Path Planning Algorithm for Initially Unknown Indoor Environment Navigation

Mohamed Emharraf(✉), Mohammed Saber, Mohammed Ghaouth Belkasmi, Ilhame El Farissi, Sara Chadli, and Mohammed Rahmoun

Laboratory Smart Information, Communication and Technologies (SmartICT), National School of Applied Sciences (ENSAO), Université Mohammed Premier (UMP), 60050 Oujda, Morocco
{m.emharraf,m.saber}@ump.ac.ma

Abstract. Allowing Robots to navigate automatically in unknown environments where human being cannot access is a needed characteristic in many fields, such as military, industry, civil engineering. This paper presents an efficient and fast path-planning algorithm for mobile robots, which navigate in a priori unknown indoor environments based on their local capacities of sensing and acting. For navigate in a priori unknown environments, re-planning the path it's necessary each time the current one cut off by an obstacle, thus the path-planning algorithm must had the ability to re-plan the path once the tracked path become unavailable (online planning). The proposed path-planning algorithm deploy a grid map to characterize the environment, where the environment described by an occupancy and trajectory grids, which smooth the planning task. In order to make the planning faster we had split up the algorithm into two modules: obstacle avoiding and path re-planning. The experiments results had shown the applicability and the fusibility of the proposed approach.

Keywords: Online path planning · Unknown environment · Indoor environment · Mobile robot

1 Introduction

Navigating in a priori unknown environment has a broad spectrum of applications in advanced robotics. Traditionally, this problem has been solved either by having the robot built a map of the environment [1] (which can be seen from the initial position) before planning the path, or by applying deterministic algorithms that are able to deal with unknown environments [2]. It is hard to apply a global trajectory planning and track it because the robot got only a few data about the global environment. In addition, the real environments are usually complex and unpredictable, which turns a planned path outdated quickly. Other approaches offer alternatives solutions, such as the potential field [3] or the fuzzy neural approaches [4, 5], however, it is hard to guarantee overall convergence of the planned path because mobile robots are likely trapped in dead end environments.

A. El Moussati et al. (Eds.): SmartICT 2019, LNEE 684, pp. 366–376, 2020.
https://doi.org/10.1007/978-3-030-53187-4_40

Path planning is an important task for intelligent mobile robot control systems, which should be carry out efficiently. Planning a path means generating a collision-free path in an environment with obstacles and optimizing it [6, 7]. The environment may be imprecise, vast, dynamic or unstructured [8]. In such environment, path planning depends on the robot sensory information about the environment, which could be associated with inaccuracy and uncertainty. Thus, to get an efficient planning in such environment, the path planning system should be adaptive with the nature of the work area. In the case of known and static environment, the path generated in advance, offline algorithm [6]. Planning is online [6] if the algorithm had the ability to re generate the path in response to environmental changes.

The frontier and virtual lens following approaches are designed to solve the problem of navigation in the literature [9–17]. Unfortunately, they still incapable to guarantee a global convergence in complex environments. In next, we present some difficulties that need to overcome. (A) When the goal is on the side of the wall, a long wall environment (Fig. 1.a) this situation could trap the robot in a wrong next boundary direction. (B) Congested and unstructured environments (Fig. 1.b) could disrupt the algorithm to recognize the typical sites of interest. (C) In a dynamic environment the structure are changing which make sensing data quickly exceed, resulting an inability to get the detection to goal or escape criteria. (D) U-shaped recursive or labyrinthine environments (Fig. 1.b) can force the robot to regress into previously visited corners. (E) An inaccurate localization estimate (i.e. odometry errors) may result in an inability to achieve the goal. (F) The sensors detection capability (i.e. sonar sensors) and noises yield difficult to determine the size or location of obstacles.

Fig. 1. (A) Long wall environment; (B) Unstructured, crowded environment and labyrinth

The presented paper describe a new approach of online path planning algorithm, which dedicated for unknown indoor environments, this algorithm allows the autonomous mobile robot to navigate in the unknown indoor environment on the basis of environmental information and robot position, without need for external or a priori informations about the environment. The algorithm operate on a grid map presentation of the environment [19]. The algorithm has two complimentary independent behaviors: obstacle avoidance behavior (EO) and the path re-planning (PC) behavior.

The rest of the paper organized as follows. Section 2 describes design of our path re-planning behavior, the obstacle avoidance behavior and the global path planned behavior. Section 3 provides a detailed discussion of global convergence as well as

method complexity performance. Section 4 presents the experimental results. The last section summarizes the paper.

