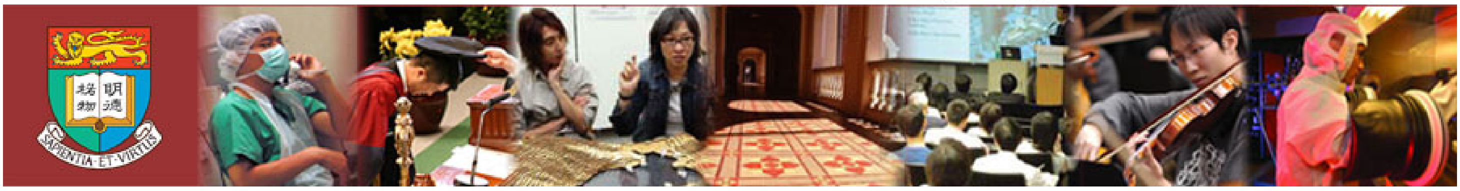

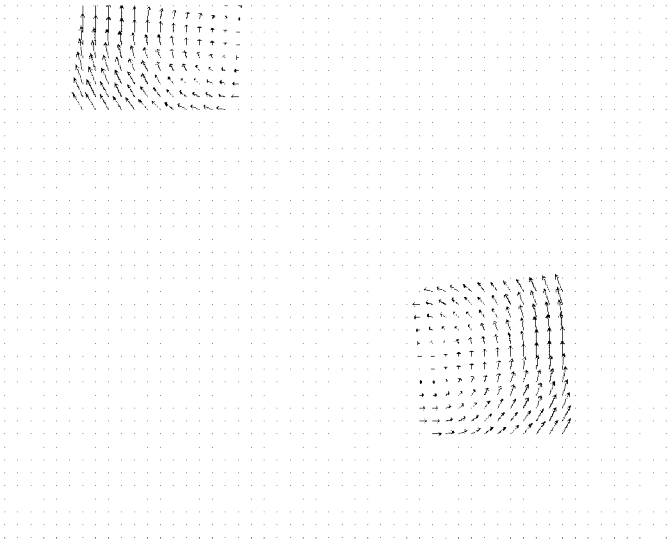

Fig. 8. X, Y components of the estimated velocity field for the multiple object image sequence.

trajectory of a moving point forms the basis of a 3-D flow constraint equation. This equation is the 3-D analog of the well known optical flow constraint equation. The velocity components at every point is computed by solving an overdetermined system of equations obtained from the neighborhood of the point. Theoretical analyses are presented to establish the noise and discontinuity robustness of the proposed 3-D flow constraint equation. Results of the experimental performance evaluation of the proposed technique support the theoretical analysis results.

The authors are currently investigating the application of the proposed invariant to correspondence based 3-D motion detection. The proposed invariant has potential applications in range image segmentation and analysis. Further research is necessary.

REFERENCES

[1] K. Arun, T. Huang, and S. Blostein, "Least square fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 698–700, 1987.

[2] H. Goldstein, *Classical Mechanics*. Reading, MA: Addison-Wesley, 1990.

[3] R. S. Millman and G. D. Parker, *Elements of Differential Geometry*. Englewood Cliffs, NJ: Prentice-Hall, 1977.

[4] B. K. P. Horn, E. Hildreth, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. Opt. Soc. Amer.*, vol. 5, pp. 1127–1135, 1988.

[5] B. K. P. Horn and J. Harris, "Rigid body motion from range image sequences," *CVGIP: Image Understanding*, vol. 53, no. 1, pp. 1–13, 1991.

[6] B. K. P. Horn and B. G. Schunk, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, 1981.

[7] B. Lucas, Ph.D. dissertation, Dept. Comput. Sci., Carnegie Mellon University, Pittsburgh, PA, 1984.

[8] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*. Cambridge, U.K.: Cambridge Univ. Press, 1988.

[9] B. Sabata and J. K. Agarwal, "Estimation of motion from a pair of range images," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 309–324, 1991.

[10] G. Strang, *Linear Algebra and Its Applications*. New York: Academic, 1980.

An Intelligent Mobile Vehicle Navigator Based on Fuzzy Logic and Reinforcement Learning

Nelson H. C. Yung and Cang Ye

***Abstract*—In this paper, an alternative training approach to the EEM-based training method is presented and a fuzzy reactive navigation architecture is described. The new training method is 270 times faster in learning speed; and is only 4% of the learning cost of the EEM method. It also has very reliable convergence of learning; very high number of learned rules (98.8%); and high adaptability. Using the rule base learned from the new method, the proposed fuzzy reactive navigator fuses the obstacle avoidance behavior and goal seeking behavior to determine its control actions, where adaptability is achieved with the aid of an environment evaluator. A comparison of this navigator using the rule bases obtained from the new training method and the EEM method, shows that the new navigator guarantees a solution and its solution is more acceptable.**

***Index Terms*—Behavior fusion, fuzzy logic, goal seeking, neural network, obstacle avoidance, reinforcement learning, vehicle navigation.**

I. INTRODUCTION

The navigation of a mobile vehicle can be considered as a task of determining a *collision free path* that enables the vehicle to travel through an obstacle course from an initial configuration to a goal configuration. The process of finding such path is also known as the *path planning problem* which could be classified into: global path planning and local path planning. *Global path planning* methods are usually conducted off-line in a completely known environment. Many attempts at solving this problem have been tried [1], where an *exact environment model* has been used for planning the path. Although these approaches have an exact solution, their time complexity grows with the geometry complexity and grows exponentially with the number of degrees of freedom in the vehicle's motion [2]. Thus they are only practical when the environment model is simple and the number of degrees of freedom is reasonably low. However, real environments are never simple enough. This fact has led to the emergence of numerous *heuristic approaches,* which rely on either calculating the *potential fields* [3] or performing a search through a *state space model* [4]. In general, these methods trade reliability for speed, in which they do not guarantee a solution even if there exists one. Worse still, all these approaches fail when the environment is not fully known.

On the other hand, the *local path planning* techniques, also known as the obstacle avoidance methods, are potentially more efficient in vehicle navigation when the environment is unknown or only partially known. It utilizes the on-line information provided by sensors such as the ultrasonic sensor, laser range finder, radar, and vision sensor, to tackle the uncertainty. An efficient local path planning method is the potential field method, which was first proposed by Khatib [3] and has been widely used in obstacle avoidance cases [5]. In spite of it's simplicity and elegance, this method has three problems: First, local minimum could occur and cause the vehicle to be stuck; second, it

Manuscript received March 4, 1997; revised January 15, 1998. This work was supported by the CRCG of The University of Hong Kong under Grant 337/062/0016. This paper was recommended by Associate Editor A. Kandel.

