Research Article

Same Fuzzy Logic Controller for Two-Wheeled Mobile Robot Navigation in Strange Environments

Awatef Aouf,1,2 Lotfi Boussaid,2 and Anis Sakly3

1National Engineering School of Sousse (ENISO), University of Sousse BP 264 Sousse Erriadh 4023, Sousse, Tunisia
2Laboratory EµE, Faculty of Sciences of Monastir, University of Monastir, Av. Ibn El Jazzar Skanes 5019, Monastir, Tunisia
3Research Unit of Industrial Systems Study and Renewable Energy (ESIER), National Engineering School of Monastir (ENIM), University of Monastir, Av. Ibn El Jazzar Skanes 5019, Monastir, Tunisia

Correspondence should be addressed to Awatef Aouf; aoufawatef@gmail.com

Received 24 September 2018; Revised 2 December 2018; Accepted 18 December 2018; Published 3 January 2019

Academic Editor: Shahram Payandeh

Copyright © 2019 Awatef Aouf et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

For any mobile device, the ability to navigate smoothly in its environment is of paramount importance, which justifies researchers’ continuous work on designing new techniques to reach this goal. In this work, we briefly present a description of a hard work on designing a Same Fuzzy Logic Controller (S.F.L.C.) of the two reactive behaviors of the mobile robot, namely, “go to goal obstacle avoidance” and “wall following,” in order to solve its navigation problems. This new technique allows an optimal motion planning in terms of path length and travelling time; it is meant to avoid collisions with convex and concave obstacles and to achieve the shortest path followed by the mobile robot. The efficiency of employing the proposed navigational controller is validated when compared to the results from other intelligent approaches; its qualities make of it an efficient alternative method for solving the path planning problem of the mobile robot.

1. Introduction

A mobile robot is able to navigate intelligently in an uncontrolled environment without the need for physical or electromechanical guidance devices using different control techniques based on sensors. This autonomous agent is being increasingly used nowadays in various fields such as security, medicine, industry, space exploration, rescue operations, disaster relief, etc. The mobile robot is flexible enough to perform these tasks satisfactorily in both static and dynamic environments.

How to avoid obstacles during robot navigation, either global or local, is the main concern of robotics researchers. Global navigation means that the environment is completely known to the mobile robot. Different approaches have indeed been applied looking for a solution to global navigation problems, notably Artificial Potential Field [1], Grids [2], Visibility Graph [3], Cell Decomposition [4], and Voronoi Graph [5]. In local navigation, by contrast, the environment is completely or partially strange to the mobile robot which controls its motion using different equipped sensors. Various researches have been conducted so far to tackle the problem of local navigation like Vector Field Histogram [6] which employs a two-dimensional Cartesian histogram grid as a world model. This model is updated continuously using onboard range sensors, Dynamic Window Approach [7]. This strategy is a local planner which calculates the optimal collision-free velocity for a mobile robot, Neural Network [8], Neurofuzzy [9], Particle Swarm Optimization [10], Genetic Algorithm [11], Ant Colony Optimization Algorithm [12], Cuckoo Algorithm [13], Simulated Annealing Algorithm [14], and Fuzzy Logic.

The fuzzy logic controller, first developed by Zadeh [15], is widely employed in various engineering applications including the mobile robotics.

Ren et al. [16] used the fuzzy logic techniques to solve navigation problems in a strange and changing environment. In [17], Pradhan et al. applied the fuzzy logic controller to ensure the navigation of one thousand robots in a wholly unknown environment. It was built using triangular, trapezoidal, and Gaussian membership functions. Then performances were compared, and Gaussian membership functions were proven to be more powerful and efficient. In their work, Yousfi et al.
constructed a fuzzy logic controller to solve the navigation problems of the mobile robot. They developed the Gradient method to optimize the variables of the membership functions of the Sugeno fuzzy controller. In [19], authors presented a behavior-based control using fuzzy logic controller for mobile robot navigation in an obscure environment. They designed four basic tasks: go to goal behavior, obstacle avoidance behavior, tracking behaviour, and deadlock disarming behavior.

