ELSEVIER

Fuzzy Sets and Systems 110 (2000) 331–340

FUZZY sets and systems

www.elsevier.com/locate/fss

# Heuristic fuzzy-neuro network and its application to reactive navigation of a mobile robot

Kai-Tai Song*,[1], Liang-Hwang Sheen[2]

*Department of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC*

Received July 1996; received in revised form November 1997

**Abstract**

A novel pattern recognition approach to reactive navigation of a mobile robot is presented in this paper. A heuristic fuzzy-neuro network is developed for pattern-mapping between quantized ultrasonic sensory data and velocity commands to the robot. The design goal was to enable an autonomous mobile robot to navigate safely and efficiently to a target position in a previously unknown environment. Useful heuristic rules were combined with the fuzzy Kohonen clustering network (FKCN) to build the desired mapping between perception and motion. This method provides much faster response to unexpected events and is less sensitive to sensor misreading than conventional approaches. It allows continuous, fast motion of the mobile robot without any need to stop for obstacles. The effectiveness of the proposed method is demonstrated in a series of practical tests on our experimental mobile robot. 

*Keywords:* Pattern recognition; Fuzzy navigation; Fuzzy-neuro network; Mobile robots

## 1. Introduction

Reactive obstacle avoidance is one of the most desirable characteristics of an autonomous mobile robot. It is the ability to free-range in an unknown environment relying only on sensory information. Fig. 1 shows a block diagram of such a motion planning and control system. In the robot navigation system, a local motion-planning module is responsible for generating steering commands in response to onboard sensory data. It is important for the robot to respond promptly to its surroundings, for instance, to avoid unexpected obstacles and continue traveling toward the target. However, available sensors are not good enough to provide accurate recognition of the environment. Very often, the measured data contain uncertainties that cause motion errors. It is therefore difficult for the mobile robot to navigate in an unknown and dynamically changing environment.

One reactive navigation approach employs the potential field or vector force field concepts [1, 4], in which a two-dimensional Cartesian grid is utilized for obstacle representation. The target exerts a virtual attractive force on the mobile robot, and the obstacles exert repulsive force. The robot-motion reaction is determined by the resultant virtual force. The shortcoming of these methods is that they require a lot of calculation. Recently, considerable work has been

* Corresponding author. Fax: 886-3-5715998.

*E-mail address*: ktsong@cc.nctu.edu.tw (K.-T. Song)

[1] This work was supported by National Science Council of the ROC under grants NSC-84-2212-E009-029.

[2] Liang-Hwang Sheen is now with Phoenixtec Power Co., Ltd., Taipei, Taiwan, ROC

0165-0114/00/$ - see front matter 
PII: S0165-0114(97)00401-6

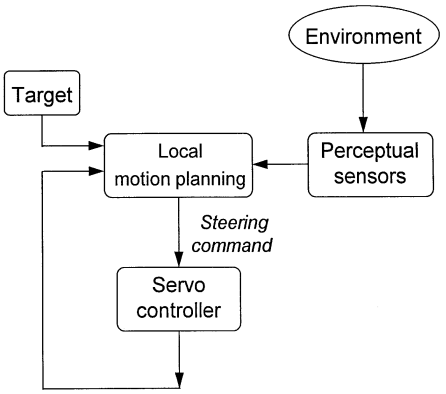


Fig. 1. System block diagram of sensor-based navigation.

reported concerning the application of artificial neural networks (ANN) and fuzzy logic for reactive control. An incremental supervised learning scheme has been proposed for reactive navigation of a mobile robot [12]. In [10], a reinforcement learning scheme was employed to train a neural network for obstacle avoidance. The advantage of this approach is the learning capacity of the neural network, however, learning convergence is very slow and generalization is not always satisfactory. On the other hand, fuzzy logic concepts was employed to handle the uncertainty problems in environmental map-building [11]. The constructed fuzzy maps from ultrasonic sensors were then used to plan a collision-free path. Several methods exploiting fuzzy control schemes have been proposed for avoiding unexpected obstacles [6, 8, 9, 14]. A rule table was established in these methods according to heuristic experience. In [8], 155 rules were employed for avoiding static and moving obstacles along a preplanned path. In [14], a fuzzy navigation controller was combined with virtual concepts for a mobile robot to navigate in an unknown environment. There were 81 rules for each right-fuzzy-logic controller and left-fuzzy-logic controller. It is noticed that there are a great many rules and some of them might not be activated during navigation. The redundant rules will increase the complexity of fuzzy inference.

In this paper, we present a novel design approach to building a rule table for reactive navigation exploiting fuzzy-neuro control. Satisfactory navigation performance can be achieved using reduced numbers of rules. The basic idea is to let the IF-part of a rule be the obstacle-configuration class and the THEN-part be the reference velocity values. A resultant velocity command to each wheel motion controller is generated through fuzzy Kohonen clustering network (FKCN) instead of by conventional fuzzy inference. FKCN is a fuzzy neural network normally used for pattern clustering. In this study this pattern-recognition structure is extended to local motion planning and control. The developed method is fast, efficient and free of the problems mentioned above. Moreover, in order to make the system robust and flexible, we adopted a behavior-based architecture for the mobile robot [5]. Consequently, the mobile robot has several ways of producing steering commands using different behavior modules. The rest of this paper is organized as follows: Section 2 describes the development of the reactive navigation algorithm based on the FKCN structure. In Section 3, we introduce a design for obstacle avoidance of an experimental mobile robot. Relevant simulation results are presented in Section 4. Section 5 illustrates practical experiments in an indoor environment. We conclude the paper in Section 6.