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: nyung@eee.hku.hk).

Publisher Item Identifier S 1083-4419(99)00910-3.

tends to cause unstable motion in the presence of obstacles; and third, it is difficult to find the force coefficients influencing the vehicle's velocity and direction in an unknown environment. The practicality of the potential field method therefore hinges on how well these problems can be resolved. One of the possible solutions that has the potential to overcome these problems is the *reactive system* proposed in [6]. The key idea of it is to build a mapping from the perceived situations to the correct actions and iterate the mapping until a goal is reached. As previous reactive systems are not able to learn and acquire situation-action rules [6]–[8], neural network and fuzzy logic approaches offer an attractive alternative for building reactive systems in recent times [9]–[13].

Fuzzy logic approach seems quite promising in tackling the problem of obstacle avoidance, as it deals with various situations without requiring to construct an analytical model of the environment. While compared with the neural network approach, it has another distinct advantage that each rule of the rule base has a physical meaning. This makes it possible to tune the rules by using expert's knowledge. For instance, Tunstel [10] proposed a hierarchical fuzzy behavior control for indoor navigation, in which the rule base for each behavior is constructed based on expert's knowledge. However, in the case of navigating the mobile vehicle in complex environments, the above method have two severe limitations that it is difficult to consistently construct the rules since there are many situations to be handled; and it is time consuming to tune the constructed rules. Song and Tai [11] tried to palliate the load on the construction of the rule base by separating the fuzzy controller into left and right and made them work in collaboration.

To tackle some of these drawbacks, supervised learning methods using neural networks were proposed in [12] and [13]. Unfortunately, these methods require a substantially large set of representative patterns to characterize the environment during training. Besides, it is also difficult to obtain these training patterns which contain no contradictory input/output pairs. Thus, reinforcement learning requiring only a scalar reinforcement signal as a performance feedback from the environment seems to be quite attractive when learning collision-free navigation is concerned. This reinforcement signal enables the navigator to tune its performance to some extent. However, reinforcement learning method has theoretically limited learning ability as it requires heavy learning phases and in some case, it might not be able to completely capture the complex features of an environment. Kröse and van Dam [14] used Kohonen maps to split the sensory input space into clusters, and associate an appropriate action to it through reinforcement learning. In theory, its performance improves with learning, although there is no criterion derived to evaluate the convergence of the learning process. Beom and Cho [15] proposed a scheme in which fuzzy logic was used to map the sensor input space to the action space and reinforcement learning was employed to construct the fuzzy rules automatically. They proposed no criterion to evaluate the convergence either. Furthermore, both of these methods are based on the environment exploration method (EEM) which explores a complex environment to obtain the correct situation–action mapping. The theoretical limitation of reinforcement learning inevitably results in a slow and uncertain convergence due to the EEM and an insufficiently learned rule base in most cases. It is this limitation that motivates us to search for a better training method that has a definite and fast convergence, and an overall navigation architecture that can tackle complex and unknown environments.

In this paper, an alternative training approach to the EEM-based training method is presented and a fuzzy reactive navigation architecture is described. The new training method employs a simple environment for constructing the fuzzy rule base and has five distinct advantages over the existing EEM:

1) 270 times faster in learning speed;
2) only 4% of the learning cost;
3) very reliable convergence of learning;
4) 98.8% of learned rules;
5) high adaptability.

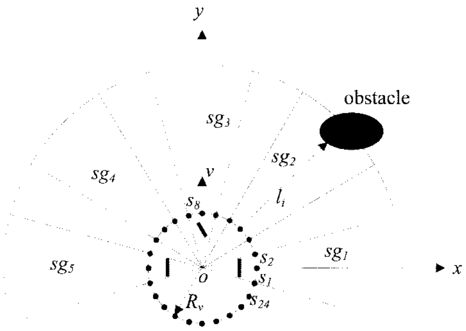

Fig. 1. Diagram of the mobile vehicle and the arrangement of the ultrasonic sensors.

Using the rule base learned from the new method, the fuzzy reactive navigator fuses the sensor information from the sensor groups, the obstacle avoidance behavior and goal seeking behavior to determine its control actions, where adaptability is achieved with the aid of an environment evaluator. Numerous simulation runs show that this new navigator is characterized by first, its ability to tackle an unknown environment without having to explore it beforehand; second, its free of local minimum; third, it has smooth changes of velocity and steering angle; fourth, its planned path is close to the shortest path; and fifth, it is both nearsighted and farsighted.

II. Overview of the Navigator

A. Vehicle Model and Sensor Arrangement

The model of the vehicle used is a cylindrical mobile platform driven by three active wheels. The radius of the mobile vehicle, R_v is approximately 20 cm, and is equipped with an ultrasonic sensor ring as depicted in Fig. 1. Assume there are N sensors evenly distributed along the ring and they are divided into eight groups where each group has at least one sensor (i.e., $N \geq 8$). For obstacle detection and avoidance purposes, five sensor groups are selected from the eight groups. The remaining sensors are used by the navigation supervisor where a sensor group is dynamically configured to determine the minimum distance to the obstacle located along the relative goal vector during navigation.

In this research, a ring of 24 ultrasonic sensors is assumed to give an angular spatial resolution of 15°. Each sensor, s_i for $i = 1, \cdots, 24$, gives a distance to the obstacle, l_i, in its field of view, where 8 cm $\leq l_i \leq$ 400 cm and each sensor covers an angular view of 10°. The five sensor groups are denoted as sg_i for $i = 1, \cdots, 5$, where each group is composed of three neighboring sensors. With this sensor arrangement, the distance, d_i, measured by the ith sensor group from the center of the vehicle to the obstacle is expressed as

$$d_i = R_v + \min(l_j | j = 3i-2, 3i-1, 3i); \quad \text{for } i = 1, \cdots, 5. \quad (1)$$

B. Coordinate Systems and Navigation Task

The coordinate systems and the control variables of the vehicle are depicted in Fig. 2. The two coordinate systems are the world coordinate denoted by XWY, and the vehicle coordinate given by xoy. Based on these two coordinate systems, a navigation task is

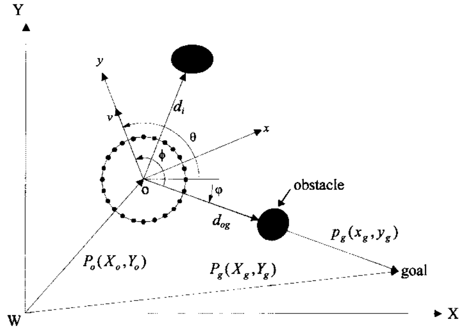

Fig. 2. Diagram of the coordinate system and the control variables.

