An Intelligent Robotic System Based on a Fuzzy Approach

TOSHIO FUKUDA, FELLOW, IEEE, AND NAOYUKI KUBOTA, ASSOCIATE MEMBER, IEEE

Invited Paper

This paper deals with a fuzzy-based intelligent robotic system that requires various capabilities normally associated with intelligence. It acquires skills and knowledge through interaction with a dynamic environment. Recently, subsumption architectures, behavior-based artificial intelligence, and behavioral engineering for robotic systems have been discussed as new technologies for intelligent robotic systems. This paper proposes a robotic system with "structured intelligence." We focus on a mobile robotic system with a fuzzy controller and propose a sensory network that allows the robot to perceive its environment. An evolutionary approach improves the robot's performance. Furthermore, we discuss the effectiveness of the proposed method through computer simulations of collision avoidance and path-planning problems.

Keywords**—***Evolutionary systems, fuzzy systems, intelligent systems, path planning, robotics.*

I. Introduction

Fuzzy logic offered by Zadeh [1] has been applied in various areas such as knowledge engineering, computer science, manufacturing systems, mechatronics, and robotics [2]–[6]. Furthermore, various fuzzy inference methods have been proposed to deal with imprecise information, human language, and nonlinear mappings [7]–[16]. With the progress of computation capabilities, techniques within the field of artificial intelligence (AI) have been developed in order to describe and build intelligent agents that perceive and sense the environment, make appropriate decisions, and take actions [17]. Recently, human intelligence and life itself have been studied in cognitive science, soft computing, artificial life (A-life), and computational intelligence (CI), in addition to AI [2]–[5], [17]–[20]. Bezdek [20] discussed intelligence from three levels: artificial; biological; and computational. In the strictest sense, CI depends on numerical data and does not rely on explicit knowledge. Furthermore, Eberhart [51] defined CI as a methodology that involves computing. It aims to construct intelligence from the viewpoints of biology, evolution, and self organization [21]. CI tries to construct intelligence by internal description, whereas classical AI tries to construct intelligence from external (explicit) description.

These intelligent methodologies have often been used for robotic systems in areas that are critical and dangerous for human beings [22]–[25]. To accomplish a given task, a robot collects or receives sensory information concerning its external environment and takes actions within the dynamically changing environment. Both the sensing system and control rules are often dictated by human operators, but ideally the robot should automatically perform the given tasks without assistance. Consequently, the robot must be able to perceive the environment, make decisions, represent sensed data, acquire knowledge, and infer rules concerning the environment. Robots that can acquire and usefully apply knowledge or skill are often called intelligent. Intelligence emerges from the close linkage of many capabilities.

An intelligent robot receives a task from a human operator and must accomplish the task in the available work space, which may include many obstacles such as people, machining centers, and other robots. The intelligent robot should therefore take into account the problem of collision avoidance. Furthermore, the intelligent robot should automatically generate its motion for performing the task. Motion planning fundamentally includes path, trajectory, and task planning [25]. Here we define these planning problems as follows. The path-planning problem requires generating the shortest path for the robot from a given starting point to a target point while satisfying the spatial constraints. The trajectory-planning problem requires generating a robot trajectory that satisfies the time constraints. The task-planning problem requires finding a sequence of primitive motion commands for solving a given task. These commands are described by robot programming languages

Manuscript received March 30, 1998; revised April 12, 1999.

T. Fukuda is with the Center for Cooperative Research in Advanced Science and Technology, the Department of Mechano-Informatics and Systems and the Department of Micro System Engineering, Nagoya University, Nagoya 464-8603 Japan (e-mail: fukuda@mein.nagoya-u.ac.jp).

N. Kubota is with the Department of Mechanical Engineering, Osaka Institute of Technology, Osaka 535-8585 Japan (e-mail: kubota@med.oit.ac.jp).

Publisher Item Identifier S 0018-9219(99)06907-8.

[23]. A task-level programming system allows a user to describe the task in a high-level language (e.g. natural languages). For example, "GO TO target_point" would be used to describe the task given to a mobile robot. To accomplish the task, the mobile robot performs spatial (geometric) map building and path planning.

Path planning and collision avoidance are closely related. The search space (environmental map) can be built by polygonal approximation, cell decomposition, and artificial potential fields [38]–[41]. In polygonal approximation, obstacles in the work space are approximated as polygons on a map. This simplifies the search space. In cell decomposition, a two-dimensional work space is divided into $N \times N$ cells to reduce the search space. In the artificial potential field method, a robot manipulator moves based on an attractive force from the goal point and a repulsive force from the obstacles in the work space [25], [41]. In the hybrid method of the cell decomposition and artificial potential fields, each cell on the divided cell space is assigned a pseudopotential value based on the distance from the obstacles in the work space [25], [38].

Evolutionary algorithms have been applied to path-planning problems [25], [28]–[31], [37], [38], [60]. Xiao *et al.* [60] proposed an adaptive evolutionary planner/navigator using various operators to evolve and improve candidate paths. The motion of a robot is constrained fundamentally by its dynamics, kinematics, and work space. The trajectory planning must satisfy the constraints, such as bounded velocity and acceleration, maximal torque, and also task dependencies. Therefore, path planning should take into account these constraints in trajectory planning. However, these methodologies are based on the map constructed *a priori* and are suitable for offline path planning, but it is difficult to build a map *a priori* because of constraints that may include moving obstacles [38]. Furthermore, in such a case, the control rules of mobile robots must be designed to handle any environmental conditions beforehand. Nowadays, various adaptive robotic systems have been proposed to adapt under dynamic or unknown environments [17], [43]–[48], [50].

