A New Genetic-Fuzzy Algorithm for Mobile Robotics Way-Finding in Environments with Any Types of Concave Obstacle

Omid Motlagh, Sai Hong Tang, Napsiah Ismail, and Razali Samin

Mechanical and Manufacturing Engineering Department, Faculty of Engineering,
University Putra Malaysia, 43400, Selangor, Malaysia
omid_motlaq@yahoo.com, {saihong,napsiah,razali}@eng.upm.edu.my

Abstract. A new behavior-based algorithm is developed for reactive navigation of mobile robots. While fuzzy logic body of the algorithm performs the main tasks of obstacle avoidance and target seeking, an actual-virtual target switching is used to resolve the problem of limit cycles in any types of concave obstacles. The overall performance of the algorithm is then enhanced by using GA optimization of the functions. In this work, concave obstacles may have any shape such as corner, U-shape cul-de-sac, snail shape, or any other complicated shape. Trajectories and behavior analysis of a *Pioneer* robot are demonstrated to prove the robustness of the proposed algorithm.

Keywords: genetic-fuzzy; mobile robot navigation; limit cycles; virtual target.

1 Introduction

In local navigation, mobile robots are capable of sensing environments, interpreting the information to obtain the knowledge of position, and planning real-time routes from initial positions to targets with obstacle avoidance, and target reaching. Some of the online approaches include; wall following, artificial potential fields [1, 2], fuzzy potential energy [3], virtual target approach [4, 5], landmark learning [6], virtual wall algorithm [7], dynamic window, neural network, and fuzzy logic [8, 9, and10]. Since the introduction of fuzzy logic control (FLC) by Zadeh of the University of California [11], this approach to robot navigation and obstacle avoidance has been investigated by several researchers. Fuzzy systems have the ability to treat uncertain and imprecise information using linguistic rules. They offer possible implementation of human knowledge and experience without requiring precise analytical models.

Although FLC simplified navigation problems, still there are local situations where a pure fuzzy logic approach fails in taking appropriate actions. Of these troublesome situations, obstacles forming a loop shape also called dead-end traps are the most common which cause the limit cycles. A mobile robot with pure fuzzy logic navigator simply gets trapped in concave obstacles where the rules that are fired for target attraction and obstacle repulsion give output actions that neutralize each other. Therefore the robot gets into an infinite loop or local minimum [6].

C. Xiong et al. (Eds.): ICIRA 2008, Part I, LNAI 5314, pp. 804–813, 2008.

1.1 Background

In the memory state method of [8], limit cycle problem is resolved by using a distance-based strategy. The variables from which the robot makes ultimate decisions i.e.; to keep turning around the obstacle or to reach the target, are the robot current distance to the target *Dc*, and the robot initial distance to the target *Dm*. The *Dm* is memorized when the obstacle is detected for the first time, while *Dc* is continuously measured and compared against *Dm*. Therefore until the robot is inside the concave obstacle where *Dc>Dm* it follows the interiors of the concave obstacle. But once the robot is out of the obstacle, *Dc* becomes less than *Dm* and therefore the robot finds its way to the target. However, distance-based decision making sometimes result in poor trajectories, because the robot has to do extra wall following even when it is outside the concave obstacle just to meet the target seeking criteria (*Dc<Dm*).

Another solution to the limit cycles problem is the method of Krishna, and Kalra [6]. This method is a real-time collision avoidance algorithm with the local minimum problem resolved by classifying the environment based on the spatio-temporal sensory data sequences. Although this method is robust and finally the robot finds its way, however it highly depends on landmark recognition and therefore needs exact coordination localization. In addition, it is difficult to choose a correct way to follow the wall boundary as shown in many cases [9].

The minimum risk method of [9] for resolving the limit cycles is based on trial-return behavior phenomenon. The method is robust in ordinary cul-de-sac and concave obstacles. However the problems with trial-return motion are; high power consumption and the time spent from start to target. These are the two important issues that are totally ignored in this approach.