2 Online Path Planning Behaviors

The general idea of the online planning algorithm is that an initial path planned based on initial information, and getting new informations while flowing the path, once the planed path blocked a re-planning task made to create a new path toward the goal. The progressive learning about the environment while finding the free path translates into better new plans. The motion control variables of the robot are the translation velocity v and the rotation angle *Y*. The safety of the robot influenced mainly by the *EO* (obstacle avoidance) behavior which is able to detect obstacles in real time and react automatically to keep robot safe. Therefore, we assume that the robot's velocity v supervised only by the *EO* behavior, in spite of (*PC)* behavior which determining the new plan through online re-planning. The robot rotation angle *Y* represented by eight angles {*Y0, Y1, Y2, Y3, Y4, Y5, Y6*}, with positive and negative terms allowing robot to turn left and right to choose from the sixteen-breakpoint neighbors.

2.1 The Path Re-planning Behavior

The re-planning path (*PC*) behavior design allows the mobile robot to navigate throw safe points in order to move toward the goal. These points are among the neighboring breakpoint point that form a circle of *RR* radius neighbors (0.5 m in our study case) around robot breakpoint position (Fig. 2). The point with minimum risk and re-planning cost is the one with minimum values of both *PMT* and *PMO* matrices in the grid map [18], which represent respectively the occupancy and how often robot passed, throw the point. The map getting update each control cycle based on sensory information from robot [18, 19]. The re-plan path with the minimum risk is the best choice for the robot to avoid the obstacles and the previous trajectories, therefore to escape dead-ends.

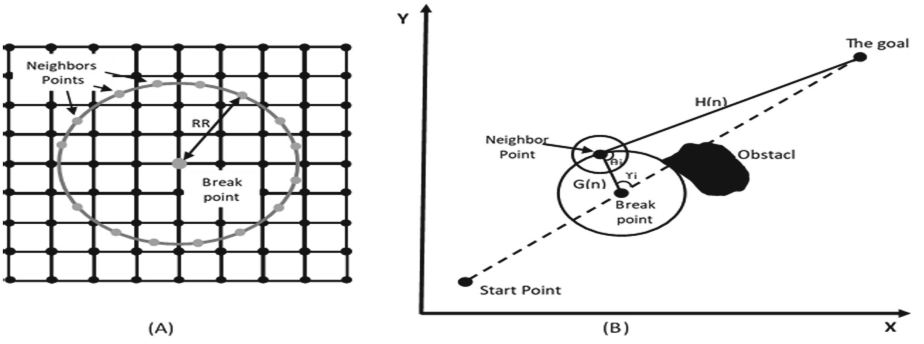

Fig. 2. Re-planning rules parameters, (A) neighboring point determinations, (B) re-planning cost function elements.

However, the *PC* behavior active only when the robot encounter obstacles so that the robot can escape from potential dead-ends. Thus, the *PC* behavior handling has to take in consideration the intensity of the obstacle point (will defined in following). The exit space point (i.e. the rotation angle) surrounded in the range (−180°, 180°) in order to get a smoother rotation. Therefore, we design the PC behavior using a multi-parameter cost function and an analysis algorithm.

To perform such behavior, the controller has to infer a risk index to breakpoint neighbors. Then get the rotation rule, which based on the risk index and cost function, and finally the analysis parte.

While robot flowing the current path, the grid map under build based on robot sensory information. In each control cycle, the *PMO* and *PMT* matrix updated to locate the current environmental obstacles and the previously traversed points. In the same time, the advanced data characteristics of breakpoint neighbors such as the Displacement Cost (*CD*), the Heuristic Path Cost (*CHC*), the trajectory point intensity (*IPT*), the obstacle point intensity (*IPO*), the collision risk (*RC*) and the iteration risk (*RI*), are calculated based on *PMO* and *PMT* at each breakpoint. The *RI* and *RC* used to derive the risk index for the PC behavior navigation rules. While *CD, CHC, IPT* and *IPO* are used to calculate the cost function. *CD, CHC, IPT, IPO, RC* and *RI* defined as follow:

Definition 1 (Travel cost): (CD) value of a point P is defined as $G(P) = RR*\theta$, where P is a breakpoint neighbor (Fig. 2), where *RR* is the radius of breakpoint neighbors circle, θ is the turn angle to the neighbor *P* while $\theta = 0$ is the angle to the old path.