The navigation of the mobile robot based on equipped sensors using fuzzy logic controller is proposed in [20, 21]. Wu et al. [22] designed a fuzzy logic with genetic algorithm controller for mobile pipeline robot in a pipe environment. The fuzzy controller furnished the initial membership function and the fuzzy rules. The genetic algorithm took the best membership value for optimizing the fuzzy controller of path planning problems. In [23], Farooq et al. conducted a comparative study between Takagi-Sugeno and Mamdani fuzzy logic model for autonomous mobile robot navigation in unpredictable environments. They found out that Mamdani fuzzy model yielded better results and that ‘Takagi Sugeno fuzzy model took less memory space in the real-time microcontroller implementation. Reference [18] used a mobile robot equipped with three sensors and designed a fuzzy logic controller which guarantees autonomous navigation in strange environment. In [24], the author developed a real-time fuzzy logic control to achieve obstacle avoidance in an unknown environment.

Several recent works in the literature are interested in solving the mobile robot navigation problem by using approaches based on fuzzy logic, but they did not focus on local minimum problem in unknown environment [25–27]. As a result, they could only handle simple environment (without deadlock) while standing helpless in more complex environment (with deadlock).

The contribution of our work is to construct a suitable architecture based on two behaviours and same fuzzy logic controller (S.F.L.C.) for solving the navigation problem in complex and unknown environments. This control architecture ensures a safe and short trajectory. A comparative study is made to illustrate the performances of our approach against PSO-PID [28], ABC-PID, and FA-PID controllers.

This paper has eight main sections. Section 2 introduces the kinematic modeling of the differential mobile robot. The proposed path planning algorithm is discussed in Section 3. Section 4 presents the PID controller and describes two evolutionary algorithms. Section 5 gives the details of the fuzzy logic controller. Section 6 presents the final results from different simulations. A comparative study is described in Section 7. Section 8 concludes and outlines the future of our work.

2. Kinematic Model of the Mobile Robot

In this work, we have used the differential mobile robot Khepera III to simulate the suggested navigation algorithms. Khepera III is equipped with nine infrared sensors used for distance measurements, two DC motors, and two encoders to give its real position in the environment. The schematic model of the robot is illustrated in the Figure I.

The mathematical kinematic model of the two-wheeled mobile robot is given by the following equations:

\[
\begin{align*}
\frac{dx}{dt} &= \frac{V_R + V_L}{2} \cos \theta \\
\frac{dy}{dt} &= \frac{V_R + V_L}{2} \sin \theta \\
\frac{d\theta}{dt} &= \frac{V_R - V_L}{L}
\end{align*}
\]

where

(i) \( V_R \) and \( V_L \) represent, respectively, the right and left velocities of the two-wheeled mobile robot.
(ii) \( \theta \) is the orientation of the two-wheeled mobile robot.
(iii) \( L \) is the distance between the two wheels of the robot.

The kinematic motion of the two-wheeled mobile robot in the discrete time is described as follows:

\[
\begin{align*}
x_{k+1} &= x_k + T \frac{V_{R,k} + V_{L,k}}{2} \cos \theta_k \\
y_{k+1} &= y_k + T \frac{V_{R,k} + V_{L,k}}{2} \sin \theta_k \\
\theta_{k+1} &= \theta_k + T \frac{V_{R,k} - V_{L,k}}{L}
\end{align*}
\]

where \( T \) is the sampling time.

3. The Proposed Path Planning Algorithm

In this work, a behavior-based technique is proposed. This method combines two behaviors, namely, “go to goal avoidance obstacle” and “wall following.” According to the information sent through the sensors in each step in the environment, an arbitration mechanism selects the convenient controller to offer the two-wheeled mobile robot a smooth motion.