## 2. Prototype pattern assignment

To achieve real-time reactive navigation, a good strategy would be to construct a perfect mapping between input sensor data and appropriate control actions. The relation, however, is very complicated and highly nonlinear. In the first place, different types of sensors have different measurement characteristics. It would be difficult to estimate the spatial parameters using onboard sensors in order to determine the configuration relationships between the mobile robot and its immediate surroundings. On the other hand, it is well recognized that artificial neural networks have impressive capacity for nonlinear mapping and pattern-recognition applications [2]. In this paper useful heuristics are combined into a fuzzy neural network to achieve the desired pattern-recognition results. The structure of the proposed reactive navigation system is illustrated in Fig. 2. It consists of two major parts: the lower is a fuzzy neural network and the upper is responsible for velocity calculation. The FKCN structure [7] was adopted for the desired pattern-recognition function. FKCN is a three-layered, pattern-clustering network. Once it is trained, there is a prototype pattern associated with each cluster.

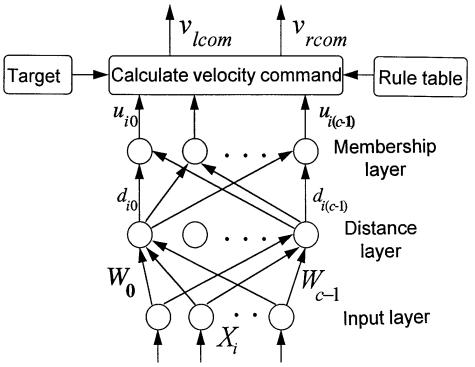


Fig. 2. The proposed heuristic FKCN for reactive navigation.

Consequently, every prototype pattern characterizes a cluster. In the neural network, all these prototype patterns are set to the weights in the distance layer. However, we do not apply the unsupervised learning algorithm of the original FKCN. The weights in the distance layer are assigned instead of being trained. There are two reasons for doing it this way. One is for simplicity and consequently a reduction in computation time. The other reason is that the clusters are known in advance in our application. The prototype patterns learned by the unsupervised learning algorithm might not be better than the assigned patterns derived from actual experimental data and human experience.

In the following, we describe the method for determining the distance and similarity between an input pattern and the prototype patterns. As shown in Fig. 2, the distance layer is responsible for comparing an input pattern with the prototype patterns. Output $d_{ij}$ of node $j$ in the distance layer equals 0 when the input pattern $X_i$ perfectly matches the prototype pattern $W_j$. The output of the distance layer is computed as follows:

$$d_{ij} = \|X_i - W_j\|^2 = (X_i - W_j)^{\mathrm{T}}(X_i - W_j), \tag{1}$$

where $W_j$ is the $j$th prototype pattern.

Formula (1) is a 2-norm equation. The larger the difference between $X_i$ and $W_j$ is, the faster $d_{ij}$ will increase by powers of 2. The membership layer is provided to map the distance $d_{ij}$ to membership values $u_{ij}$. If an input pattern does not match any prototype pattern, then the similarity between the input pattern and each individual prototype pattern is represented by a membership value from 0 to 1. The determination of the membership value is given in [3] and can be summarized by the following equations:

$$u_{ij} = \begin{cases} 1 & \text{if } d_{ij} = 0, \\ 0 & \text{if } d_{ik} = 0,\ (k \neq j,\ k \geqslant 0,\ j \leqslant c-1), \end{cases} \tag{2}$$

where $c$ denotes the number of prototype patterns, otherwise

$$u_{ij} = \left( \sum_{l=0}^{c-1} \left( \frac{d_{ij}}{d_{il}} \right) \right)^{-1}. \tag{3}$$

The larger the $u_{ij}$ is the more input pattern $X_i$ is similar to some prototype pattern $W_j$. Since each prototype pattern is associated with a rule, the membership value represents the degree of activation of a rule. The sum of the outputs of the membership layer equals 1.

In this study, ultrasonic range sensors are employed for obstacle detection. This type of sensor is simple and efficient for measuring distances to obstacles. Sixteen ultrasonic transducers were fixed in a ring on our experimental mobile robot. A detailed discussion on the system will be found later. In practice, we can get sonar vectors from these sensor readings corresponding to each different obstacle-configuration class. For example, the vector $W_j = \{w_1, w_2, \ldots, w_p\}$ is the sonar vector for the $j$th obstacle-configuration class; where $w_i$ is the $i$th sensor reading, and $p$ transducers are used to construct the sonar map. Using the concept of pattern recognition, we view each sonar vector as a pattern. Hence, we can determine several prototype patterns corresponding to various obstacle-configuration classes. These prototype patterns are then assigned to be the weights of the neural network. After the network weights have been assigned, they can be recalled on-line when the sensory input data are provided during navigation. The recall procedure is described briefly below.

*Step* 1: A quantized sonar vector (input pattern) constructed from current ultrasonic sensor readings is presented to the neural network input.

