Solving the Local Minima Problem for a Mobile Robot by Classification of Spatio-Temporal Sensory Sequences

K. Madhava Krishna, Prem K Kalra*
Department of Electrical Engineering
Indian Institute of Technology
Kanpur, India
e-mail: kalra@iitk.ac.in

Received 3 January 2000; accepted 28 June 2000

The local minima problem occurs when a robot navigating past obstacles towards a desired target with no priori knowledge of the environment gets trapped in a loop. This happens especially if the environment consists of concave obstacles, mazes, and the like. To come out of the loop the robot must comprehend its repeated traversal through the same environment, which involves memorizing the environment already seen. This paper proposes a new real-time collision avoidance algorithm with the local minima problem solved by classifying the environment based on the spatio-temporal sensory sequences. A double layered classification scheme is adopted. A fuzzy rule base does the spatial classification at the first level and at the second level Kohonen's self-organizing map and a fuzzy ART network is used for temporal classification. The robot has no prior knowledge of the environment and fuzzy rules govern its obstacle repulsing and target attracting behaviors. As the robot traverses the local environment is modeled and stored in the form of neurons whose weights represent the spatio-temporal sequence of sensor readings. A repetition of a similar environment is mapped to the same neuron in the network and this principle is exploited to identify a local minima situation. Suitable steps are taken to pull the robot out of the local minima. The method has been tested on various complex environments with obstacle loops and mazes, and its efficacy has been established.

* To whom all correspondence should be addressed.

1. INTRODUCTION

Path planning is one of the key issues in mobile robot navigation. Autonomous mobile robots are used in various applications such as in automatic freeway driving,[1] cleaning of hallways,[2] and exploration of dangerous regions.[3] These applications demand robust and adaptable methods for path planning.

Path planning is traditionally divided into two categories, global path planning and local path planning. In global path planning, prior knowledge of the workspace is available. Some of the global approaches include the configuration space method,[4] potential field method,[5] the generalized Voronoi diagram,[6] and free space representations.[7] The planning is done offline and the robot has complete knowledge of its work area and its path when it starts. One of the main issues in this area of work is to reduce the time complexity of these algorithms and minimize the cost of the search.[8–10]

Local path planning methods use ultrasonic sensors, laser range finders, and on-board vision systems to perceive the environment and planning is done online. The workspace for the navigation of the mobile robot is assumed to be unknown and consisting of stationary obstacles. One of the earliest implemented strategies of this type is the wall following method,[11] where the robot motion is based on moving adjacent to the walls at a finite distance. The drawback is that the algorithm fails in cluttered environments and if the walls are in the form of loops. Other online approaches to path planning include edge detection,[12,13] virtual force field,[14] and fuzzy logic[15–17] methods. Fuzzy rulebase techniques have an advantage in that they do not require an analytical model of the environment, but designing the rulebase is heuristic. Though fuzzy path planning schemes work well in cluttered environments they fail in environments where the rules that are fired for target attractor and obstacle repulsor modules give output actions that neutralize each other and the robot gets into an infinite loop or a local minima. There are methods in literature that tackle the local minima problem such as the *Bug* algorithms of Lumelsky[18,19] and their improvements.[20,21] Recently a virtual obstacle approach[22] and a virtual target approach[23] have been proposed as possible solutions for the local minima problem. Wall following is a common strategy adopted in general for surmounting the local minima situation and many algorithms slip into a wall following mode upon encountering the first obstacle en route to the target. However, the approaches[20,22,23] do try to evaluate whether the robot is in a trapped situation before a suitable strategy is invoked to overcome the minima. Among this the approach of Huang[20] makes an empiric guess to identify the robot's trapped condition while the recent strategies[22,23] are more robust in determining the same.

The main contribution of this article is in the identification of the local minima situation during the robot's traversal, much akin to the way a human might understand his trapped state by recollecting some of the landmarks he had seen in his last traversal of the same environment. This remembrance is provided by a classifier network, which classifies the spatio-temporal sequences of sensor readings. A two layered classification scheme is employed. At the first layer fuzzy rulebase does the spatial classification. At the second layer Kohonen's SOM[24] and the fuzzy ART[25] network is used for learning and classifying the temporal sequences of spatial patterns classified by the first layer. The classification scheme imparts an understanding of the robot's local environment and correlates the same with previous experiences of a similar environment. The robot's local environment is classified in terms of landmarks through each neuron in the SOM or ART layer whose weight vector represents a landmark. When the robot sees a similar landmark at the same spatial location where it had seen it previously it understands its entanglement in a loop. Suitable actions are then taken to pull the robot out of its trap. The SOM interprets the local environment of the robot in terms of landmarks learnt offline and can recall previously stored patterns but is not plastic enough to new situations. The online classification property of fuzzy ART makes the classifier plastic to new patterns but its ability to understand the environment is poor compared to the SOM. Hence the SOM and ART networks are employed in the second layer to avail the advantages of both.

The organization of the article is as follows. A fuzzy navigation scheme for guiding the robot past unknown obstacles is discussed in section 2. The learning of the spatio-temporal sequence of sensory inputs is dealt in section 3. The robustness of the algorithm is shown through simulation in section 4. Section 4 also compares the present algorithm with an existing approach,[20] which is an extension of

Lumelsky's *Bug* algorithm. Section 5 presents the conclusions and future directions.

2. REAL-TIME NAVIGATION PROBLEM

The objective of the sensor-based navigation of a mobile robot is to reach the target in any unknown workspace, cluttered with obstacles of any shape, size, and orientation. The decision for the proper turn angle of the mobile robot is taken based on sensory information and the angular difference between the robot's current direction of motion and the goal orientation with respect to origin of the reference frame. The input space, U, of the mobile robot can be represented as

$$u = [\, u_s \quad u_{df} \,]^T \qquad u \in U$$

where $u_s^T = [\, u_0 \;\; u_1 \;\; u_2 \;\; u_3 \;\; u_4 \;\; u_5 \;\; u_6 \,]^T$, the sensor vector, denotes the range readings obtained by the seven sensors placed in the form of an arc on the circumference of the robot, subtending an angle of 105 degrees at the center. The input u_{df} denotes the angular difference between the mobile robot's instantaneous direction vector and the vector joining the robot's center to the target, also called as the difference angle.

Ultrasonic sensors are in general used to obtain information regarding the local environment. In our simulation we have used an array of seven ultrasonic sensors (0–6). They have been grouped as the left (sensors 0–2), center (3), and right (4–6) arrays. The actual intricacies involved in the measurement of time of flight values using ultrasonic sensors have not been modeled in our simulation algorithm.

2.1. Fuzzy Inference Engine

The sensor inputs which are in the form of pixels from the current location of the robot to the obstacle position are normalized to $[0, 1]$. The inference engine partitions the problem into two main modules. One module governs the target reaching action and the other governs the obstacle avoidance action of the robot.

Target Reaching Module

The input difference angle in the range of $[-180, 180]$ is fuzzified using the membership function, which can be represented as

$$\mu_{\text{left}}(u_{df}) = \begin{cases} \dfrac{-u_{df}}{90} & \text{for } -90 \le u_{df} \le 0 \\ 1 & -180 \le u_{df} \le -90 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\mu_{\text{right}}(u_{df}) = \begin{cases} \dfrac{u_{df}}{90} & \text{for } 0 < u_{df} \le 90 \\ 1 & 90 \le u_{df} \le 180 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The defuzzified output turn angle due to the target reaching behavior, y^t, is obtained as

$$y^t = \frac{c_l \times \mu_{\text{left}}(u_{df}) + c_r \times \mu_{\text{right}}(u_{df})}{\mu_{\text{left}}(u_{df}) + \mu_{\text{right}}(u_{df})} \tag{3}$$

where c_l, c_r are constants whose values take -9 and $+9$, respectively.