Brooks [43] proposed a subsumption architecture, and later a behavior-based artificial intelligence (BBAI) for robotics [50]. The subsumption architecture uses a layered finite state machine to represent the agent. The design is decomposed into behaviors such as obstacle avoiding, photo tracing, and wall following. In addition, reinforcement learning methods such as the bucket brigade algorithm, the temporal difference method, and Q-learning have been discussed as methods for inducing rules of behavior through the interaction with environment [34], [44]–[48]. This kind of BBAI stresses the importance of the direct interaction between robot and environment. In contrast, classic AI is based on the representation and manipulation of explicit knowledge. However, the behavior-based approach needs to design a new controller for each task [17]. Recently, other methods described as behavior analysis and training, the MonaLysa architecture, model-based learning, hierarchical intelligent control, and others, have been proposed [44]–[48]. The MonaLysa architecture relies on a hierarchical classifier system including reactive and planning modules [45]. In the reactive module, a simulated robot has many possible rules and decides what action to perform next. A given task is then logically decomposed into a series of subtasks in the planning module. On the other hand, hierarchical intelligent control for robotic systems comprises: 1) a learning level of knowledge; 2) a skill level based on fuzzy rules; and 3) an adaptation level to adjust rules according to environmental conditions [35]. Each level has a feedback loop for information processing between levels.

The intelligence of a robotic system depends on the structure of hardware and software for processing information, i.e., structure influences the potential intelligence. Furthermore, the intelligence of the robotic system can evolve through learning and adaptation in dynamic environments. This paper proposes a robotic system with structured intelligence [21], [49] that emphasizes the importance of the entire integrated system architecture. Based on perceptual information, a robotic system with structured intelligence makes decisions and actions deriving from four different levels in parallel. In addition, the robot generates its motion through interaction with its environment and at the same time acquires its skill by learning. To realize the structured intelligence on robotic systems, we synthetically apply fuzzy systems (FS), neural networks (NN's), and evolutionary computation (EC).

This paper is organized as follows. In Section II, we describe the synthetic methodologies of FS, NN's, and EC. Furthermore, we propose the implementation of structured intelligence for robotic systems. Section III offers a mobile robotic system based on a fuzzy controller and describes the learning method. Section IV introduces evolutionary learning for the mobile robotic system based on the fuzzy controller and shows the results from simulated collision avoidance problems. Section V proposes a path-planning method for the mobile robot with a fuzzy controller and shows results of simulated path planning. Finally, Section VI concludes the paper and suggests future work.

II. CI and Robotic Systems

A. Emerging Synthesis of FS, NN's, and EC

A synthesis of various techniques is required to build a highly intelligent system. Fig. 1 shows the synthesis of NN's, FS, and EC. Each technique plays a specific role in intelligent systems. NN's are useful for recognizing patterns, classifying input, and adapting to dynamic environments by learning. Unfortunately, the internal mapping structure of an NN is a black box. The resulting behavior is therefore too difficult to explain. FS can cope easily with human knowledge and can be used to perform inference, but FS do not fundamentally incorporate any learning mechanism. Neurofuzzy computing has been developed to overcome this disadvantage [2]–[5]. In general, the NN is used for learning, while the fuzzy logic is used for representing knowledge. The learning is performed via an

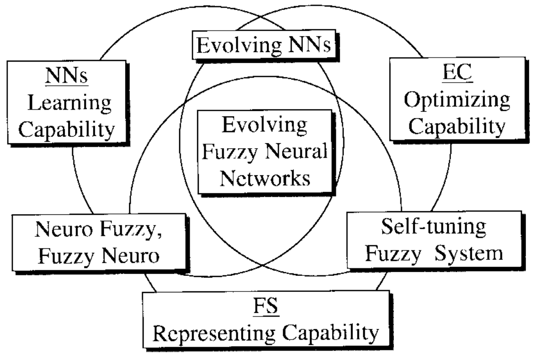


Fig. 1. Emerging synthesis of NN's, FS, and EC. Each technique plays a specific role in intelligent systems. These techniques are combined to realize greater intelligence.

incremental learning, back propagation, or the delta rule based on error functions (i.e., a measure of misclassification). EC can also tune NN's and FS. Furthermore, EC has been used for optimizing the structure of NN's and FS [2]–[4].

The goal is for an intelligent system to quickly adapt to a dynamically changing environment by using NN's and FS via a back propagation method or the delta rule. Further, the structure of the intelligent system can globally evolve by EC, leading to a more intelligent system.

B. Structured Intelligence for a Robotic System

People make decisions and take actions based on sensed information and their own internal states. In addition, people can learn by acquiring or perceiving information concerning rewards or penalties for different behaviors. Thus, we repeat such perception, decision making, and action. These behaviors are deeply dependent on information processing related to intelligence. Key technologies for intelligence are knowledge representation, inference, search, planning, learning, prediction, and so on [17]. Intelligence emerges from the synthesis of these system capabilities. Therefore, we should consider an entire structure of intelligence for processing information "flowing" over the hardware and software. We have proposed a structured intelligence with three features: 1) neurodynamics; 2) skill; and 3) recursive "consciousness" [49], [58] (Fig. 2). A skill is defined as generalized knowledge and motion. To represent skill, we apply a fuzzy inference system as the logical information processing [2]–[5]. Consequently, skill is the explicit representation of human intelligence. We cannot, however, adapt to dynamic environments by skills. Instead, we use intuitive information processing and induction, but it is difficult to realize this on a computer. Neurodynamics can be applied to represent intuitive information processing. The nonlinear property of NN's has been used for representing various relationships to be learned [2]–[5], [18]. We assume NN's make intuitive outputs based on the learned relationship. However, the intuitive inference sometimes makes ineffective outputs. To treat ineffective outputs, we need a controller, i.e., recursive "consciousness" (see Fig. 2). EC is used as the recursive consciousness that combines several outputs and generates a

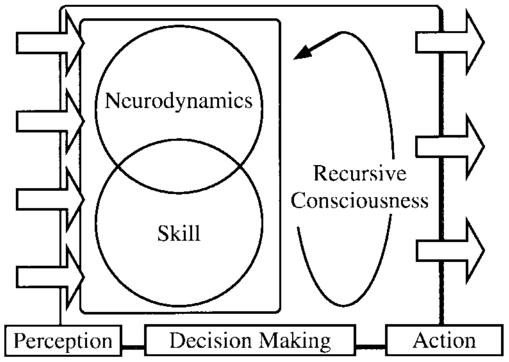


Fig. 2. Architecture of structured intelligence. The structured intelligence is composed of three parts: neurodynamics (NN); skills (FS); and recursive consciousness (EC). The EC as the recursive consciousness makes valid decisions by combining the outputs of NN and FS.