*Step* 2: The distances between the input pattern and every prototype pattern are computed using (1).

*Step* 3: The similarities between the input pattern and every prototype pattern are calculated using (2) and (3). The similarities are represented by membership values from 0 to 1, according to their distance values.

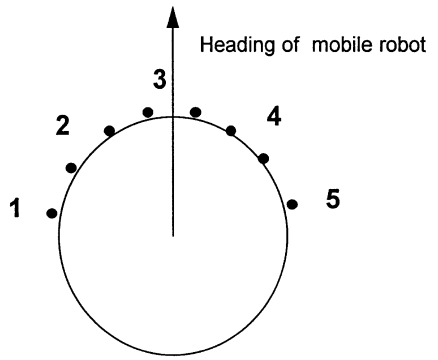


Fig. 3. Grouping of ultrasonic sensors.

Our experimental mobile robot is equipped with two independent drive wheels. Its course is determined by the relative velocities of the left and right wheels. Therefore, in this design, each prototype pattern is associated with a pair of reference wheel velocities. This association is determined in a preset manner. In other words, we have to provide a rule table for this mapping (see Fig. 2). The number of rules equals that of prototype patterns. Notably, it will be shown that satisfactory reactive navigation results can be obtained employing a considerable reduced number of rules compared with using a conventional fuzzy logic controller. In this manner, the mobile robot can perform on-line obstacle avoidance using onboard ultrasonic sensors. The complete navigation design is presented in the next section.

## 3. Design for obstacle avoidance

The proposed navigation scheme was developed for an experimental mobile robot. Sixteen ultrasonic sensors are mounted in a circle on the robot 22.5° apart, alternating in height at 30 cm or 75 cm to cover more detection space. It takes 150 ms to complete an updating cycle of all sixteen sensors [15]. Only eight sensors mounted on the front of the mobile robot were used in this study. Those on the back side were not included because backward motion commands were beyond the scope of the current experiments. These eight sensors were divided into five groups, as shown in Fig. 3. Only one sensor in group 1 and another in group 5 were used to detect obstacles at the left or right side of the mobile robot, leaving two sensors in each of the other three groups. In these three groups, the smaller of the two transducer readings was used in each sampling instant. These five sensor-group values were quantized before sending into the neural network. The quantization formula for groups 1, 2, 4 and 5 is as follows:

$$x_i = \begin{cases} 1 & \text{for } 0 < d_i \leqslant 100\,\text{cm}, \\ 2 & \text{for } 100\,\text{cm} < d_i \leqslant 150\,\text{cm}, \\ 3 & \text{for } 150\,\text{cm} < d_i \leqslant 200\,\text{cm}, \\ 4 & \text{for } d_i \geqslant 200\,\text{cm}, \end{cases} \tag{4}$$

where $d_i$ is the sensor value of the $i$th group.

The two sensors in group 3 were responsible for detecting head-on obstacles. The quantization of this group sensor data must take into account the response time for preventing from collision:

$$x_i = \begin{cases} 1 & \text{for } 0 < d_3 \leqslant 150\,\text{cm}, \\ 2 & \text{for } 150\,\text{cm} < d_3 \leqslant 200\,\text{cm}, \\ 3 & \text{for } 200\,\text{cm} < d_3 \leqslant 250\,\text{cm}, \\ 4 & \text{for } d_3 \geqslant 250\,\text{cm}, \end{cases} \tag{5}$$

where $d_3$ is the sensor value of the 3rd group.

Too many grades of quantization would have resulted in a complicated rule table, but too few grades would have led to unclear representation of the obstacle-configuration classes. Quantized sensory data are used in the FKCN and $u_{ij}$, $j = 0 \sim c - 1$ is calculated according to (1)–(3). The mobile robot is always trying to reach the assigned target. Therefore, the target direction is also taken into account during the reactive navigation (see Fig. 2). The target direction is defined relative to the heading of the mobile robot. It is divided into 5 levels as shown in Fig. 4. The details of this division are as follows:

$$t = \begin{cases} 1 & \text{for } 180^\circ < \phi \leqslant 270^\circ, \\ 2 & \text{for } 120^\circ < \phi \leqslant 180^\circ, \\ 3 & \text{for } 60^\circ < \phi \leqslant 120^\circ, \\ 4 & \text{for } 0^\circ < \phi \leqslant 60^\circ, \\ 5 & \text{otherwise}, \end{cases} \tag{6}$$

where $\phi$ is the direction of the target with respect to the current heading of the mobile robot. In (6), the range in each level affects the stability of navigation in avoiding an obstacle in a long corridor. This phenomenon has been examined in experiments.

