Neural Comput & Applic (2014) 24:1569–1581
DOI 10.1007/s00521-013-1393-z

ORIGINAL ARTICLE

Automatic navigation of mobile robots in unknown environments

Omid Motlagh · Danial Nakhaeinia · Sai Hong Tang · Babak Karasfi · Weria Khaksar

Received: 19 June 2012 / Accepted: 21 March 2013 / Published online: 2 April 2013

Abstract Online navigation with known target and unknown obstacles is an interesting problem in mobile robotics. This article presents a technique based on utilization of neural networks and reinforcement learning to enable a mobile robot to learn constructed environments on its own. The robot learns to generate efficient navigation rules automatically without initial settings of rules by experts. This is regarded as the main contribution of this work compared to traditional fuzzy models based on notion of artificial potential fields. The ability for generalization of rules has also been examined. The initial results qualitatively confirmed the efficiency of the model. More experiments showed at least 32 % of improvement in path planning from the first till the third path planning trial in a sample environment. Analysis of the results, limitations, and recommendations is included for future work.

Keywords Local navigation · Reinforcement learning · Associative memory

O. Motlagh (✉)
Department of Robotics and Automation, Faculty of Manufacturing Engineering, University Teknikal Malaysia Melaka (UTeM), 76100 Melaka, Malaysia
e-mail: omid@utem.edu.my

D. Nakhaeinia · S. H. Tang · B. Karasfi · W. Khaksar
Department of Mechanical and Manufacturing, Faculty of Engineering, University Putra Malaysia (UPM), 43400 Serdang, Malaysia

1 Introduction

There are tens of various techniques to materialize the notion of artificial potential fields (APFs) in mobile robot local navigation. Fuzzy logic is indeed one of the most fundamental techniques as described in [1]. However, the focus of this article is not on fuzzy models as they involve expert-based definition of rules which entails reduction in robot's own level of intelligence to cope with uncertainties. Automatic construction of navigation rules has been a challenging topic in local path planning. It was shown that neural network (NN) or hybrid models such as fuzzy cognitive map (FCM) [2] are more appropriate for construction and modification of rules which are initially overlooked in expert-based designs. Learning as the essential element in local navigation could not be simply included in pure fuzzy models. Several attempts have been made to modify fuzzy models using complementary techniques such as memory-based fuzzy models [3–5], landmark learning [6], virtual obstacle [7], virtual target [8], and other search strategies [9, 10], yet they mainly have lack of coherency among algorithms' building blocks besides lack of learning capability compared to NN-based models. More importantly, such models are rather dependent on experts for initial settings, for example, for rules or algorithm conditions and in handling uncertainties and unexpected scenarios. A robot navigating through an unfamiliar environment requires to have at least two behaviors known as (1) obstacle avoidance and (2) target seeking. Without experts or predefined navigation algorithms such as APF [11], the robot is supposed to learn on its own how to reach for a target while avoiding collision with obstacles. Automatic construction of navigation rules has remained as a challenging problem in mobile robotics.

2 Basic navigation behaviors

In contrast to global navigation where the knowledge of the entire configuration space, that is, including location and orientation of all obstacles, targets, and cul-de-sac, is available to the robot, in local navigation, the robot has to detect the space locally relying on its own sensors such as cameras, sonar range finders, touch sensors, etc. Online path planning involves the challenge of exploration throughout the entire environment in seek of a target while avoiding collision with obstacles. Online path planning is indeed fully based on local knowledge, and therefore, in order to find the target, the robot has to explore the entire space usually based on complex search techniques [12, 13]. However, in order to make the problem size smaller, in most local navigation models, the knowledge of the target, that is, merely coordinates of target position but not obstacles, is given to the robot. The robot therefore knows where the target is located and can always keep track of this information along the path by using its wheels' counters and other internal measurement mechanisms, for example compass.

Let us imagine the target is placed at coordinates (x, y). To enable target-seeking behavior, that is, to make the target attracting the robot all the time, the angle of difference [1] or "rotational difference" (RD) must approach 0° for which the robot must continually make clockwise or counterclockwise rotations to maintain its own orientation angle (θ) parallel to the orientation of the (x, y) vector drawn from the robot's center of shape as shown in Fig. 1. The shortest RD between the robot's orientation angle (θ) and the orientation of the (x, y) vector is concerned. Therefore, the obtained value is always in the range $(-180°, +180°)$, which determines the angle that the robot should rotate to stay pointing toward the target. Accordingly, at any time instance, a positive value of RD means that the target is oriented at the left (TL), while a negative value indicates that the target is at right-hand side of the robot (TR).

In the context of APF, while the target is the attractor, that is, enabling target-seeking behavior, obstacles are assigned repeller potentials for the robot to exhibit obstacle avoidance behavior. These two are the very basic behaviors a robot needs for secure local navigation in an unknown environment. The robot's range finder sensors return values indicating the robot's distance to obstacles situated within the coverage of the sensors' radiation cones. The examined robot of this research as shown in Fig. 2 has been equipped with three range finders at right, front, and left with 60° intervals and 60° radiation cone. By setting the range of obstacle detection to, for example, 1 m for each of the sensors, obstacles may or may not be detected at either direction: obstacle at left (OL), obstacle in front (OF), obstacle at right (OR), or no obstacle (NO).

3 Expert-based navigation

As described above, the target (if only one target is aimed) gets two possible states either target at left (TL) or target at right (TR), while obstacles get four possible states. Accordingly, the following rules as given in Table 1 could be defined to satisfy the required behaviors for efficient path planning toward the target. The outputs are the left and right wheels' velocities shown as (LWV) and (RWV) to be applied to the robot's differential wheels. The rules are generated based on a couple of simple principles described as follows:

- Obstacle avoidance is given higher priority than target seeking.
- When an obstacle is in front, both wheels' velocities should be reduced.
- When there is an obstacle at left, left wheel velocity shall increase.
- When there is an obstacle at right, right wheel velocity shall increase.
- When target is at right, left wheel speeds up, and when target is at left, right wheel speeds up.