The method of Xu, and Tso [4] has good properties in minimum avoidance. Due to actual-virtual target switching, the robot avoids limit cycles. Even in a loop shape like in Fig. 1 (a), still the robot can find the opening at point *C* due to fuzzy logic target seeking and obstacle avoidance behaviors. However, this method shows its major weakness when it fails to reach the goal in such a concave and recursive U-shape environment shown in Fig. 1 (b). "This is because the robot encounters another local minimum at locations *B* and *C* when it is still working under influence of the previous virtual sub goal [location *A*]" [9].

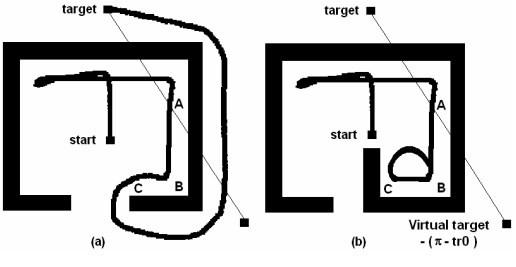

Fig. 1. (a) Simple minimum situation, (b) Inability in multiple minimum situation

2 The Proposed Approach

Unlike other virtual target methods, here, target switching is a smooth clockwise or counter-clockwise shift from actual target direction to tangent direction to the walls. Therefore the robot turns and follows the interiors of the dead-end. This process has no conflict with the FL controller that is controlling the two tasks of obstacle avoidance and target seeking regardless to the target nature. Fig. 2 shows by counter-clockwise target shift, the robot follows the front wall from point *A* to *B*. At point *B*, due to presence of the two walls at front and right, the target again will be gradually shifted *90* degree to left (towards *C*). The robot keeps following the circular motion of the virtual target resulting with wall following behavior until the opening at point *D*.

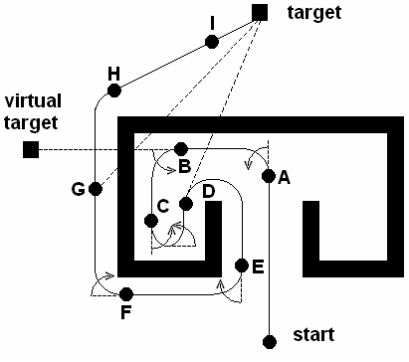

Fig. 2. The proposed method; ideal performance in a sample environment

At this point, the robot must decide either to switch back to the real target, or to keep following the walls. Here, if the sum of turn angles throughout the way is nearly *0*, the robot will decide to go toward the real target. If the sum is positive, the robot generally has had a clockwise motion therefore it has to compensate it with a counter-clockwise motion. And if the sum is negative, a clockwise compensatory motion is required at the opening. In Fig. 2, until point *D*, the robot has had a total of *-360* degrees turning angles. Therefore instead of going towards the real target which results in limit cycles, the robot turns to right due to the virtual target shift, and continues to wall following. This way the robot turns *180* degrees clockwise at the opening and another *90* degrees clockwise from point *E* to *F*.

The last compensation is done by turning right at point *F* where the total clockwise turn becomes equal to *+360* degree. In other words, the sum of turned angles becomes *0*. Therefore, at point *H* where the robot detects the opening it switches the target back from its virtual to real amount, and goes through point *I* until reaches the target.

The algorithm structure consists of FLC and target switch conditions. Basically an FL controller makes decisions for the robot speed, steering, and the amount of next target shift. Then, as soon as the robot makes a turn, the amount of actually turned angle will be added to the sum of the turning angles so far the robot has turned. The sum of turned angles then will be fed back to the upper layer for target switching. The

decision will be to either go towards the next virtual target, or shifting the virtual target towards the actual one, or immediately switching back to the actual target.

Genetic algorithm is used for tuning of the membership functions. However, generation of genetic-fuzzy rule base is not included yet. Therefore an extension to this work would be inclusion of an evolutionary algorithm for generation and optimization of the rule bases. In addition there are other parameters to be optimized such as range of the sonar sensors for obstacle detection, and sensor grouping strategy. As for the algorithm itself, inclusion of a search strategy for better path productions at decision points is also within the future scope of this research.