Obstacle Avoidance Module

The membership functions that determine the degree of farness or nearness to the obstacle are defined as follows. The inputs which are distances of the obstacles to the sensors are measured in terms of pixels and normalized to $[0, 1]$.

$$\mu_{\text{near}}(u_i) = \begin{cases} -2.5 \times \mu_{\text{near}}(u_i) + 0.75 & 0.3 \le u_i \le 0.7 \\ 1 & 0.7 \le u_i \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $i \in I \cong \{0, 1, \ldots, 6\}$

$$\mu_{\text{far}}(u_i) = \begin{cases} 2.5 \times \mu_{\text{far}}(u_i) + 0.75 & 0.3 \le u_i \le 7 \\ 1 & 0.7 \le u_i \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $i \in I \cong \{0, 1, \ldots, 6\}$

The output defuzzified turn angle due to the obstacle avoidance behavior, y^o, is given by

$$y^o = \frac{\sum_i n_i \, \mu_{\text{near}}(u_i) + \sum_i f_i \, \mu_{\text{far}}(u_i)}{\sum_i \mu_{\text{near}}(u_i) + \sum_i \mu_{\text{far}}(u_i)} \tag{6}$$

where $i \in [0,6]$ and n_i, f_i are constants determined in a heuristic fashion.

The net angle by which the robot must turn is decided by assigning weights to the individual modules depending on the closeness of the obstacle to the robot. The weights ascribe the proportion of importance to either of the modules. The angle y by which the robot shall finally turn after assignment of weights can be represented as

$$y = \mu_{\text{far}}(u_l)\mu_{\text{far}}(u_c)\mu_{\text{far}}(u_r)y^t + \bigcup(\mu_{\text{near}}(u_l), \mu_{\text{near}}(u_c), \mu_{\text{near}}(u_r))y^o \quad (7)$$

where $\bigcup(A, B, C) = A + B + C - AB - BC - CA + ABC$ and the factors by which the angles due to target reaching, y^t, and obstacle avoidance behavior, y^o, are multiplied represent the weights assigned to the modules.

The robot turns by the angle y, obtained as above, and traverses some distance in that direction before sampling the environment for its next input. The distance s (measured in pixels) the robot moves between any two samples of the environment is obtained as

$$s = w_1 s^t + w_2 s^o \quad (8)$$

where w_1, w_2 are the weights assigned in the same vein as in Eq. (7) and s^t, s^o are the distances due to target reaching and obstacle avoidance behavior, which are given by

$$s^t = \frac{s_1 \times \mu_{\text{near}}(u_t) + s_2 \times \mu_{\text{far}}(u_t)}{\mu_{\text{near}}(u_t) + \mu_{\text{far}}(u_t)} \quad (9)$$

where u_t represents the Euclidean distance of the target from the current position of the robot's center and $s_1 = 1$, $s_2 = 3$.

$$s_o = \frac{s_n \times \max_{i \in I}(\mu_{\text{near}}(u_i)) + s_f \times \min_{i \in I}(\mu_{\text{far}}(u_i))}{\max_{i \in I}(\mu_{\text{near}}(u_i)) + \min_{i \in I}(\mu_{\text{far}}(u_i))} \quad (10)$$

where $I = \{0, 1, \ldots, 6\}$.

Though the above approach works well when the environment is highly cluttered it fails when the obstacles are in form of loops and bends due to the contradicting actions imposed by the target attracting and goal repulsing modules. The robot consequently is trapped in an infinite loop. This is because a purely fuzzy approach fails to provide some kind of a *memory or remembrance of the environment the robot has already traversed*, which is needed for the robot to come out of such loops. Hence there is the need for incorporating memory in the navigation algorithm, the topic of the next section.

3. SPATIO-TEMPORAL CLASSIFIER

Artificial neural networks have proved through the last decade their powerful classification properties as a result of their inherent abstraction and generalization capabilities. Kohonen's self-organizing maps (SOM) and ART networks are one of the most popular unsupervised learning mechanisms. The SOM has "*a special property of effectively creating spatially organized internal representations of various features of input signals and their abstractions*"[24] while the advantage of fuzzy ART is that it is very stable to previously stored inputs and plastic to new inputs. Both the schemes have been used in various applications such as speech recognition,[26] vector quantization,[27] and robotics.[28,29] A brief review regarding the theory and learning of a SOM network is given in Appendix A and the fuzzy ART is given in Appendix B.

3.1. Spatial Classification

The need for an initial spatial classifier can be understood as follows. At any instant t when the robot samples its environment it obtains the sensory vector $u_s(t) = [u_0(t)\ u_1(t)\ u_2(t)\ u_3(t)\ u_4(t)\ u_5(t)\ u_6(t)]^T$. At each instant t a vector of dimension seven is present as the input and a sequence of such vectors must be learnt and stored by the classifier. In other words the classifier network must learn various combinations of sequences $u_s(t), u_s(t+1), \ldots, u_s(t+n-1)$. The complexity of such a learning algorithm increases tremendously in space and time. The classifier would need a network of over 20,000 neurons to store the various combinations without loss of significant data and the learning time involved is huge. In order to avoid this each of the seven dimensional patterns at a given instant is mapped into a particular class without losing essential data. This can be considered as a vector quantization problem and a fuzzy classification scheme is employed for this. Then the SOM or the ART can be trained on the sequence of quantized vectors or classes rather than on a sequence of the input vectors itself. The classification is done as follows. Ini-

tially the sensory vector $u_s(t)$ is represented as $u_s(t) = [u_l(t)\ u_c(t)\ u_r(t)]^T$ where l, c, and r are interpreted as left, center, and right sensor readings. The sensors 0, 1, and 2 are grouped together as left and their minimum reading is taken as the reading of the left sensor; i.e., $u_l(t) = \min\{u_0(t), u_1(t), u_2(t)\}$. Similarly $u_r(t) = \min\{u_4(t), u_5(t), u_6(t)\}$ and $u_c(t) = u_3(t)$. Now based on these left, center, and right readings a fuzzy classification scheme is employed as shown in the table below.

The left, center, and right sensors readings are given to the fuzzy rule bases (described in section 2) and a reading is classified as very near if the nearness degree of the reading is more than 0.9, as near if the nearness degree is in $[0.3, 0.9]$, and as far if nearness degree is less than 0.3. Thus each input vector $u(t)$ is classified into one of the nine classes (as shown in Table I) and the SOM is trained over a sequence of such classes.

The next issue in consideration is the upper bound on n, the number of classes in the sequence over which the second layer of the classifier shall be trained. To obtain information regarding the environment it is essential to keep track of changes in the classes of a sequence. Often the changes in the sequence are more important than simply the sequence itself. It has been observed through experience with various simulation environments that all the important landmarks of an environment can be represented by one or two changes of a sequence. For example, in Figure 1(a) a robot meeting a dead end while going through a narrow corridor is represented as 333333333333110 over 15 time instants whereas in Figure 1(b) the robot traverses to a dead corridor of shorter length and hence the sequence obtained is 333333110 within nine instants.

Hence a robot passing through a corridor to hit a dead end is represented by the following temporal

Table I. Spatial classification of the sensory input space by fuzzy rule base.

Right sensor	Center sensor	Left sensor	Class
Very near	Very near	Very near	0
Near	Near	Near	1
Near	Near	Far	2
Near	Far	Near	3
Near	Far	Far	4
Far	Near	Near	5
Far	Near	Far	6
Far	Far	Near	7
Far	Far	Far	8

Figure 1. (a) Robot passing through a long corridor with a dead end. (b) A similar shorter coorider.

sequence $[3\ 1\ 0]^T$. The landmark is represented by two changes in the sequence viz. $3 \rightarrow 1, 1 \rightarrow 0$. Hence the bound on the number of classes in the temporal sequence is fixed to be 3. In general a two or three dimensional vector of classes can represent a landmark. The structure of the double layered classifier network is shown in Fig. 2. The structure represents the process of extracting the temporal order of the classes before giving it to the second layer. As discussed in the previous paragraph any two consecutive classes in the final sequence to be classified must be different. The comparator (Fig. 2) which compares the present class $C(t)$ with the class of the previous instant $C(t-1)$ does this. When they are different then $C(t-1)$ is extracted to form the first class in the sequence to be classified by the second layer. The process is repeated and when three such classes are extracted the sequence, y_1, y_2, y_3 (see Fig. 2) is input to the second layer of the classifier.

