obstacles, minimize the traveling distance, etc. Thus, the optimization problem is based on the minimization of functions which adequately reflects these goals. In this paper we proposed the method for solving navigation problem using modified PSO method. In this method various fitness sub-functions and their imposed constraints are defined in order to minimize the robot path length toward the goal position and maximize the safety of robot in both unstructured static and dynamic environments. Also, this method prevents the robot to get trapped in a local minimum with simultaneous satisfying the real-time constraints imposed by an environment. The proposed method was tested on the kinematic model of mobile robot within the MobileSim program environment.

This paper is organized as follows. Section II describes the standard PSO method. The proposed modifications of the standard PSO method for the mobile robot motion in static and dynamic environments are given in Section III. Section IV presents the the obtained results in different scenarios, while Section V summarizes the paper with brief conclusion.

II. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a stochastic method based on the population of particles, where the particles

978-1-5090-0658-8/16/$31.00 ©2016 IEEE

change their own positions within the search space over time [20]. The position of a particle is changed according to its own experience and those of its neighbors. Each particle memorizes its best position, called a personal best and compares its current position to it. If the current position is better than the personal best, then particle's personal best is updated. For the whole population (swarm), there is one best solution, called a global best, which is also memorized and updated over time. An appropriate fitness function is used to evaluate the position of particles.

Each particle is described with three vectors: the position in n-dimensional search space $\mathbf{x}_i = (x_{i_1}, x_{i_2}, ..., x_{i_n})^T$, the personal best $\mathbf{p}_i = (p_{i_1}, p_{i_2}, ..., p_{i_n})^T$ and the velocity $\mathbf{v}_i = (v_{i_1}, v_{i_2}, ..., v_{i_n})^T$. The velocity presents the change of particle position over time. The fitness function for this optimization method is $f : R^n \to R$. Let us assume that there is m particles with position $\mathbf{x}_i \in R^n$ and velocity $\mathbf{v}_i \in R^n$. Further, let us assume that $\mathbf{p}_i$ is the personal best for each particle and $\mathbf{g}$ is the best possible solution among those particles at the instant k (the global best). The velocity and position of each particle are updated according to the following equations [21]:

$$\mathbf{v}_i^{k+1} = w\mathbf{v}_i^k + c_1\mathbf{r}_1\left(\mathbf{p}_i^k - \mathbf{x}_i^k\right) + c_2\mathbf{r}_2\left(\mathbf{g}^k - \mathbf{x}_i^k\right), \quad (1)$$

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1}, \quad (2)$$

where i denotes index of particle, c_1 and c_2 are control coefficients defined by the user ($0 \le c_1 \le 2$, $0 \le c_2 \le 2$), $\mathbf{r}_1$ and $\mathbf{r}_2$ are vectors of random numbers from [0,1] produced at each updated velocity and w_i is an inertia weight ($0 \le w_i \le 1.2$). The chosen ranges of c_i and w_i are typically used [20]. The coefficients $c_i, i = 1,2$, control the effect of the personal and global guides. The inertia weight controls the trade-off between the global and local experience. Large values of this parameter leads to global search, while small values leads to fine local search, which is suitable when the algorithm converges [22].

The PSO method involves the following steps [20]:

1) Initialize the random population with random positions and velocities.
2) Determine the personal best for each particle in the swarm and the global one for whole population using a convenient fitness function $\mathbf{p}_i \leftarrow \mathbf{x}_i$, $\mathbf{g} \leftarrow \underset{x}{\text{argmin}} f(x), i = 1,..,m$.
3) Create random vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ for each particle and dimension of the search space.
4) Update the position for each particle (Fig. 1):

$$\mathbf{x}_i^{k+1} \leftarrow \mathbf{x}_i^k + \mathbf{v}_i^{k+1}. \quad (3)$$

5) Update the velocity for each particle:

$$\mathbf{v}_i^{k+1} \leftarrow w\mathbf{v}_i^k + c_1\mathbf{r}_1\left(\mathbf{p}_i^k - \mathbf{x}_i^k\right) + c_2\mathbf{r}_2\left(\mathbf{g}^k - \mathbf{x}_i^k\right). \quad (4)$$

6) If the current fitness of the particle is better than its personal best, memorize the current position as the particle's new personal best, $f(\mathbf{x}_i) < f(\mathbf{p}_i)$, for $i = 1,...,m$.

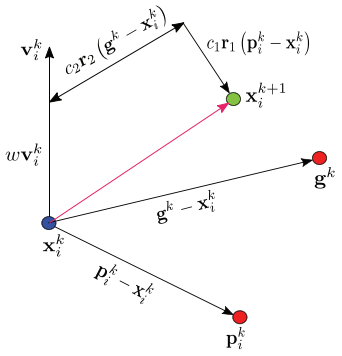

Fig. 1. Position and velocity update of the particle

7) If the current fitness of any particle in the swarm is better than the global best fitness of the swarm, then position of this particle is memorized as new global best position of the swarm, $f(\mathbf{x}_i) < f(\mathbf{g})$, $\mathbf{g} \leftarrow \mathbf{x}_i$, for $i = 1,...,m$.
8) Return to step 3 if neither stopping condition for the algorithm is reached.