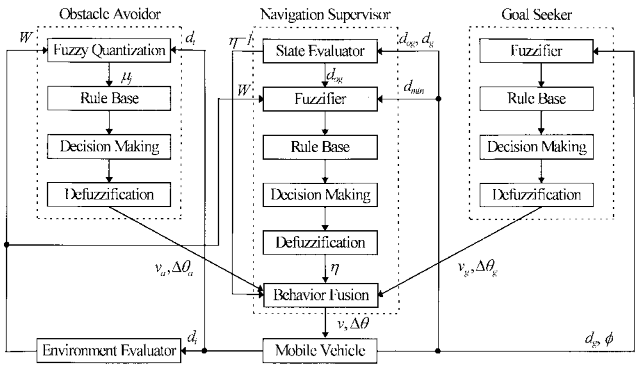

Fig. 3. Diagram of the proposed navigator.

defined as navigating the vehicle from a start coordinate to its goal without colliding with the obstacles in between. Each navigation task is specified in the world coordinate, where the vehicle configuration is represented by $S = (X_o Y_o \theta)^T$, where X_o and Y_o are coordinate of the vehicle's center, and θ stands for the heading angle of the vehicle. Without considering the vehicle dynamics, we assume the control variables are its linear velocity v and the change in the heading angle (steering angle), $\Delta\theta$.

In order to navigate the mobile vehicle to it's goal, it is assumed that the current configuration of the mobile vehicle is always known at each time step t. Therefore, a navigation task is to obtain the environment information, d_i and $P_g(X_g, Y_g)$, and the vehicle's configuration $S(t)$ at each time step t, where $t = 0, 1, \cdots, k, \cdots$; determine the output variables $v(t)$ and $\Delta\theta(t)$; then update the vehicle's configuration; and iterate this situation–action mapping process until the goal is achieved.

C. Architecture of the Navigator

A navigation task can be achieved by two behaviors: *obstacle avoidance* and *goal seeking*. The former behavior is inherently *nearsighted* as it only considers how to avoid obstacles and ignores whether it causes the vehicle to deviate from the goal; whereas the latter behavior is inherently *farsighted* as it enables the vehicle moves toward the goal and neglects if it causes a collision. When the vehicle encounters a obstacle which obstructs the goal, these two behaviors are in conflict, where an arbitrator is required to mediate between the two behaviors. To resolve this conflict, an overall navigation architecture is depicted in Fig. 3 [17].

It consists of four main modules: an *obstacle avoider* (OA), a *goal seeker* (GS), a *navigation supervisor* (NS), and an *environment evaluator* (EE). The OA determines the action, v_a and $\Delta\theta_a$ for the behavior of obstacle avoidance, while the GS determines the action, v_g and $\Delta\theta_g$ for the behavior of goal seeking. These two behavioral modules work independently and their actions are fused by the NS to produce v and $\Delta\theta$ for the eventual navigation. The role of the *environment evaluator* is vital here. It's purpose is to perform an evaluation of the environment based on the distances sensed by the ultrasonic sensors and determine the appropriate value of W based on the sensor readings, where W is used for fuzzification by the fuzzy quantization inside the OA. With this ability, the navigator is able to develop both nearsighted and farsighted decisions. It is also due to the EE that the rule base used by the OA can be learned using a simple corridor-like training environment, where the training converges quickly with extremely small number of blank rules.

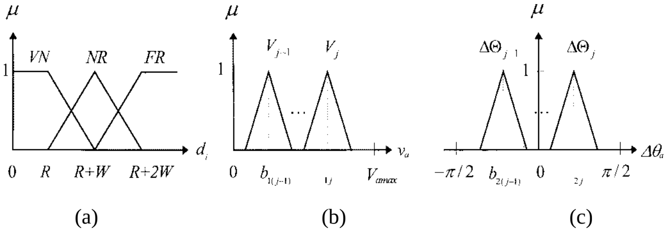

Fig. 4. Membership functions of the input/output variables for the OA.

III. Obstacle Avoider

The OA is a reactive fuzzy controller, in which the fuzzy rule base determines the vehicle's action, v_a and $\Delta\theta_a$, based on the input sensor readings d_i for $i = 1, \cdots, 5$. Being a reactive system, the key to the OA is the construction of the rule base.

A. Fuzzy Control of Obstacle Avoidance

As depicted in Fig. 3, the input variables of the OA are the sensor input variables, d_i, while the control outputs are v_a and $\Delta\theta_a$. The design steps of this module are as follows:

1) definition of membership functions;
2) fuzzification of the input variables;
3) rule base construction through reinforcement learning;
4) fuzzy inference;
5) defuzzification of the output variables.

The membership functions of the input and output variables are given in Fig. 4.

In Fig. 4(a), the crisp value of each input variable, d_i is fuzzified and expressed by the fuzzy sets—VN, NR, FR, referring to very near, near, and far, respectively. The universe of discourse of each fuzzy set is determined by R and W. Here, $R = R_v + l_{\min}$, where $l_{\min} = 8$ cm is the minimum distance that the ultrasonic sensor can detect and $R_v = 20$ cm is the radius of the vehicle. Therefore, the fuzzy sets are fully determined by the variable, W. Each sensor reading, d_i is mapped to a set of different membership function values and further mapped to a different action for a different W. In accordance with the fuzzification of the input variables, d_i, the fuzzy rule base consists of 243 rules and it requires 243 fuzzy sets, $V_j\,(j = 1, \cdots, 243)$, to represent the velocity v_a; and 243 fuzzy sets, $\Delta\theta_j\,(j = 1, \cdots, 243)$, to represent the steering angle $\Delta\theta_a$. The fuzzy sets of the output variables v_a and $\Delta\theta_a$ take the triangular membership functions as shown in Fig. 4(b) and (c), respectively, while their center positions, b_{1j} and b_{2j} for $j = 1, \cdots, 243$, are determined by the reinforcement learning. The upper bound of the velocity for the reinforcement learning is $V_{a\,\max}$.