The frames in Fig. 3a through d are taken from the MobotSim robotic simulator [14] showing the robot

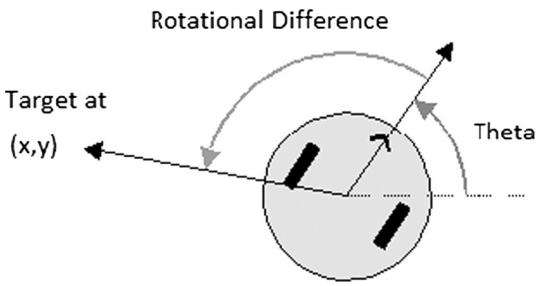

Fig. 1 Calculation of rotational difference: positive value indicates anticlockwise rotation [14]

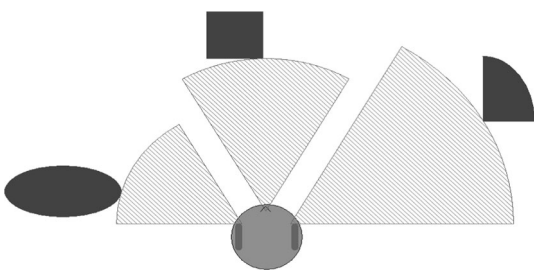

Fig. 2 The robot equipped with three range finder sensors to detect obstacles

Table 1 Simple rules satisfying two behaviors of obstacle avoidance and target seeking

IF	OBSTACLE	TARGET	THEN	LEFT WV	RIGHT WV
IF	OL	TL	THEN	HIGH	–
IF	OF	TL	THEN	LOW	LOW
IF	OR	TL	THEN	–	HIGH
IF	NO	TL	THEN	–	HIGH
IF	OL	TR	THEN	HIGH	–
IF	OF	TR	THEN	LOW	LOW
IF	OR	TR	THEN	–	HIGH
IF	NO	TR	THEN	HIGH	–

performance under the developed rules in a sample environment without dead ends.

The reason for exclusion of concave obstacles and dead ends is that the robot does not realize limit cycles. Besides, the rules do not provide the robot with minimum avoidance behavior. Additional techniques such as the method in [1] are therefore needed to incorporate such a behavior. Based on the existing behaviors, that is, obstacle avoidance and target seeking, the robot can navigate around obstacles and toward the target. The problem is that the navigation system is critically influenced by the designers' setup for rules. Another designer, for example, could develop the rules based on other principles, for example, target given higher priority than obstacles. Therefore, besides dependency on experts, another disadvantage is that there are various possible setups of rules, and yet, none of them could guarantee the best performance and results.

4 The proposed strategy

Through utilization of neural networks, expert-independent navigation is made possible as robot learns navigation behaviors on its own based on the principles of reinforcement learning. Regardless of initial expert setup of rules and robot's own past experiences, learning is a continuous process to ensure improvement in robot performance over time along the path. However, over training is not a concern as the reward and punishment principles lead to distinct behaviors of target seeking and obstacle avoidance. It also enables for dealing with uncertainties as the robot gets more matured. To model the robot's work space, first a set of neurons should be assigned to each of the possible situations that may occur. As described in the APF model,

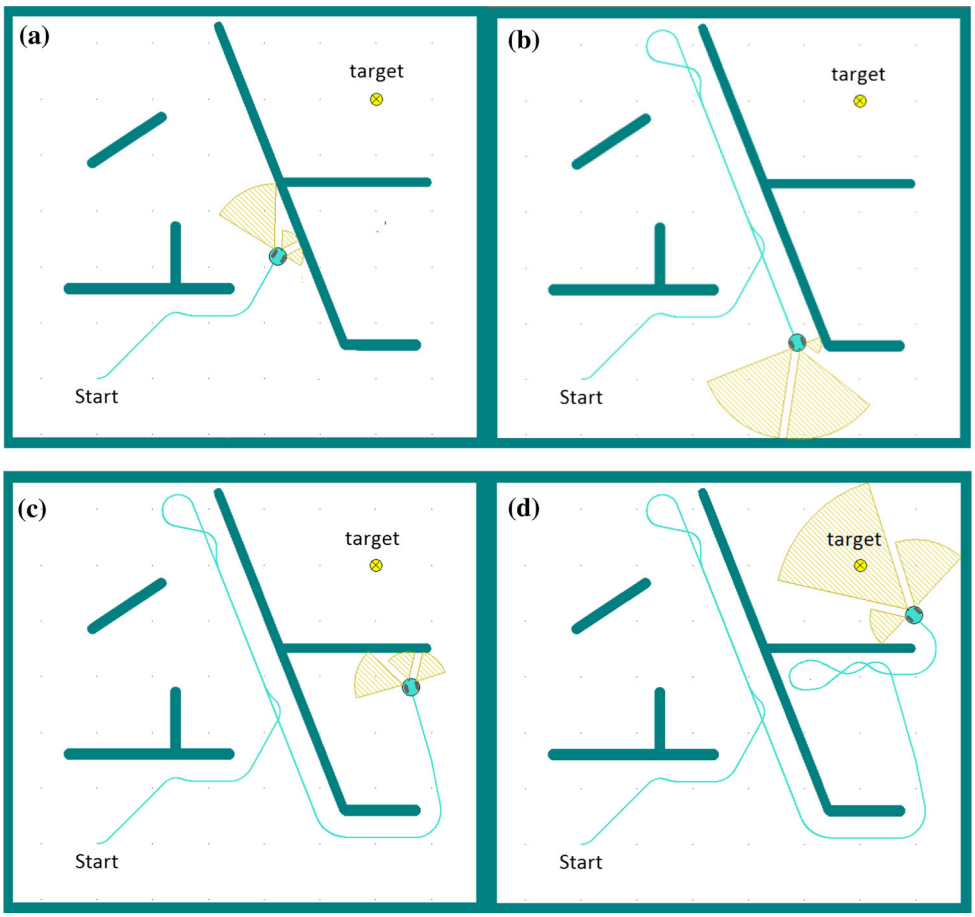

Fig. 3 Robot navigation under the rules given in Table 3

depending on the number and orientations of sensors, obstacles could be detected at different directions relative to the robot. The target as well, it could be detected at different orientations.

For the examined robot as shown in Fig. 2, possible situations for orientation of target and obstacle are listed as follows: obstacle at left (OL), obstacle in front (OF), obstacle at right (OR), no obstacle (NO), target at left (TL), and target at right (TR). Therefore, at any instance of time, depending on the situation being observed through the robot's sensory channels, the robot decides to take an action of whether to increase or decrease its wheels' velocities. Therefore, the possible actions the robot may take due to occurrence of a situation include left wheel velocity high (LWH), left wheel velocity low (LWL), right wheel velocity high (RWH), and right wheel velocity low (RWL). A neural model is then required to associate such situations with respective actions. The associative memory (AM) [15, 16] of Fig. 4 is proposed as a perfect match for this purpose. While the connections between the left and right sets of neurons are the counterparts of the navigation rules, their weights are initially unknown without need for initial setup. In other words, assignment of initial random weights allows for independency from expert design.