Firstly, the wheeled mobile robot proceeds in a strange environment towards the fixed goal. In this case, the problem of the path planning does not arise. When the mobile robot
detects the presence of the obstacles, it should move far away without hitting with obstacles.

Once the robot enters into a deadlock, it will be trapped. In this case, the “wall following” controller interferes in order to allow the two-wheeled mobile robot to follow the boundary of the deadlock and take it out from the obstacle. In this point, the robot exists and ultimately starts to move toward the destination; “wall following” is not demanded anymore.

3.1. “Go to Goal Obstacle Avoidance” Behavior. The aims of “go to goal obstacle avoidance” controller are to ensure the safety of the mobile robot during its navigation in the unknown environment and to steer the robot to the desired goal.

There are many ways to avoid obstacles. In order to clarify the strategy used in our work, we need to follow the upcoming steps:

(i) Step 1: Transform the IR distance measured by each sensor to a point in the reference frame of the robot.

As shown in Figure 2, the point $P_i$ measured by the sensor $i$ can be written as a vector (in the reference frame of the sensor):

$$v_i = \begin{bmatrix} d_i \\ 0 \end{bmatrix}, \quad i = 1..9 \quad (7)$$

Use the location and the orientation of the sensor in order to transform this point in the reference frame of the robot. This transformation is defined by the following equation:

$$v_i' = M_t \begin{bmatrix} x_s \\ y_s \\ \theta_s \end{bmatrix} v_i \quad (8)$$

where $M_t$ is the transformation matrix that makes a translation by $(x, y)$ and a rotation by $\theta$:

$$M_t (x, y, \theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \quad (9)$$

(ii) Step 2: Transform the point $P_i$ again to determine its location in the reference frame of the world. For that, we need the pose of the two-wheeled mobile robot $(x, y, \theta)$ in order to transform the robot from the robot’s reference frame to the world’s reference frame. This transformation is given by the following equation:

$$v_i'' = M_t (x, y, \theta) v_i' \quad (10)$$

(iii) Step 3: Compute a vector $u_i$ to each point from the robot as shown in Figure 3.

(iv) Step 4: Choose a weight $\alpha_i$ for each vector depending on how much importance you attach to a particular sensor to avoid obstacle.

The weighted vectors will be summed into a single vector $u_{ao}$ as illustrated in Figure 4.

The value of this vector is given by the following equation:

$$u_{ao} = \sum_{i=1}^{9} \alpha_i u_i \quad (11)$$

(v) Step 5: Employ the vector $u_{ao}$ and the pose of the mobile robot to calculate the heading that steers the mobile robot away from any object.

$$\theta_{ao} = \tan^{-1} \frac{u_{ao}(y)}{u_{ao}(x)} \quad (12)$$

(vi) Step 6: Compute the vector $u_{gtg}$ that points from the robot to the goal as shown in Figure 5.

The vector $u_{gtg}$ is written as follows:

$$u_{gtg} = \begin{bmatrix} x_p - x \\ y_p - y \end{bmatrix} \quad (13)$$

(vii) Step 7: Combine the two vectors $u_{ao}$ and $u_{gtg}$ into a single vector $u_{ao-gtg}$ that points the wheeled mobile robot both away from any obstacle and towards the target as shown in Figure 6. It is given by the following equation:

$$u_{ao-gtg} = \alpha u_{gtg} + (1-\alpha) u_{ao} \quad (14)$$

where $\alpha$ is the weight of the vector. It should be carefully tuned to get the best response.

(viii) Step 8: Use the vector $u_{ao-gtg}$ and the pose of the mobile robot to compute the heading that steers the mobile
robot away from any obstacle and allows the robot to reach its target:

\[ \theta_{ao} = \tan^{-1} \frac{u_{ao-gtg}(y)}{u_{ao-gtg}(x)} \]  

The input of the fuzzy controller of this behavior is the error \( (\Phi_{ao-gtg}) \) between the angle of “go to goal and avoidance obstacle” \( (\theta_{ao-gtg}) \) and the current heading of the robot \( (\theta) \).

\[ \Phi_{ao-gtg} = \theta_{ao-gtg} - \theta \]  

The outputs of this controller are the linear \( (v) \) and angular velocity \( (w) \).