The main problem of PSO method is the local minima problem. The local minima prevents the robot to reach the goal position since the robot gets trapped in an infinite loop. Despite this problem, this method is easy to implement and capable to converge very fast toward the global optimum. It implements only a few parameters and updates only a few variables over time. With suitably selected parameters and with the modification of the main algorithm, this method is often used in mobile robotics for navigation problems.

III. PROPOSED PSO-BASED METHOD

The first step in the proposed method is to detect the type of obstacles (static or dynamic) in the mobile robot's environment. The position and type of obstacles are a prior unknown for the robot and will be determined based on sensor's measurements during the robot motion. It will be assumed that 3D environment can be presented as 2D space, because all mobile robot's sensors are placed in one plane. The main objective of the mobile robot navigation is to avoid all obstacles and safely reach a goal position. It is possible to extract information about a moving obstacle by comparing data sets obtained from two consecutive scans. In our approach those data sets are acquired at the same robot location. It means that the robot does not change its location between two consecutive sensor measurements. If two consecutive sensor measurements are the same, then the detected obstacle is considered to be static. Otherwise, the obstacle is moving and the environment is a time-varying.

The time interval $t_2 - t_1$ between computing two adjacent local goals (x_{p_1}, y_{p_1}) and (x_{p_2}, y_{p_2}) is very short and the motion of obstacles is approximated by a straight-line trajectory $y = y_0 + kx$ with constant velocity v under this time interval. In accordance with two consecutive sensors measurements,

the path of the dynamic obstacle and its velocity are:

$$y = y_{p_1} + \frac{y_{p_2} - y_{p_1}}{x_{p_2} - x_{p_1}} (x - x_{p_1}), \tag{5}$$

$$v = \frac{\sqrt{(y_{p_2} - y_{p_1})^2 + (x_{p_2} - x_{p_1})^2}}{t_2 - t_1}. \tag{6}$$

Also, it is important to take into consideration the size of the mobile robot so it does not collide with any obstacles. The mobile robot is observed as a material point and around each detected obstacle a restricted area is constructed and placed based on the dimensions of the mobile robot (Fig. 2).

With proper simplifications, the mobile robot can be observed in a 2D environment. Therefore, the fitness function is represented as $f : R^2 \to R$. In 2D space each particle can be described with its position in the plane $\mathbf{x}_i = (x_i, y_i)^T$, its personal best $\mathbf{p}_i = (p_{i_1}, p_{i_2})^T$ and its velocity $\mathbf{v}_i = (v_{i_1}, v_{i_2})^T$. To address the local minima problem, this paper introduces the following modification of the standard PSO method. The global best is not updated according to step 7 of the standard PSO method. It is assumed to be global best of the whole swarm for the current iteration (state). The previous global states are forgotten for the sake of being adaptive to the changes in the environment. Therefore, the global best position is updated as follows:

$$\mathbf{g} \leftarrow \underset{\mathbf{x}_i}{\operatorname{argmin}} f(\mathbf{x}_i), i = 1, .., m. \tag{7}$$

The main purpose of the fitness function is to provide an avoidance of the obstacles and to minimize a distance from the goal position. It is also important to generate optimal trajectories when the robot senses obstacles. In this paper a new fitness function is proposed to satisfy conditions in unstructured static and dynamic environments. The proposed fitness function is divided into sub-functions which are responsible for fulfillment of different tasks.

Regardless of the environment, the mobile robot has to reach its destination and the fitness function should maintain the minimal distance from robot's destination. The sub-function responsible for implementation of this task is directly proportional to the distance between particles and the robot destination (goal position). If the position of the particle is (x_i, y_i) and the goal position is (x_g, y_g) then this sub-function is represented as:

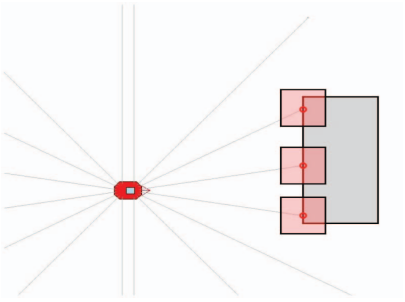

Fig. 2. Restricted areas placed around detected obstacles

$$f_1(\mathbf{x}_i) = \sqrt{(x_i - x_g)^2 + (y_i - y_g)^2}. \tag{8}$$

A. Sub-functions for Static Environment

In static environments the fitness function should maintain the maximal distance of the robot from detected obstacles. The part of the overall fitness function related to this is inversely proportional to the distance between the robot and the nearest detected obstacle. Let us assume that there are n_{do} detected obstacles with coordinates (x_{p_i}, y_{p_i}) in 2D space. The coordinates of the mobile robot (the center of mass) are (x_c, y_c). The distance between each obstacle and the mobile robot is computed as follows:

$$d_i = \sqrt{(x_c - x_{p_i})^2 + (y_c - y_{p_i})^2} \text{ for } i = 1, ..., n_{do}. \tag{9}$$