Definition 2 (Path heuristic cost): (*CHC*) of a neighboring point P is defined as $H(P) = distance(P_v, P_b)$, where *P* is a breakpoint neighbor (Fig. 2), P_v present *P* coordinates *P*, P_b present the goal coordinates. The Euclidean distance is used to calculate the distance between P_v and P_b.

Definition 3 (Path Point Intensity): (*IPT*) of a neighboring point *P* is defined as $T(P) = \sum_{(i,j)\in A} V_{PMT}(i,j)$, where A is a circle-shaped region having as center the neighbor P coordinates (Fig. 2), $V_{PMT}(i, j)$ is the value of the *PMT* of the cell (i, j) involved in the region *A*.

Definition 4 (Obstacle Point Intensity): (*IPO*) of a neighboring point P is defined as $M(P) = \sum_{(i,j)\in A} V_{PMO}(i,j)$, where $V_{PMO}(i, j)$ is the *PMO* value of the cell (i, j) involved in the region *A*.

Definition 5 (Risk of collision): risk of collision (*RC*) of a neighboring point P is defined as $O(P) = V_{PMO}(i, j)$

Definition 6 (Risk of iteration): (*RI*) of a neighboring point P is defined as $N(P) = V_{PMT}(i, j)$.

The value α of *RI* is converted into three values (0, 50, 100) which can translate by {never visited, little visited, too visited). The value β of the *RC* is converted as well into three values (0.50, 100) which can translate by {free, unknown, occupied) respectively.

Risk Index. The risk index rule combine two risks parameters into a single indicator, which indicates the robot safety to go over the point. We represent the risk index by three values (0, 50, 100). The risk index *r* defined in terms of *RI* and *RC* risks collision

by a set of intuitive relationships shown in Table 1. For example, the element (3, 3) of Table interpreted as if *RI* is weak and *RC* is low, then r index is SAFE ($r = 0$). Naturally, we define that the neighboring point with a minimum of risk is the point that has a risk index SAFE (r = 0). Risk indexes for breakpoint neighbors derived using the risk index rules.

Table 1. Risk index rules.

RC / RI	100	50	0
100	100	100	100
50	100	50	50
0	100	50	0

The multi-valued nature of the proposed risk index representation provide robustness and significant tolerance to the large amount of uncertainty and inaccuracy inherent in one-point obstacle sensor detection. This robustness is because the output system based on rules and a function of input variables values.

Re-planning Rules. The re-planning *PC* behavior tend to choose the closest direction towards the goal, thus the robot deny all unnecessary rotations. The re-planning behavior steps described by Fig. 3.

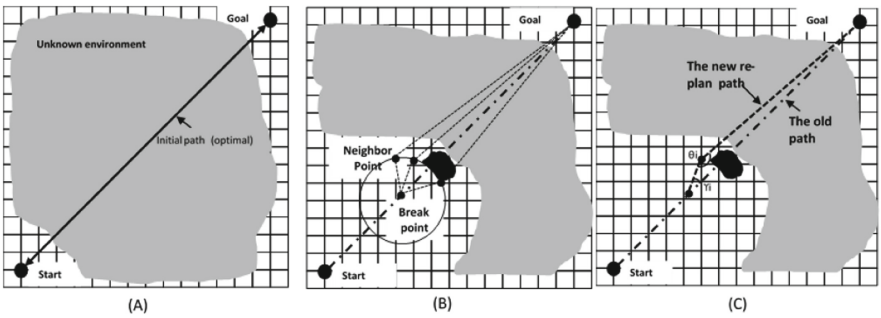

Fig. 3. The path re-planning steps. (a) Neighbors points identification, (b) weigh up the risk index and the cost function for each neighbor, ((c) determination of the new plan with minimum risk and cost.

The PC behavior cost function defined as $F(n) = G(n) + CHC(n) + IPO(n) + IPT(n) + r(n)$, where:

G (n): the cost of travel. H (n): the heuristic cost of path.
IPO (n): Intensity of obstacle point. IPT (n): Intensity of trajectory point.
R (n): Risk index

Important note: a rotation to a neighboring point is not engaged when all the points have the same dangerous risk indices (r = 100). The re-planning rule does not force the robot to arbitrarily choose between points, but maintain the steering angle at zero at this point. The final selection will made using the algorithm presented below.