2.1 Fuzzy Logic Controller

The FL strategy adopted here is to make a compromise between target reaching i.e.; target attracting the robot, and obstacle avoidance i.e.; obstacles repelling the robot. A basic fuzzy logic controller is applied to coordinate the various behaviors. Here, there is no possibility of conflicting behaviors, and implementation of preference-based FL structure [12] is not necessary. The smooth shift of virtual target confines the steering changes to small amounts, and therefore has no conflict with obstacle avoidance. On the other hand, the next amount of target shift itself is affected by obstacle avoidance behavior which results in overall coherency of the system. Like other standard FLs, the behaviors influence the controls directly, so that the response of an individual behavior itself can be a control command that can drive the robot [12].

By dividing the work space *S* into six subspaces with overlaps, fuzzy rules can be defined accordingly. The subspaces for obstacle position are; right *R*, right-front *RF*, front *F*, left-front *LF*, and left *L*, each with *45* degrees of overlap with the next one, and finally no-obstacle *NO* which is the area beyond detection. Also there are three subspaces for the target orientation; right *ur*, front *uf*, and, left *ul*. Therefore, *18* if-then rules are used for defining the outputs of robot's speed, and heading angle.

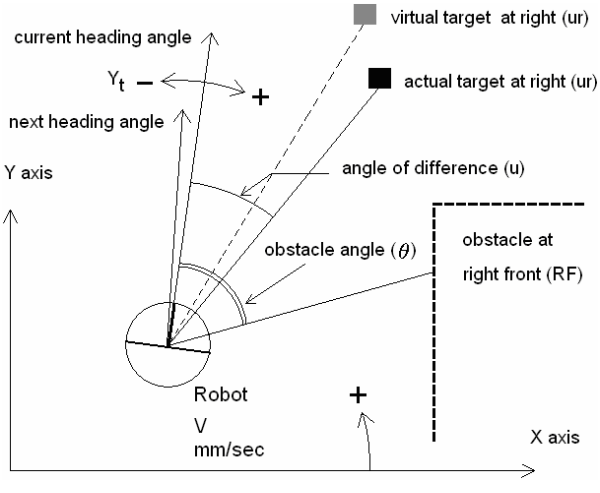

Fig. 3. An obstacle is detected at right front

The array of eight sonar sensors *S0* to *S7* on the robot circumference are mounted at *-90*, *-50*, *-30*, *-10*, *+10*, *+30*, *+50*, *+90* degrees. For more accuracy in sonar readings and therefore more reliable obstacle avoidance, the maximum sonar range is set to *30cm*. However, the robot can not detect any obstacle beyond this distance which is assumed as the no-obstacle area. Fig. 3 depicts the robot in a test environment. The inputs to the fuzzy controller are the obstacle orientation angle θ relative to the robot heading, and the target orientation *u* which is defined as the angle between the robot heading direction and the robot-to-target direction. These two angles at any time are obtained from the sensors by which the robot perceives its environment. The outputs are the steering angle *Yt*, and the robot linear velocity *V* to be applied to the robot left and right wheel actuators.

2.2 Development of Fuzzy Rules

Both input functions, are normalized to *0-1*. The amounts of overlap between the regions are decided with respect to the positions of the sonar sensors on the robot, and are optimized accordingly. The *8* sonar sensors on the robot can be grouped as left sensors *S0-S1*, left-front sensors *S0-S1-S2-S3*, front sensors *S2-S3-S4-S5*, right-front sensor *S4-S5-S6-S7*, and right sensors *S6-S7*. The fuzzy logic operation includes mainly fuzzification, inference, aggregation, and defuzzification. The fuzzification converts the input variables into input grades by means of the membership functions shown in Fig. 4. The inference and aggregation generate resultant output membership functions with respect to fuzzy rules. And finally the defuzzification finds the center of gravity of the output membership functions as the robot's steering angle and linear velocity. The output membership functions are shown in Fig. 5.