3.2. Wall Following Behavior. The mobile robot drives from a starting point to a target one without hitting with any obstacles. These obstacles are fairly easy if they are convex but rather hard to overcome when they are concave in shape. If the robot starts from a point A to reach a point B existing behind the concave obstacle, the mobile robot will be trapped. To save the robot, we need to use another behavior known as the “wall following.” This behavior allows the robot to follow the contour thus avoiding the obstacle and reaching its target.

The algorithm of the wall following is explained in the following steps:

(i) Step 1: Calculate a vector \( u_{fw,t} \) that estimates a section of the wall, using the IR sensors of the mobile robot as shown in Figure 7.

The vector \( u_{fw,t} \) is determined as follows:

\[ u_{fw,t} = [P_2 - P_1] \]  

\[ u_{fw,t} = \frac{u_{fw,t}}{||u_{fw,t}||} \]  

(ii) Step 2: Compute a vector \( u_{fw,p} \) that points from the mobile robot to the closest point on \( u_{fw,t} \) as illustrated in Figure 8. It can be calculated as follows:

\[ u_p = \begin{bmatrix} x \\ y \end{bmatrix}, \]  

\[ u_a = P_1 \]

\[ u_{fw,p} = (u_a - u_p) - (u_a - u_p) u_{fw,t}' u_{fw,t}' \]
The input of the fuzzy controller of this behavior is the error \( \Phi_{fw} \) between the angle of wall following (\( \theta_{fw} \)) and the current heading of the mobile robot (\( \theta \)). It is given by the following equation:

\[
\Phi_{fw} = \theta_{fw} - \theta
\]

The outputs of this controller are the linear and angular velocity.

Inputs and outputs variables of this controller have the same partition of membership functions and the same rules of the previous behavior “go to goal avoidance obstacle.”

4. Design of the PID Controller

In the literature, numerous researches use the PID controller for the path planning. However, the parameters of the PID controller can affect the navigation steps. For that, its accuracy can be enhanced by tuning the parameters \( k_p, k_i, k_d \).

This shows that the parameters of the PID controller are not chosen in optimal way. In order to improve the performances of the navigation of the mobile robot, we have applied the Firefly and the Artificial Bee Colony algorithms to the PID controller.

The PID discrete law that assists the mobile robot to reach the desired goal without hitting with obstacles can be written as follow:

\[
u_t (k) = k_D e_t (k) + k_I \frac{1}{T} \sum_{j=0}^{k} e_t (j) + k_P \frac{1}{T} \Delta e (k)
\]

where \( k \) and \( T \) are, respectively, the current discrete instant time and the sampling period.

The objective function employed to get an optimized PID controller is the integral square error (ISE).

4.1. Artificial Bee Colony: Overview. Artificial bee colony is inspired by the intelligent foraging behavior of honey bee swarm. It is developed in 2005 [29] by Derviş Karaboga. In this clever algorithm, there are three types of bee colony that is based on the role it plays: the onlooker, employed (forager), and the scout bee.

The onlooker bees wait on the area of the waggle dance in purpose to choose a source of food. The forager bees keep visiting the food sources that are visited previously to get nectar. The scout bees perform random searches to discover novel sources of food.

The artificial bee colony can be divided into four essential steps.

(i) Initialization phase: The random initial population is given by the following equation:

\[
x_i = x_{min} + \text{rand} (0, 1) \ast (x_{max} - x_{min})
\]

(ii) Employed bee phase: Each employed bee was affected to a food source. After that, each employed bee discovers a novel neighboring food source of its presently attributed food source throughout (26) and calculates for each iteration the nectar amount of the novel food source. The employed bee is
Firefly algorithm is a new metaheuristic algorithm inspired by the social lighting behavior of fireflies. It was first introduced by Yang [30].