3.2. Temporal Classifier

The second layer as mentioned earlier classifies the sequence of classes of dimension three. The fundamental difference in the way the SOM and ART networks learn and classify gives rise to different results. The SOM models the local environment of the robot into well defined landmarks such as corners (meeting of two walls), mazes, and blind ends that are encountered in a typical environment. Each neuron in the SOM lattice codes a particular landmark. This requires that the SOM be trained for

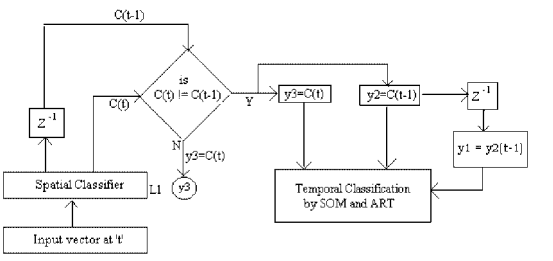

Figure 2. Structure of the Spatio-temporal Classifier Network.

various kinds of landmarks offline before using the trained lattice during real time. During real-time navigation the robot can encounter a temporal sequence of classes for which the SOM was not trained. This can under some exceptional situations still give rise to an infinite loop situation, which is discussed in section 4. One way to circumvent this problem is by training the SOM offline for an exhaustive combination of sequences. Since each sequence is of dimension three, no two consecutive classes in a sequence are same, and the total number of classes as classified by the spatial classifier is nine, the SOM shall be a map of 648 (9X8X9) neurons. But most of the neurons represent combinations of sequences, which have no possibility of occuring in any real-time environment. As a matter of fact experiments suggest 70% of such a lattice or 454 neurons do not win at all in various real-time environments. Another solution for this is to incorporate an online learning network in the second layer. Fuzzy ART with a complement coding scheme serves as a good alternative. Fuzzy ART is capable of learning new patterns and simultaneously remembers the earlier patterns to a reasonable accuracy. It eliminates the need for maintaining a map of large neurons, most of which are idle. The ART network can dynamically add new patterns according to the local environment of the robot and can afford to forget patterns that have not come within a recent time interval, say in the last 100 samples of the environment, thus memorizing the environment to the extent needed. But the SOM is not entirely useless for it can correlate the local environment seen at real-time in terms of landmarks learnt offline. This kind of classification by the SOM provides the algorithm with an immediate memory (IM) and is discussed in section 4. The ART, however, cannot provide for this, as it does not understand a classified pattern to represent a particular landmark. It just learns a pattern, but is not sure what it represents in the real world. It should be noted that the terms immediate and distant memory used in this article have nothing to do with the long term and short term memories of the standard ART architectures. The classifier network has been incorporated with both the SOM and ART network in its second layer to avail the advantages of both. The SOM is a lattice of 49 neurons with their weights representing some of the common landmarks, while the ART is used as a kind of backup when the robot encounters a new pattern not learnt by the SOM. It has been found that the ART needs to maintain not more than 15 neurons whose weights represent the most recent patterns not identified by the SOM. Thus ART also serves to reduce the space required by the algorithm from 648 vectors, each of dimension 3, to 64 vectors (49 from SOM + 15 due to ART) of the same dimension.

3.2.1. Learning Landmarks and the Vector of Lower Bounds

Since the SOM identifies a real-time sequence to a predefined landmark the minimum number of times (the lower bound) each class must occur in a sequence is to be known. This prevents misidentification of spurious sequences as landmarks. For example, in landmarks such as a narrow corridor leading to a dead end (Fig. 1), the first class in the sequence, 3, must occur at least four times, the second class must occur at least twice, and the last should occur once. The minimum number of times each class occurs in a sequence is termed the vector of lower bounds (Fig. 3).

A simulation environment consists of various landmarks, some of which are shown in Figure 4, consisting of corners (meeting of two walls) of various orientations and shapes, a long wall with a narrow slit, half open doors, narrow corridors with dead ends, etc. These are landmarks that can be

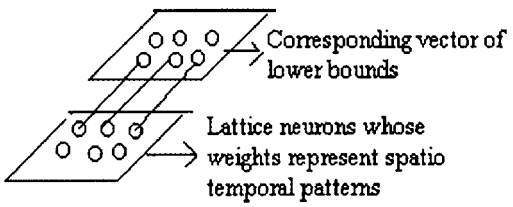

Figure 3. Lattice neurons and the corresponding vector of lower bounds.

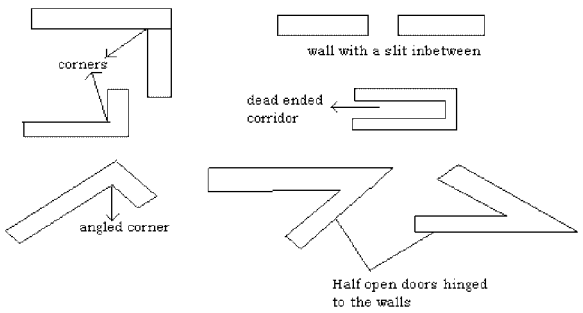

Figure 4. A simulation environment with some of the landmarks.

expected in any typical environment. The robot is made to navigate by placing starting and target locations across the landmarks. The robot navigates using the fuzzy algorithm and the temporal sequences formed as described in Section 3.1 are stored as an array of vectors. Each vector in the array is a vector of classes. An example is shown in Figure 5, where the start and target locations are given across a landmark in the form of a right bend.

The SOM is initialized with a lattice of 7×7 Kohonen neurons with normalized random values. From the array a particular vector is selected at random and presented to the lattice as the input. The winner is selected and the winner and the neighboring neurons are updated as described in Appendix A. The network is trained for several iterations. The number of iterations should be of two orders more than the total number of neurons in the lattice[30] for the SOM to capture the essential topology of the input space, in this case to identify clearly various kinds of landmark. Once the SOM has been trained for various landmarks the robot is again made to navigate across the same landmark with reduced sizes for obtaining the vector of lower bounds for that landmark. The smallest of such landmarks that the SOM can recall is identified. The number of occurrences of each class in the sequence for the smallest landmark is stored as a lattice neuron whose weights represent the lower bounds. The neuron is stored in the position corresponding to the lattice position of the winning neuron in the SOM. Thus for each neuron in the SOM representing a particular landmark by the vector $[c1\ \ c2\ \ c3]^T$ there exists a corresponding neuron which stores the vector of lower bounds $[t1\ \ t2\ \ t3]^T$ where $t1$ represents the minimum number of times $c1$ must occur in a sequence (Fig 3). Otherwise, for example, the sequence $[3, 1, 0]$ without considering the lower bounds can possibly be something other than a narrow dead-ended corridor.

Figure 6 shows how the SOM has been able to map similar landmarks to neighboring positions in the lattice upon training. Training the SOM for a particular landmark involves considerable visualization by the user with respect to the particular shape of a landmark and various sizes of that shape. It becomes an involved affair to train the SOM for the exhaustive combination of 648 sequences and their corresponding lower bounds. Without the vector of lower bounds a map of 648 neurons can identify any sequence in real-time but the obvious advantage of SOM is lost. The same can be done with much less memory by the fuzzy ART architecture and without offline learning.

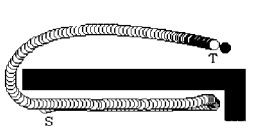

Figure 5. Sensor sequences obtained for SOM learning by navigating the robot across a right bend.