The distribution of both input and output functions were optimized based on a genetic-fuzzy approach. Another alternative to improve the membership functions would be based on representing the membership function distributions for each of the variables. Robot navigation is actually governed by rules in the fuzzy controller which represent qualitatively the human driving heuristics. For any N-rules system, it is important that what kind of information needs to be supplied to the FLC and what kind of decisions it needs to provide. In the proposed method, each of the rules gets a value according to positions of the target and obstacles at any time. For example if an obstacle is exactly at right front of the robot, and the target is exactly in front, then *RF=1* and *uf=1* will be obtained from the input membership functions in Fig. 4. Therefore the rule *#8* in Table 1 gets the amount of *1* while the others get *0*.

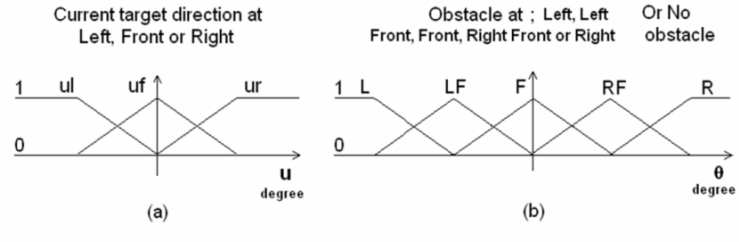

Fig. 4. Input membership functions. (a) Target direction, and (b) Obstacle position.

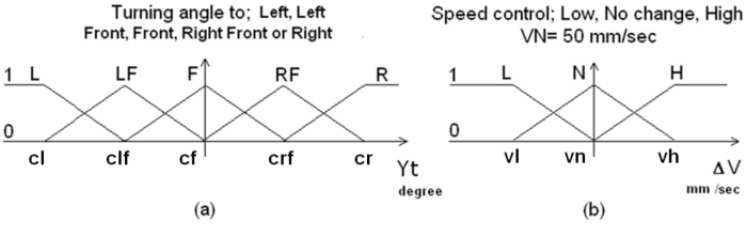

Fig. 5. Output functions. (a) Amount of turning angle, and target shift, (b) Speed control.

The fuzzy rules are given in Table 1, where *ur*, *uf* and *ul* are the weights of the target at right, front and left. As for the obstacles, they can be either at right, right-front, front, left-front or at left all relative to the current heading angle. In case there is no obstacle around, the state will be *NO* or no-obstacle. Since the target i.e.; actual, or virtual, takes *3* and the obstacles take *6* states, therefore only *18* rules have been developed which is advantageous in short execution time and high frequency of sensor readings. As a result, this method is suitable for dynamic environments where fast response to changes is required.

Table 1. If-Then fuzzy rules

Rule #	If	Current virtual target direction		Obstacle position	Then Steering direction	Then Robot velocity
1		ur	&	R	F	L
2		ur	&	RF	LF	L
3		ur	&	F	R	L
4		ur	&	LF	R	N
5		ur	&	L	R	H
6		ur	&	NO	R	H
7		uf	&	R	F	N
8		uf	&	RF	LF	L
9		uf	&	F	F	L
10		uf	&	LF	RF	L
11		uf	&	L	F	N
12		uf	&	NO	F	H
13		ul	&	R	L	H
14		ul	&	RF	L	N
15		ul	&	F	L	L
16		ul	&	LF	RF	L
17		ul	&	L	F	L
18		ul	&	NO	L	H

Defuzzified output for turning angle can be obtained from Eq. 1, where *cr*, *crf*, *cf*, *clf*, and *cl* are constants of the output membership function in Fig. 5 (a). Although these constants are adjustable during programming, however an optimization algorithm resulted to the best amounts with respect to the robot's actual limits for steering. Similarly the defuzzified output for robot velocity can be obtained from Eq. 2, where *vh*, *vn*, and *vl* are constants of the output membership function in Fig. 5 (b). Based on GA optimization, the robot normal speed *VN* was set to *50mm/sec* for highest efficiency and with respect to speed limits, slippage, and power consumption issues.