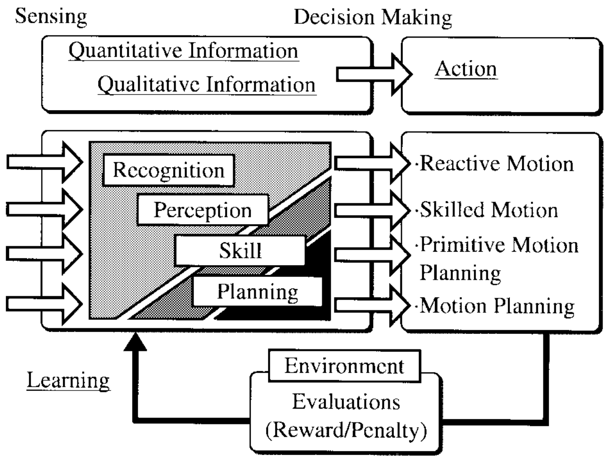


Fig. 3. The architecture of a robotic system with structured intelligence. The robotic system makes decisions according to information sensed from its environment. After taking actions, the robotic system receives an evaluation (reward or penalty). The decision-making rules are trained by the learning mechanism.

new output because the individuals of candidate outputs can evolve through interaction with the environment. Therefore, the structured intelligence has candidate solutions of outputs generated by NN and FS and generates valid outputs by EC [49], [58].

Fig. 3 shows a robotic system with structured intelligence. Based on information perceived from the environment, the robot makes decisions and takes actions from four levels in parallel. A sensory network is applied to perceive the environment (Fig. 4). The robot recognizes quantitative information of the environment. Next, the robot perceives its external environment through the interpretation of selective attention into qualitative information by sensor fusion/integration and focus/release. This action comprises a reactive motion (reflex), skilled motion, primitive motion planning, and motion planning. When the robot recognizes dangerous quantitative information from the environment, it makes a reactive motion without exact decision making. At the skilled motion level, the robot recognizes the state of its environment and selects a fundamental motion such as locomotion, tracing, or running. The robot must generate its suitable motion under different environmental conditions.

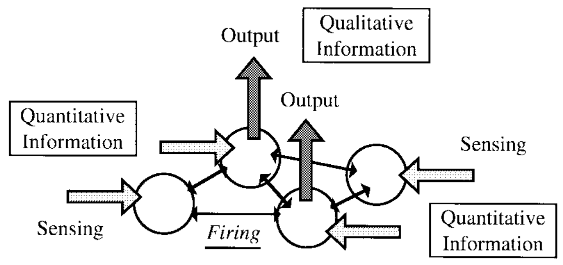


Fig. 4. A sensory network for perceiving environment. According to the internal state of the robotic system, the sensed information is interpreted into qualitative information.

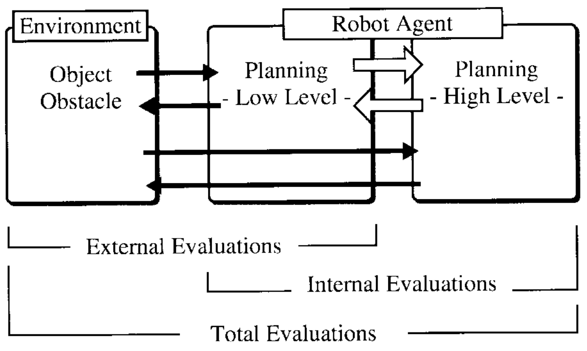


Fig. 5. External and internal evaluations for motion planning. The external evaluation is received directly from the environment and the internal evaluation quantifies the state of robotic system based on external evaluations.

If it can simply combine the skills or motions already acquired, it can generate its new motion by combining those skills or motions [58]. However, if it cannot apply the acquired motions and skills, the robot then must generate new motions based on inverse kinematics and dynamics. Thus, a robot has no skill initially but gradually acquires skills and motions through interaction with the environment.

The robot requires external and internal evaluations to acquire these skills (Fig. 5). An external evaluation is received directly from the environment and used for generating primitive motions in low-level planning. The internal evaluation quantifies the state of robotic system based on external evaluations received in low-level planning. High-level planning is based on internal evaluations that combine the primitive motions generated by low-level planning [49], [57]. Furthermore, a complicated behavior can be generated by combining a series of primitive behaviors. In this way, the robot can acquire skills and motions based on the internal and external evaluations through interaction with the environment. Next, we consider how to realize the structured intelligence for a robotic system with a fuzzy controller.

III. A Mobile Robotic System Based on a Fuzzy Controller

A. Target Trace and Collision Avoidance for a Mobile Robotic System

A mobile robot may be required to move to a target point while avoiding obstacles in a work space. This requires information regarding the distance between the robot and

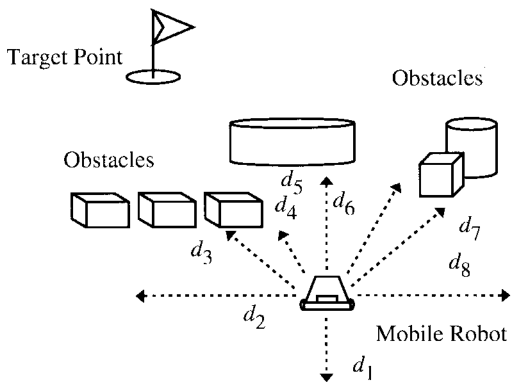


Fig. 6. Sensing direction d_i for detecting obstacles. The mobile robot has eight range sensors for measuring distance to the obstacles.