The coordinates of the nearest obstacle are found $(x_{p_{nr}}, y_{p_{nr}})$. According to this, the sub-function is given by:

$$f_2(\mathbf{x}_i) = \frac{1}{\sqrt{(x_i - x_{p_{nr}})^2 + (y_i - y_{p_{nr}})^2}}. \tag{10}$$

Large obstacles, such as walls, can often induce unwanted oscillatory motion. To mitigate this, a sub-function is introduced to maximize the distance of the robot to the previously detected nearest obstacle, (x_{p_p}, y_{p_p}). This sub-function is inversely proportional to this distance as follows:

$$f_3(\mathbf{x}_i) = \frac{1}{\sqrt{(x_i - x_{p_p})^2 + (y_i - y_{p_p})^2}}. \tag{11}$$

B. Sub-functions for Dynamic Environment

In case that the mobile robot detects moving obstacle in its environment, avoidance of this obstacle becomes the main problem. First, it is necessary to determine if this obstacle is going to affect the mobile robot. If that is the case, the mobile robot has to leave its current position due to a collision avoidance. The fitness sub-function responsible for departure of the mobile robot from its current position is inversely proportional to distance between the particle and the robot's current position:

$$f_4(\mathbf{x}_i) = \frac{1}{\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}}. \tag{12}$$

To avoid a motion of the mobile robot towards a dynamic obstacle, the fitness sub-function is introduced which provides that next local goal of the mobile robot is as far as possible from the central point of the dynamic obstacle's trajectory. Coordinates of that point are $x_s = \frac{x_c + x_{d_2}}{2}$ and $y_s = \frac{y_c + y_{d_2}}{2}$, where (x_{d_2}, y_{d_2}) are coordinates of dynamic obstacle obtained during last sensor's measurement. This fitness function is represented as:

$$f_5(\mathbf{x}_i) = \frac{1}{\sqrt{(x_i - x_s)^2 + (y_i - y_s)^2}}. \tag{13}$$

Besides these sub-functions, proper constraints are imposed to the fitness function in order to generate the collision-free path. These constraints are included to prevent the robot from penetrating into restricted (prohibited) areas around detected obstacles or to avoid that particle moves too far away from the mobile robot's current position. They are defined by parameters k_1 and k_2, respectively. For k_1 and k_2 it is desirable to choose very large values, therefore it is certain that particles near obstacles and particles outside range of robot's sensors are discarded. In our case we have chosen $k_1 = 10000$ and $k_2 = 10000$. The restricted areas around detected obstacles are squares that are 0.8 meters high and wide. These dimensions are determined based on performed simulations. To determine if a particle is too far from the mobile robot, a distance $d_{c_i} = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2}$ for $i = 1, ..., m$ is compared with a predefined maximum distance.

Taking into account all fitness sub-functions and constraints, the whole final fitness function becomes:

$$f(\mathbf{x}_i) = \sum_{j=1}^{5} w_j f_j(\mathbf{x}_i) + w_6 k_1 + w_7 k_2. \quad (14)$$

Values of parameters w_j, where $j = 1, ..., 5$, depend on the requirements of a certain task. If it is the priority to minimize the length of the path towards the goal position, then the parameter w_1 should has the largest value. The safe path is generated in the case of the largest values of w_2 and w_3. If dynamic obstacles are the main concern, the highest values for w_4 and w_5 should be selected. Parameters w_6 and w_7 are equal to zero if there is no need for constraints and equal to one if particle enters any restricted area.

Considering the purpose of each sub-function of the whole fitness function, it can be concluded that the proposed fitness function, besides enabling the mobile robot to the goal position and avoid static obstacles, is also taking care of various types of dynamic obstacles and resolving problem of oscillatory motion. By eliminating oscillatory motion of the robot, the occurrence of the local minima is minimized.

The pseudo code of the proposed modification is listed in Algorithm 1. The effectiveness and time-consuming performance of the proposed method will be verified through simulations in the next section.

Algorithm 1 PSO based algorithm for path generation

```
Input: Number of particles
Output: New robot location
for each particle in population do
    Initialize particle
    Calculate the fitness function according to (14)
    Determine personal best
end for
Determine global best
Move robot to new position
while destination is not reached do
    Carry out two consecutive sensor measurements
    Determine environment's behavior
    for each particle in population do
        Create random vectors r1 and r2
        Update the position for each particle according to (3)
        Update the velocity for each particle according to (4)
        Calculate the fitness function according to (14)
        Update the personal best
    end for
    Update the global best according to (7)
    Move the robot to new position
end while
```

IV. SIMULATION RESULTS

The simulations were implemented within the MobileSim program package with C++ function library ARIA and PSO-based algorithm was implemented using MATLAB. The kinematic model of the Pioneer 3-DX mobile robot and used adaptive controller are taken from [23]. In order to verify the effectiveness and quality of the proposed algorithm the following parameters will be considered: number of iterations, length of the obtained path and mean distance.