$$Turning\ \ Angle\ \ (Y_i) = \frac{cr\sum R + crf\sum RF + cf\sum F + clf\sum LF + cl\sum L}{\sum R + \sum RF + \sum F + \sum LF + \sum L} \tag{1}$$

$$Robot\ \ Velocity\ \ (\Delta v) = \frac{vh\sum H + vn\sum N + vl\sum L}{\sum H + \sum N + \sum L} \tag{2}$$

3 Experimental Results

A *Pioneer* robot platform was used to show the performance of the proposed algorithm. In simulation investigation, the starting location of the robot *home*, and the location of the target *goal* were given for each navigation task. The robot is equipped with *8* sonar sensors; *2* sensors at sides, and other *6* at front with *20* degrees interval. A standard simulation software from *ActivMedia* Robotics was used for graphic representation and programming of the robot. The programming language was *Colbert* which is kind of *C++* added with special functions for robot motion and self localization. In this method, unlike the memory state method of [8] the robot does not have extra wall following and goes directly towards the target as soon as it is out of a dead-end trap. And because the motion is not based on trail and return, the robot makes more logical trajectories in compression with the minimum risk approach [9].

Way-finding behaviors of the robot are discussed through an example as depicted in Fig. 6. Starting from start position *S* at zero coordinates, the robot takes a straight path towards the real target with heading angle of about *45* degrees. As the robot is approaching the barrier at point *A*, the target starts to virtually shift from *45* degrees to about *0* degree (graph 1 in Fig. 8 (a)).

Due to the target shift, the steering control (graph 2) which is meant to direct the robot towards the target decreases the heading angle to about *0* degree (Fig. 8 (b)). Therefore the robot follows the barrier until detects the opening. As the robot is passing through the opening, the target angle shifts back to the actual amount, so does the heading angle. Therefore the robot continues its motion at a slightly increased angle towards the target (about *50* degrees) and proceeds to point *B*.

3.1 Local Minimum Avoidance

At points *B*, and then *C*, again due to target switching, the heading angle will smoothly vary from *55* to *0* and respectively from *0* to *-90* degrees. Due to this, the robot follows the interiors of the subspace with heading angle of about *-90* degrees maintained from point *C* to *D*.

The sensitivity of the FL controller system i.e.; obstacle avoidance and target seeking, cause flickering of the virtual target direction as the robot moves along the wall. However it does not significantly affect the heading angle and the robot follows the wall at overall direction of about *-90* until point *D*. Here the target again will be shafted another *90* degrees clockwise to guide the robot to point *E*. The difference between the virtual target angle, and the actual target direction, is now at its maximum (*-180* and *+60* degrees). Therefore, as the robot is coming out from the subspace (point *F*), this difference starts to compensate in counterclockwise direction. This will

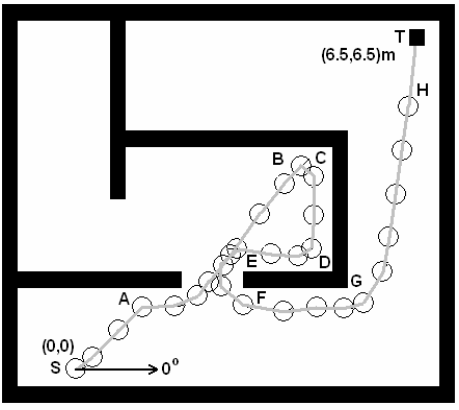

Fig. 6. An example of robot path planning in an environment with a dead-end subspace

result in increase of the heading angle from *-180* to *-90* (from point *E* to *F*), and then from *-90* to *0* (point *F* to *G*) respectively. At point *G* the virtual target has already shifted back to its actual direction which is at about *75* degrees relative to the robot coordinates. Therefore the heading angle smoothly increases to *75* degrees leading the robot to point *H* and therefore to the target (Fig. 8 (b)).