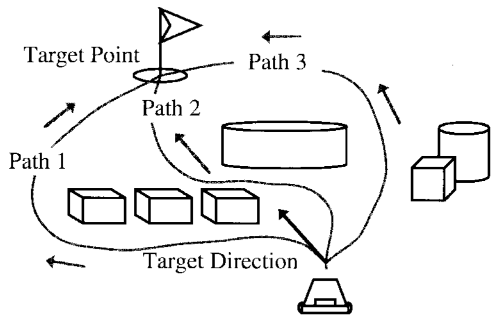


Fig. 7. Candidate paths avoiding collision with obstacles. In this case the mobile robot must pass through a narrow area if it selects a target direction reactively.

the obstacles in each sensing direction as input variables for collision avoidance (Fig. 6). It also requires the direction toward the target point as input variables for target trace (Fig. 7). We consider an example of planning the shortest path in the work space of Fig. 7. At least three types of path candidates can be generated and the direct target direction is included in path 2. The best path depends on the robot's hardware and software performance. If the robot has very precise sensors, actuators, and well-designed control rules, it may easily reach the target point via path 2 without colliding with any obstacles. Otherwise, it may collide with obstacles along the path, i.e., path 2 is narrow and confined. Since it is difficult to change the robot's hardware to meet the posed environmental conditions, designing a suitable control rule for the robot hardware in such an environment is very important. In addition, the mobile robot cannot receive rewards or penalties for its behaviors without reaching the target point. Therefore, we divide the control rules into two behaviors: collision avoidance and target trace, much like a subsumption architecture [43]. The design of these behaviors follows closely with the environmental conditions. We should take into account controlling the velocity in light of the density of the obstacles in the work space. If there are many obstacles, the mobile robot should move more slowly to the target point while avoiding collision. In contrast, if there are few obstacles, it can move easily and unrestricted to the target point. The mobile robot should concentrate on the surrounding area if there are many nearby obstacles, but

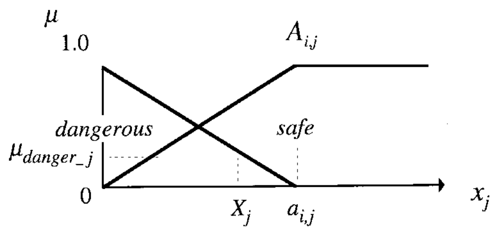


Fig. 8. A triangular membership function concerning the distance x_j between the mobile robot and obstacles in the ith rule. μ_{danger_j} is the grade (degree) of danger and $a_{i,j}$ is central value of a membership function $A_{i,j}$ corresponding to safe, respectively.

it should otherwise pay attention to the distant area. We therefore apply a sensory network with a scalable attention range. When we assume the scalability of control rules, the sensory network can change the output of the controller according to the time series of sensed information. In the following sections, we describe a fuzzy controller and the sensory network for this collision avoidance behavior.

B. Collision Avoidance by a Fuzzy Controller

We apply a simplified fuzzy inference system to a fuzzy controller for collision avoidance because of its simple architecture and low computational cost. In general, a fuzzy if–then rule using the simplified fuzzy inference method is described as follows [4], [5]:

IF x_1 is $A_{i,1}$ and ... and x_j is $A_{i,j}$
and ... and x_n is $A_{i,n}$
THEN y_1 is $w_{i,1}$ ... and y_j is $w_{i,j}$
and ... and y_o is $w_{i,o}$

where $A_{i,j}$ is a membership function for the jth input of the ith rule, $w_{i,j}$ is a singleton for the jth output of the ith rule, and n, o, and r are the numbers of inputs, outputs, and rules, respectively. There are various types of membership functions such as triangular, trapezoidal, and Gaussian [4]–[6]. Here we use triangular membership functions in order to reduce computational time as well as the parameter settings of the fuzzy controller. To simplify, two linguistic values of dangerous and safe are used to represent the degree of danger as a function of the distance as the jth input x_j (Fig. 8). Here, μ_{danger_j} corresponds to the degree of danger. The collision avoidance behavior of the mobile robot should take actions that reduce the value of μ_{danger_j}. A triangular membership function is generally described as

$$\mu_{Ai,j}(x_j) = \begin{cases} 1 - \frac{|x_j - a_{i,j}|}{b_{i,j}}, & |x_j - a_{i,j}| \le b_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $a_{i,j}$ and $b_{i,j}$ are the central value and the width of the membership function, $A_{i,j}$, respectively. Consequently, the activation degree (firing strength) of the ith rule $(i = 1, 2, \ldots, r)$ is calculated by

$$\mu_i = \prod_{j=1}^{n} \mu_{Ai,j}(x_j). \tag{2}$$

Next, we obtain the jth resulting output $(j = 1, 2, \ldots, o)$ by a weighted average as follows:

$$y_j = \frac{\sum_{i=1}^{r} \mu_i \cdot w_{i,j}}{\sum_{i=1}^{r} \mu_i}. \tag{3}$$

This simplified fuzzy inference system can be regarded as an adaptive fuzzy neural network [5]. When y_j^* is the jth target output $(j = 1, 2, \ldots, o)$ of the controlled system in online learning, the error function is defined as

$$E_j = \frac{1}{2}(y_j^* - y_j)^2. \tag{4}$$

To construct a fuzzy controller, we must minimize the error function E_j. When the condition parts (membership functions) are fixed, we can easily train the output, $w_{k,j}$, of the kth rule according to the delta rule based on the error function. The partial derivative of E with respect to $w_{k,j}$ is as follows:

$$\frac{\partial E}{\partial w_{k,j}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_{k,j}} = -\left(y_j^* - \frac{\sum \mu_i \cdot w_{i,j}}{\sum \mu_i}\right)\frac{\mu_k}{\sum \mu_i}. \tag{5}$$

Consequently, we can update the jth output value of the consequence of the kth rule by the following equation:

$$\begin{aligned} w_{k,j}(t+1) &= w_{k,j}(t) - \tau \cdot \frac{\partial E}{\partial w_{k,j}} \\ &= w_{k,j}(t) + \tau \cdot \left(y_j^* - \frac{\sum \mu_i \cdot w_{i,j}}{\sum \mu_i}\right)\frac{\mu_k}{\sum \mu_i} \end{aligned} \tag{6}$$

where τ is the learning rate satisfying $0 < \tau < 1.0$, and t is the learning iteration.