For the proposed PSO-based method it is necessary to determine which parameters to use. Based on our performed simulations it was observed that satisfactory results were obtained with 100 particles in the swarm and 100 iterations of the algorithm. In order to enable convergence of the algorithm, inertia weight values are set to be lower than 1 in regard to (4). To achieve a trade-off between local and global searching, the inertia weight is set to linearly decrease from the value $w_{max} = 0.9$ to $w_{min} = 0.4$. Thus, the value of the inertia weight per iteration is computed as:

$$w = w_{max} - i\frac{(w_{max} - w_{min})}{i_{max}}.$$

Then, parameters c_1 and c_2 should be determined. Their values have to be large enough to enable successful exploration of the search space. Also, for unimodal problems it is desirable to select larger value for c_2 then for c_1 [24]. To prove that better results are obtained with larger value for c_2 than for c_1, the motion of mobile robot will be observed in two different environments in the next subsection. Distance traveled from the start to the goal position and a mean distance from the robot to obstacles will be used as indicators of the better selection.

A. Influence of Control Parameters for Unknown Static Environments

In the case of larger value of the parameter c_2, particles tend to converge to the best solution of the whole swarm rather than their particular solutions. This provides particles to be retained within a neighborhood near the current mobile robot location. The results obtained for different values of the parameter c_1, with fixed values of c_2, in two different environments are represented in Figs. 3-6.

From the obtained results, it can be concluded that for larger value of c_1, particles diverge too far away from current positions of the mobile robot. Furthermore, it reduces the number of the particles that actively participate in exploration

of the search space. With larger value for parameter c_1, the number of iterations of the algorithm is larger. Also, the path is longer while the mean distance between the robot and the obstacles becomes shorter (see Tables I and II). Further simulations were performed with stated values of the PSO method parameters w_i in the next subsections.

TABLE I

CHARACTERISTIC INDICATORS FOR THE FIRST SCENARIO

Parameters		Characteristic indicators		
c_1	c_2	No. iterations	Path length [m]	Mean distance [m]
1	2	26	24.2	2.49
2	2	28	25.1	2.36

TABLE II

CHARACTERISTIC INDICATORS FOR THE SECOND SCENARIO

Parameters		Characteristic indicators		
c_1	c_2	No. iterations	Path length [m]	Mean distance [m]
1	2	39	34.60	1.60
2	2	50	39.43	1.57

B. Influence of Fitness Function Parameters for Unknown Static Environments

Now, it is necessary to determine the parameters of the fitness function and investigate their influence on the path planning and execution performance. For static environments, parameters w_1, w_2 and w_3 are of concern. It is estimated that optimal value for parameter w_3 is between 60 and 80 percentages of parameter w_2. For each scenario 12 simulations with different w_i parameters values were performed. Results of these simulations for the first scenario are presented in the Table III. Dash in the table means robot's inability to reach the goal position, as a result of the collision with obstacles or the local minima problem.

The minimal number of iterations and the minimal path length were accomplished for parameter values $w_1 = 1$, $w_2 = 0.4$ and $w_3 = 0.24$. The mobile robot's motion is the safest with parameters $w_1 = 0.4$, $w_2 = 1$ and $w_3 = 0.8$. Furthermore, trade-off between these two demands is obtained for parameter values $w_1 = 1$, $w_2 = 1$ and $w_3 = 0.8$. For these three set of parameters, mobile robot's trajectory is presented in Figs. 7-9.

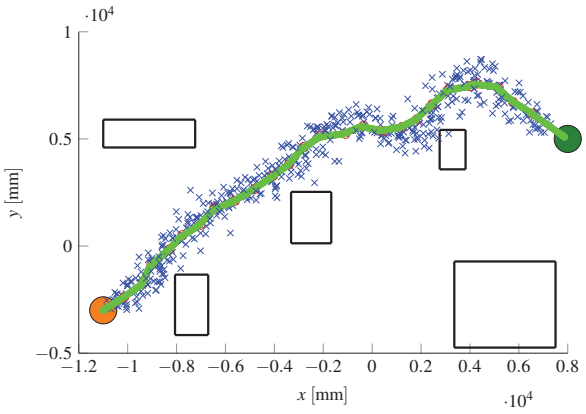

Fig. 3. Robot path for $c_1 = 1$ and $c_2 = 2$ (first scenario). The orange and green colored circles represent the start and the goal position, respectively.

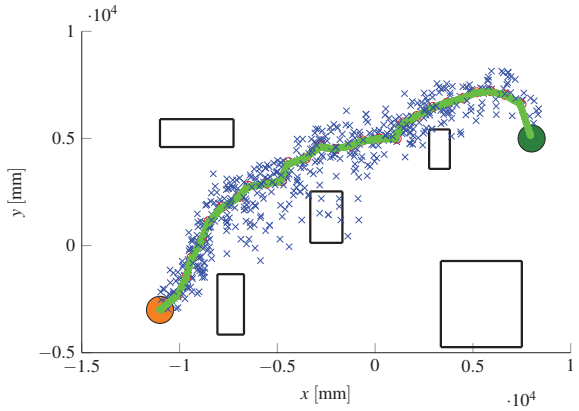

Fig. 4. Robot path for $c_1 = 2$ and $c_2 = 2$ (first scenario)