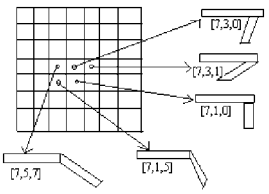

Figure 6. Some of the landmarks coded by the Kohonen neurons. The weight vector is shown along with landmarks.

The classification by fuzzy ART architecture is a relatively simpler affair. As the robot navigates through the environment the sensory inputs are obtained and classified by the first layer of the classifier network. The temporal order of classes is extracted. A sequence of three such classes with no two consecutive classes being identical forms the input vector for the fuzzy ART. The fuzzy ART maps the sequence with an earlier one or adds a new spatio-temporal pattern to the existing set of patterns. The decision for adding a new pattern is decided by the vigilance parameter ρ, which is set to 0.9 in our algorithm. A review of the fuzzy ART algorithm is given in Appendix B.

4. SIMULATION RESULTS AND COMPARISON

To test the efficacy of the algorithm a graphical simulator has been developed on a Pentium machine. The robot has been modeled as a circle of size 5 pixels and imaginary sensors, seven in number, are placed in the form of an arc along the circumference of the robot subtending an angle of 105 degrees at the center. The robot rotates about its center which is a reasonable assumption considering the fact that real robots such as the LABMATE and ROVER do rotate about their centers. Each sensor sends rays within a cone of 15 degrees. The minimal distance obtained within the cone of each sensor is

considered as the distance of the obstacle from that sensor and is available as input to the algorithm.

According to the input vector the fuzzy algorithm makes its decision. In our simulation the position of all the obstacles in the workspace is unknown to the robot. The robot is aware of only its start and final positions. The purely fuzzy algorithm performs well in a highly cluttered environment as shown in Figure 7. This is because the goal attracting and the obstacle repulsing modules work in tandem and steer the robot to the target.

4.1. Immediate Memory Due to SOM

The immediate memory (IM) can be termed as the property by which the robot understands and remembers its most recent environment.

It can reduce path lengths as seen from Figures 8 and 9. Figure 8 is a landmark in the form of a right angled corner such as meeting of the two walls. The target lies on the other side of the bend. The locations marked ''a'' and ''b'' (Fig. 8) represent the two ends of a stretch of the wall, while ''b'' also represents the corner. The robot is unable to get attracted to the target due to the presence of a stretch of the wall ''ab.'' Along the side ''ab'' of the wall the obstacle avoidance behavior offsets the target attracting behavior. This prevents the robot from banging into the wall but follow it. Until the robot arrives at the corner b the sensors on the left side keep detecting the wall while the center sensor and those on the right do not. The presence of the wall on the left forces the obstacle repulsing module to turn the robot to its right. But the presence of the target on the robot's left makes the target attracting module turn the robot to its left. Thus the action spaces of the target attracting and obstacle repulsing module offset each other and the robot moves straight, following the wall. When it reaches the corner b all the sensors begin to detect the obstacles. But because of the corner on the left, the left sensor readings become slightly larger than those on the right; i.e., at the bend, the corner on the left is slightly further than the wall (bc) which is in front and extending to the right of the robot. These larger readings can also arise due to multiple reflections in real world implementations. When this occurs both the obstacle avoidance and target reaching behaviors force the robot to turn to its left only to meet the wall again. This results in a navigation path as shown in Figure 8. Thus *the robot seems to have forgotten* that it had been seeing the wall all this while on its left, being captivated by a slight increase in the sensor reading on its left side and turning to its left. This scenario is avoided by the IM provided by the SOM. The SOM automatically identifies the meeting of the two walls at right angles on the robot's left as the temporal sequence $[7\ 1\ 0]^T$. This sequence essentially can be interpreted as ''*wall on the left of the robot that bends at near about right angles to its right*.'' The classification of the seven dimensional sensory vector $u_s(t)$ into a single group or class filters the minor variations in the sensor readings. Near the corner the sensory vectors are classified as 1 or 0. This feature of the classifier network is particularly helpful as it gives a generalized classification of the environment the robot has recently witnessed. When the lattice neuron with

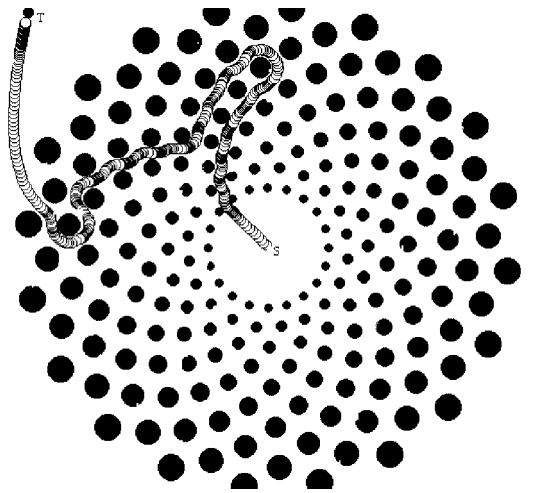

Figure 7. Navigation in a densely cluttered environment.

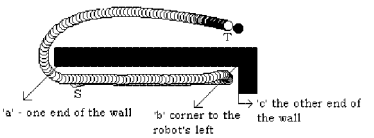

Figure 8. Path traversed by the robot without incorporating IMAS. 'S' and 'T' represent start and target positions.

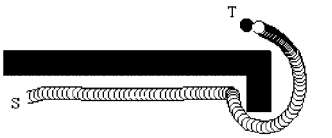

Figure 9. Robot traversal of landmark in Fig. 8 after incorporating IMAS.

the weight vector $[7\ 1\ 0]^T$ is triggered the immediate memory action space (IMAS) of the neuron that won fires the rules which turns the robot sharply to its right (in Fig. 9) and results in a shorter path.

4.2. The Infinite Loop Problem

When the obstacles are long with many bends and kinks the target attracting and goal repulsing behavior conflict and the robot gets itself into an infinite loop [Fig. 10(a)]. Along the side ab of the wall the obstacle avoidance behavior offsets the target attracting behavior. When at b the IMAS of the SOM turns the robot right and the robot continues to see the wall on its left after its right turn at b. Henceforth both the target attracting and obstacle repulsing behaviors make turn the robot to its right. After this the robot sees no obstacle and only the target attracting module influences the overall behavior of the robot. Thus the robot begins to rush toward the target and encounters the same wall again. This continues ad infinitum and is termed the local minima problem. A possible means by which the robot can come out of this loop is to recognize its repeated traversal in the same environment and execute a sequence of steps that pulls it out of the trap, discussed in the following section.

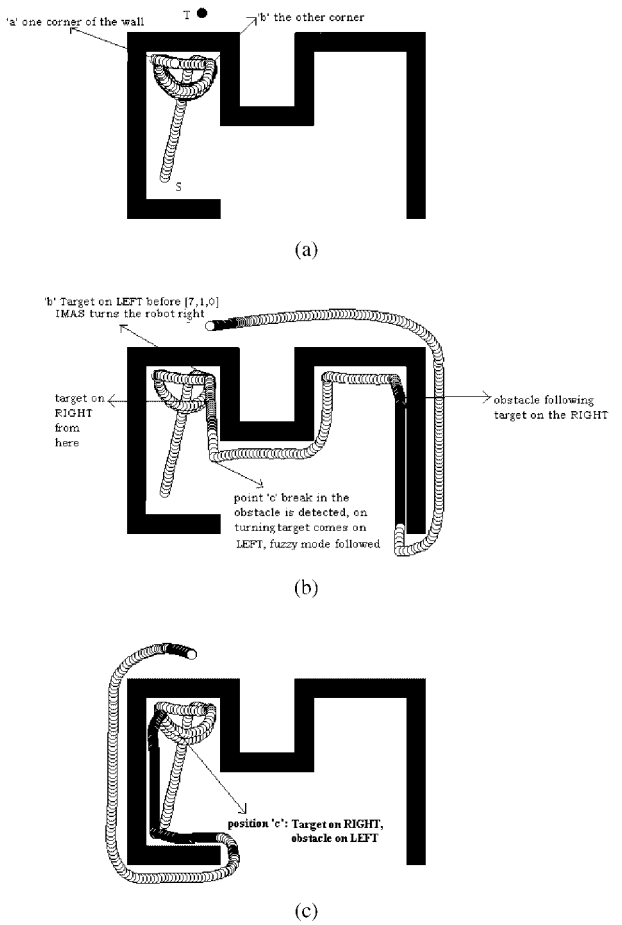

Figure 10. (a) The robot getting trapped between the two corners 'a' and 'b' though IMAS acts making the robot turn appropriately at 'b'. (b) Robot guided out by DMASc. (c) Fuzzy ART detects local minima ([7, 8, 5]) earlier than SOM, invokes DMAS. A shorter path results.