3.2 Results Comparison

Fig. 7 (a) depicts the robot performance in the same type of dead-end as discussed in [4, 6, 8, and 9] as well as in Section 2 of this article. Compared to other methods, the robot accomplished the task more efficiently and without extra field search and wall following. Here it must be noted that, the robot shows the same obstacle avoidance behavior when encountering any types of obstacle. While in some other methods, at least two types of obstacle avoidance behavior, one for convex obstacles, and one for concave ones have been developed [13] (Fig. 7 (b)). The fact that this algorithm highly depends on wall following is shown in Fig. 7 (c). This has the advantage of secure path planning; however, is time-taking compared to more intelligent search strategies.

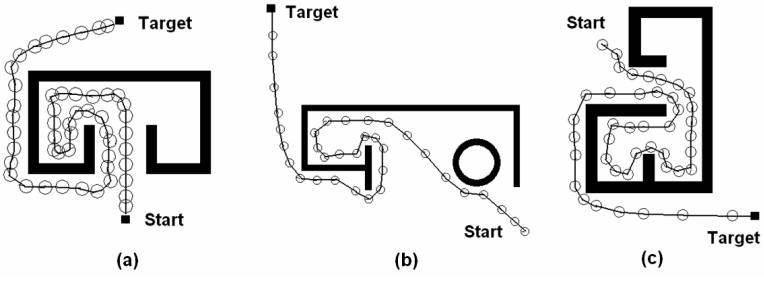

Fig. 7. (a) A typical dead-end, (b) Convex/ concave obstacle avoidance, (c) Wall following

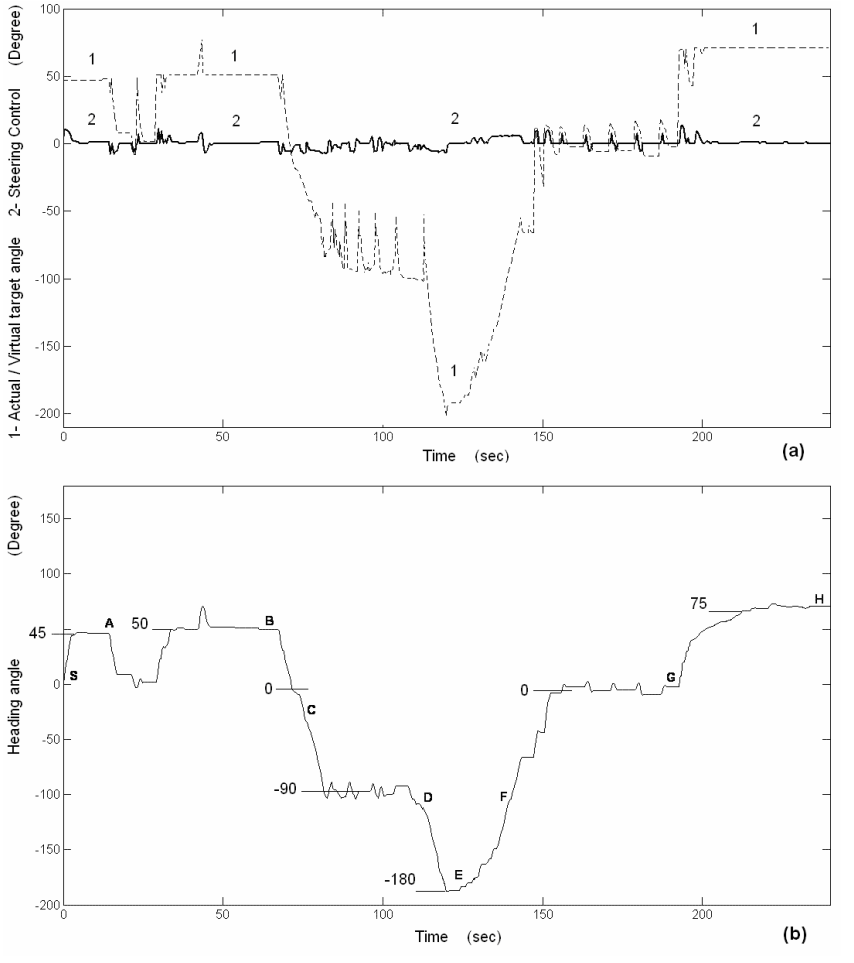

Fig. 8. (a) Graph 1: Target angle (Dotted), Graph 2: Steering control (Solid), (b) Heading angle