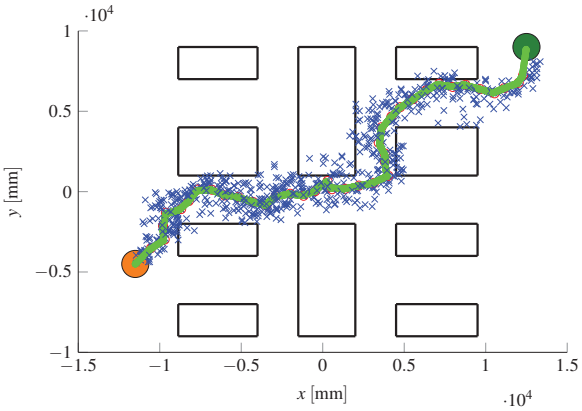

Fig. 5. Robot path for $c_1 = 1$ and $c_2 = 2$ (second scenario)

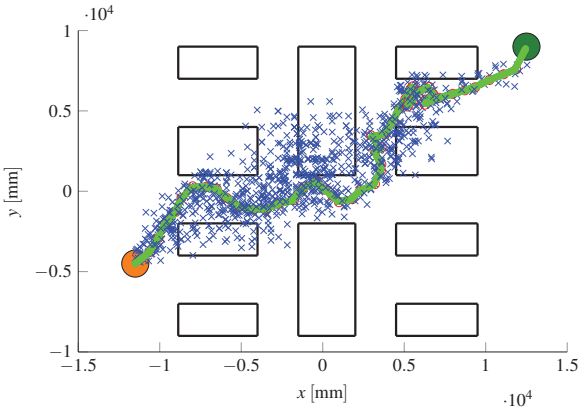

Fig. 6. Robot path for $c_1 = 2$ and $c_2 = 2$ (second scenario)

As in the first scenario, in the second one the same results were obtained (Table IV). The shortest path is generated for parameter's values $w_1 = 1$, $w_2 = 0.4$ and $w_3 = 0.24$ (Fig. 10). The safest path is obtained for parameters $w_1 = 0.4$, $w_2 = 1$ and $w_3 = 0.8$ (Fig. 11) and a trade-off is accomplished for parameters $w_1 = 1$, $w_2 = 1$ and $w_3 = 0.8$ (Fig. 12).

TABLE III

CHARACTERISTIC INDICATORS FOR THE FIRST SCENARIO

Parameters			Characteristic indicators		
w_1	w_2	w_3	Iterations	Path length [m]	Mean distance [m]
1	0.3	0.18	-	-	-
1	0.4	0.24	23	22.70	1.86
1	1	0.6	24	23.80	2.22
1	1.6	0.96	24	23.90	2.48
1	0.3	0.24	-	-	-
1	0.4	0.32	25	23.50	1.89
1	1	0.8	25	23.80	2.3
1	1.6	1.28	25	24.20	2.65
0.4	1	0.6	26	24.50	2.97
1.6	1	0.6	-	-	-
0.4	1	0.8	26	25.30	3.03
1.6	1	0.8	-	-	-

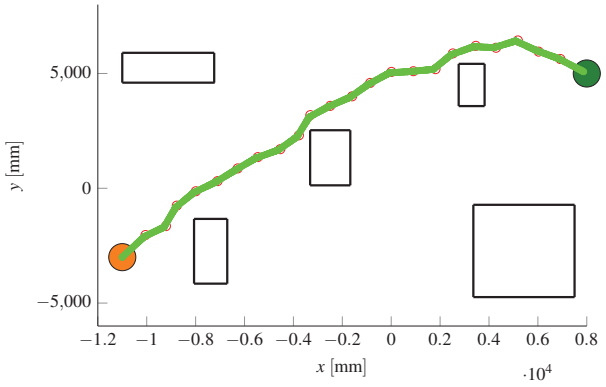

Fig. 7. Robot path for $w_1 = 1$, $w_2 = 0.4$ and $w_3 = 0.24$ (first scenario)

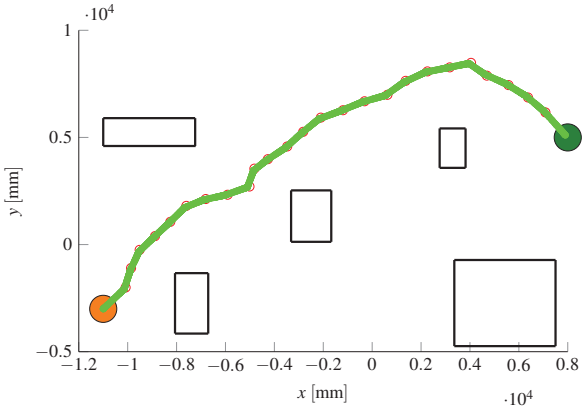

Fig. 8. Robot path for $w_1 = 0.4$, $w_2 = 1$ and $w_3 = 0.8$ (first scenario)

C. Comparative Study for Unknown Static Environment

In order to assess the effectiveness of the proposed method in an unknown static environment, the proposed PSO based method method will be compare with the representative approaches, such as A*, RRT, PRM and potential field (PF) method. In the comparison, the following parameters are considered: the path length from the start to the goal position, the computation time (CT) and the pulse time (PT). The pulse time refers to average computational time per iteration. The comparison will be done for the environment represented in