The fuzzy rule base plays a central role in mapping the sensor input space d_i to the mobile vehicle's action space v_a and $\Delta\theta_a$. It

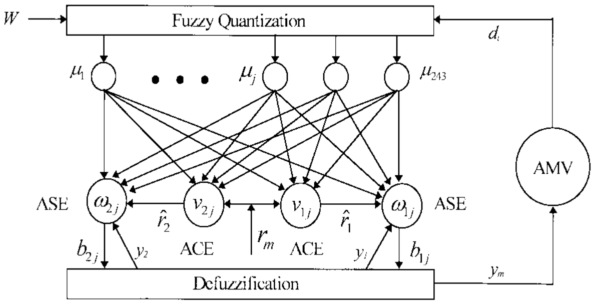

Fig. 5. Diagram of the neural network to learn obstacle avoidance.

consists of 243 fuzzy rules, each of which is denoted by

$$\text{Rule } j\text{:}\quad \text{IF } d_i \text{ is } D_{j1} \text{ AND } \cdots \text{ AND } d_5 \text{ is } D_{j5}$$
$$\text{THEN } v_a \text{ is } V_j,\ \Delta\theta_a \text{ is } \Delta\Theta_j$$

for $j = 1, \cdots, 243$, where D_{ji} for $i = 1, \cdots, 5$, are the fuzzy sets for d_i in the jth rule, which take the linguistic value of VN, NR or FR; v_a and $\Delta\theta_a$ denote the output variables; and V_j and $\Delta\Theta_j$ are the fuzzy sets for v_a and $\Delta\theta_a$ in the jth rule. Let the fire strength of the jth rule be denoted by μ_j. For the inputs $d_i = d_i'$, the fire strength of the jth rule, μ_j can be written as

$$\mu_j = \mu_{D_{j1}}(d_1') \wedge \mu_{D_{j2}}(d_2') \wedge \mu_{D_{j3}}(d_3') \wedge \mu_{D_{j4}}(d_4') \wedge \mu_{D_{j5}}(d_5'). \tag{2}$$

If the Mamdani's minimum operation [16] is used for fuzzy implication, the memberships of the inferred fuzzy control action, V and $\Delta\Theta$ are calculated by

$$\mu_V(v_a) = \bigcup_{j=1}^{243} \mu_j \wedge \mu_{V_j}(v_a)$$

and

$$\mu_{\Delta\Theta}(\Delta\theta_a) = \bigcup_{j=1}^{243} \mu_j \wedge \mu_{\Delta\Theta_j}(\Delta\theta_a). \tag{3}$$

For the reason of limiting the computing cost, the method of height defuzzification is used. The crisp control action is given by

$$v_a = \frac{\sum_{j=1}^{243} \mu_j b_{1j}}{\sum_{j=1}^{243} \mu_j}$$

and

$$\Delta\theta_a = \frac{\sum_{j=1}^{243} \mu_j b_{2j}}{\sum_{j=1}^{243} \mu_j}. \tag{4}$$

B. Rule Learning for Obstacle Avoidance

In essence, the problem engaged in constructing the rule base is to determine the values for b_{1j} and b_{2j}. In the case of learning to avoid obstacle, the input/output pairs are not available, thus this becomes an unsupervised learning problem. In this paper, we employ the reinforcement learning method using the Sutton and Barto's model [18] to fulfill this learning requirement (Fig. 5).

In principle, the vehicle begins the learning with an initial v and $\Delta\theta$ at time step $t = 0$, then the vehicle moves into a new position at time step $t = 1$, and so on, until a collision occurs at the time step $t = k$. The whole process, until a collision occurred, is called a trial and the time step t, $(t > 0)$ is called the t^{th} learning step. For instance, if a trial ends at $t = k$ where a collision occurs, then a failure signal is fed back to the learning network, and the rules which were used at the previous time steps k, $k - 1$, $k - 2$, that have contributed to this failure would be changed in order to get an improvement on the vehicle's performance. In Fig. 5, this task is accomplished by two adaptive neuron-like elements in which one is for updating the rules concerning v and the other one for updating the rules concerning $\Delta\theta$. Each element consists of an associative search element (ASE) and an associative critic element (ACE) [17]. After the rules are updated, a new trial begins at the $(k+1)^{\text{th}}$ learning step. The process is iterated and terminated until no more collisions occur.

Suppose that the current configuration of the vehicle is $S(t) = (X_o(t)\, Y_o(t)\, \theta(t))^T$, and the five sensor group readings are encoded into μ_j. In order to give the associativity in learning the rules, the trace, $\overline{\mu}_j(t)$ of the fired jth rule, is used. The trace at time step, $t + 1$ is given by

$$\overline{\mu}_j(t+1) = \lambda\overline{\mu}_j(t) + (1 - \lambda)\mu_j(t) \tag{5}$$

where λ, $0 \le \lambda < 1$, is the trace decay rate. Each ACE receives the external reinforcement signal, $r_m(t)\,(m = 1, 2)$, as a performance feed back from the environment, and generates the internal reinforcement signals, $\hat{r}_m(t)\,(m = 1, 2)$, which are fed into the ASE for updating their weights. The external reinforcement signal is determined by

$$r_m(t) = \begin{cases} -1, & \text{if } \min(d_i \,|\, i = 1, 2, \cdots, 5) \\ & \quad < R_v + V_{a\,\max} \times \Delta T \quad \text{for } m = 1, 2 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where, ΔT is the time interval between two learning steps. In order to define the internal reinforcement signal, the temporal difference learning theory is used [19], where if the external reinforcement signal is $r_m(t)$ at time step t, then $P_m(t)$ may be used to predict the discounted sum

$$z_m(t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_m(t'+1) \tag{7}$$

where the discount-rate parameter γ, $0 < \gamma < 1$, determines the extent of the prediction. If the predictions are accurate, then from (7) we have:

$$p_m(t-1) = r_m(t) + \gamma p_m(t). \tag{8}$$

In practice, the ACE learns to make the predictions where $p_m(t)$ is implemented as a weighted sum of $\mu_j(t)$ and given by

$$p_m(t) = G\left(\sum_{j=1}^{243} v_{mj}(t)\mu_j(t)\right) \tag{9}$$

where $G(x) = 2/(1 + e^{-\xi x}) - 1$. Thus the mismatch or time difference error between the two sides of (8) is defined as the internal reinforcement signal and can be expressed as

$$\hat{r}_m(t) = r_m(t) + \gamma p_m(t) - p_m(t-1) \tag{10}$$

where v_{mj} for $m = 1, 2$; $j = 1, \cdots, 243$, are the weights of the ACE. In order to predict $p_m(t)$ correctly, the weights of ACE must be updated, which can be expressed as

$$v_{mj}(t+1) = v_{mj}(t) + \beta\hat{r}_m(t)\overline{\mu}_j(t) \tag{11}$$

where β is a positive constant determining the rate of change of v_{mj}. Similarly, the weights of the ASE, ω_{mj} for $m = 1, 2; j = 1, \cdots, 243$, are updated by

$$\omega_{mj}(t+1) = \omega_{mj}(t) + \alpha\, \hat{r}_m(t) e_{mj}(t) \tag{12}$$

where α, $0 < \alpha \leq 1$, determines the learning rate, and $e_{mj}(t)$ is the eligibility trace of the jth rule at time t, which is updated by