Nine typical obstacle-configuration classes were considered in this study as depicted in Fig. 5. Through fuzzifying and combining these nine classes in the

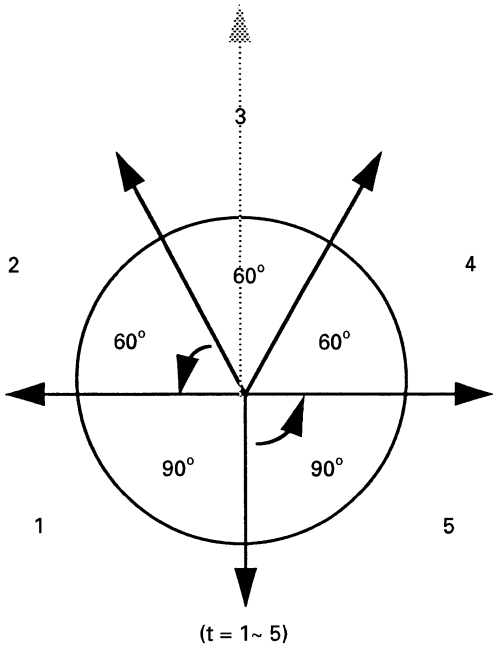


Fig. 4. Quantization of target direction.

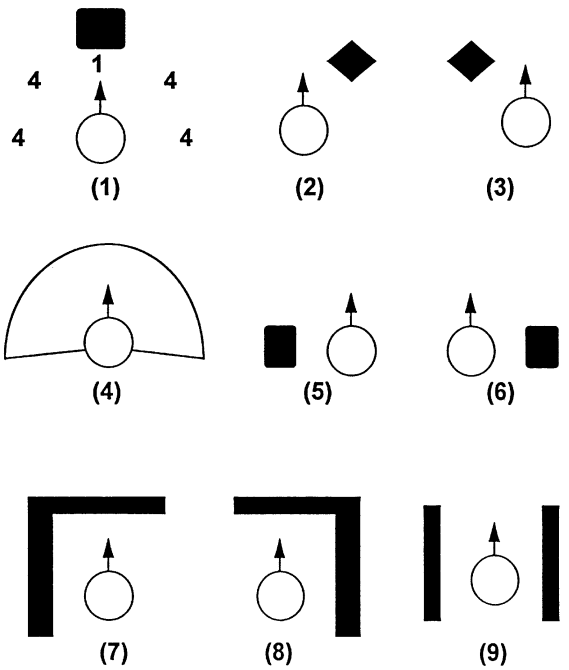


Fig. 5. Obstacle-configuration classification used in this design.

heuristic FKCN, we practically had no need to consider other obstacle configuration. This is the main purpose of the membership layer. Consequently, the number of the prototype patterns can be kept few. This resulted in considerable fewer control rules than otherwise employed in conventional fuzzy control methods. We take the first obstacle-configuration class in Fig. 5 as an example to illustrate the idea of forming control rules. In this case, there is an obstacle in front of the mobile robot and the corresponding prototype pattern is

$$W_j = \{4\ \ 4\ \ 1\ \ 4\ \ 4\}.$$

The corresponding rule can be set in the following manner:

For $t = 1$, IF $W_j = \{4\ \ 4\ \ 1\ \ 4\ \ 4\}$,

THEN $v_{lj} = 2.0\,\text{cm/s},\ v_{rj} = 10.0\,\text{cm/s}$,

or

For $t = 2, 3$, IF $W_j = \{4\ \ 4\ \ 1\ \ 4\ \ 4\}$,

THEN $v_{lj} = 3.0\,\text{cm/s},\ v_{rj} = 10\,\text{cm/s}$,

or

For $t = 4, 5$, IF $W_j = \{4\ \ 4\ \ 1\ \ 4\ \ 4\}$,

THEN $v_{lj} = 10.0\,\text{cm/s},\ v_{rj} = 3.0\,\text{cm/s}$,

where $v_{lj}$ and $v_{rj}$ are the output (reference) left and right wheel velocities, respectively. The rule table was constructed exploiting the representative prototype patterns and heading levels. We employed altogether 16 rules in the present study as shown in Table 1. Notably, the elements of prototype patterns $W_j$ are mostly of quantized value 1 or 4. This is because it would be beneficial if the obstacle-configuration classes could be represented as sharply as possible. The quantized values 2 and 3 are only used to represent special configuration classes.

The algorithm for generating velocity commands is described below. First of all, if an input pattern is identical to one of the prototype patterns ($d_{ij^*} = 0$), then the firing of the corresponding rule to this input pattern is equal to one. The generated velocity command will be equal to the reference velocities of the excited rule. However, in most situations, the calculated smallest distance $d_{ij^*}$ is only less than a pre-defined