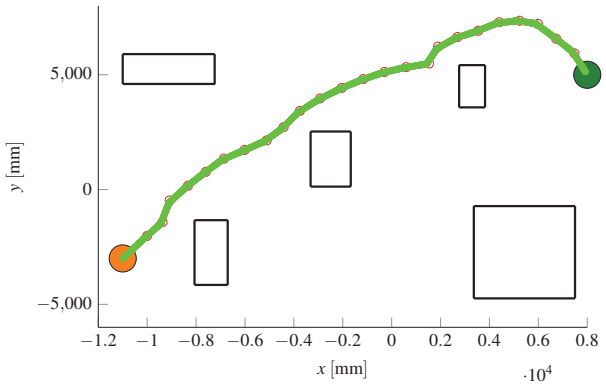

Fig. 9. Robot path for $w_1 = 1$, $w_2 = 1$ and $w_3 = 0.8$ (first scenario)

TABLE IV

CHARACTERISTIC INDICATORS FOR THE SECOND SCENARIO

Parameters			Characteristic indicators		
w_1	w_2	w_3	Iterations	Path length [m]	Mean distance [m]
1	0.3	0.18	-	-	-
1	0.4	0.24	34	31.67	1.39
1	1	0.6	34	32.22	1.58
1	1.6	0.96	38	34.90	1.69
1	0.3	0.24	-	-	-
1	0.4	0.32	34	31.93	1.45
1	1	0.8	37	33.60	1.61
1	1.6	1.28	39	35.30	1.69
0.4	1	0.6	42	38.00	1.87
1.6	1	0.6	-	-	-
0.4	1	0.8	45	38.90	2.24
1.6	1	0.8	-	-	-

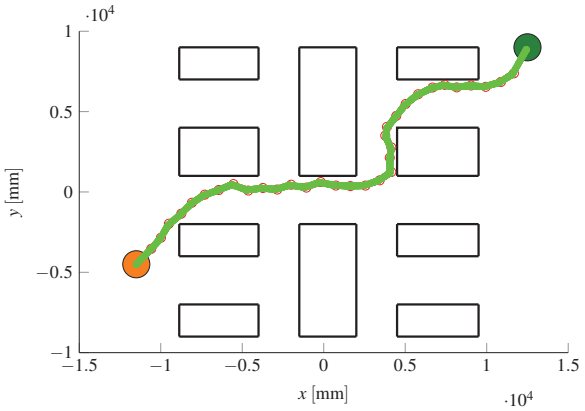

Fig. 10. Robot path for $w_1 = 1$, $w_2 = 0.4$ and $w_3 = 0.24$ (second scenario)

TABLE V

REAL-TIME CONSTRAINTS ANALYSIS FOR SCENARIO IN FIG. 12

Method	Path length [m]	CT [s]	PT [ms]
PSO based	12.6092	0.3814	12.9452
A*	14.2710	0.4352	14.2791
RRT	13.5138	0.3918	13.1239
PRM	13.7091	0.3970	13.2861
PF	13.4279	0.3823	12.9765

(Fig. 12). The obtained results are presented in Table V. From obtained results is obvious that the proposed PSO

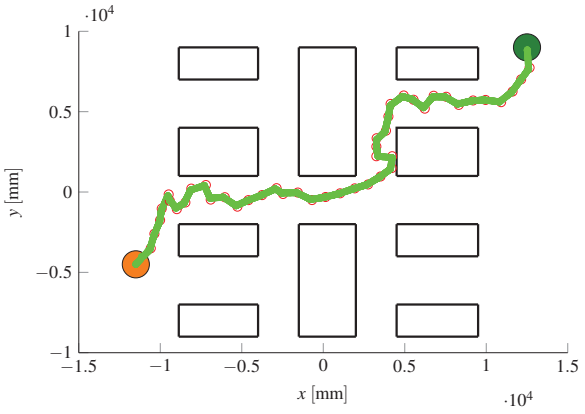

Fig. 11. Robot path for $w_1 = 0.4$, $w_2 = 1$ and $w_3 = 0.8$ (second scenario)

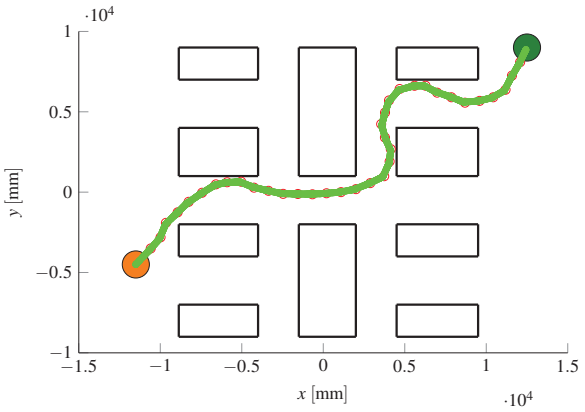

Fig. 12. Robot path for $w_1 = 1$, $w_2 = 1$ and $w_3 = 0.8$ (second scenario)

method produces the shortest path with the smallest computation and pulse time. Therefore, the proposed method helps the robot to avoid a local minimum with satisfying imposed real-time constraints, that will be demonstrated in the next subsection.