The proposed re-planning rules may work well in most situations to offer recommendations. On the other hand, when the robot get in an extreme situation (for example, all neighbor points had the same index risk with high iteration), the re-planning rules can't judge the point to re-plan. Despite of the high collision risk, under some extreme situation the robot may choose a point that has a high iteration risk but present a minimal value comparing the other neighborhood points.

The idea of re-planning algorithm is that if the recommended point does not have a SAFE risk index ($r = 0$), the collision and iteration risk of all neighboring points are compared again (by threshold comparison) to recommend a direction that has a safe collision risk and a minimum iteration risk. The thresholds for the *RI* and *RC* are respectively $\alpha 1$ and $\beta 1$. Both the re-planning rules and the algorithm made a complete framework for getting the re-plane point of the *PC* behavior. Therefore, the turn angle recommended by the *PC* behavior can prevent the robot from iterating its previous trajectory as little as possible, so the robot chooses to explore a new point as a way to escape the local hazard.

2.2 The Obstacle Avoidance Behavior

The Obstacle Avoidance (*EO*) behavior is a sensor-based behavior that keeps the robot safe without risk of collision with obstacles. It activated if an obstacle come closer.

For this behavior, we are interested in the robot translation velocity; the speed rules for *EO* behavior are simple. When the robot approaches an obstacle in the direction of movement, the behavior decreases the speed once the distance is less than the threshold $\lambda 1$. When the distance to the obstacle is less than the threshold $\lambda 2$ the robot stops completely, and waits for a change in direction to avoid the obstacle.

2.3 The Overall Behavior

The global online planning behavior based on two behaviors, the *EO* behavior (obstacle avoidance) which secure from obstacles by controlling the translation velocity, the *PC* behavior (re-planning of path) which re-plan the path at each breakpoint by proposing another path which will passed throw by a neighbor point present minimum of risk. The global behavior algorithm recommends the determination of the velocity and rotation angle.

Algorithm : (the global behavoir)
Input: (x1, y1) = goal coordinates; (x0, y0) = robot currant coordinates;
cp0 = robot current angle;
Di (i = 0, 1, ..., 5) = sonar read (6 rotation angles).
Output: (v, ϒ, θ) = velocity and rotation angle.
BEGIN:
Step 1.0. Update sensory data, including (x0, y0), (d0, d1, d2, d3, d4, d5);
Step 1.1. Update the MDG and TMD matrix using the re-plan algorithm;
Step 1.2. If the distance between the robot's current postion (x0, y0) and the goal (x1, y1) is below the predefined threshold (distance tolerance), THEN the objective reached and the robot stopped, otherwise go to step 1.3;
Step 1.3 : if the robot stopped go to step 1.4, otherwise keep the current path and go to step 2
Step 1.4. Update RI and RC values for all neighbor point;
Step 1.5. Update CD and CHC values for all neighbor point;
Step 1.6. Update IPO and IPT values ;
Step 1.7. Calculate the risk index of the neighbor points using the rules of table 1;
Step 1.8. Calculate the passage cost for the neighbor points, and classify the points according to the cost.
Step 1.9. Calculate the rotation angle ϒi, θi to the neighbor point and goal respectively, recommended by the PC behavior using the re-planning rules;
Step 1.10 plot the path re-planned by the neighbor point Pi.
Step 2. Compute the translation velocity v recommended by the EO behavior
Step 3. Execute motor control commands (v, ϒ, θ).
End Algorithm

3 Performance Analysis

3.1 Convergence Analysis

At each breakpoint, the risk index rules choose the neighbor point present a lower risk of collision *RC* and iteration risk *RI* (safe point). The *PC* behavior re-planning rules ensure the recommendation to a point with a safe risk index and a lower cost function value if such a point exist. If not, when no point with a safe risk index exist, the re-planning algorithm solve the situation by ensuring that the choosing re-plan point has a safe *RC* (below the $\beta 1$ threshold), or a strong *RC* and a minimum *RI* by applying a threshold comparison.

Therefore, if a path needed for a navigation task in an unknown environment, the method will guarantee the global convergence at the given goal location. Thus the path planning task always contain collision and iteration risk, whereas the method guarantees that the choosing point to re-plan has minimum risk in order to escape potential local minima and achieved overall navigation by coordinating two behaviors (*EO*, *PC*).