Table 1
Implemented rule table

| Rule | IF-part prototype pattern | | | | | THEN-part reference velocity | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | $t=1$ | | $t=2$ | | $t=3$ | | $t=4$ | | $t=5$ | |
| No. | | | | | | vl | vr | vl | vr | vl | vr | vl | vr | vl | vr |
| 1 | 4 | 4 | 1 | 4 | 4 | 2.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 10.0 | 3.0 | 10.0 | 3.0 |
| 2 | 3 | 4 | 1 | 4 | 3 | 2.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 10.0 | 3.0 | 10.0 | 3.0 |
| 3 | 4 | 4 | 4 | 1 | 4 | 3.0 | 10.0 | 5.0 | 10.0 | 5.0 | 10.0 | 5.0 | 10.0 | 5.0 | 10.0 |
| 4 | 4 | 1 | 4 | 4 | 4 | 10.0 | 5.0 | 10.0 | 5.0 | 10.0 | 5.0 | 10.0 | 5.0 | 10.0 | 3.0 |
| 5 | 1 | 1 | 1 | 4 | 4 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 |
| 6 | 4 | 4 | 1 | 1 | 1 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 |
| 7 | 1 | 4 | 4 | 4 | 1 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 |
| 8 | 1 | 4 | 4 | 1 | 1 | 8.0 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 |
| 9 | 1 | 1 | 4 | 4 | 1 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 | 8.0 |
| 10 | 4 | 4 | 4 | 4 | 4 | 3.0 | 10.0 | 5.0 | 10.0 | 10.0 | 10.0 | 10.0 | 5.0 | 10.0 | 3.0 |
| 11 | 4 | 1 | 1 | 4 | 4 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 1.0 |
| 12 | 4 | 4 | 1 | 1 | 4 | 1.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 | 3.0 | 10.0 |
| 13 | 1 | 1 | 4 | 4 | 4 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 | 5.0 | 10.0 | 3.0 |
| 14 | 4 | 4 | 4 | 1 | 1 | 3.0 | 10.0 | 5.0 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 | 8.0 | 10.0 |
| 15 | 4 | 4 | 4 | 4 | 1 | 3.0 | 10.0 | 5.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 |
| 16 | 1 | 4 | 4 | 4 | 4 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 5.0 | 10.0 | 3.0 |

value *mindist* for the current obstacle configuration. This means the input pattern is similar to one or more prototype patterns. Consequently, if $d_{ij^*} \leqslant mindist$ for an input pattern $X_i$, then only those neurons with distances $d_{is}$'s not larger than *maxdist* are fuzzified using (3). In this case, the sum of the firing of rules will still equal one. The velocity command is calculated by the weighted sum of all firing rules. Here, the parameter *maxdist* is employed to reduce the influence of less important prototype patterns. For a $d_{ij}$ greater than *maxdist*, the input pattern is recognized as quite different from the prototype pattern. On the other hand, when an input pattern does not match or similar to any prototype pattern, i.e., $d_{ij^*}$ (the smallest $d_{ij}$) $>$ *mindist*, then it is treated as a special pattern that cannot be recognized. The velocities will not change in this situation. The threshold values *mindist* and *maxdist* are determined by experimental observation. The algorithm is summarized in Table 2.

In the present study, we implemented two behaviors for local navigation, namely *obstacle avoidance* behavior and *danger* behavior. *Obstacle avoidance* behavior was designed using the fuzzy-neuro network described above. *Danger* behavior is activated when the mobile robot is either trapped in a cul-de-sac, or the mobile robot is navigating in a special obstacle configuration that temporarily cannot be resolved by the ultrasonic sensors. A rule was designed to deal with such circumstances [13]. It instructs the mobile robot to spin around until it finds a direction in which to escape from the unfavorable situation. The direction of spin is determined according to the onboard sensory data. However, when *obstacle avoidance* behavior and *danger* behavior are triggered simultaneously, the latter has priority.

## 4. Simulation results

In order to verify the effectiveness of the proposed method, we set up a simulation program on a personal computer. The ultrasonic sensors were modeled by taking into consideration the characteristics of wide beam-angle and specular reflections [13]. The sampling period for ultrasonic sensor data updating was 0.5 s in the simulation. The velocity commands to the motor controller were assumed to be perfectly executed. Motion errors due to wheel slippage, surface irregularities, etc., were not considered.

Figs. 6–9 present several simulation results of the proposed fuzzy-neuro navigation algorithm. In these figures, the label 'S' denotes the start point and the label 'T' denotes the target position. The mobile robot is represented by a circle in proper proportion to the environment. The executed navigation route is depicted with a sequence of circles, where the positions

Table 2
The navigation algorithm

Define:

$mindist = 5,\ maxdist = 12$

$d_{ij^*} = \min(d_{i0}, d_{i1}, \ldots, d_{i(c-1)}),\ c = 16$

IF $d_{ij^*} = 0$
THEN

$u_{ij^*} = 1$

$u_{ij} = 0,\ j \neq j^*,\ (j = 0 \sim 15)$

$v_l = v_{lj^*},\ v_r = v_{rj^*}$

where $v_l$: left wheel velocity command
$v_r$: right wheel velocity command
$v_{lj^*}$: the left reference velocity of the $j^*$th rule
$v_{rj^*}$: the right reference velocity of the $j^*$th rule

ELSE IF $0 < d_{ij^*} \leqslant mindist$
THEN

$$u_{ij} = \left( \sum_{s \subset \bar{S}} \left( \frac{d_{ij}}{d_{is}} \right) \right)^{-1},\ j \subset \bar{S}$$

$$d_{ij^*} \leqslant d_{is} \leqslant maxdist \quad (7)$$

$\bar{S}$: the set of neurons satisfying (7)

$$v_l = \sum_{s \subset \bar{S}} v_{ls} u_{is} \quad (8)$$

$$v_r = \sum_{s \subset \bar{S}} v_{rs} u_{is} \quad (9)$$

ELSE

$v_l = v_l^*,\ v_r = v_r^*,$

where $v_l^*$ the left velocity command of previous sample instant
$v_r^*$ the right velocity command of previous sample instant

of the mobile robot was plotted for every four sampling periods. A darker-colored circle was plotted for every 40 sampling periods to enable easier determination of velocity profiles. The darker line-segment in each circle denotes the heading of the mobile robot. The size of the outside rectangle area shown in Figs. 6–8 is 12 m×12 m. Fig. 6 presents the robot's ability to avoid obstacles directly in front of it. Fig. 7 illustrates a situation where a local minimum would be faced using the potential field method, thus trapping the robot. As shown in the figure, the current design can handle this situation if the rectangles are not so wide. The robot will not move into the concave region and therefore navigate successfully to the target. However, for deeper traps or a closer target position to the obstacle as shown in Fig. 8, the mobile robot will move into