D. Unknown Static Environment with Local Minima

A particular attention is paid to avoid a local minimum that is accomplished using the proposed modification. It neglects the previous best solution of the whole swarm and takes the best solution of the particle which has the best fitness in that time instant. The results obtained using the proposed PSO based method is shown in Fig. 13, where the goal position is achieved escaping the local minima.

E. Dynamic Environments

In the time-varying environment, besides above defined parameters, parameters w_4 and w_5 are also of the great significance. The larger values of these parameters tend to drag the mobile robot further away from the assumed trajectory of dynamic obstacle. However, too large values might distract the mobile robot from the motion directed towards the goal position. Values of these two parameters should be larger than value of the parameter w_2 producing stronger reaction of the mobile robot to the dynamic obstacle than to the static ones. Experimentally it was determined that

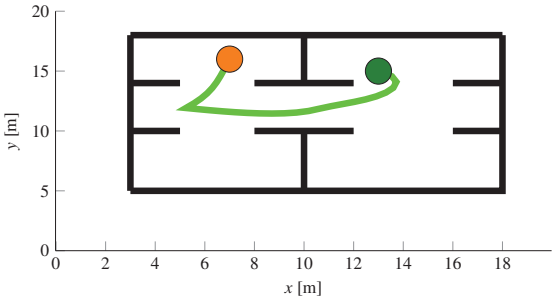

Fig. 13. Path planning within static environment with C shaped obstacles.

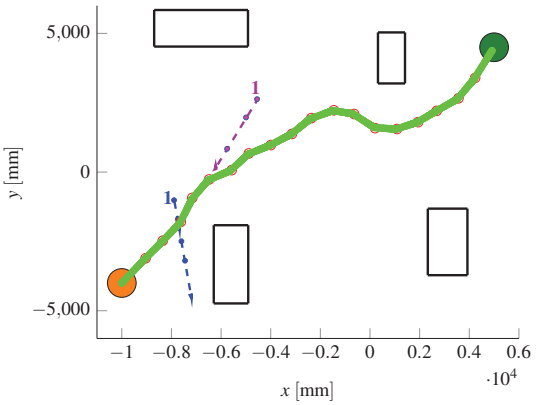

Fig. 14. Path of the mobile robot in case of two dynamic obstacles

optimal values for these two parameters are $w_4 = 1.5w_2$ and $w_5 = 1.5w_2$.

For simulations that were carried out for unstructured dynamic environments, parameters values are: $w_1 = 1$, $w_2 = 1$, $w_3 = 0.8$, $w_4 = 1.5$ and $w_5 = 1.5$. The results for the scenario where dynamic obstacles move along a straight line toward the mobile robot are shown in Fig. 14. The start position of moving obstacles is denoted by the number 1. During its motion the robot first detects a dynamic obstacle (blue colored) that moves from its sideways and whose motion can discourage the robot. Then the robot stops between two consecutive sensor's readings to decide whether detected obstacle is approaching or running away from the robot. After several readings the robot is assured that can continue towards the goal position. The next dynamic obstacle (pink colored) comes straight ahead from front of the robot. In order to avoid this obstacle, the mobile robot determines a new subgoal point that is as far away as possible from an estimated obstacle trajectory and moves with the maximum speed to reach it. The mobile robot has successfully avoided this obstacle and navigated efficiently to the goal position.

In the next scenario two robots move toward their goals whereas they paths may cross each other, where a motion of one robot is unknown for another. Both robots successfully move to their goals while avoiding each other (Fig. 15).

The advantages of the proposed PSO method over other existing methods for dynamic and online path planning, such as D* and Dynamic Window Approach (DWA), will be investigated by simulations for the scenario shown in

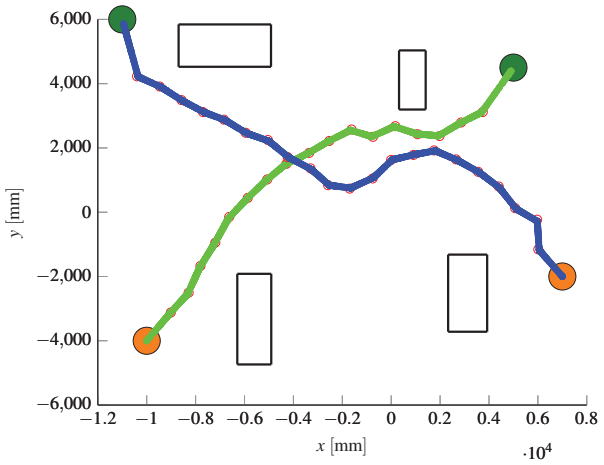

Fig. 15. Mobile robots motions with paths crossing

TABLE VI

REAL-TIME CONSTRAINTS ANALYSIS

Method	Path length [m]	CT [s]	PT [ms]
PSO based	22.4176	0.4719	14.4960
D*	23.3864	0.4680	15.1724
DWA	23.2650	0.4498	14.2307

Fig. 15. In this case, the green curve represents the robot motion whereas the blue one represents the obstacle motion. The obtained results indicate that the PSO based method generates the shortest path with satisfactory computation time, while the smallest computation time is achieved by DWA method (Table VI).