In order to implement the learning strategy, the algorithm of Fig. 5 has been developed. First, the software setup is completed including configuration of the robot, for example, with three sensors at 60° intervals and 60° radiation cone in the experiments and construction of a random AM matrix of weights (W). There are eight possible arrangements of obstacle and target orientations relative to the robot, that is, $X1 \ldots X8$, while X_i is a vector representing (OL, OF, OR, NO, TL, TR). These are in fact the minimum number of situations as only two possible orientations are considered for target with only four possible orientations for obstacle. While the eight situations are known and could be detected anytime, the respective actions are unknown in the beginning, and therefore, arbitrary amounts have to be assigned. Therefore, eight random sets of values are assigned to the output variables, that is, $Y1 \ldots Y8$, where Y_i represents (LWH, LWL, RWH, RWL). Construction of the weights is based on AM learning principle as given in Eq. 1. A common activation function for AM is the *sign* function. It also has the advantage of reducing the number of output neurons from four to only two neurons representing left wheel velocity (LWV) and right wheel velocity (RWV) with positive or negative values to enable differential steering.

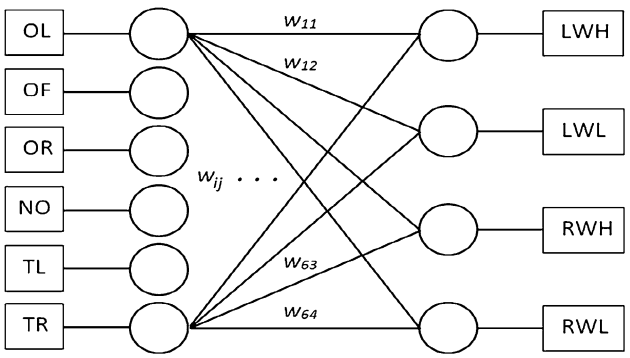

Fig. 4 The developed associative memory (AM) for relating inputs and outputs

$$W = \sum_{n=1}^{8} X_n Y_n^T \tag{1}$$

The robot is then put to experience a sample constructed environment with randomly scattered convex obstacles and a known target. As the experiment starts, the robot detects the current situation X. The respective action is then retrieved from the AM according to the activation rule given in Eq. 2.

$$Y = \text{sign}(W^T X) \tag{2}$$

Upon applying the action, that is, *StepForward* command in MobotSim, the robot will be either rewarded which is due to getting closer to the target whether distance-wise or direction-wise or will be punished due to getting too close to an obstacle, going far from the target, or both. The extent of reward or punishment is determined according to the extent of target seeking and obstacle avoidance being fulfilled or unfulfilled which then will be used to decide for an alternative action. The key issue is that a reward encourages the robot to proceed with the existing sequence of situation–actions, while a punishment makes the robot retreat and reconsider alternative actions.

In the event of punishment, as the robot is retreating to the previous position, an alternative action must replace the existing one. The matrix of weights (W) should be therefore reconstructed based on the current situation, the altered action, and the other seven pairs of situation–actions to memorize the decided alternative action for the current situation. The entire process starts from the beginning, and the learning cycles never end throughout the whole experiment until reaching the target. The robot then stores the last obtained weights, that is, memorizes the experiment, to utilize it for better performance in future. The reason for retreating is that the robot should first scape the risky situation that might entail a collision. Another reason is that regardless of the actual size of the robot, and despite differential steering capability, still some distance is needed for any left or right turn (i.e., to alter the action) to be safely made. In other words, since at many occasions wheels' velocities are not equally opposite, the robot