4.3. The Distant Memory (DM)

The distant memory feature is common to both the ART and SOM networks. It is called so because of the ability of the robot to correlate an immediate environment that it sees to a similar environment experienced earlier in its traversal. To become aware that it is passing through the same environment again the robot keeps a record of its spatial position every time the neuron coding a landmark wins. For every neuron in the output lattice there is a queue which is dynamically allocated whenever a neuron wins. An element of the queue is the instantaneous spatial location of the robot when the neuron wins. The queue stores a maximum of six such spatial locations of the robot, a spatial location characterized by the Cartesian coordinates of the center of the robot. If a neuron wins more than six times the least recent location in the queue is dropped from the end of the queue and the most recent spatial location is stored at the beginning of the queue. Whenever a neuron wins more than once the robot understands it is seeing another landmark of the same category, like the robot may be seeing another ''half open door'' in its traversal. But to make sure that it is seeing the *same half open door* and *not an another one* it must compare its previous spatial locations in the queue with its present one. If any one of the previous locations stored in the queue matches the current one the robot understands its encounter of the same landmark. The distant memory action space (DMAS) of the winning neuron fires a sequence of steps that guides the robot out of the trap. The sequence of steps can be understood as follows through Figure 10(b).

At the instance DMAS is activated a record is made on the direction of the target with respect to the robot's left or right flanks. Similarly the sensor which obtains the nearest range reading determines the direction of the closest obstacle. In Figure 10(b) both the obstacle and the target are on the robot's

left at point b, the instance of DMAS activation. There arise four cases and accordingly the robot executes a sequence of steps.

Case (i)

Target and closest obstacle are on the left.

The following steps are executed until the robot reaches the target.

1. Fuzzy rule base guides the robot until the target is on the left.
2. The robot switches to obstacle following when the target comes on the right while the obstacle continues to appear on the left.
3. During the course of obstacle following if a break in the obstacle is detected either because the obstacle ends or has turned in a direction away from the robot [location c in Fig. 10(b)], the robot turns around the break to continue following the obstacle.
4. If during such a break while turning around the obstacle the target appears on the left the fuzzy rule base is activated again or else obstacle following is continued through step **2**. If the robot comes again under the control of fuzzy rules and both the target and obstacle are on the right steps **1**–**4** are executed according to Case (ii) below; i.e., the occurrences of left in the steps **1**–**4** are replaced with right and vice versa. This occurs at position c in Figures 11(d) and 12(b).

Case (ii)

Target and obstacle are on the right.

The algorithm executes steps **1**–**4** of the previous case with the occurrences of ''left'' replaced with ''right'' and vice versa.

Case (iii)

Target is on the left and obstacle is on the right.

The robot turns in such a manner that the target comes on the right and obstacle on the left. Steps **1**–**4** of the first case are repeated until the target is reached.

Case (iv)

Target is on the right and obstacle is on the left.

The robot turns in such a manner that the target comes on its left and obstacle on its right. Steps **1**–**4** of the first case are executed with a swap in ''left'' and ''right'' occurrences until robot reaches the target. This occurs, for example, at position c in Figure 10(c).

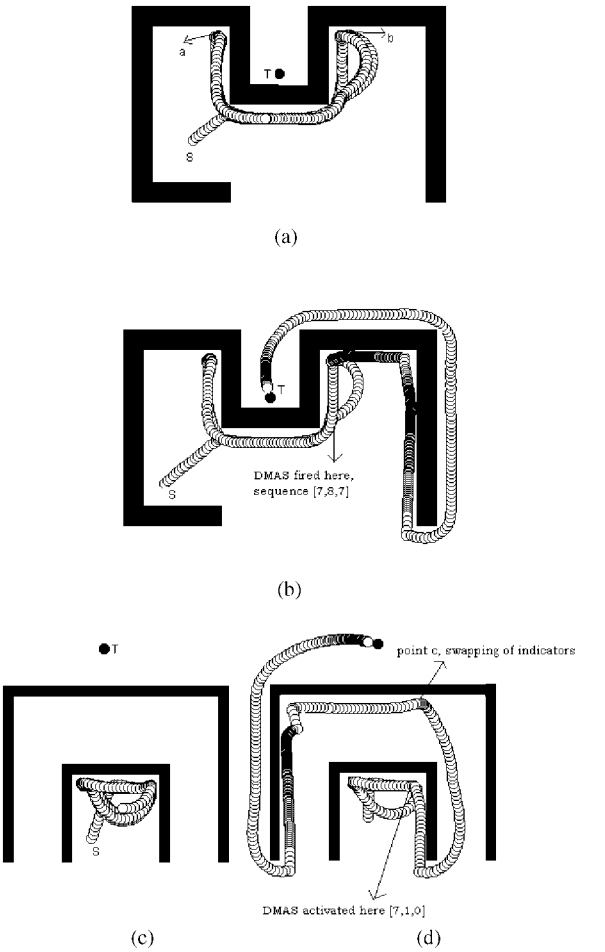

Figure 11. (a) Another infinite loop. Robot oscillating between 'a' and 'b'. (b) Robot guided out of oscillations. (c) Double walled obstacle. (d) Guiding out of the double layered wall.

The above sequence of steps is competent to carry the robot out of complicated meshes and loops, as Figures 10–12 illustrate. The sequences of steps are the same irrespective of whether the DMAS is invoked by the SOM or by the ART. In all these figures the infinite loop is shown in part (a) where only the fuzzy algorithm is implemented, and the guidance out of the loop when the local minima is detected by the SOM is shown in (b). The temporal sequence of classes that leads to the identification of the landmark is also shown in the figures. Figure 13(a) shows a simulation where the robot is unable to come through an opening in the maze at location

c due to conflicting behaviors of the two modules. Figure 13(b) illustrates circumvention of the trapped situation without the DMAS invoked. The IM capacity of the SOM understands through the weight vectors the unduly long traversal through the maze and execution of IMAS pulls the robot through the hole in the maze [Fig. 13(b)]. While DMAS is exe-