4 Conclusion

A new genetic-fuzzy system is developed for reactive navigation of mobile robots. While the fuzzy logic body of the algorithm performs the main tasks of obstacle avoidance, and target seeking, an actual-virtual target switching strategy enables the robot to wall following behavior when needed. This significantly results in resolving the problem of limit cycles in concave obstacles. The distribution of input and output functions are optimized using genetic algorithm for overall efficiency of the system.

In the proposed method, no specific search strategy is adopted, and motion of the robot is merely according to relative direction of the target, and position of detected obstacles. Therefore, despite simpler structure of the FL controller, this may result to longer travel time. Another extension to this work therefore would be integration of search strategies such as the method presented in [14]. Target seeking can be based on

following a series of immediate sub-goals which are in fact vertices of Voronoi diagram of the space. Then an A* search algorithm can be used to identify least-cost navigational paths between the initial and final sub-goals.

References

1. Khatib, O.: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. J. Robotics Research 5(1) (1986)
2. Koren, Y., Borenstein, J.: Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In: The IEEE Conference on Robotics and Automation, pp. 1398–1404 (1991)
3. Tu, K.-Y., Baltes, J.: Fuzzy Potential Energy for a Map Approach to Robot Navigation. J. Robotics and Autonomous Systems 54(7), 574–589 (2006)
4. Xu, W.L., Tso, S.K.: Sensor-Based Fuzzy Reactive Navigation for a Mobile Robot through Local Target Switching. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 29(3), 451–459 (1999)
5. Yang, X., Moallem, M., Patel, R.V.: A Layered Goal-Oriented Fuzzy Motion Planning Strategy for Mobile Robot Navigation. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 35(6), 1214–1224 (2005)
6. Krishna, K.M., Kalra, P.K.: Perception and Remembrance of the Environment During Real-Time Navigation of a Mobile Robot. J. Robotics and Autonomous Systems 37, 25–51 (2001)
7. Ordonez, C., Collins Jr., E.G., Selekwa, M.F., Dunlap, D.D.: The Virtual Wall Approach to Limit Cycle Avoidance for Unmanned Ground Vehicles. J. Robotics and Autonomous Systems (2007)
8. Zhu, A., Yang, S.X.: A Fuzzy Logic Approach to Reactive Navigation of Behavior-Based Mobile Robots. In: The IEEE International Conference on Robotics and Automation, vol. 5, pp. 5045–5050 (2004)
9. Wang, M., Liu, J.N.K.: Fuzzy Logic Based Robot Path Planning in Unknown Environments. In: The International Conference on Machine Learning and Cybernetics, vol. 2, pp. 813–818 (2005)
10. Parhi, D.R.: Navigation of Mobile Robots Using a Fuzzy Logic Controller. J. Intelligent and Robotic Systems 42(3), 253–273 (2005)
11. Zadeh, L.A.: Fuzzy Sets as a Basis for a Theory of Possibility. J. Fuzzy Sets and Systems 1, 3–28 (1978)
12. Selekwa, M.F., Dunlap, D.D., Shi, D., Collins Jr., E.G.: Robot Navigation in Very Cluttered Environments by Preference-Based Fuzzy Behaviors. J. Robotics and Autonomous Systems (2007)
13. Maaref, H., Barret, C.: Sensor-Based Navigation of a Mobile Robot in an Indoor Environment. J. Robotics and Autonomous Systems 38(1), 1–18 (2002)
14. Huq, R., Mann, G.K.I., Gosine, R.G.: Mobile Robot Navigation using Motor Schema and Fuzzy Context Dependent Behavior Modulation. J. Applied Soft Computing 8(1), 422–436 (2008)