Next, we consider the output variables for the mobile robot. The steering angle $\Delta\theta(t)$ and velocity $v(t)$ are used as output variables of the fuzzy controller for collision avoidance. When correct exemplar data are presented to the fuzzy controller, it can be trained using the delta rule (6). However, because the mobile robot has no global map of the unknown work space, it is difficult to generate such global data. Therefore, the mobile robot calculates local "teaching" data by evaluating the measured distance to obstacles in online learning. A heuristic rule is proposed for the collision avoidance problem in order to select a local moving direction toward collision-free space. Consequently, the mobile robot searches relatively safe space according to the measured distance and then selects the safe moving direction. The state evaluation of the sensing direction, d_i, should include the measured distance of neighboring sensors (Fig. 9). The state of the sensing direction is evaluated by the following equation:

$$\text{state\_}d_i = \sum_{j=1}^{5} Wd_j \cdot x_{i+j-3} \quad i = 1, 2, \ldots, n \tag{7}$$

where Wd_i is a weight coefficient (see Fig. 9). These weight coefficients are predefined as $Wd_i = \{0.1, 0.3, 1.0, 0.3, 0.1\}$. The evaluation value for a direction d_i becomes greater as the robot increases its distance from the obstacles. The outputs of fuzzy controller are trained using the delta rule based on the direction with the highest evaluation value.

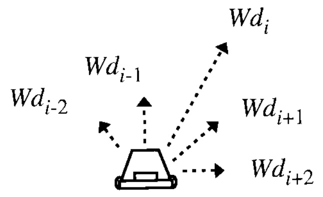


Fig. 9. State evaluation of the sensing direction d_i by weight coefficients: $Wd_i = \{0.1, 0.3, 1.0, 0.3, 0.1\}$. The collision-free direction is evaluated by the weighted average.

C. Sensory Network for the Mobile Robotic System

A fuzzy controller for collision avoidance can be trained by using the teaching data that are suitable for a given environmental condition. However, if the fuzzy rules are not generalized, the fuzzy controller must be retrained when the environmental conditions change. The fuzzy rules that work well under various environmental conditions can be described exactly, but the number of fuzzy rules can be very large. To construct compact and useful fuzzy rules, we use a sensory network with scalable attention ranges, which adjust the shape of membership functions. As mentioned before, we should modify the control rules in accordance with the density of obstacles in the work space, i.e., the attention range and velocity of the mobile robot should be changed according to the density of the obstacles. The attention range corresponds to $a_{i,j}$ of the membership function in fuzzy rules. The attention range $A\_\mathrm{rng}(t)$ is changed as follows:

$$A\_\mathrm{rng}(t) = \mathrm{sprs}(t) \cdot S\_\mathrm{rng} \tag{8}$$

$$\mathrm{sprs}(t+1) = \begin{cases} \gamma^{-1} \cdot \mathrm{sprs}(t), & \text{if all } x_i \geq A\_\mathrm{rng}(t) \\ \gamma \cdot \mathrm{sprs}(t), & \text{otherwise} \end{cases} \tag{9}$$

where $\mathrm{sprs}(t)$ is the degree of sparseness of obstacles satisfying $0 < \mathrm{sprs}_{\min} \leq \mathrm{sprs}(t) \leq 1.0$ for perception of the work space, S_rng is the maximal sensing range, and γ $(0 < \gamma < 1.0)$ is a perception coefficient. Fig. 10 shows the membership functions corresponding to the maximal and minimal attention ranges. Even if the input data x_1 and x_2 are the same, the resulting outputs μ_1 and μ_2 differ in the scaled membership functions of Fig. 10. Consequently, the internal state for the perception is updated by the time series of sensed information recursively. In the simplified fuzzy inference for the collision avoidance, x_i is regarded as $A\_\mathrm{rng}(t)$ if x_i is larger than $A\_\mathrm{rng}(t)$. Furthermore, the velocity $v(t)$ of the inference result is also scaled by the internal state of the perception

$$V(t) = \mathrm{sprs}(t) \cdot v(t). \tag{10}$$

$V(t)$ is used for the actual speed control of the mobile robot. On the other hand, $\Delta\theta(t)$ of the inference result is used without scaling. When $\mathrm{sprs}(t)$ is small, the actual speed is reduced and the mobile robot is capable of small sharp turn.

Next, we consider control rules for the target trace behavior. The control rules for velocity and steering angle of the mobile robot are

$$v(t) = f_v(\theta_T(t) - \theta(t)) \tag{11}$$

$$\Delta\theta(t) = f_\theta(\theta_T(t) - \theta(t)) \tag{12}$$

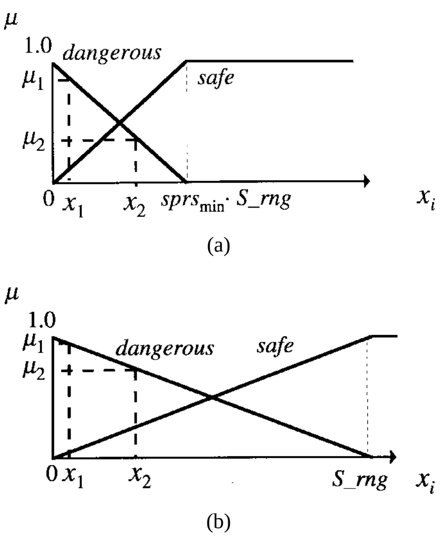


Fig. 10. Membership functions corresponding to linguistic variables of dangerous and safe based on the attention range. The membership functions are adjusted according to the time series of sensed information concerning the sparseness of obstacles in a work space. Even if the input data (x_1 and x_2) are same, the resulting outputs μ_1 and μ_2 are different in the scaled membership functions of (a) minimal attention range and (b) maximal attention range.