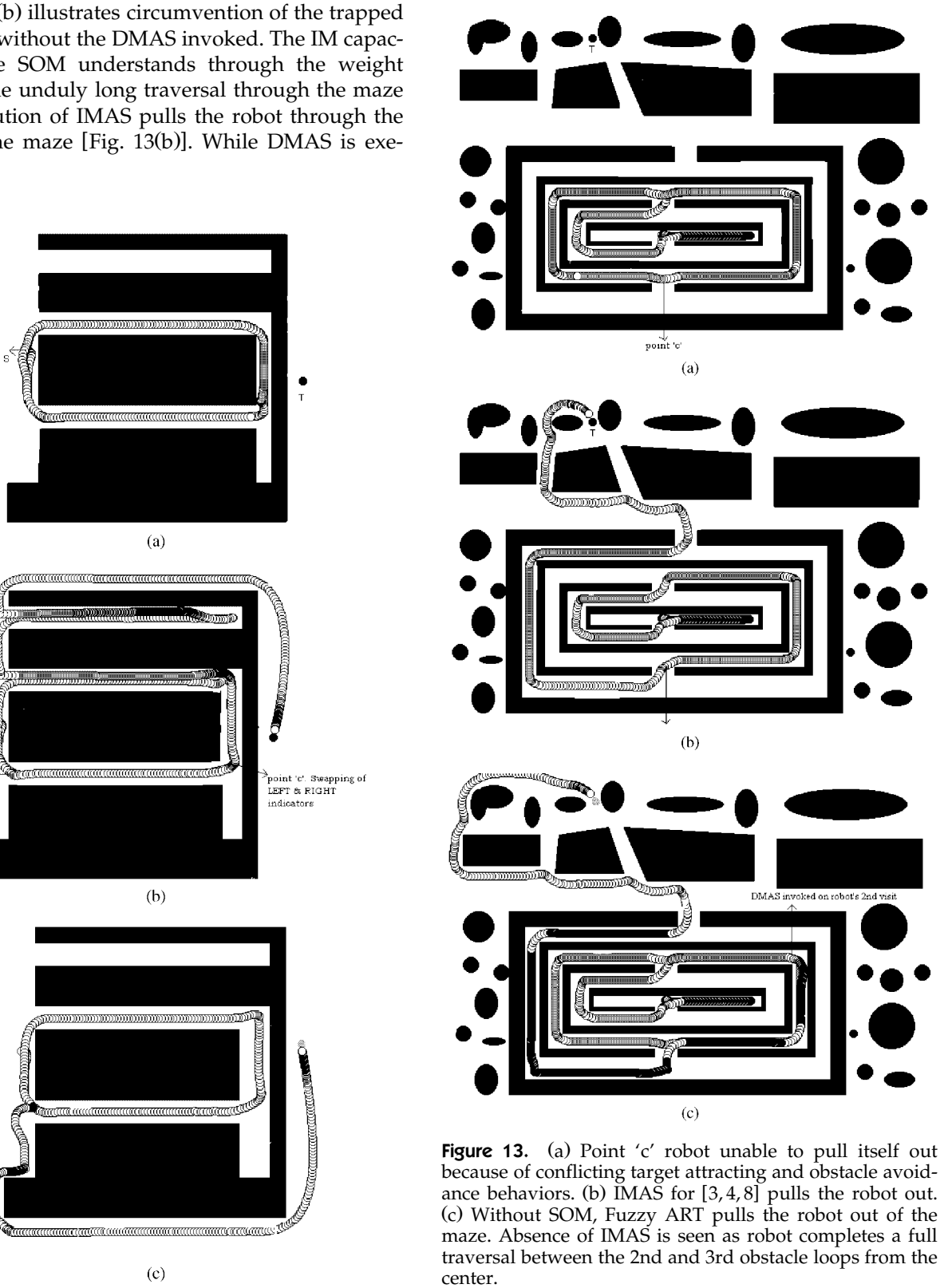

Figure 13. (a) Point 'c' robot unable to pull itself out because of conflicting target attracting and obstacle avoidance behaviors. (b) IMAS for [3, 4, 8] pulls the robot out. (c) Without SOM, Fuzzy ART pulls the robot out of the maze. Absence of IMAS is seen as robot completes a full traversal between the 2nd and 3rd obstacle loops from the center.

Figure 12.

cuted the obstacle following behavior is also modeled through a separate set of fuzzy rules which has not been discussed since it is not the theme of the paper.

4.4. The Role of Fuzzy ART

Figure 14(a) shows a simulation environment where the SOM is unable to detect the local minima situation. This is because the SOM is unable to map the real-time patterns to a particular landmark learned during offline. To overcome this the fuzzy ART network has been incorporated to classify new patterns online. Repeated occurrences of such a pattern trigger the DMAS when the spatial locations of the robot match. Here the ART network invokes the DMAS in Figure 14(b) and pulls the robot out of the local minima. Another advantage of the ART is shown in Figures 10(c) and 12(c). Here the ART is able to detect a repeated occurrence of a pattern earlier than SOM. This results in a reduced path. The SOM has to wait for a pattern that matches a landmark stored by its lattice neurons through the weight vectors. This can result in a longer traversal. This, however, cannot be generalized as it depends on the kind of environment present once the robot gets out of the local minima through the two routes. But it can be intuitively seen that online learning by fuzzy ART helps in detection of the local minima situation earlier than the SOM, which can probably lead to shorter traversals. Figure 13(c) shows an example when only the ART network is present in the second layer. Hence the IMAS advantage of SOM is absent. Still the ART network alone is sufficient to detect patterns online and pull the robot out of the maze. The absence of IMAS results in the robot making a complete traversal of the free space between the second and third obstacle loops from the center of the mazelike structure with four obstacle loops.

To summarize the ART network in the second layer alone is capable of detecting the local minima and invokes the DMAS. The SOM provides for modeling and understanding the local environment based on landmarks learned offline. When the SOM is also present in the second layer of the classifier network the ART plays the role of a backup, detecting the local minima when the SOM is unable to do. It also helps in detecting the local minima earlier than the SOM.

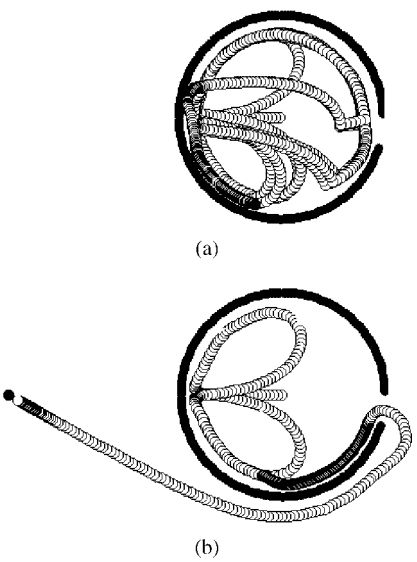

Figure 14. (a) An environment where the SOM is unable to detect the local minima. (b) Fuzzy ART pulls the robot out of the local minima.

4.5. Comparing with an Earlier Approach

In a previous approach[20] the determination of the local minima situation is done in an empiric way by comparing the difference in orientation of the robot between successive instants. If this orientation difference has a value greater than 160 degrees the robot is considered trapped or boxed. This can result in a situation where the robot begins to track the obstacle contour though it is not trapped. We take an example from the same paper[20] for comparison. Figure 15(a) portrays the path obtained by the authors in their paper while Figure 15(b) is the path obtained by the current method which is shorter as the robot does not experience a similar scenario twice during its traversal. Another issue, which merits consideration, is the instant when the robot switches from obstacle following mode to fuzzy rule base mode or from the track mode to Heuristic mode in the terminology of the previous paper. The previous approach says the robot switches from the T mode to H mode when a, b, c are collinear and b is between a and c. Here 'a' is the point where robot switches from H mode to T mode, 'b' is the point where the robot finishes tracking the obstacle, and 'c' is the target location. In the present approach the robot finishes tracking the obstacle according to step **4** discussed in section 4.3. According to the present approach the robot switches from obstacle following mode to fuzzy rule base mode when the sensors detect the end of the obstacle, and during the course of turning around the obstacle's end, the target