$$e_{mj}(t+1) = \delta\, e_{mj}(t) + (1-\delta) y_m(t) \mu_j(t) \tag{13}$$

where δ, $0 \leq \delta < 1$, is the decay rate of the eligibility. The eligibility trace is a trace of what rules have been used and what control actions have been applied to the vehicle. Here $y_m(t)$, are the control action defined as $(y_1(t)\, y_2(t))^T = (v_a\, \Delta\theta_a)^T$. The center positions of the fuzzy sets at each time step are determined by

$$b_{mj}(t) = b_m + \frac{\omega_{mj}(t) f_m}{k\, \max(|\omega_{mj}(t)|) + |\omega_{mj}(t)|} \tag{14}$$

where b_1 is the initial center position for the fuzzy sets, V_j; b_2 is the initial center position for the fuzzy sets, $\Delta\theta_j$; f_m is a positive constant that determines the range of $b_{mj}(t)$; and the positive constant k is used to guarantee the fuzzy sets of the output variables to be within their universe of discourse. According to (14), $b_{mj}(t) \in [b_m - f_m/(1+k), b_m + f_m/(1+k)]$ and $V_{a\,\max}$ is $(b_1 + f_1)$. With known control actions of the current time step, the vehicle configuration at the next time step is updated by

$$S(t+1) = \begin{pmatrix} \theta(t) + \Delta\theta_a(t) \\ X_o(t) + v_a(t)\Delta T \cos(\theta(t) + \Delta\theta_a(t)) \\ Y_o(t) + v_a(t)\Delta T \sin(\theta(t) + \Delta\theta_a(t)) \end{pmatrix}. \tag{15}$$

Eventually, if the rules are sufficiently learned in a specific environment, the weights of the ASE converge to a set of fixed values. When the learning process is terminated, the learned set of b_{mj} is used as the rule base for the OA.

C. Simulation of Rule Learning for Obstacle Avoidance

In a reinforcement learning problem, the only piece of available information is presented by the reinforcement signal received from the environment. To obtain the gradient information, a reinforcement system probes the environment through the combined use of trials and errors and delayed reward. This form of exploration searches for directional information on the basis of the environment properties. In so doing, however, the reinforcement learning system is slowed down. This phenomenon is known as the conflict between exploration and exploitation [20], which can be stated as a conflict between

1) the desire to use the rule base already learned;
2) the desire to acquire more knowledge about the consequences of the actions so as to make further improvement on the rule base in the future.

For efficient learning, a tradeoff between exploration and exploitation should be achieved to maximize the effect of learning and minimize the costs of exploration.

1) Environment Exploration Method: The EEM is a straightforward and simple method. It explores and converges slowly in a complex environment for learning and compiling the rule base. For the purpose of simulation, a slightly more complex, computer generated environment is used (Fig. 6). The parameters used for the simulation are shown in Table I. It is possible that other sets of parameters could be used for this purpose. As the new training method is compared with the EEM on the same set of parameters, the difference in parameter values should not have any significant effect on the simulation results.

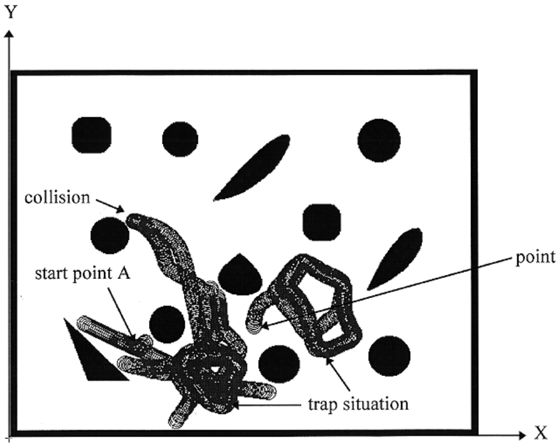

Fig. 6. Training by the EEM ($W = 60$ cm, size 1000 cm × 800 cm).

At the start, $v_{mj}(t)$ are set to some small nonzero values, while $\omega_{mj}(t)$, $\overline{\mu}_j(t)$, $p_m(t-1)$, $e_{mj}(t)$ are set to zero. The vehicle begins at an arbitrary initial configuration of $S(0)$ with nonzero initial control action. In order to facilitate the learning process, a prespecified rule in the rule base is used which is IF $d_i = FR$ for $i = 1, \cdots, 5$, THEN $b_{1,243} = 27.5$ cm/s and $b_{2,243} = 0$. The learning process is iterated as follows: 1) the current distance readings from the five sensor groups are fed into the fuzzy quantization module, where they are encoded into $\mu_j(t)$; 2) if $\max(|\omega_{mj}(t)|) = 0$, then the initial control actions, v_a and $\Delta\theta_a$, are used as the control outputs for this situation; otherwise the control outputs are determined by (4); 3) the external reinforcement signal is calculated by (6), the current prediction value $p_m(t)$ is calculated by (9), and the internal reinforcement signal is calculated by (10); 4) the weights of the ACE and ASE are updated by (11) and (12), while the trace of the rule and the eligibility trace are updated by (5) and (13), respectively; and 5) finally, if there is no collision, the configuration of the vehicle is changed by (15) and the learning process returns to Step 1. If a collision occurs, i.e., $r_m(t) = -1$; $v_{mj}(t)$, $\overline{\mu}_j(t)$, $p_m(t-1)$ and $e_{mj}(t)$ are reset to zero. The vehicle is backtracked 4 steps and its heading direction is reversed. The weights of the ASE, $\omega_{mj}(t)$ which are learned just before the collision are then used for the next trial. The next trial begins by repeating Step 1 through Step 5 again.

As the learning process continues, the center positions of the membership functions of the output fuzzy sets are tuned from the initial values, b_1 and b_2 to the correct values, b_{1j} and b_{2j}. In the example shown in Fig. 6, the start configuration of (280 cm 196 cm $-45°)^T$ was used. After a number of learning steps, the vehicle went into a trap situation and failed to get out. Then the vehicle was moved to a new start configuration of (520 cm 240 cm $0°)^T$ manually and a new trial began. The weights of the ASE were tuned as the learning process continued. If we consider the change of the ACE's weights are calculated: $\Delta\omega_{mj}(t) = \omega_{mj}(t+1) - \omega_{mj}(t)$, the norms of the vector $\Delta\omega_m = (\Delta\omega_{m1}, \cdots, \Delta\omega_{mj}, \cdots, \Delta\omega_{m243})^T$, $\|\Delta\omega_m\|$, is large when there is a collision, and small otherwise. Therefore, it is not viable to determine the convergence using this term. Neither is it reliable to set a large number of learning steps as extensive simulation shows that the EEM still has collision at up to 100 000 steps.