In the case of dynamic environment (i.e. the existence of mobile objects), the navigation method could guarantee the global convergence. Because our mapping behavior could detect the environment changes in real time and update the grid map accordingly. Based on the continuously updated map, the use of *PC* behavior allows choosing the safest direction to escape obstacles. At the same time, the *EO* behavior maintains robot safety and avoid collision with possible fixed or dynamic obstacles.

3.2 Complexity Analysis

A) **Space complexity:**
Our re-planification system require a fixed memory space to save the grid map if the goal location is determined. Importantly, this space requirement does not change with the navigation time or the complexity of the environment. The map should cover the physical areas that include the beginning, purpose, and the path. When the cell size of the card is determined, the size of the entire grid card is determined. Suppose that the length of the cell size is λ, so that an $M \times N$ grid map covers a physical space whose area $M \times \lambda \times N \times \lambda$. For example, if $\lambda = 0.1$ m (for our robotic system), a 100×100 grid map can cover a real space whose area is $(100 \times 0.1) \times (100 \times 0.1) = 100\ m^2$.

B) **Time complexity**
The time complexity is fixed and effective. As discussed in space complexity section, the decision is determined based on a small range of map information and sensory data. As a result, feature calculations (iteration risk, etc.) involve very few addition operations. In addition, the navigation algorithm based on fast and efficient calculation rules. Other calculations such as the cost function require simple operations and threshold comparisons for the re-planning algorithm.

4 Experimental Results

To expose the final control output provided by the different behaviors, we had used a visualization of each behavior data. Each behavior produces an output. The path indicated by serial rectangles. The program draws update the map every 0.5 s.

4.1 Performance Analysis for a Long Wall Environments

The purpose of this experiment is to analyze the decision-making process. The robot has to move from the beginning point *S* to the goal *T*. While the robot flowing the path with normal velocity (maximum), the *PMO* and *PMT* values updated in the grid map, so the *IPO* and *IPT* values are low. The *EO* behavior recommends maximum speed while the obstacle ahead still far. Therefore, the *PC* behavior recommends the direction towards the goal (Fig. 4 (1)) once the path blocks by an obstacle the robot decreases its speed, the number of memory points increases and the *IPO* or *IPT* becomes medium or large. Therefore, the PC behavior is activated ((2) (3) (4) (5) (6) Fig. 4); When the distance to the obstacle becomes lower than the threshold $\lambda 2$, the *EO* behavior stops the robot to re-plan the path ((2) (3) (5) Fig. 4).

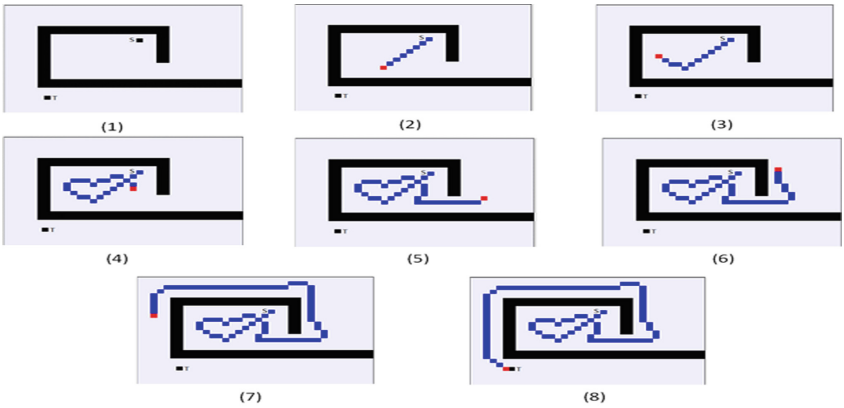

Fig. 4. Navigation results for a long wall environment.

4.2 Complex Environments Performance

We had also tested our online re-planning method in unknown complex environments. (Figure 5(a), (b), (c)) show the results for, respectively, circular, unstructured and congested, and labyrinthine environments. Each time the method of navigation, find the path to the goal.

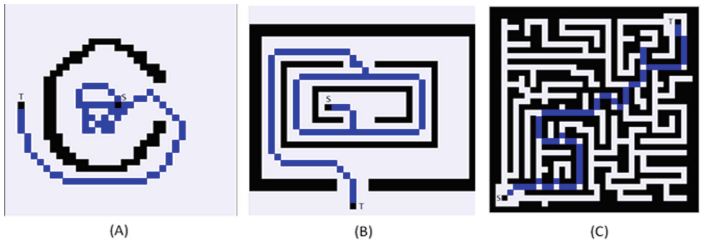

Fig. 5. (A) circle-shape environment. (B) Labyrinth environment. (C) Very complex environment.