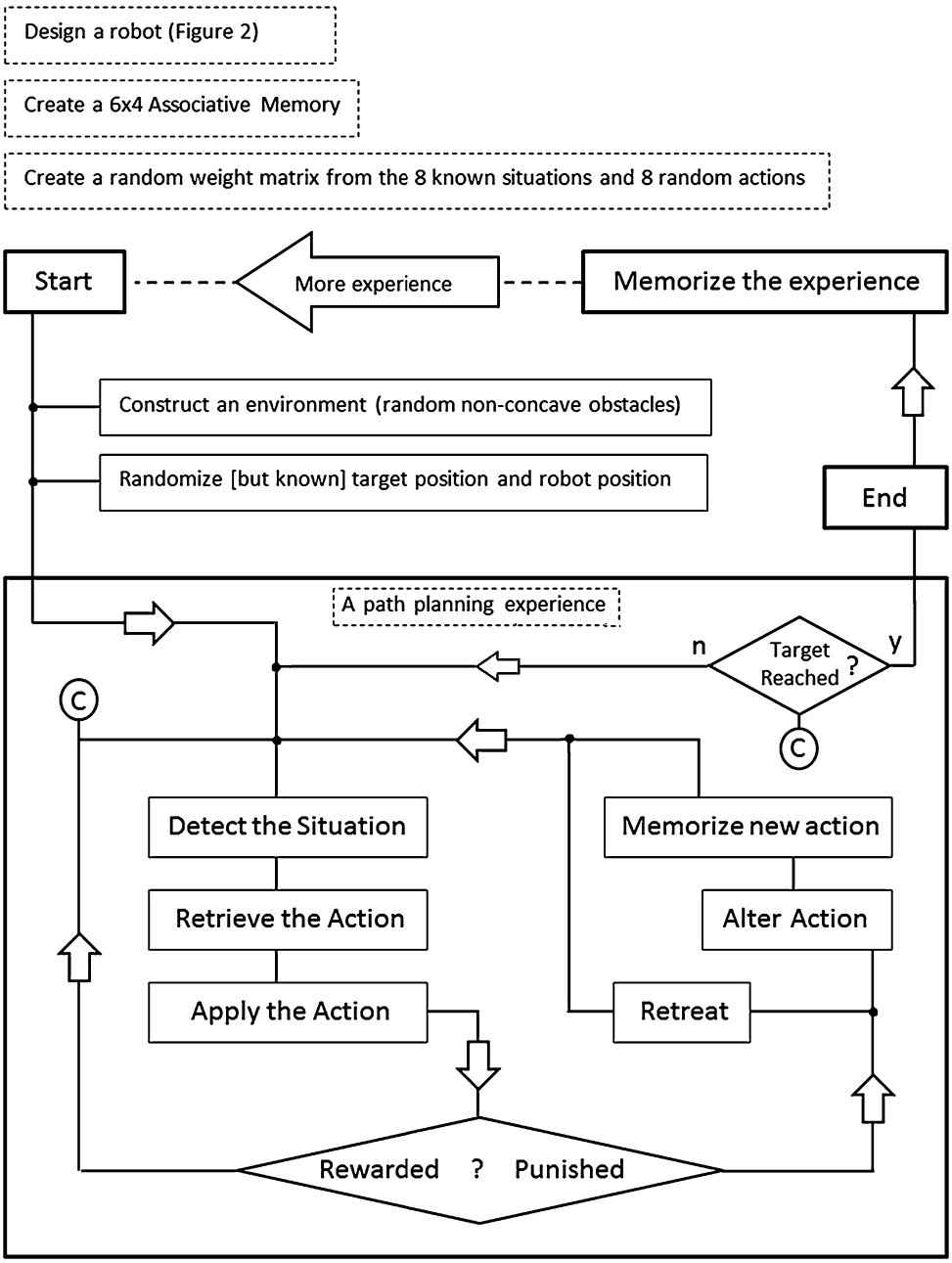

Fig. 5 The learning strategy implemented through reinforcement learning principles

cannot make immediate turns and therefore requires some distance from the obstacle in front to successfully bypass it.

5 Simulation results and analysis

To conduct navigation experiments using the developed learning strategy, the following steps have been taken. At first, the application of AM in robot navigation has been tested with one and then two basic behaviors. Then, capability of the model in generalization of rules, that is, performing beyond the defined problem size, has been evaluated. And lastly, as the third step, the algorithm's learning capacity has been analyzed through a number of random experiments.

To accomplish steps one and two, the *basic* code has been developed for MobotSim to navigate the robot. Two types of expert-based designs, that is, excluding the learning phase, were examined merely to test the feasibility of AM in robot navigation and to compare its performance against fuzzy methods. In the first example, the basic behavior of target seeking was examined according to the following situation–actions given in Table 2, where $+1$ and -1 denote activation and deactivation of a sensor in input or positive and negative activation of wheels' actuators in output. The applied AM's weight matrix is shown with W.

Table 2 Expert-based target-seeking rules using associative memory

Possible Situations	X1	X2	X3	X4	X5	X6	X7	X8
	↓	↓	↓	↓	↓	↓	↓	↓
OL :	-1	-1	-1	1	-1	-1	-1	1
OF :	-1	-1	1	-1	-1	-1	1	-1
OR :	-1	1	-1	-1	-1	1	-1	-1
NO :	1	-1	-1	-1	1	-1	-1	-1
TL :	-1	-1	-1	-1	1	1	1	1
TR :	1	1	1	1	-1	-1	-1	-1
Expert-Defined Actions	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8
	↓	↓	↓	↓	↓	↓	↓	↓
LWV :	1	1	1	1	-1	-1	-1	-1
RWV :	-1	-1	-1	-1	1	1	1	1

$$W = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -8 & +8 \\ +8 & -8 \end{bmatrix}$$

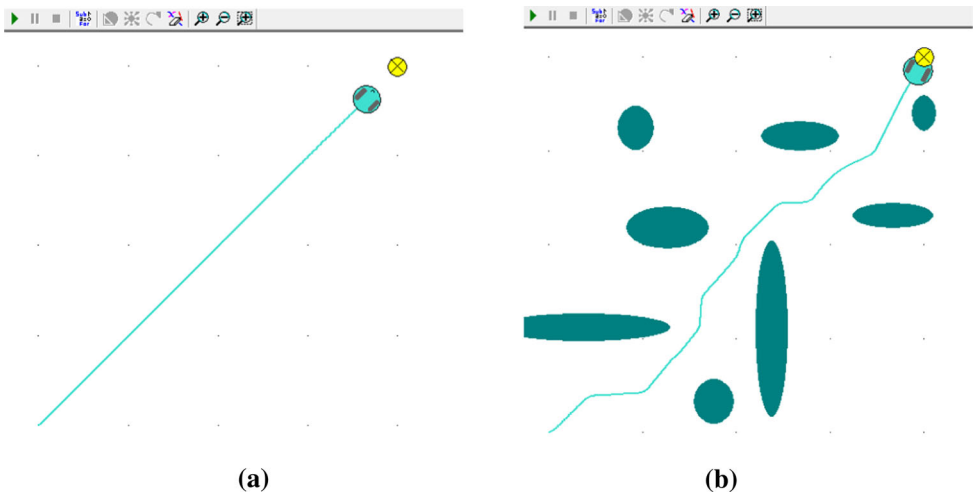

Fig. 6 **a** Target-seeking behavior under AM's stored situation–actions and **b** both target-seeking and obstacle avoidance behaviors under AM's memorized situation–actions

As shown in Fig. 6a, the robot showed satisfactory performance by directly navigation toward the target, that is, the circled cross, under the AM's expert-defined weights. A more complex situation, however, was to examine the system with both behaviors of target seeking and obstacle avoidance. The fuzzy rules given in Table 2 were exactly transformed into their NN counterpart as shown in Table 3.

Figure 6b shows the algorithm performance under the expert AM of Table 3. The robot successfully managed to reach for the target while avoiding obstacles along the path. This signifies that AM is a perfect replacement for traditional fuzzy models except for the path shape which is not as smooth as fuzzy-generated paths. This is due to the discrete nature of binary input and output of the neural networks. However, there are advantages including generalization of navigation rules, and the learning capacity, which far outweigh this single disadvantage.

An advantage of NN-based models is in dealing with uncertainties through generalization of certain rules. In the experiment shown in Fig. 6b, it was observed that not only the system is able to perform as good as fuzzy models, but it could also handle situations which have not been initially included among its predefined navigation rules. For example, Fig. 7a, b shows situations along the path in Fig. 6b, where more than one sensor is triggered at a time. While these situations are not included among the eight possible situations as given in Table 3, the robot could handle the obstacles. Although the same phenomenon happens in aggregation of fuzzy rules, but the difference here is that NN models do not need extra instruction to perform such aggregation processes.