In order to study the impact of W on the learned rule bases, the termination condition was set at 100 000 learning step. Simulation runs were conducted under the same set of parameters but of different W. The impact of W is that a smaller W maps all the five sensor readings to a situation of $\mu_{VN}(d_i) = 0$, $\mu_{NR}(d_i) = 0$, and

TABLE I
PARAMETERS USED FOR SIMULATION

$R = 28cm$	$\Delta T = 0.3$	$\lambda = 0.5$	$b_1 = 15cm/s$	$b_2 = 0$	$\beta = 0.8$	$\alpha = 0.8$
$W = 20cm$	$\delta = 0.85$	$\gamma = 0.95$	$f_1 = 15cm/s$	$f_2 = \pi/2$	$k = 0.2$	$\xi = 1.5$

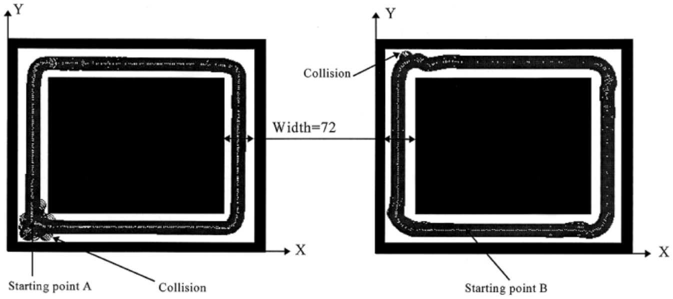

Fig. 7. Learning in a simple environment ($W = 20$ cm).

$\mu_{FR}(d_i) = 1$ more frequently meaning that only $\omega_{m,243}(t)$ of each ASE is updated. On the contrary, a larger W reduces the values of $\mu_{FR}(d_i)$. If any one of them is reduced to zero, the associated rule is not updated and therefore, more blank rules are produced with the increase of the value of W. In an extreme situation in which the value of W causes $\mu_{VN}(d_i) = 1$ for all the five sensor readings throughout the whole training process, then only $\omega_{m,1}(t)$ is learned.

We have also observe that: first, in the case that d_1 and d_3 are FR and $\mu_{FR}(d_2) = \mu_{FR}(d_4) = \mu_{FR}(d_5) = 1$, v is very small and $\Delta\theta$ is large. This action could cause the vehicle to turn around without being able to move in an unobstructed situation. Second, in the case that d_1 and d_3 are VN and $\mu_{NR}(d_2) = \mu_{NR}(d_4) = \mu_{NR}(d_5) = 1$, v is very large and $\Delta\theta$ is nearly zero; also, when d_1 and d_3 are VN and $\mu_{FR}(d_2) = \mu_{FR}(d_4) = \mu_{FR}(d_5) = 1$, v is very large and $\Delta\theta$ is very small. These actions could cause a collision easily. Third, when W increases, the above-mentioned problems improved. The further findings of the simulation are: 1) no matter which W is used, the vehicle frequently goes into trap situation and 2) the learned rules are difference if different start configurations are used. This indicates that the convergence of the learning method is not unique.

2) New Training Method: Instead of using a complex training environment, the new training method was conducted in a simple corridor-like environment as depicted in Fig. 7. As the environment is regular, the vehicle trajectory remains unchanged while the learning process converges, where the criterion for terminating the training process can be based upon. The new training method is divided into two phases according to the moving direction of the vehicle. First, the vehicle begins its training from an arbitrarily chosen start configuration, where the vehicle moves in a clockwise (CW) or counter-clockwise (CCW) direction. In this phase, the learning is iterated as in the EEM. This training phase is completed when the vehicle maintains a constant trajectory without collision. In its second phase, the vehicle then learns to navigate in the opposite direction with a new start configuration. Upon collision, the vehicle backtracks 40 steps and turns an additional steering angle $(-\pi/30)$ if in CW direction or $(\pi/30)$ if in CCW direction. This training phase is completed when the vehicle keeps a constant trajectory without collision.

The set of parameters given in Table I is also used in this simulation. With $W = 20$ cm, it is found that a high number of learned rules (only three rules blank) was obtained when the width of the corridor is 72 cm. From the start configuration A of (60 cm, 70 cm, 0°), after 11 collisions, it navigated successfully in the CCW direction. The CW direction training phase starts at configuration B (226 cm, 60 cm, 90°) from which, after two collisions, the vehicle navigated successfully. The learning process converged at 360 learning steps with 11 collision, while the convergence of the learning process of the EEM method is uncertain at up to 100 000 learning steps with 267 collision. Therefore, the new training method is able to construct the rule base with less cost and less time. If the number of learning steps taken before a collision is used to measure the performance of a learning process. The rules were almost sufficiently learned (1.2% of the rules were blank) in our method after 13 trials. On the other hand, the rules of the EEM were far from sufficiently learned (30% of the rules were blank) up to 76 trials and it collided in less than 6000 steps.

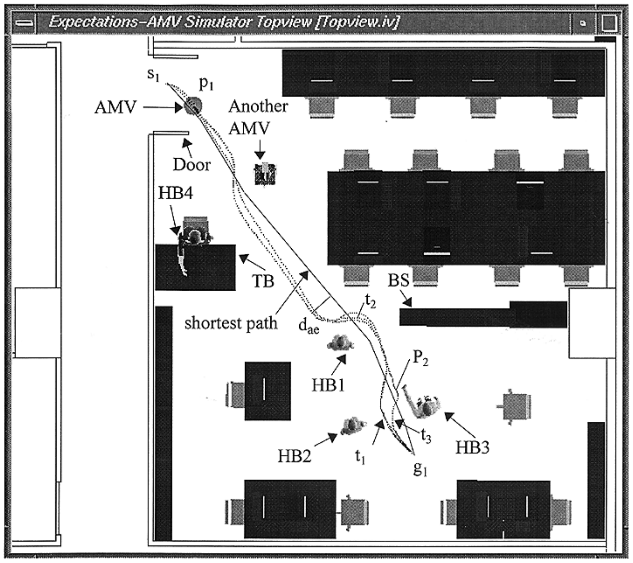

Fig. 8. Top view of a laboratory and the trajectory from s_1 to g_1.

IV. PERFORMANCE ANALYSIS

To further evaluate the performance of the navigator, it was embedded in a fully integrated and interactive simulator developed on the SGI IRIX operating system and the OpenInventor platform. The environment is an indoor floor space including offices and laboratories. Scene objects including tables, chairs, book shelves, human beings, and other mobile vehicles have been constructed according to the true dimensions of these objects and incorporated into it. The simulator displays a top view of the complete environment, and a three-dimensional (3-D) camera view on top of the vehicle. The start and goal configurations for each navigation task can be defined by the mouse keys and/or the keyboard. In Fig. 8, the vehicle trajectory from s_1 to g_1 is shown by the dotted line drawn after the navigation task has completed (t_1). Three cases are compared in Fig. 8, where t_1 and t_2 are the trajectories determined by the vehicle while using the rule base constructed by the new training method with and without the EE, respectively; and t_3 is the trajectory determined by the vehicle while using the rule base constructed by the EEM (terminated at 100 000 learning step with $W = 60$ cm).