5 Conclusion

The present paper introduce a new behavior-based navigation method based on online re-planning. The method addresses the problem of goal-oriented navigation in initially unknown indoor environments. This method experimentally demonstrated by simulation a global convergence to the goal location even in the case of long wall, a large concave, recursive U-shaped, unstructured, congested and labyrinth environments. One of the future works is to develop a motion planning system to test the practical navigation approach on a real robot, and to improve and formulate the method as well to theoretically prove global convergence.

The proposed navigation method is not suitable for outdoor navigation, because of accumulated odometer localization system errors corrected, which are acceptable for an area of 100 m^2. The method is currently particularly suitable for short navigation applications between crossing points in complex environments. The supervised robotics falls into this type of application where the human operator can give a number of sub-goals to allow the remote robot to explore unknown environments.

References

1. Meryer, J.A., Filliat, D.: Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies. Cogn. Syst. Res. **4**, 283–317 (2003)
2. Ferguson, D., et al.: A guide to heuristic-based path planning. In: Proceedings of the Workshop on Planning Under Uncertainty, for Automated Planning and Scheduling (ICAPS-05) (2005)
3. Tsourveloudis, N.C., Valavanis, K.P., Hebert, T.: Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic. IEEE Trans. Robot. Autom. **17**(4), 490–497 (2001)
4. Rusu, P., Petriu, E.M., Whalen, T.E., et al.: Behavior-based neuro-fuzzy controller for mobile robot navigation. IEEE Trans. Instrum. Meas. **52**, 1335–1340 (2003)
5. Godjevac, J., Steele, N.: Neuro fuzzy control for basic mobile robot behaviours. In: Driankov, D., Saffiotti, A. (eds.) Fuzzy Logic Techniques for Autonomous Vehicle Navigation, pp. 98–118. Physica-Verlag, Heidelberg (2000)
6. Howard, A., Matari, M.J., Sukhatme, G.S.: An Incremental self-deployment algorithm for mobile sensor networks, autonomous robots. Special Issue Intell. Embed. Syst. **13**(2), 113–126 (2002)
7. Florczyk, S.: Robot Vision Video-Based Indoor Exploration with Autonomous and Mobile Robots. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim (2005)
8. Janglova, D.: Neural networks in mobile robot motion. Int. J. Adv. Robot. Syst. **1**(1), 15–22 (2004). ISSN 1729-8806
9. Huang, H.P., Lee, P.C.: A real-time algorithm for obstacle avoidance of autonomous mobile robots. Robotica **10**, 217–227 (1992)
10. Kamon, I., Rivlin, E.: Sensory-based motion planning with global proofs. IEEE Trans. Robot. Autom. **13**(6), 814–822 (1997)
11. Lim, J.H., Cho, D.W.: Sonar based systematic exploration method for an autonomous mobile robot operating in an unknown environment. Robotica **16**, 659–667 (1998)
12. Krishna, K.M., Kalra, P.K.: Perception and remembrance of the environment during real-time navigation of a mobile robot. Robot. Auton. Syst. **37**, 25–51 (2001)
13. Maaref, H., Barret, C.: Sensor-based navigation of a mobile robot in an indoor environment. Robot. Auton. Syst. **38**, 1–18 (2002)
14. Chatterjee, R., Matsuno, F.: Use of single side reflex for autonomous navigation of mobile robots in unknown environments. Robot. Auton. Syst. **35**, 77–96 (2001)
15. Pin, F.G., Bender, S.R.: Adding memory processing behavior to the fuzzy behaviorist approach: resolving limit cycle problems in mobile robot navigation. Intell. Autom. Soft Comput. **5**(1), 31–41 (1999)
16. Xu, W.L.: A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behavior-based mobile robot. Robot. Auton. Syst. **30** (2000)
17. Xu, W.L., Tso, S.K.: Sensor-based fuzzy reactive navigation for a mobile robot through local target switching. IEEE Trans. Syst. Man Cybern. **29**(3), 451–459 (1999)

18. Emharraf, M., et al.: Mobile robot: SLAM implementation for unknown indoor environment exploration. JCS **12**(2), 106–112 (2016)
19. Emharraf, M., et al.: Mobile robot unknown indoor environment exploration using self-localization and grid map building. In: 2014 9th International Conference on Intelligent Systems: Theories and Applications (SITA-14). IEEE (2014)