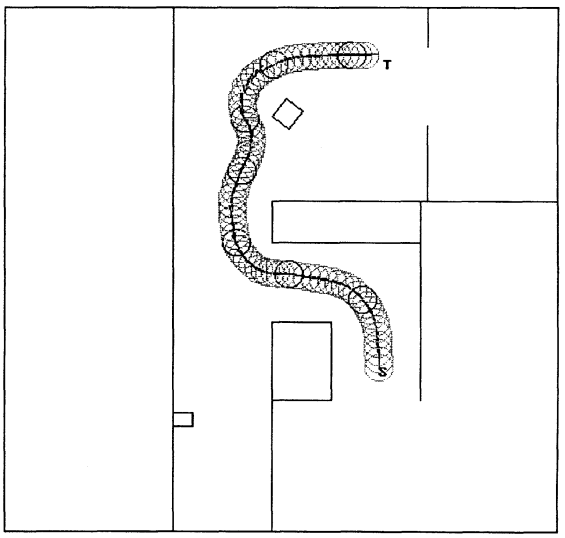


Fig. 6. Simulation result of avoiding head-on obstacles.

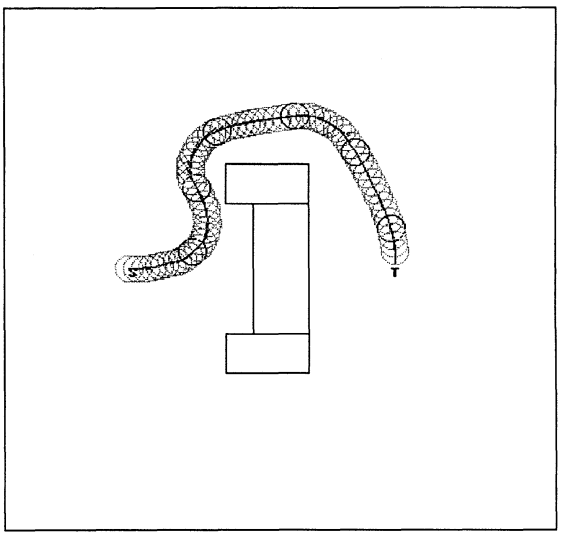


Fig. 7. Simulation result of navigating in an environment where a local minimum exists.

the concave region and be trapped. In such situations, the *danger* behavior will come into action and bring the robot out of the trap (see Fig. 8). A *wall-following* behavior can be added to the navigation system, allowing the mobile robot to travel along obstacle's contour for escaping from the trap [9]. Fig. 9 presents the simulation result of navigating in a long corridor. In this case, the surrounding area was 40 m × 40 m, an example of our laboratory environment. This simulation

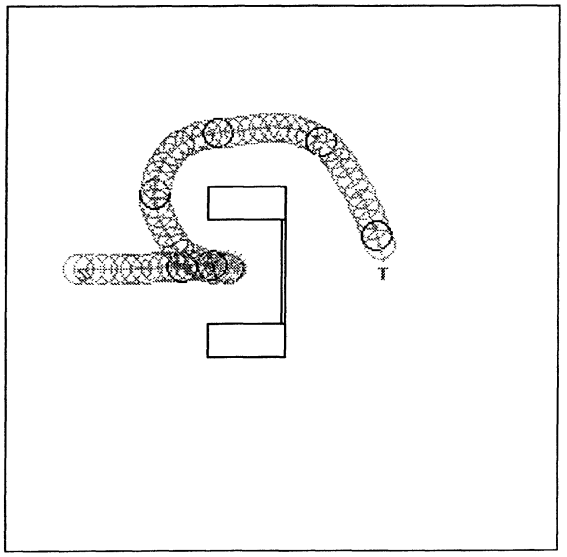

Fig. 8. Simulation result of encountering a trap.

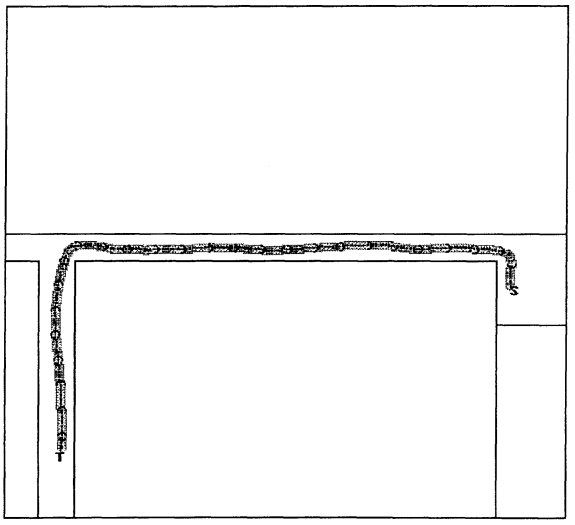

Fig. 9. Simulation result of navigation in a long corridor.

depicts that the mobile robot can explore an unknown environment exploiting onboard sensory information.