The third stage of the experimental work has been about evaluation of the algorithm's learning capabilities. Through reinforcement learning, the robot showed continuous learning behavior and improving path planning skills in variety of constructed environments. The principles of learning lie on possible punishments on the one hand and immediate or delayed rewards on the other hand. The punishments occurred anytime the robot approached an obstacle too closely, while the rewards were given anytime the robot approached the target or turned toward the target's direction. However, giving higher priority to obstacle avoidance, in the events of concurrent punishment and

Table 3 Expert-based target seeking and obstacle avoidance using associative memory

Possible Situations	X1	X2	X3	X4	X5	X6	X7	X8
	↓	↓	↓	↓	↓	↓	↓	↓
OL :	-1	-1	-1	1	-1	-1	-1	1
OF :	-1	-1	1	-1	-1	-1	1	-1
OR :	-1	1	-1	-1	-1	1	-1	-1
NO :	1	-1	-1	-1	1	-1	-1	-1
TL :	-1	-1	-1	-1	1	1	1	1
TR :	1	1	1	1	-1	-1	-1	-1

Expert-Defined Actions	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8
	↓	↓	↓	↓	↓	↓	↓	↓
LWV :	1	-1	-1	1	-1	-1	-1	1
RWV :	-1	1	-1	-1	1	1	-1	-1

$$W = \begin{bmatrix} 6 & -2 \\ -2 & -2 \\ -2 & 6 \\ 2 & 2 \\ -2 & 2 \\ 2 & -2 \end{bmatrix}$$

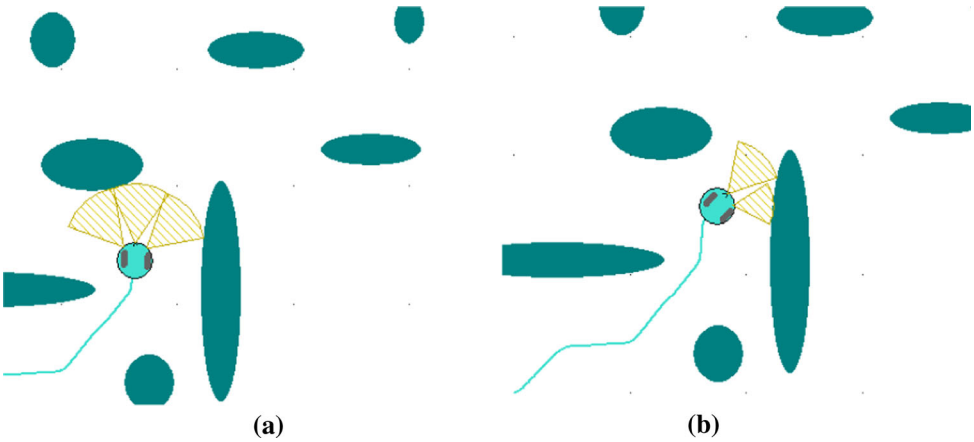

Fig. 7 **a** An undefined situation along path due to simultaneous occurrence of OL, OF, and OR. **b** An undefined situation with simultaneous OF, and OR

reward, the rewards were delayed as they encourage for target-seeking behavior which could overcome the obstacle avoidance behavior.

Figure 8a shows an experiment in a sample environment in MobotSim, while Fig. 8b shows the second trial in the same environment. In both experiments, the knowledge of the robot is limited to position of the target only. The position and orientation of obstacles have been detected locally with sonar sensors set to a relatively short range of 0.4 m to allow online local navigation. To start learning of obstacle-related and target-related rules simultaneously, the knowledge of target position was kept hidden from the robot until the first punishment (point P in Fig. 8a).

While the first experiment (Fig. 8a) starts with a random matrix of weights, the second experiment (Fig. 8b) starts with a relatively tuned matrix from the previous experiment. Therefore, the improvement in navigation skills is clear in experiment 8b compared to the first experiment with exactly similar setting of parameters. In Fig. 8c, going through the third experiment, the robot takes a new path starting from point C. It is obviously seen that the path is smoother, and despite taking a new path, the decisions are more firm. In fact, the robot in experiments "(Fig. 8a, b)" is still not well trained. It tends to make many mistakes and takes inefficient paths. However, during experiment (Fig. 8c), thanks to sufficient training, the robot takes a better path with less redundancy. This new path turns out to be the robot's ultimate choice which is also obtained in experiment (Fig. 8f).

In the fourth experiment as shown in Fig. 8d, the robot is put into a new challenge as the initial setting in terms of sensitivity to punishments was increased so that the robot would retreat more intensely. The robot showed more refractory behaviors in the beginning. However, along the path its motion changed to be smoother except for the last two obstacles which might have been due to an unknown computation issue or a newly discovered situation.

Based on generalization principle, however, such problems are resolved in the next experiment. Therefore, it is seen that in experiment of Fig. 8e, despite making another change, that is, reducing sensitivity to punishment, and taking a new path, the robot performs well. It could be said that in both experiments shown in Fig. 8c, f, the robot seldom gets a punishment as it is already well trained and easily navigates toward the target based on continuous rewards it receives.

In fact, Fig. 8c, f shows the robot's best performance that could be obtained with existing setup, including