In nature, there are many species of fireflies and most of them produce rhythmic and short flashes. The rate and the rhythmic flash and the amount of time form a part of the signal system which brings both sexes together. Therefore, the main part of the firefly’s flash is to act as a signal system to attract other fireflies.

The FA adopts three particular rules based on few of the features of real fireflies:

(i) All fireflies are unisex. They will proceed towards the most attractive and brightest ones independently of their sex.

(ii) The degree of their attraction is proportional to their luminosity which diminishes as the distance from other fireflies raises. The firefly will move arbitrarily if there is not any shinier or more attractive one.

(iii) The brightness of a firefly is given by the value of the objective function of a specific problem.

Firefly algorithm has three important aspects, attractiveness to light intensity seen by adjacent fireflies and the third term is employed for the arbitrary motion of a firefly when no shining ones exist.

(iv) Population elimination: In this stage, if we assume that some solution gains do not get improvement after updates, it is then supposed to be taken into local optimum and is dispensed; after that, the corresponding onlooker bees transformed into scouting bees and generate a novel solution to substitute the removed solution.

where fit(xᵢ) represents the fitness value of the ith firefly and d represents the number of dimensions.

\[
V_{ij} = x_{ij} + \text{rand}(-1, 1) (x_{ij} - x_{kj}) \quad (27)
\]

\[
p_i = \frac{\text{fit}(x_i)}{\sum_{n=1}^{m} \text{fit}(x_n)} \quad (28)
\]

where fit(xᵢ) represents the fitness value of the nth solution.

Fitness is given by the following equation:

\[
\text{fit}(x_n) = \begin{cases} 
1 & \text{if } f(x_n) \geq 0 \\
1 + \text{abs}(f(x_n)) & \text{if } f(x_n) < 0 
\end{cases} \quad (29)
\]

where f(xᵢ) is the objective function value of bee source xᵢ.

\[
r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \quad (31)
\]

where rᵢⱼ represents the distance between two fireflies i and j, at positions xi and xⱼ, respectively, the distance between these two ones can be determined as follows:

\[
x_i = x_i + \beta_0 e^{-r(x_i)}^m (x_i - x_j) + \alpha \left( \text{rand} - \frac{1}{2} \right) \quad (32)
\]

where the first term represents the actual position of a firefly; the second term is used for considering a firefly’s attractiveness to light intensity seen by adjacent fireflies and the third term is employed for the arbitrary motion of a firefly when no shining ones exist. α is a randomization variable and rand represents a random number generator uniformly distributed in the space (0,1).

5. Design of the Fuzzy Logic Controller

The optimized PID controller allows the two-wheeled mobile robot to find a safe path but not the shortest one.

To obtain better results, we developed, for the two behaviors, a Same Fuzzy Logic Controller (S.F.L.C.) that helps the robot to follow the shortest trajectory. It is more expressive and interpretable; the results of rules are more simple and intuitive.

The constructed fuzzy logic controller uses only one input depending on the scenario in the strange environment. In other words, if the “go to goal and obstacle avoidance” behavior is activated, the input of the fuzzy controller is the error between the angle of “go to goal and obstacle avoidance” and the current heading of the robot (Φ<sub>go-ogv</sub>). If the “wall following” behavior is activated, the input of the fuzzy controller is the error between the angle of “wall following” and the current heading of the mobile robot (Φ<sub>fw</sub>).

The range of inputs of this fuzzy logic controller is divided into eleven linguistic variables.

Figure 10 illustrates the distribution of membership functions for input variables of the two basic behaviors: “go to goal and obstacle avoidance” and “wall following.”

They have been extracted after several trials based on their possible variation as to obtain the best system response.