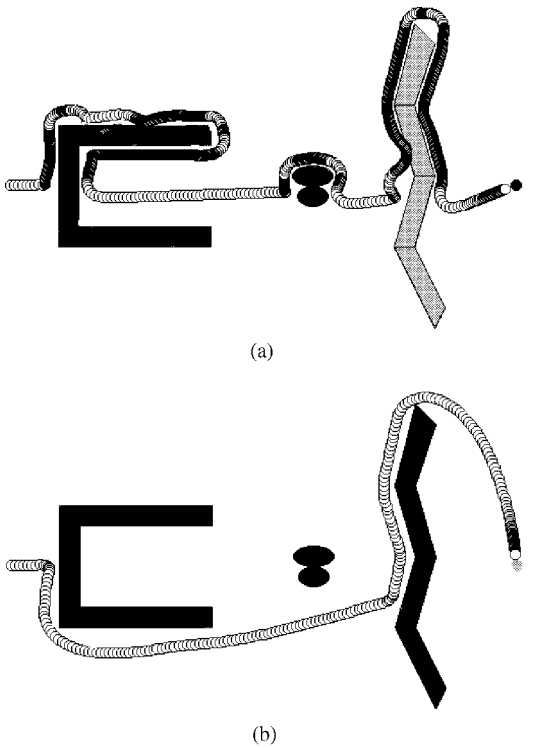

Figure 15. (a) An example from a previous paper. Heuristic determination results in misidentification of a situation to a local minima. (b) The robot traverses a shorter path by the current method.

position relative to the current direction of motion gets swapped. A switch also occurs if the target position relative the robot gets swapped during obstacle following mode and is maintained until the robot reaches the end of the obstacle. It has been observed that if there exists no further obstacle(s) beyond the concave obstacle into which the robot gets trapped and the path tracked by the robot from the instant of obstacle following is considered in either methods, the present method gives a shorter path. This is verified from Figures 16(a) and 16(b) which is once again an example from the previous paper. In Figure 16(b) the robot turns around the end of an obstacle between the pair of points a, b and c, d. At b after turning around the end there is no change in target position and hence the obstacle following mode is continued. At d a change in position occurs and the fuzzy rule base is executed but is unable to pull the robot to the target (due to the wall on the left). A change in target position once again occurs at e while the robot follows the wall on its left and is maintained until f. At f the robot reaches the end of the obstacle and fuzzy rules are able to pull the robot toward the target, resulting in a highly optimal path compared to Figure 16(a). Also Figures 16(c) and (d) illustrate the same point when the robot begins tracking in the counterclockwise direction. In Figure 16(d) the earlier algorithm has been modified with the criteria for breaking from the T mode set according to the present algorithm. Thus it is observed that the present criteria gives shorter traversals as the robot stops tracking the obstacle much earlier than the previous approach, and in the absence of further obstacles the fuzzy target reaching module steers the robot to its destination.

4.6. Limitations and Practicality Issues

During navigation the robot can witness similar scenes along different paths. To avoid getting confused between two similar scenes and a same scene seen twice the robot compares its spatial location at the instant of registering a scene with its spatial position when it saw a similar scene earlier. If the spatial positions coincide the robot understands its encounter of the same landmark again. In real-time implementation errors creep up due to dead reckoning and other sources and the estimates of the robot's position in a global frame can vary from its actual position. This can limit the performance of the algorithm especially if the robot encounters similar scenes at rapid intervals such that the robots global position does not vary much between these intervals.

The algorithm, however, can tackle a problem commonly encountered during implementations, that of errors in range readings due to multiple reflections at the corners. This has been discussed in section 4.1.

In our experience with real world sensory data, a system of seven sensors has been found to be adequate for identification of typical landmarks. The algorithm does not demand storage of a long sequence of sensory data as only changes in sequence are to be processed and considerable memory is saved due to the dimensionality reduction by the spatial classifier. The net space required is a lattice of 64 neurons that identify the landmarks along with 49 neurons of lower bounds, each neuron characterized by a three dimensional vector of integers. Spatial positions of the robot are stored only at instances of registering a landmark and not more

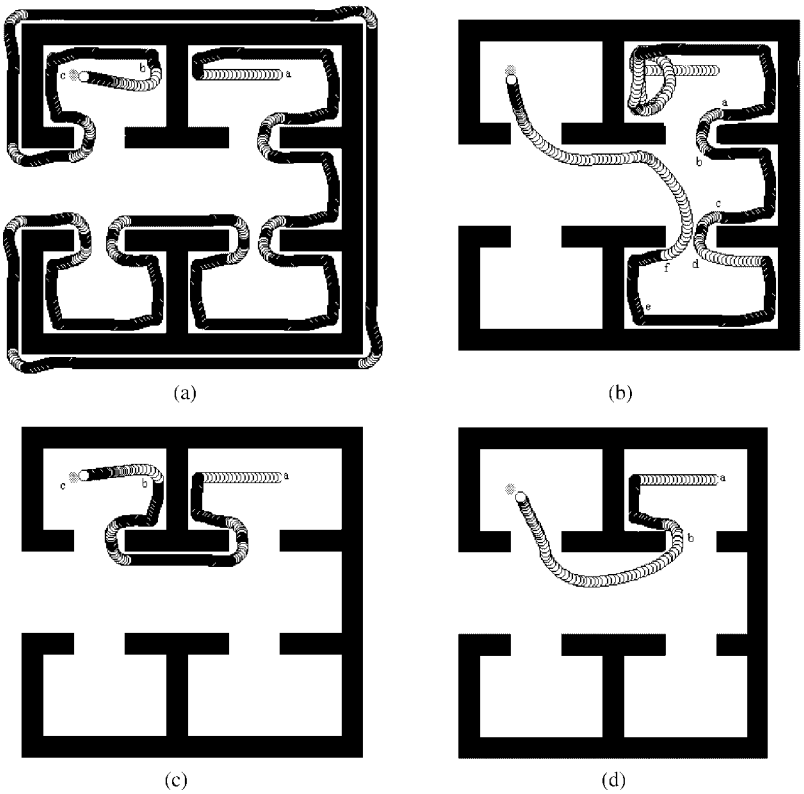

Figure 16. (a) According to the earlier method the robot switches from 'T' to H mode at point 'b' when points a, b and c are almost collinear. (b) A more reduced path results by the current method. (c) The robot tracks the obstacle in counter clockwise direction. (d) Path obtained when the earlier algorithm modified with the new criteria for switching from T mode to H mode.

than 20 spatial positions need to be stored on an average at any instant during navigation.

5. CONCLUSION

Real-time navigation involves decision making according to the perception of the local environment. The fuzzy inferencing method has been shown to be successful in real-time navigation with cluttered environments. But when the environment is filled with obstacles in the form of loops, mazes, and other complicated structures the robot tends to lose track of direction and gets trapped.

The paper proposes a new approach for identifying the robot's trapped state that seems more consistent with how a human would normally understand his trapped condition in an environment by recalling the landmarks he had seen earlier in his traversal. A spatio-temporal classifier network is employed for learning and classifying temporal sequences of spatial sensory data that enables the robot to comprehend its immediate environment in terms of landmarks and remember previous experiences of a similar environment. In this way the algorithm differs from other methods that surmount the local minima problem by recollecting previous experiences to understand its trapped condition. The algorithm has been tested on cluttered, concave, and mazelike environments and its efficacy has been established.

A possible extension of this work could be to recognize dynamic objects in the vicinity of the robot based on similar classification of range patterns.

APPENDIX A: SOM NETWORKS AND KOHONEN LEARNING

Kohonen developed the SOM to transform an input signal of arbitrary dimension into a lower (one or two) dimensional discrete representation preserving topological neighborhoods. Let $\Phi: U \rightarrow A$ denote the

SOM mapping from an input space U and the discrete output space A. The SOM defines in Kohonen's words ''an elastic net of points A that are fitted to the input signal space U to approximate its density function in an ordered way.'' In order to achieve this goal the discrete grid A of neurons indexed by $i \in A$ is described by reference vectors w_i which take their values in the input space, U. The response of a SOM to an input $u \in U$ is determined by the reference vector w_{iw} of the two dimensional lattice which produces the best match to the input.

$$iw = \arg\min_i \text{dist}(w_i - u), \qquad i = 1, \ldots, N,$$

where iw refers to the winning neuron of the two dimensional lattice and dist$(\cdot)$ is the Euclidean metric. The training of the SOM can be accomplished generally with a competitive learning rule as

$$w_i(n+1) = w_i(n) + \eta h_\sigma(i, iw)(n)(u(n) - w_i(n))$$

where $h_\sigma(i, j)$ is an unimodal function that decreases monotonically for increasing $\|i - j\|$ with a characteristic decay constant σ. The usual choice for such a function is a Gaussian given by $h_\sigma(i, j) = e^{-\|(r-s)\|/2\sigma^2}$, where r and s represents respective lattice position vectors corresponding to ith and jth neural units, respectively. h_σ is also known as the neighborhood function that adjusts appreciably the neurons close to the winner and those far away with little change. Both the learning parameter η and the neighborhood parameter σ are annealed with time.