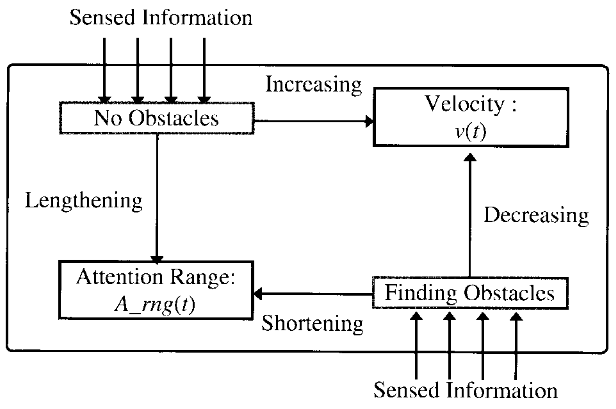


Fig. 11. Adjusting mechanism of sensory network in the collision avoidance. The velocity and attention range are adjusted according to the sensed information.

where f_v and f_θ are control functions based on the error between the moving direction $\theta(t)$ and target direction $\theta_T(t)$. Here we use a simple increasing function for the steering control and a simple decreasing function for the speed control with increasing error. The velocity $v(t)$ for target trace is also scaled by (10). Accordingly, the mobile robot moves to the target point without slowdown if it does not detect obstacles in the work space. After collision avoidance, it slowly moves to the target point because of the small $\mathrm{sprs}(t)$. Therefore, the mobile robot can quickly turn to the target direction in a small work area with a slow speed after the collision avoidance.

The relationship in the sensory network is shown conceptually in Fig. 11. The translation from sensed information to linguistic variables depends on the time series of sensed information recursively, i.e., the parameters for the perception are self scaled. The sensory network adjusts the outputs of the inference results according to the sensed

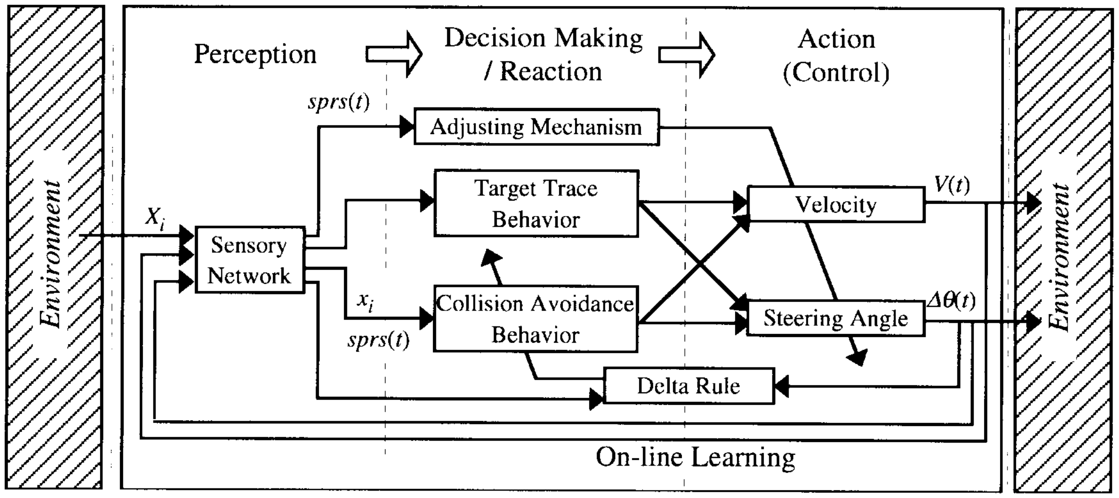


Fig. 12. The architecture of perception, decision making, and action for the mobile robot. The sensory network receives the distance x_i to obstacles in the sensing direction d_i and perceives the state of its environment as the degree of sparseness $\mathrm{sprs}(t)$ of obstacles. According to x_i and $\mathrm{sprs}(t)$, the mobile robot decides the velocity $V(t)$ and $\Delta\theta(t)$ using the fuzzy rules for collision avoidance, the control rules for the target trace, and the adjusting mechanism.

information. Fig. 12 shows the perception, decision making, action, and online learning mechanisms for the mobile robot. Based on the sensed information, the mobile robot performs the collision avoidance or target trace behavior with the adjusting mechanism. Furthermore, the fuzzy rules for the collision avoidance are trained locally by heuristic evaluation.

IV. Evolution of the Mobile Robotic System

Section III described an online learning mechanism for fuzzy controllers. In addition to the delta rule, various self-tuning methods have been proposed such as a learning fuzzy controller with radial basis functions, a fuzzy logic controller using EC for deciding the shapes of membership functions and fuzzy rules, and incremental learning methods based on the error function [6]–[16]. However, the learning capability and approximation accuracy depend largely on the number and shape of the membership functions. A fuzzy inference system that is composed of many membership functions and fuzzy rules has a corresponding high learning capability (it can learn diverse behaviors), but at the same time, it often includes some redundant membership functions and fuzzy rules. Basically, the number of fuzzy rules is the nth power of the number of membership functions for each input variable in order to cover all of the input space when the number of input variables is n. The number of rules increases exponentially with the number of the input variables. We must pay considerable attention to determining the structure of the membership functions, but it is very difficult to determine their structure before learning. Therefore, we apply a self-tuning method based on EC to construct the fuzzy controller for a mobile robot [16].