Finally, it can be concluded that mobile robot moves successfully among static and dynamic obstacles and found a collision-free path from the start to the goal position. Therefore, the proposed method tends to minimize total path planning time and the path length while avoiding obstacles.

V. CONCLUSIONS

The novelty of this paper concerns the modification of the standard PSO method to be successfully used in unknown static and dynamic environments. The modification of the fitness function is based on the navigation objectives and imposed constraints. These include the maximization of the mobile robot safety, avoidance of the dynamic obstacles and the local minima problem solving. With a suitably chosen parameters, the proposed method successfully overcomes the obstacles and finds a collision-free path from the start to the goal position in dynamic environments.

The future work will involve expanding the suggested fitness function with additional factors, such as minimum path length and acceleration. Also, the proposed method will be compared with others for a real robot, considering the following performance indexes: the final cost value, less chance to fall into local minimum and computation time.

REFERENCES

[1] M. C. Mora and J. Tornero, "Path planning and trajectory generation using multi-rate predictive artificial potential fields," in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 2008, pp. 2990-2995.

[2] M. Gao, J. Xu, J. Tian and H. Wu, "Path planning for mobile robot based on chaos genetic algorithm," in Proc. International Conference on Natural Computation, Jinan, China, 2008, pp. 409-413.

[3] I. Saboori, M. Menhaj and B. Karimi, "Optimal robot path planning based on fuzzy model of obstacles," in Proc. IEEE Annual Conference on Industrial Electronics, Paris, France, 2006, pp. 383-387.

[4] S. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, vol. 34, no. 1, pp. 718-724, 2004.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotic Research*, vol. 30, no. 7, pp. 846-894, 2011.

[6] D. Ferguson, M. Likhachev and A. Stentz, "A guide to heuristic-based path planning," in Proc. ICAPS Workshop on Planning under Uncertainty for Autonomous Systems, 2005, pp. 1-10.

[7] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in Proc. IEEE International Conference on Robotics and Automation, 1991, pp. 1398-1404.

[8] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613-1643, 2008.

[9] D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no, 1, pp. 2333, 1997.

[10] C. Park, J. Pan and D. Manocha, "Real-time optimization-based planning in dynamic environments using GPUs," IEEE International Conference on Robotics and Automation, 2013, pp. 4090-4097.

[11] D. N. Subramani, T. Lolla, P. J. Haley Jr. and P. F. J. Lermusiaux, "A stochastic optimization method for energy-based path planning," in book *Dynamic Data-Driven Environmental Systems Science*, pp. 347-358, 2015.

[12] E. Masehian and D. Sedighizadeh, "An improved particle swarm optimization method for motion planning of multiple robots," in book *Distributed Autonomous Robotic Systems*, pp. 175-188, 2013.

[13] T. Zhang and J. D. Cai, "A novel hybrid particle swarm optimisation method applied to economic dispatch," *International Journal of Bio-Inspired Computation*, vol. 2, no. 1, pp. 9-17, 2010.

[14] J. G. Lu, L. Zhang, H. Yang and D. Jie, "Improved strategy of particle swarm optimization algorithm for reactive power optimization," *International Journal of Bio-Inspired Computation*, vol. 2, no. 1, pp. 27-33, 2010.

[15] D. Gong, L. Lu and M. Li, "Robot path planning in uncertain environments based on particle swarm optimization," in Proc. IEEE Congress on Evolutionary Computation, 2009, pp. 2127-2134.

[16] Y. Zhang, D. Gong and J. Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," *Neurocomputing*, vol. 103, pp. 172-185, 2013.

[17] M. Yarmohamadi, "Improvement of robot path planning using particle swarm optimization in dynamic environments with mobile obstacles and target," *Advanced Studies in Biology*, vol. 3, no. 1, pp. 4353, 2011.

[18] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas and M. N. Vrahatis, "Improving the particle swarm optimizer by function "Stretching", in book: *Advances in Convex Analysis and Global Optimization*, pp. 445-457, 2001.

[19] G. I. Evers and M. B. Ghalia, "Regrouping particle swarm optimization: a new global optimization algorithm with improved performance consistency across benchmarks," in Proc. IEEE Conference on Systems, Man and Cybernetics, 2009, pp. 3901-3908.

[20] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in Proc. IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.

[21] Y. Fukuyama, "Fundamentals of particle swarm optimization techniques," in book: *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*. New York: Wiley, 2008.

[22] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in Proc. IEEE International Conference on Evolutionary Computation, 1998, pp. 69-73.

[23] D. Osmanković and J. Velagić, "Gradient based adaptive trajectory tracking control for mobile robots," In Proc. ICAT, 2013, pp. 1-6.

[24] R. Poli, J. Kennedy and T. Blackwell,"Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33-57, 2007.