The range of the first output of this fuzzy logic controller (angular velocity) is divided into eleven linguistic variables while the range of the second output (linear velocity) is divided into four linguistic variables.

Figures 11 and 12 show the distribution of membership functions for outputs variables of the two developed behaviors.
Since the fuzzy membership functions of inputs and outputs share similar names, common abbreviations have been used and given in Table 1.

Table 2 summarized the obtained fuzzy rule base for the linear and angular velocities. These rules are manually developed after several simulations.

6. Simulation Results

This part is dedicated to test the effectiveness of the proposed approach. For this aim, we consider a robot which starts from an initial position \((x, y) = (1.3, 1.3)\) and tries to move directly to the final position \((-1.2, -1.2)\).

Firstly, we considered an environment which contains scattered obstacles without deadlocks, using the S.F.L.C. approach. Figure 13 shows the trajectories covered by the two wheeled mobile robot.

It is obvious that the two-wheeled mobile robot can proceed towards the goal until reaching the desired target, which confirms the efficiency of the developed method.

In the next simulation, the mobile robot navigates in an environment containing deadlock. To choose the correct reactive behavior, the suggested path planner controller should study all information. As it is illustrated in Figure 14, at the beginning of navigation, the mobile robot turns and proceeds towards the goal. When the robot senses an object in front of it, and the goal is located on the other side of the object, the arbitration mechanism realizes that the robot does not progress towards the goal and switches to the wall following behavior. The robot changes its orientation angle when it is close to the obstacles. At the end, the track leading to the goal is clear; the robot is safe and reaches its desired target.

7. Comparison with Other Approaches

The main contribution in this work is that the designed control method based on S.F.L.C. deals with the complexity of the environment and guarantees the optimization of the process.

To introduce the validity of our proposed technique, we have compared it with three other methods based on PID controller tuned with the PSO [28] and ABC and FA algorithms (PSO-PID, ABC-PID and FA-PID). Performances
Figure 13: Trajectory of the mobile robot using S.F.L.C. without deadlock.

Figure 14: Trajectory of the robot using S.F.L.C. in an environment with deadlock.

Figure 15: Path planning of the mobile robot in an environment without deadlock using S.F.L.C. (red), PSO-PID (yellow), ABC-PID (blue), and FA-PID (green).

Figure 16: Path planning of the mobile robot in an environment with deadlock using S.F.L.C. (red), PSO-PID (yellow), ABC-PID (blue), and FA-PID (green).

Table 1: Abbreviations of linguistic values.

<table>
<thead>
<tr>
<th>Linguistic value</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL</td>
<td>Negative Large</td>
</tr>
<tr>
<td>NM</td>
<td>Negative Medium</td>
</tr>
<tr>
<td>NS</td>
<td>Negative Small</td>
</tr>
<tr>
<td>N</td>
<td>Negative</td>
</tr>
<tr>
<td>ZN</td>
<td>Zero Negative</td>
</tr>
<tr>
<td>Z</td>
<td>Zero</td>
</tr>
<tr>
<td>ZP</td>
<td>Zero Positive</td>
</tr>
<tr>
<td>PS</td>
<td>Positive Small</td>
</tr>
<tr>
<td>P</td>
<td>Positive</td>
</tr>
<tr>
<td>PM</td>
<td>Positive Medium</td>
</tr>
<tr>
<td>PL</td>
<td>Positive Large</td>
</tr>
<tr>
<td>S</td>
<td>Small</td>
</tr>
<tr>
<td>M</td>
<td>Medium</td>
</tr>
<tr>
<td>VL</td>
<td>Very large</td>
</tr>
</tbody>
</table>