A. Representation of Candidate Solutions

In this section, we consider the optimization problem of fuzzy controllers as numerical and combinatorial optimiza-

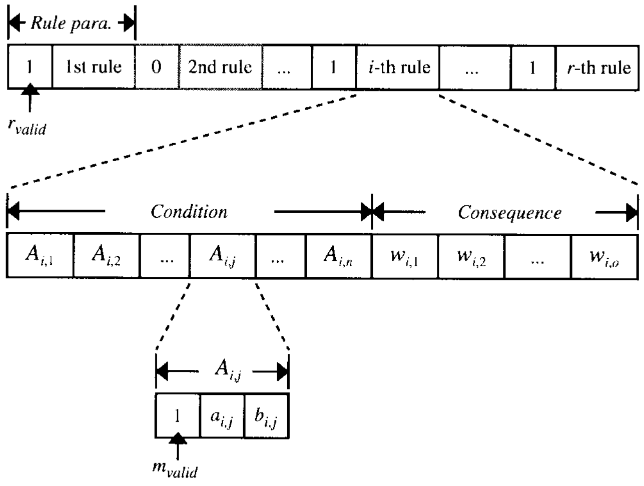


Fig. 13. A candidate solution representing fuzzy rules. A fuzzy rule is composed of condition and consequence parts. A membership function $A_{i,j}$ is represented by the central value $a_{i,j}$ and the width $b_{i,j}$. In the consequence part, a singleton $w_{i,j}$ is used as each output parameter.

tion problems. The optimization problem has the decision variables of the shape of membership functions and the combinations of membership functions and fuzzy rules. We use a triangular membership function (1). Consequently, the decision variables are $a_{i,j}$, $b_{i,j}$, and $w_{i,k}$ where the maximal number of fuzzy rules, the numbers of input variables, and output variables, are r, n, and o, respectively. Furthermore, we must determine the combination of input variables of the condition part in each fuzzy rule. Based on these optimization parameters, a set of fuzzy rules as a candidate solution is represented as in Fig. 13. The combination of fuzzy rules is determined by the validity of each rule (r_{valid}) and the combination of membership functions is determined by the validity of each membership function (m_{valid}). By changing these validity parameters,

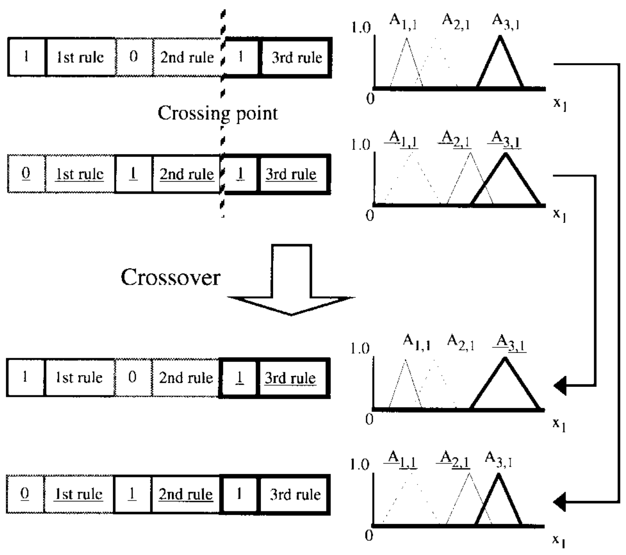


Fig. 14. A one-point crossover operation. In this example, the membership functions of x_1 in the third rules are exchanged between two candidate solutions.

we can obtain the optimal combination, thereby reducing the numbers of fuzzy rules and membership functions while maintaining the inference performance. The string length of a candidate solution is $(3 \times n + o + 1) \times r$. A population is composed of candidate solutions and evolves toward optimal fuzzy rules through genetic operators and selection. Next, we consider crossover and mutation for search.

B. Genetic Operators

Genetic operators generate new candidate solutions. Crossover exchanges the combination of fuzzy rules and membership functions between two candidate solutions. Fig. 14 shows an example of one-point crossover acting on fuzzy rules. Membership functions corresponding to the input variablex_1 in each rule are depicted in Fig. 14, but other membership functions and output variables are also exchanged between two candidate solutions. Other crossover operators such as multipoint crossover and uniform crossover are often used for exchanging information between candidate solutions [53]–[55].

In the optimization problem, two types of combinatorial mutation operations can be used for the search. The first one is to change the validity parameters for fuzzy rules and membership functions. The simple mutation for the validity parameters is shown in Fig. 15. The validity of the second rule is changed from one to zero. The other is to exchange membership functions between two fuzzy rules in a candidate solution (called exchanging mutation). Fig. 16 shows an example of an exchanging mutation. Here, $A_{2,1}$ and $A_{3,1}$ are exchanged between the second and third fuzzy rules. Thus, crossover and mutation operators can change the global and local combinations of fuzzy rules and membership functions, respectively.

Next, numerical mutation is used for optimizing the shape of the membership functions and output variables. Various types of numerical mutation operators have been proposed

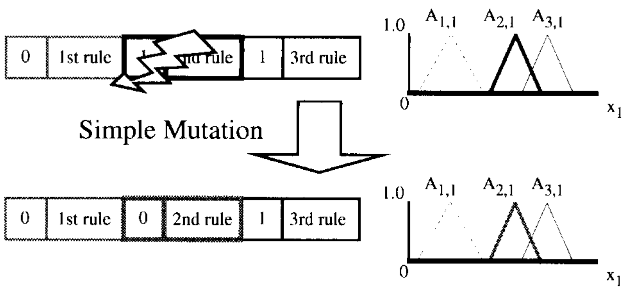


Fig. 15. A simple mutation. The validity of the second rule is changed from one to zero.

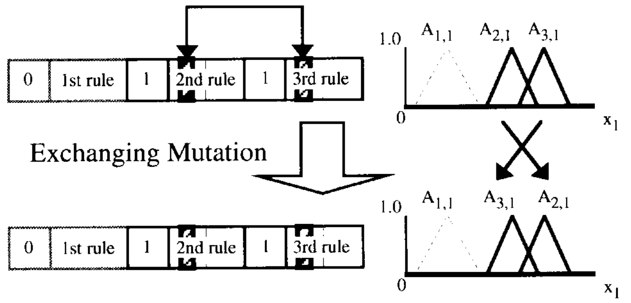


Fig. 16. An exchanging mutation. The membership functions are exchanged between fuzzy rules.