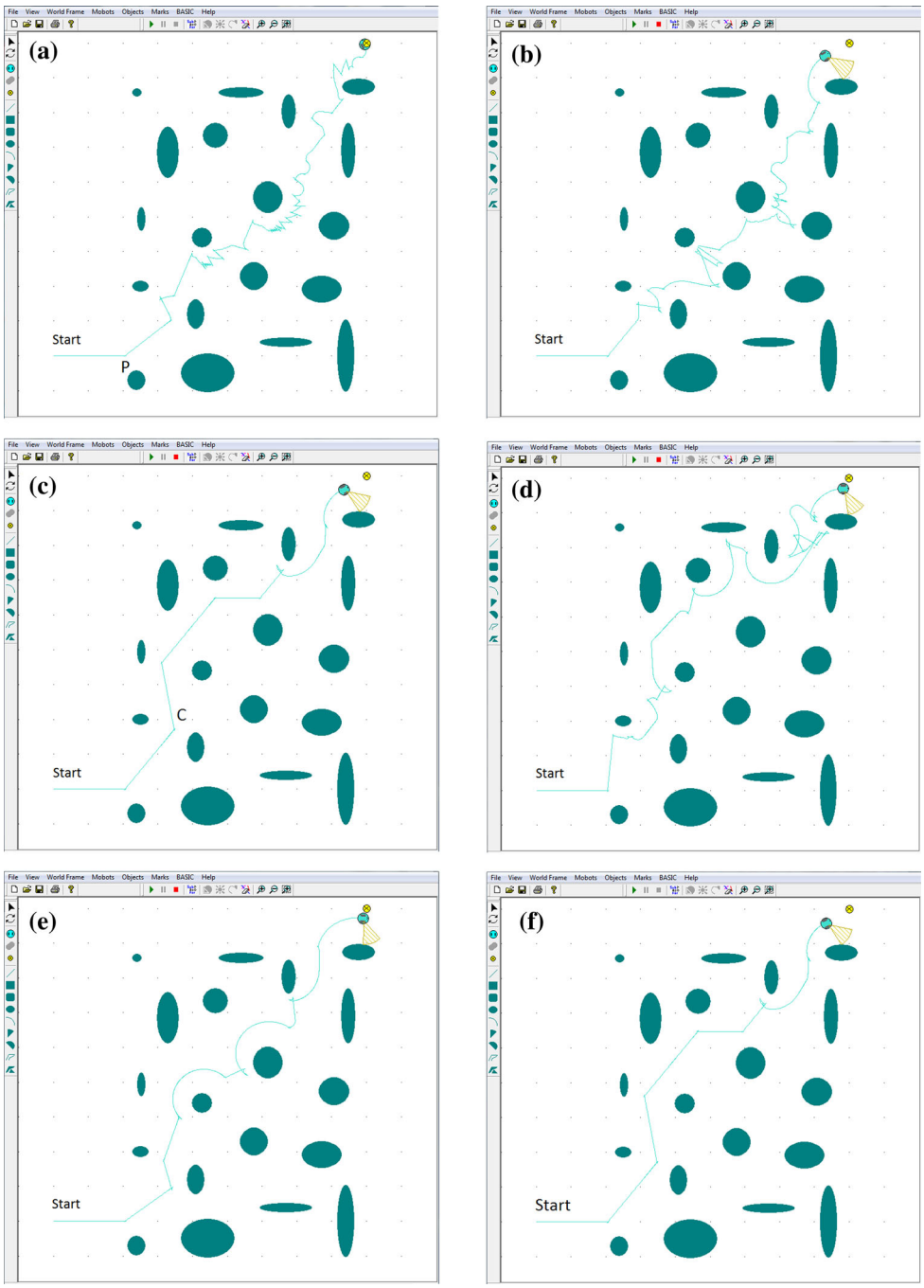

Fig. 8 A training experiment starting from **a** and repeating with updated *W* through **b** and **c**. Setting changes are made in **d** and **e** which will again converge to the same path as **c** in **f**

number of sensors, their orientation and range, range of obstacle detection for punishment, punishment sensitivity or the extent of reaction in the event of punishment, and other simulator parameters and features of the robot geometry.

Performance graphs: Robot performances in Fig. 8a, f are compared. Velocity graph, for example, velocity graph in Fig. 9a for the experiment shown in Fig. 8a, is not very informative as it is too crowded. However, other measures, such as distance to target over time in Figs. 9b and 11, target direction in Figs. 10 and 12, and the total traveling time or total number of decision instances, are more suitable measures to compare the performance of the robot before and after sufficient training in Fig. 8a, f, respectively.

From comparison of the graphs in Figs. 9b and 11, it could be seen that the robot in experiment of Fig. 8a needs to make 2,565 path decisions to travel from the start point to the target, while the same distance is covered by the robot in experiment of Fig. 8f with only 1,755 decision instances. This signifies 32 % improvement in terms of obtaining shorter traveling time. The actual amount of the

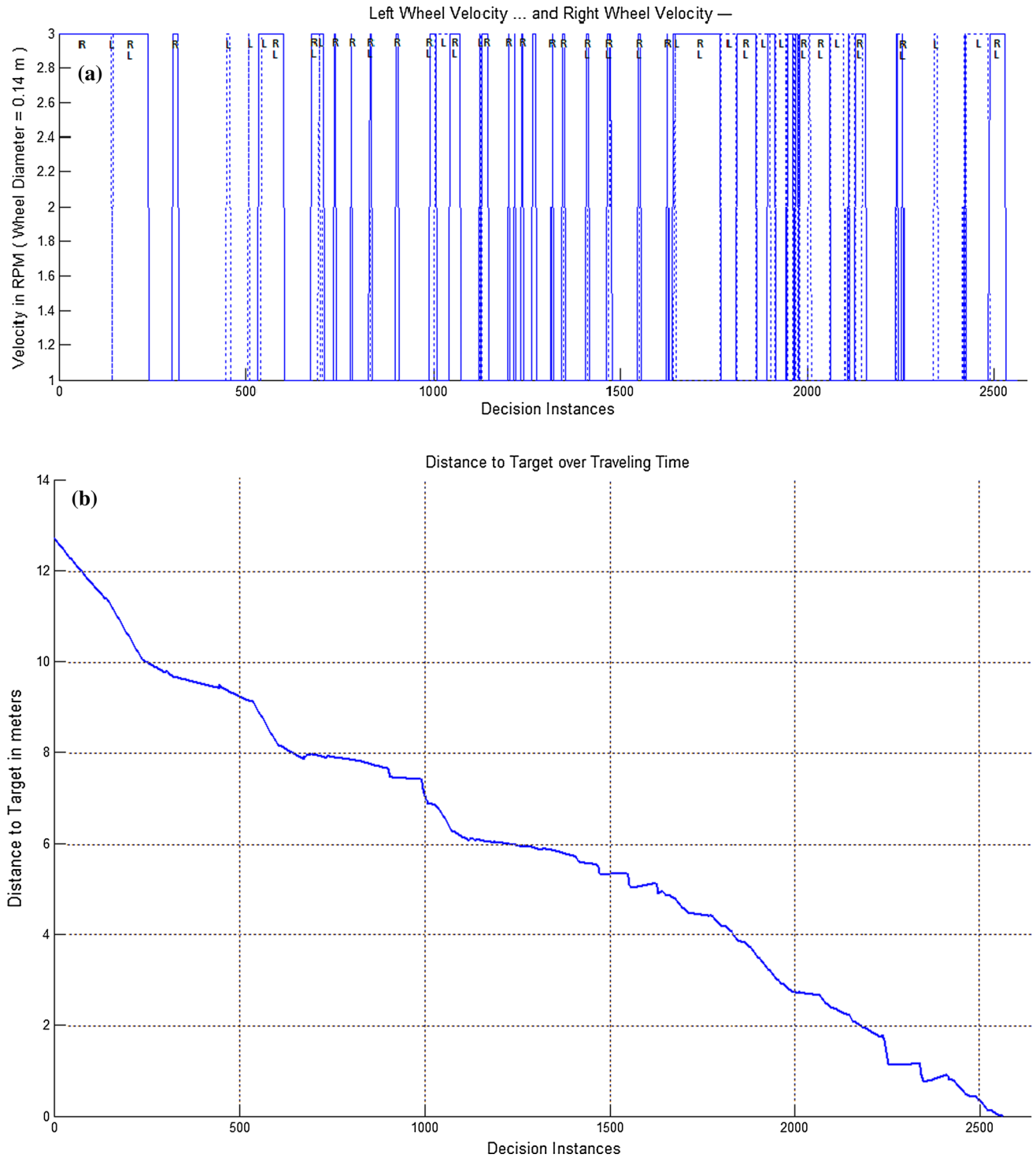

Fig. 9 **a** Left and right wheels' velocities and **b** distance to target, for the experiment of Fig. 8a

travel time itself depends on the speed of the processor, and therefore, with similar processors, less number of decision instances means shorter travel time and therefore shorter path length in experiment of Fig. 8f compared to Fig. 8a. And lastly, from comparison between target direction (rotational difference) in the graphs of Figs. 10 and 12, it could be observed that the robot has made a smoother trajectory in experiment of Fig. 8f compared to Fig. 8a. While the graph of Fig. 10 has too much of fluctuation and at many points crosses zero, the graph of Fig. 12 has much