Table 2: Fuzzy rule base of the two reactive controllers.

```
<table>
<thead>
<tr>
<th>( \Phi_{\text{w-dir}} ) or ( \Phi_{\text{f-w}} )</th>
<th>Linear Velocity</th>
<th>Angular Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL</td>
<td>S</td>
<td>NL</td>
</tr>
<tr>
<td>NM</td>
<td>S</td>
<td>NM</td>
</tr>
<tr>
<td>N</td>
<td>S</td>
<td>N</td>
</tr>
<tr>
<td>NS</td>
<td>M</td>
<td>NS</td>
</tr>
<tr>
<td>ZN</td>
<td>L</td>
<td>ZN</td>
</tr>
<tr>
<td>Z</td>
<td>L</td>
<td>Z</td>
</tr>
<tr>
<td>ZP</td>
<td>L</td>
<td>ZP</td>
</tr>
<tr>
<td>PS</td>
<td>M</td>
<td>PS</td>
</tr>
<tr>
<td>P</td>
<td>S</td>
<td>P</td>
</tr>
<tr>
<td>PM</td>
<td>S</td>
<td>PM</td>
</tr>
<tr>
<td>PL</td>
<td>S</td>
<td>PL</td>
</tr>
</tbody>
</table>
```

are illustrated in Figures 15, 16, and 18. It is clear that the S.F.L.C. approach performs better.

Figures 17(a) and 17(b) show the evolution of the linear and angular velocities in the two unknown environments.

In order to evaluate the performance of the two-wheeled mobile robot using the PSO-PID [28], ABC-PID, FA-PID, and S.F.L.C., two criteria have been considered which are the navigation time and the travelled distance. Numerical results are illustrated in Tables 3 and 4. The comparison between these results proves the efficiency of S.F.L.C. controller.
Figure 17: Evolution of the two velocities (w and v) in the environment (a) without deadlock and (b) with deadlock.

Figure 18: Path planning of the mobile robot in an environment with deadlock using S.F.L.C. (red), PSO-PID (yellow), ABC-PID (blue), and FA-PID (green).

Table 3: Comparison between navigation time using PSO-PID, ABC-PID, FA-PID, and S.F.L.C.

<table>
<thead>
<tr>
<th>Method</th>
<th>Figure 15</th>
<th>Figure 16</th>
<th>Figure 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO-PID</td>
<td>37.1</td>
<td>23.9</td>
<td>12.2</td>
</tr>
<tr>
<td>ABC-PID</td>
<td>36.35</td>
<td>22.45</td>
<td>9</td>
</tr>
<tr>
<td>FA-PID</td>
<td>33.95</td>
<td>21</td>
<td>7.5</td>
</tr>
<tr>
<td>S.F.L.C.</td>
<td>18</td>
<td>17</td>
<td>6.5</td>
</tr>
</tbody>
</table>

Table 4: Comparison between travelled distance using PSO-PID, ABC-PID, FA-PID, and S.F.L.C.

<table>
<thead>
<tr>
<th>Method</th>
<th>Figure 15</th>
<th>Figure 16</th>
<th>Figure 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO-PID</td>
<td>5.565</td>
<td>3.585</td>
<td>1.83</td>
</tr>
<tr>
<td>ABC-PID</td>
<td>5.4525</td>
<td>3.3675</td>
<td>1.35</td>
</tr>
<tr>
<td>FA-PID</td>
<td>5.0925</td>
<td>3.15</td>
<td>1.125</td>
</tr>
<tr>
<td>S.F.L.C.</td>
<td>3.843</td>
<td>2.784</td>
<td>0.936</td>
</tr>
</tbody>
</table>
8. Conclusion

In this work, we have applied a new method, S.F.L.C., for controlling the navigation of the mobile robot in a strange environment. We have mainly used the two primitive reactive behaviors: “go to goal” obstacle avoidance” and “wall following” as well as an arbitration mechanism responsible for switching to the suitable behavior according to the circumstances in the unknown environment. The mobile robot is now able to avoid obstacles and to escape from deadlocks, reaching the target successfully. The proposed path planner controller has been compared with other related works, and it has been deduced that the current motion controller gives better results. For future work, we will focus on testing our proposed method on the real Khepera III mobile robot.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References