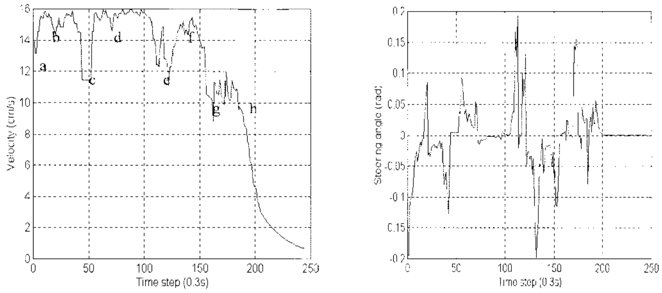

Fig. 9. Velocity and steering angle functions of t_1.

From t_1, it is observed that the velocity and steering angle changes are smooth. To study this particular property, the velocity and steering angle functions are plotted in Fig. 9. At point s_1, the vehicle was at a velocity of about 14 cm/s. When it reached time step "a," the vehicle turned its heading direction slightly toward the goal g_1 with a small drop in velocity. After this turn, it accelerated and passed by the door on its right at "b." Its velocity dropped slightly when it was near the door. Beyond this point, it encountered another AMV at "c," which caused the vehicle to slow down to below 12 cm/s before making a relatively large steering change to avoid the AMV. It then accelerated to top speed when passing the table (TB) at "d." It continued to travel at this speed with little change in its steering angle before encountering the human being (HB1) at "e." The vehicle detected the presence of the human being, decelerated, and steered to the left. It accelerated when passing the bookshelf (BS) at "f," and decelerated when approaching the two human beings at "g." The vehicle slowed down to about 9 cm/s when it was directly in front of HB3, before making a turn to the right, approaching "h." At this point, it selected the path between HB2 and HB3 and navigated through. When the vehicle approached the goal, it decelerated gradually until coming to a stop at point g_1.

From Fig. 9, it can be observed that

1) the range of acceleration/deceleration is small when it passes by an obstacle but large when the obstacles are in its path;
2) there is no abrupt change of velocity (±3 cm/s);
3) there is no abrupt change in the steering angle ($\pm 11.5^\circ$).

These properties have obvious benefit for practical application when the vehicle's dynamics become an important consideration. On t_2, the velocity and steering angle functions are very similar to t_1 even though the EE was not used. As for t_3, it is found that the changes in velocity and steering angle are more abrupt than t_1 and t_2. In this case, the velocity varied between ±6 cm/s and the steering angle varied between +40° and −29°. Although these cannot be considered as large changes, and their impact to the vehicle dynamics cannot be ascertained, smooth and small changes are desired. On an important note, the vehicle collided with HB3 at p_2.

To evaluate the path quality determined by the navigator, the visibility graph method [21] was used to find the shortest path for each navigation task which is shown by the solid line in Fig. 8. At each time step, the deviation of the vehicle's position from the shortest path is denoted by d_{ae}. The length of the actual path and the shortest path are represented by p_a and p_e, respectively, and the relative error between the actual path length and the shortest path length, $(p_a - p_e)/p_e$ is denoted by E_r. On the same floor plan, six navigation tasks were conducted and the errors are tabulated (Table II).

It can be seen that

1) the navigator achieves a path reasonably close to the shortest path;

TABLE II
NAVIGATION UNDER THE RULE BASE CONSTRUCTED BY THE NEW METHOD

T	p_a/cm	p_e/cm	E_r	average d_{ae}	max. d_{ae}	time	obstacles	collision.
1	833.7	785.6	+6.1%	14.8cm	40.1cm	73.8s	7	0
2	795.0	759.8	+4.6%	9.8cm	33.8cm	70.2s	6	0
3	682.5	660.1	+3.4%	7.6cm	28.6cm	62.1s	5	0
4	584.3	573.0	+2.0%	4.3cm	13.2cm	54.6s	4	0
5	493.7	485.0	+1.8%	3.8cm	10.0cm	49.5s	2	0
6	424.1	422.9	+1.2%	3.1cm	8.5cm	45.6s	0	0

TABLE III
NAVIGATION UNDER THE RULE BASE CONSTRUCTED BY THE EEM

T	p_a/cm	p_e/cm	E_r	average d_{ae}	max. d_{ae}	time	obstacles	collision.
1	834.0	785.6	+6.2%	13.1cm	35.1cm	78.0s	7	1
2	794.0	759.8	+4.5%	25.9cm	95.5cm	73.2s	6	1
3	682.0	660.1	+3.3%	7.5cm	26.5cm	64.2s	5	1
4	584.0	573.0	+1.9%	4.1cm	12.9cm	56.4s	4	0
5	493.7	485.0	+1.8%	3.8cm	10.0cm	50.4s	2	0
6	424.1	422.9	+1.2%	3.1cm	8.5cm	45.6s	0	0

2) the less obstacles the vehicle tackled, the closer the path is to the shortest path;
3) the relative error and the path deviation are proportional to the number of obstacles.

For the first point, the largest relative error is only 6.1% for Task 1, whereas the smallest error is 1.2% when there is no obstacle. For the second point, we can see that the relative error decreases as the number of obstacles decreases. For the third point, the trend is clear that the navigated path deviates a lot more from the shortest path if there are more obstacles present in the environment. This is to be expected because of the use of the EE which tends to give larger clearance from the obstacle.

For comparison purpose, the same six navigation tasks were repeated using the rule base constructed by the EEM, and the results are given in Table III. It is observed that collision occurred for the first three tasks, while the remaining three tasks were successful. It is also observed that first, the time consumed in each case is similar to the preceding cases. Second, there is a large deviation (95.5 cm) from the shortest path due to a wrong turn the vehicle made at one point of the path. It subsequently navigated back on course. Third, for the rest of the tasks, the values for p_a, E_r, average d_a, and max. d_{ae} are roughly the same as the preceding cases.

V. CONCLUSION

We have presented a fuzzy navigator that performs well in complex and unknown environments, using a rule base that is learned from a simple corridor-like environment. The principle of the navigator is built on the fusion of the obstacle avoidance and goal seeking behaviors aided by an environment evaluator to tune the universe of discourse of the input sensor readings and enhance its adaptability. For this reason, the navigator has been able to learn extremely quickly in a simple environment, and then operate in an unknown environment, where exploration is not required at all.