Fig. 10 Target direction over time for the experiment shown in Fig. 8a

less fluctuation and does not cross zero unless it is necessary as dictated by the obstacles as encountered along the path.

Other experiments: More experimental work was conducted in several other environments. Figure 13a–d shows two different environments each of which experienced twice by the robot: second trial and third trial. While experiments all confirmed the effectiveness of the developed training model, there were drawbacks discovered at some points. The most annoying issue was that despite sufficient training, due to discrete binary outputs applied as wheels' velocities, the robot was not able to make smoother trajectories. In fact, a minimum non-zero offset has been necessary for forward velocity to avoid equal positive and negative values at wheels which would make the robot stop and rotate around itself at some points. These are disadvantages never seen in fuzzy models as their outputs are continuous and seldom generate such stoppage and futile rotations of differential wheels. However, the coherency of the system in terms of structure, that is, merely consisting of neurons, makes it possible to incorporate more neurons for obtaining digital outputs with more bits to generate scaled velocity controls at wheels. Incorporation of neuro-fuzzy models or other strategies for utilization of gray inputs and outputs would also make the trajectories smoother and more efficient in terms of path length and power consumption.

6 Conclusion

An effective strategy was presented for automatic generation of navigation rules required for local path planning of mobile robots. By associating possible situations (as they occur and are sensed by the robot along the path) with respective actions (applied to robot actuators) through supervised learning, the robot was made capable of concurrent learning and path planning. Currently, the robot behaviors are limited to target seeking and obstacle avoidance, while inclusion of necessary wall following and minimum avoidance behaviors