## 5. Experimental results

Practical navigation experiments were conducted employing a self-constructed mobile robot. It is of cylindrical shape with a diameter about 60 cm. Two drive wheels are placed at the ends of its central axis, and there are two free casters at the front and rear for balance. Motion control is accomplished by differential-velocity steering using the independent drive wheels. This motion control method has two advantages. One is that the robot can spin 360° in place; the other is that we can control the robot simply by velocity commands to the two drive wheels. The robot has a maximum travel speed of $V_{\max} = 43$ cm/s. An industrial personal computer AT-486 is carried onboard for navigation control. Two HCTL-1100 motion control chips from Hewlett-Packard are used for motor servo control. The merit of employing these chips is that they accept velocity commands, decreasing the computation burden of the onboard computer. The specified traveling speed in these experiments was 20 cm/s and the sampling time was 250 ms. It takes less than 1 ms for *obstacle avoidance behavior* calculation in the current implementation. In the experiments, only the start and target positions were specified. The mobile robot had to find a collision-free path to the target employing onboard sensory information.

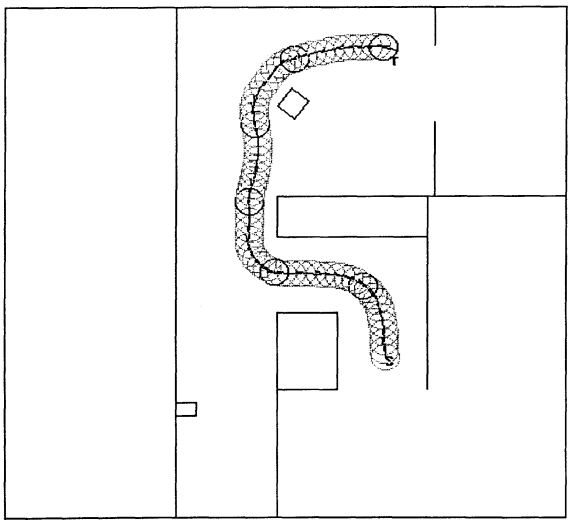

Fig. 10. Experimental result of obstacle avoidance.

Fig. 10 presents an experimental result in which the navigation path depicts two occasions of turning away from walls and avoidance of a rectangular carton. The recorded trajectory reveals that the robot can avoid

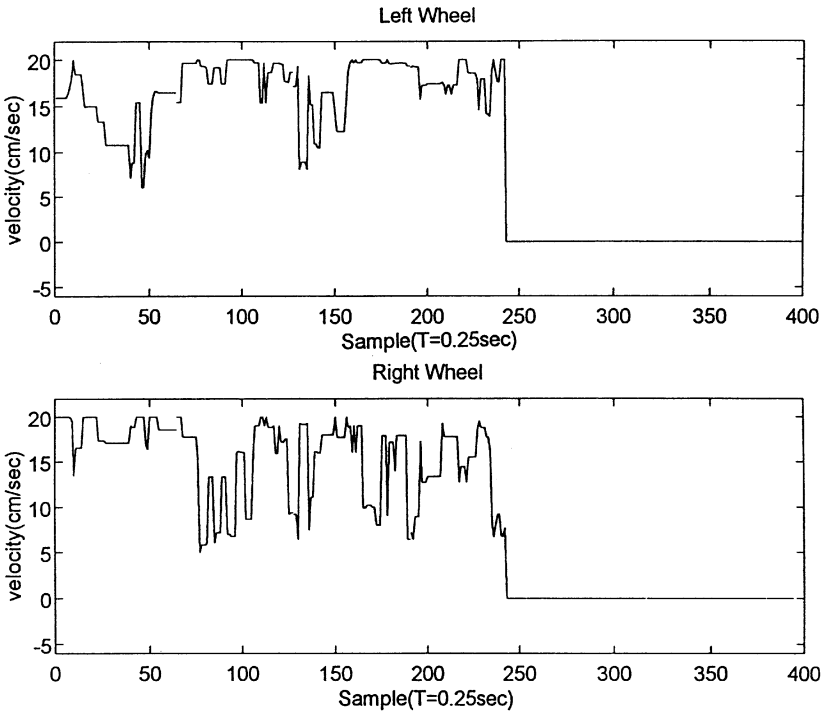


Fig. 11. Recorded wheel velocities of the experiment of Fig. 10.

obstacles safely and in an efficient manner. Fig. 11 presents the recorded velocity profiles of both wheels in this run. They can be used to check the maneuvering of the robot navigation as it encounters obstacles. We see that the *danger behavior* was not activated in this case. Consequently, a smooth trajectory was executed in obstacle avoidance and traveling to the target.