APPENDIX B: FUZZY ART ALGORITHM

The fuzzy ART algorithm is capable of unsupervised classification of both binary and analog inputs in real-time. The main advantage of this network is that it is very stable to previously stored inputs and plastic to new inputs. This implies that new classes can be allocated dynamically and the number of classes that can be formed is only limited by the total memory available.

The fuzzy ART network consists of two processing layers. The first layer corresponds to the input layer while the second layer consists of neurons where each neuron represents a self-organized category of the input. The two layers are connected by adaptive weights Z. Initially the components of the weight vector Z are set at 1.0. Then the input vector and its complement y^c are stored as $Y = (y, y^c)$ in the first layer where $y^c = 1 - y$. This process of storing both the input and its complement is called complement coding and helps in preventing a category proliferation problem.[25]

The input activates the node in the second or category layer as

$$T_j = \frac{|Y \wedge Z_j|}{\delta + |Z_j|} = \frac{\sum_i \min(Y_i, Z_{ji})}{\delta + \sum_i z_{ji}}.$$

Here the conjunction operator represents the fuzzy AND operator and the norm used is an L1 norm. The network then makes a hypothesis by selecting the node J that has the maximum T_j to be the category that stores the presented input. In case of a tie, the node with the least index is chosen.

Then this hypothesis is tested using a similarity test called the vigilance criterion as $|Y \wedge Z_j|/|Y_j| \geq \rho$, where ρ is called the vigilance parameter. The ratio on the left of the above similarity test represents the degree of match between Y and the category J in the second layer. If the node J satisfies the vigilance criterion then the weights are updated as $Z_j = \beta|Z_j \wedge Y| + (1 - \beta)Z_j$, where $\beta(< 1.0)$ is called the forgetting factor. In our simulations we have fixed β to be 0.85.

If node J does not satisfy the vigilance criterion, it is shut down. Then, a search process is initiated in the category layer to see if any other node satisfies the vigilance criterion. If there exists one such node, then its weights are updated as above. If there exists no such node then a new node is recruited to store the spatio-temporal input. Thus the vector Y is classified into a category node in the second layer.

REFERENCES

1. L.L. Wang and W.H. Tsai, Car safety driving aided by 3-D image analysis techniques, Proc. Micro Elect. and Info. Sci. and Tech. workshop, Hsinchu, Taiwan, R.O.C., 1986, pp. 687–701.
2. J. Borenstein and Y. Koren, Real-time obstacle avoidance for fast mobile robots, IEEE Trans SMC 19 (1989), 1179–1186.
3. H.G Tilleme (Ed.), An overview of the mobile autonomous robot twente project MART 93, Univ. Twente, WA-315, 1993, pp. 315–319.
4. T. Lozano-Perez and M.A. Wesley, An algorithm for planning collision-free paths among the polyhedral obstacles, Commn ACM 22 (1979), 560–570.

5. O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int J Robot Res 5 (1986), 90–98.
6. O. Takahashi and R.J. Schilling, Motion planning in a plane using generalized Voronoi diagrams, IEEE Trans Robot Automat 5 (1989), 143–150.
7. S. Hutchinson and M. Barbehenn, Efficient search and hierarchical motion planning by dynamically maintained single source shortest path trees, IEEE Trans Robot Automat 11 (1995), 198–214.
8. J. Storer and J. Reif, Shortest paths in a plane with polygonal obstacles, Dept of Comp Sci, Brandies Univ. Tech Rep, 1993.
9. J.A. Janet, R.C. Luo, and M.G. Gray, Autonomous global motion planning using traversability vectors, IEEE Trans Robot Automat 13 (1997), 132–140.
10. K.M. Krishna, K.D. Rajasekhar, and L. Behera, On fast computation of optimal paths from the visibility graph for the minimal workspace, Int Symp on Intelligent Rob Sys, CAIR, Bangalore, 1998.
11. G. Bauzil, M. Briot, and P. Ribes, A navigation subsystem using ultrasonic sensors for the mobile robot Hilare, 1st Int Conf on Robot Vision and Sensory Controls, Stratford-upon-Avon, UK, 1981, pp. 47–58, 681–698.
12. C.R. Weibsin, G. de Saussure, and D. Kammeer, Self-Controller: A real-time expert system for an autonomous mobile robot, Comp Mech Eng (1986), 12–19.
13. J.L. Crowley, Dynamic world modeling for an intelligent mobile robot, Proc IEEE Seventh Int Conf Pattern Recognition, Montreal, Canada, 1984, pp. 207–210.
14. J. Borenstein and Y. Koren, The Vector field histogram —Fast obstacle avoidance for mobile robots, IEEE Trans Robot Automat 7 (1991).
15. S. Ishikawa, A method of indoor mobile robot navigation by fuzzy control, Proc Int Cong Intell Robot and Sys, Osaka, Japan, 1991, pp. 1013–1018.
16. P.S. Lee and L.L. Wang, Collision avoidance by fuzzy logic for AGV navigation, J Robot Syst 11 (1994), 743–760.
17. H.R. Beom and H.S. Cho, A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning, IEEE Trans SMC 25 (1995), 464–477.
18. V.J. Lumelsky and A.A. Stepanov, Path planning strategies for a point mobile automaton moving amidst obstacles of arbitrary shape, Algorithmica 2 (1987), 403–430.
19. V.J. Lumelsky, A comparative study on the path performance of maze-searching and robot motion planning algorithms, IEEE Trans Robot Automat 7 (1991), 57–66.
20. H.P. Huang and P.C. Lee, A real-time algorithm for obstacle avoidance of autonomous mobile robots, Robotica 10 (1992), 217–227.
21. I. Kamon and E. Rivlin, Sensory-based motion planning with global proofs, IEEE Trans Robot Automat 13 (1997), 814–822.
22. F.G. Pin and S.R. Bender, Adding memory processing behaviors to the fuzzy behaviorist approach (FBA): Resolving limit cycle problems in autonomous robot navigation, Intell Automat Soft Comput 5 (1999), 31–41.
23. W.L. Xu, A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behavior-based mobile robot, Robot Autonomous Syst 30 (2000), 315–324.
24. T. Kohonen, Self-organizing map, Proc IEEE 78 (1990), 1460–1480.
25. G.A. Carpenter, S. Grossberg, and D.B. Rosen, Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, Neural Networks 4 (1991), 759–771.
26. T. Kohonen, The neural phonetic typewriter, Computer 21 (1988), 11–22.
27. N. Nasrabadi and Y Feng, Vector quantization of images based on Kohonen self organized feature maps, Proc IEEE Int Conf on Neural Networks, ICNN'88, San Diego, CA, 1988, pp. 101–108.
28. A. Dubrawski and J.L. Crowley, Learning locomotive reflexes: A self supervised neural system for a mobile robot, Robot Autonomous Syst 12 (1994), 133–142.
29. I.J. Nagrath, L. Behera, K.M. Krishna, and K.D. Rajasekhar, Real time navigation of a mobile robot using Kohonen's topology conserving neural networks, Proc Int Conf on Advanced Robotics, 1997, Monterey, CA, pp. 359–364.
30. M.H. Hasoun, ''Fundamentals of artificial neural networks,'' MIT Press, Cambridge, MA, 1995.