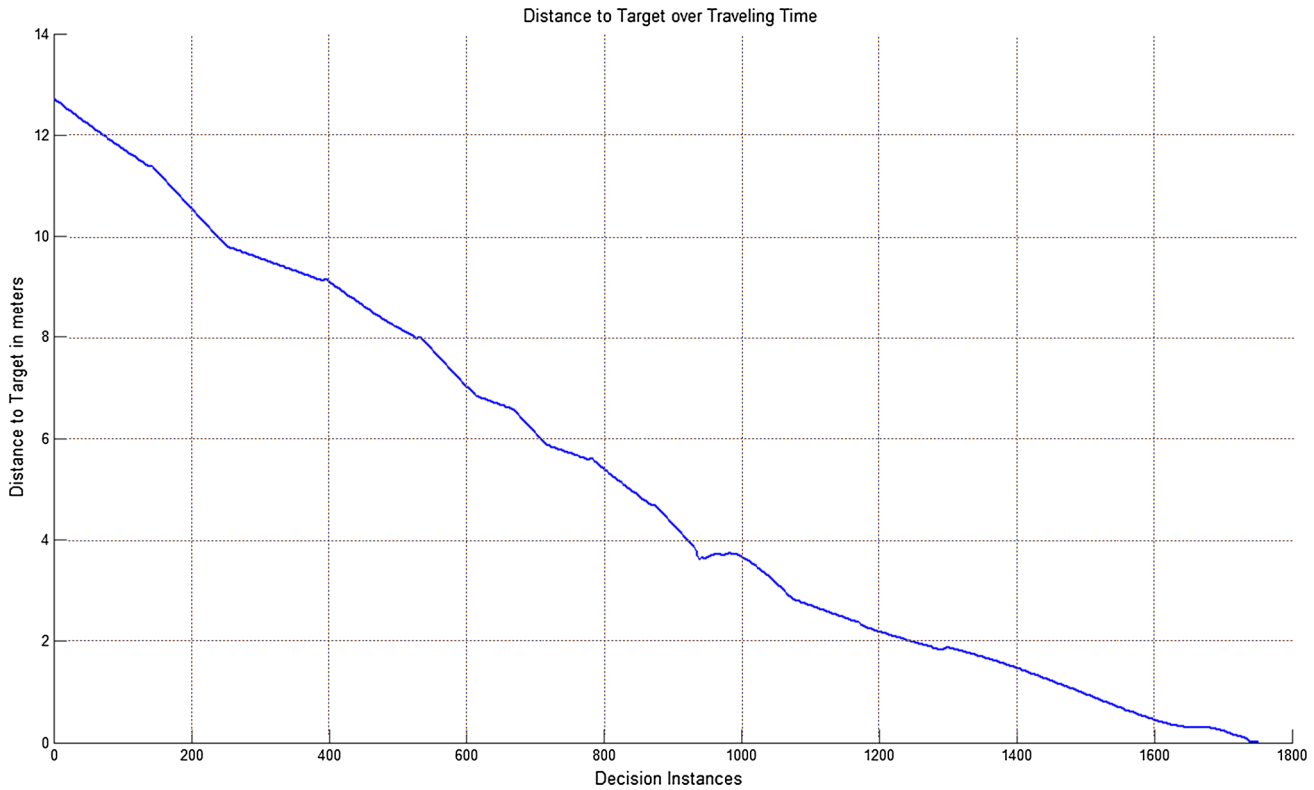

Fig. 11 Distance to target for the experiment of Fig. 8f

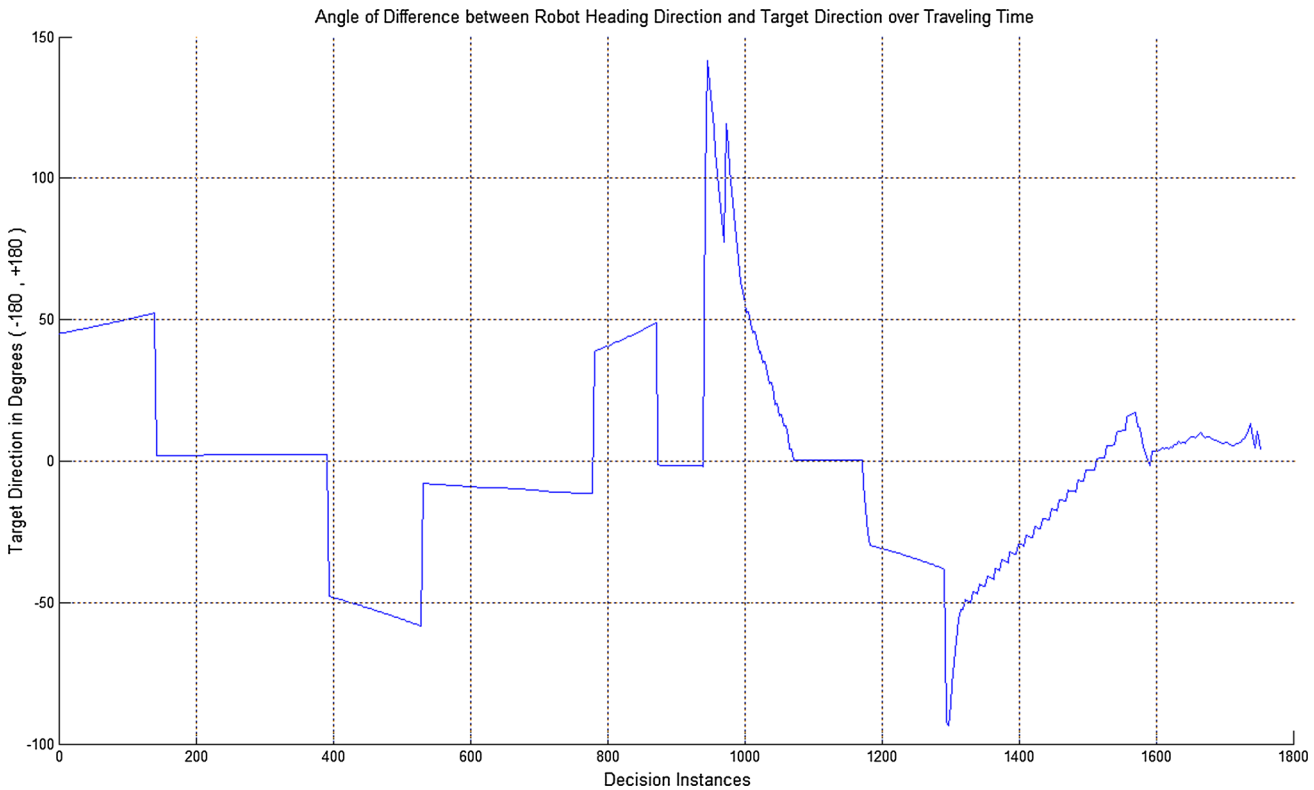

Fig. 12 Target direction over time for the experiment shown in Fig. 8f

Fig. 13 **a** Environment 1 after trial 2, **b** environment 1 after trial 3, **c** environment 2 after trial 2, **d** environment 2 after trial 3

are left as the future works using other techniques such as fuzzy cognitive map [17]. Issues such as incorporation of fuzzy inputs and outputs as well as compatible (NN-based) search techniques for unknown targets are also within the future scope of this research. Another technique involves path prediction when environment is dynamic and motion of all objects including the robot itself must be predicted for robust obstacle avoidance [18].

References

1. Motlagh O, Tang SH, Ismail N (2009) Development of a new minimum avoidance system for a behavior-based mobile robot. Fuzzy Sets Syst 160(13):1929–1946
2. Motlagh O, Tang SH, Ismail N, Ramli AR (2012) An expert fuzzy cognitive map for reactive navigation of mobile robots. Fuzzy Sets Syst 201:105–121
3. Zhu A, Yang SX (2004) A fuzzy logic approach to reactive navigation of behavior-based mobile robots. In: Proceedings of the 2004 IEEE international conference on robotics and automation, 5:5045–5050
4. Hui NB, Mahendar V, Pratihar DK (2006) Time-optimal, collision-free navigation of a car-like mobile robot using neuro-fuzzy approaches. Fuzzy Sets Syst 157(16):2171–2204
5. Selekwa MF, Dunlap DD, Shi D, Collins EG Jr (2007) Robot navigation in very cluttered environments by preference-based fuzzy behaviors. Rob Auton Syst 56(3):231–246
6. Krishna KM, Kalra PK (2001) Perception and remembrance of the environment during real-time navigation of a mobile robot. Rob Auton Syst 37:25–51
7. Ordonez C, Collins EG Jr, Selekwa MF, Dunlap DD (2007) The virtual wall approach to limit cycle avoidance for unmanned ground vehicles. Rob Auton Syst 56(8):645–657
8. Xu WL, Tso SK (1999) Sensor-based fuzzy reactive navigation for a mobile robot through local target switching. IEEE Trans Syst Man Cybern C Appl Rev 29(3):451–459
9. Huq R, Mann GKI, Gosine RG (2008) Mobile robot navigation using motor schema and fuzzy context dependent behavior modulation. Appl Soft Comput 8(1):422–436
10. Tu K-Y, Baltes J (2006) Fuzzy potential energy for a map approach to robot navigation. Rob Auton Syst 54(7):574–589
11. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Int J Rob Res 5(1):90–98
12. Acar EU, Choset H (2002) Sensor-based coverage of unknown environments: incremental construction of morse decompositions. Int J Rob Res 21:345–366
13. Fekete SP, Schmidt C (2010) Polygon exploration with time-discrete vision. Comput Geom 43:148–168
14. MobotSim Version 1.0 (2001) Fully functional trial version of MobotSim. Retrieved from http://www.mobotsoft.com/?page_id=70
15. Stamova IM, Ilarionov R, Krustev K (2011) Asymptotic behavior of equilibriums of a class of impulsive bidirectional associative

memory neural networks with time-varying delays. Neural Comput Appl 20(7):1111–1116
16. Leung ACS, Pui FS, Ho K (2011) The effect of weight fault on associative networks. Neural Comput Appl 20(1):113–121
17. Motlagh O, Tang SH, Ramli AR (2010) An FCM modeling for using a priori knowledge: application study in modeling quadruped walking. Neural Comput Appl 21(5):1007–1015
18. Motlagh O, Tang SH, Ismail N, Ramli AR, Samin R (2009) A new path estimation strategy for predicting blind persons' motion in indoor environments. J Asian Arch Build Eng 8(2):371–377