Fig. 12 presents the experimental result of exploration in a long corridor. As in the simulation, only the target position was specified in this experiment; the mobile robot had to explore in an unknown environment employing its onboard sensors. We see in the figure that the recorded route deviates from the actual locations in the corridor. This discrepancy is mainly due to wheel slippage which causes accumulated errors in the onboard odometer. Notably, the mobile robot explored the environment safely in this long journey without any collision. The experimental result in Fig. 12 reveals more velocity variations than the simulation result shown in Fig. 9. This is because that there are doors and extinguisher-boxes along the corridor wall, which are concave and convex regions for the ultrasonic sensors. Moreover,

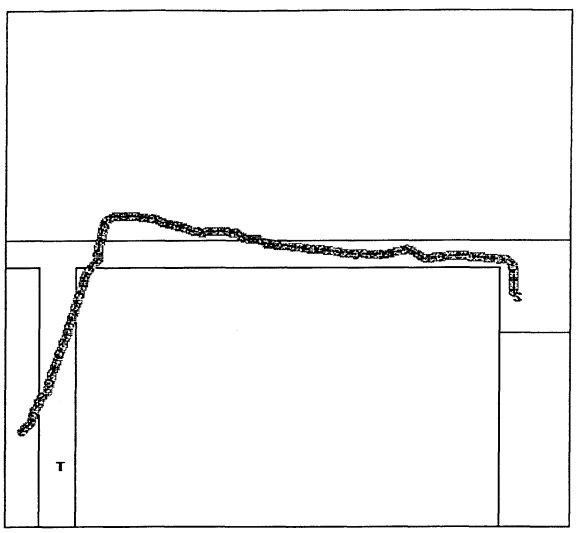


Fig. 12. Experimental result of exploration in a long corridor.

actual ultrasonic sensor data contain measurement errors caused by specular reflections and wide beam-opening angle; this also causes velocity variation during navigation. It should be mentioned that this

method is good for local reactive navigation, the mobile robot is controlled by perceptual sensors. For practical applications, a global path planner should be provided for optimal performance.

## 6. Conclusion

A pattern-recognition approach to reactive navigation based on real-time sensory information has been developed and successfully implemented on an autonomous mobile robot. Through the process of prototype-pattern assignment, the proposed fuzzy neural network can be adapted for reactive motion control. By employing a small number of rules, satisfactory performance has been achieved. The amount of computation is therefore reduced a great deal and this enhances the real-time performance of reactive control. Furthermore, this method offers a straightforward mapping between control rules and human-like heuristics. The mobile robot can therefore demonstrate human-like tendencies for continuous motion without stopping for obstacles. Many aspects of this method are worth further investigation in the future. On the one hand, other sensors such as CCD cameras can be used for better detection of obstacles. Although the present design can cope with moving as well as stationary obstacles, more accurate perception sensors are required for more complex environmental configurations. On the other hand, optimization of the prototype-mapping can be made to play a more important role in the algorithm.

## References

[1] R.C. Arkin, Motor schema-based mobile robot navigation, Int. J. Robotics Res. 8 (4) (1989) 92–112.

[2] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.

[3] J.C. Bezdek, Fuzzy models for pattern recognition, in: J.C. Bezdek, A.K. Pal (Eds.), Fuzzy Models for Pattern Recognition, IEEE Press, New York, 1992, pp. 1–27.

[4] J. Borenstein, Y. Koren, The vector field histogram-fast obstacle avoidance for mobile robot, IEEE Trans. Robotics Automat. 7 (3) (1991) 278–288.

[5] R.A. Brooks, A robust layered control system for a mobile robot, IEEE Trans. Robotics Automat. 2 (1) (1986) 14–23.

[6] S.G. Goodridge, M.G. Kay, R.C. Luo, Multilayered fuzzy behavior fusion for real-time reactive control of systems with multiple sensors, IEEE Trans. Industrial Electron. 43 (3) (1996) 387–394.

[7] T. Huntsberger, P. Ajjimarangsee, Parallel self-organizing feature maps for unsupervised pattern recognition, Intentional J. General Systems 16 (4) (1990) 357–372.

[8] S. Ishikawa, A method of indoor mobile robot navigation by using fuzzy control, Proc. IEEE/RSJ IROS'91, Osaka, 1991, pp. 1013–1018.

[9] C.H. Lin, L.L. Wang, Intelligent collision avoidance by fuzzy logic control, Robotics and Autonomous Systems 20 (4) (1997) 61–83.

[10] S. Mahadevan, J. Connell, Automatic programming of behavior-based robots using reinforcement learning, Artificial Intelligence 55 (1992) 311–365.

[11] G. Oriolo, G. Ulivi, M. Vendittelli, Fuzzy maps: A new tool for mobile robot perception and planning, J. Robotics Systems 14 (3) (1997) 179–197.

[12] P. Reignier, V. Hansen, J.L. Crowley, Incremental supervised learning for mobile robot reactive control, Robotics Autonomous Systems 19 (1997) 247–257.

[13] L.H. Sheen, A fast path-planning method for a mobile robot in an unknown environment, Master Thesis, National Chiao Tung University, 1994.

[14] K.T. Song, J.C. Tai, Application of virtual concepts and fuzzy control for mobile robot navigation, Proc. NSC – Part A: Phys. Sci. Eng. 18 (4) (1994) 400–411.

[15] Y.H. Suen, Intelligent motion planning and control for a mobile robot, Master Thesis, National Chiao Tung University, 1995.