Specifically, the rule base for obstacle avoidance has been constructed through reinforcement learning using a cost-effective new training method. The new method is reliable and has no uncertainty of convergence compared with the EEM. In addition, the new training method has five distinct advantages over the existing EEM:

1) 270 times faster in learning speed;
2) only 4% of the learning cost;
3) very reliable convergence of learning;
4) 98.8% of learned rules;
5) high adaptability.

Numerous simulation runs show that this new navigator is characterized by first, it's ability to tackle an unknown environment without having to explore it beforehand or being supervised; second, it's free of local minimum; third, its smooth changes of velocity and steering angle; fourth, its planned path is close to the shortest path; and fifth, it's both nearsighted and farsighted. A comparison of the navigator using the rule base obtained from the new training method and the EEM, shows that the new navigator guarantees a solution when the EEM-based navigation fails and its solution is more acceptable.

Acknowledgment

The 3-D real-time simulator used in this research work was developed by a number of students involved with the EXPECTATIONS project, of which the contributions of F. P. Fong and Y. L. Lee are acknowledged.

References

[1] J. T. Schwartz and M. Shirir, "A survey of motion planning and related geometric algorithm," *Artif. Intell. J.*, vol. 37, pp. 157–169, 1988.

[2] J. F. Canny, *The Complexity of Robot Motion Planning.* Cambridge, MA: MIT Press, 1988.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[4] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 108–120, 1983.

[5] J. Borestein and Y. Koren, "Real-time obstacles avoidance for fast mobile robot," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 1179–1187, Sept./Oct. 1989.

[6] M. J. Schoppers, "Universal plans for reactive robots in unpredictable environments," in *Proc. 10th Int. Joint Conf. Artificial Intelligence,* 1987, pp. 1039–1046.

[7] R. C. Arkins, "Motor schema based navigation for a mobile robot: An approach to programming by behavior," in *Proc. IEEE Int. Conf. Robotics Automation,* 1987, pp. 264–271.

[8] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, pp. 14–23, Feb. 1986.

[9] S. Ishikawa, "A method of indoor mobile robot navigation by using fuzzy control," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robotics Systems,* 1991, pp. 1013–1018.

[10] E. Tunstel, "Mobile robot autonomy via hierarchical fuzzy behavior control," in *Proc. 6th Int. Symp. Robotics Manufacturing,* Montpellier, France, 1996.

[11] K. T. Song and J. C. Tai, "Fuzzy navigation of a mobile robot," in *Proc. of IEEE/RSJ Int. Conf. Intelligent Robotics Systems,* 1992, pp. 621–627.

[12] P. K. Pal and A. Kar, "Mobile robot navigation using a neural net," in *Proc. IEEE Int. Conf. Robotics Automation,* 1995, pp. 1503–1508.

[13] C. Kozakiewwicz and M. Ejiri, "Neural network approach to path planning for two dimension robot motion," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems,* 1991, pp. 818–823.

[14] B. J. A. Krose and J. W. M. Van Dam, "Adaptive state space quantization for reinforcement learning of collision-free navigation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems,* 1992, pp. 1327–1332.

[15] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 464–477, Mar. 1995.

[16] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller-part II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 419–435, 1990.

[17] N. H. C. Yung and C. Ye, "An intelligent navigator for mobile vehicles," in *Proc. Int. Conf. Neural Information Processing,* 1996, pp. 948–953.

[18] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. 13, no. 5, pp. 834–846, 1983.

[19] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.

[20] S. B. Thrun, "The role of exploration in learning control," in *Handbook of Intelligent Control.* New York: Van Nostrand Reinhold, 1992, pp. 527–559.

[21] J. C. Latombe, *Robot Motion Planning.* Norwell, MA: Kluwer, 1991.

Network-Based Approach to Online Cursive Script Recognition

Bong-Kee Sin, Jin-Yong Ha, Se-Chang Oh, and Jin H. Kim

***Abstract*—The idea of combining the network of HMM's and the dynamic programming-based search is highly relevant to online handwriting recognition. The word model of HMM network can be systematically constructed by concatenating letter and ligature HMM's while sharing common ones. Character recognition in such a network can be defined as the task of best aligning a given input sequence to the best path in the network. One distinguishing feature of the approach is that letter segmentation is obtained simultaneously with recognition but no extra-computation is required.**

***Index Terms*—Cursive script, hidden Markov model, ligature, network search, online character recognition, segmentation, Viterbi algorithm.**

I. Introduction

In spite of the difficulties due to variability and ambiguity of unconstrained handwriting, attempts to automatic recognition are now emerging armed with mature technologies such as neural network [1], [2], hidden Markov model (HMM) [3]–[5], or other structural or mathematical modeling techniques [6]. Hybrid methods also exist where individual approaches complement each other so that even higher performance can be achieved [7]. Many researchers report promising performance with sophisticated architectures. However, they usually suffer from the weakness in their framework for modeling cursive patterns or are limited to tasks of small vocabulary. And systems are not rare that demand external segmentation in input stream, i.e., supplying a set of candidate letter boundaries prior to shape classification [8].

The primary focus of the paper is the word recognition and segmentation method which works over a network of HMM's. In such a network an individual HMM models a letter or a ligature. We consider ligatures as separate entities just like letters and designed a set of ligature HMM's. The two kinds of HMM's are then combined to form a finite state network, a word model.

We have designed two script models, one for Korean Hangul characters and the other for English words [5]. The experimental recognizers based on the network models use a chain of direction codes that describes the complete locus of stylus from the first pen-down, including intermediate pen-up loci, to the last pen-up of a word. Based on this coding scheme, the networks can model both or mixed run-on and cursive words.

In the network-based approach, the character recognition is defined as the task of aligning the observation sequence optimally to the best path, which can be performed by using a dynamic programming algorithm. A great advantage of the method is that the recognizer performs letter segmentation simultaneously with recognition by fast backtracking of the best alignment.

Manuscript received October 8, 1994; revised June 7, 1997. This paper was recommended by Associate Editor M. S. Obaidat.

B.-K. Sin is with the Information Retrieval Team, Multimedia Research Labs, Korea Telecom, Seoul, Korea (e-mail: bkshin@pine.kotel.co.kr).

J.-Y. Ha is with the Computer Science Department, Kangwon University, Chunchon, Kangwon, Korea (e-mail: jyha@cc.kangwon.ac.kr).

S.-C. Oh is with the Information Technology Laboratory, LG Electronics Research Center, Seoul, Korea.

J. H. Kim is with the Computer Science Department, Korea Advanced Institute of Science and Technology, Taejon, Korea (e-mail: jkim@cs.kaist.ac.kr).

Publisher Item Identifier S 1083-4419(99)00911-5.