[53]. One is a standard numerical mutation using normal random values as follows:

$$a_{i,j}(t+1) = a_{i,j}(t) + N(0, \sigma^2) \tag{13}$$

where t is the generation (iteration times) and σ^2 is variance. This is the most basic numerical mutation, but the variance is fixed *a priori*. To overcome this problem, an adaptive mutation has been proposed [53]. The adaptive procedure changes the variance according to the fitness record. Rechenberg [53] suggested the following.

> The ratio of successful mutations to all mutations should be 1/5. If this ratio is greater than 1/5, increase the variance; if it is less, decrease the variance.

This ratio has been often discussed in previous studies [53], but the adaptive mutation can change the variance of the normal random mutation according to the success ratio based on the landscape of an arbitrary fitness function. While this adaptive mutation refers to its own fitness record, an alternative adaptive mutation can refer to the average, maximum, and minimum of fitness values of the candidate solutions in the population, i.e., this adaptive mutation changes the variance according to the fitness values of the candidate solutions. The most-used adaptive mutation for a minimization problem is

$$\begin{aligned} &a_{i,j}(t+1) \\ &= a_{i,j}(t) + \left(\alpha_j \cdot \frac{f - \min f}{\max f - \min f} + \beta_j\right) \cdot N(0,1) \end{aligned} \tag{14}$$

where f is the fitness value of a candidate solution and α_j and β_j are the coefficients for scaling and offset, respectively. The variance of normal random values in the adaptive mutation is scaled according to the distribution of the fitness values at the iteration times t. Consequently, the adaptive mutation can scale the focus of the searching range according to the evolution of the population. To summarize, numerical mutation operations can be conceptually classified into three types: 1) numerical mutation without using fitness values; 2) adaptive mutation based on the fitness record of a candidate solution; and 3) adaptive mutation based on the relative evaluation to other candidate solutions. In this paper, we apply adaptive mutation based on the relative evaluation for optimizing fuzzy controllers because the genotype of the candidate solutions can be altered greatly by using crossover with high probability.

C. Evaluation Function for the Fuzzy Controller

The objective of the optimization problem is to find a fuzzy controller that minimizes errors between the target outputs and the inference results, while simultaneously reducing redundant fuzzy rules and membership functions. The evaluation function $\text{fit}_{\text{controller}}$ generally consists of the error function E, the number of the membership functions M_{NUM}, and the number of fuzzy rules R_{NUM} as follows:

$$\text{fit}_{\text{controller}} = W_1 \cdot E + W_2 \cdot M_{\text{NUM}} + W_3 \cdot R_{\text{NUM}} \quad (15)$$

$$M_{\text{NUM}} = \sum_{i=1}^{r} \sum_{j=1}^{n} m_{\text{valid\_}i,j} \quad m_{\text{valid\_}i,j} = \{0, 1\} \quad (16)$$

$$R_{\text{NUM}} = \sum_{i=1}^{r} r_{\text{valid\_}i} \quad r_{\text{valid\_}i} = \{0, 1\} \quad (17)$$

where $m_{\text{valid\_}i,j}$ is a validity of the membership function for the jth input of the ith rule, $r_{\text{valid\_}i}$ is a validity of the ith rule, and W_1, W_2, and W_3 are weight coefficients. A fuzzy controller for collision avoidance is evaluated after reaching the target point. Consequently, the error function is evaluated by the performance factors: time steps (P_{time}); moving length (P_{length}); and average of the degree of danger (D_{average}), and maximum of the degree of danger $(D_{\max})$ on the trajectory generated by the fuzzy controller of a candidate solution as follows:

$$\begin{aligned} E = W_4 \cdot P_{\text{time}} + W_5 \cdot P_{\text{length}} \\ + W_6 \cdot D_{\text{average}} + W_7 \cdot D_{\max} \end{aligned} \quad (18)$$

where W_4, W_5, W_6, and W_7 are weight coefficients. D_{average} is the average of the degree of danger $(=\mu_{\text{danger}})$ from starting point to the target point, which is calculated as follows:

$$D_{\text{average}} = \frac{1}{p_{\text{time}}} \cdot \frac{1}{n} \cdot \sum_{t=1}^{p_{\text{time}}} \sum_{j=1}^{n} \left(1.0 - \frac{x_j(t)}{A\_\text{rng}(t)}\right) \quad (19)$$

where $x_j(t)$ is the sensed distance to obstacles in a sensing direction d_i (see Fig. 6), n is the number of inputs (sensors), and $A\_\text{rng}(t)$ are the attention range at the discrete time step t. As mentioned, $x_j(t)$ is regarded as $A\_\text{rng}(t)$ if $x_j(t)$ is larger than $A\_\text{rng}(t)$. Also, we must consider penalties in two cases where the robot does not reach the target point: 1) it collides with the obstacles and 2) it cannot avoid local dead ends. In the former case, a large penalty is added to $D_{\max}$. In the latter case, a large penalty is added to P_{time} and P_{length} because it cannot reach the target point in a predefined finite time. Consequently, this optimization results in a multiobjective minimization.

D. Evolutionary Process with Learning

The evolution of candidate solutions depends on the combination of genetic operators and selection mechanism. We apply a steady-state genetic algorithm (SSGA) for optimizing fuzzy controllers. In SSGA, only a few existing solutions are replaced by new candidate solutions generated by genetic operators in each generation [59]. Generally, the worst candidate solutions are eliminated. Since the objective of the above evaluation function is minimization, the candidate solution with the maximal value (i.e., the greatest error) is eliminated in the selection.

The evolutionary learning of fuzzy controllers often takes considerable time because a GA or other form of EC is a stochastic search method. Furthermore, the fine tuning of fuzzy controllers is difficult when performed only by the GA because of the lack of local search. Therefore, we incorporate human knowledge before learning as well as the delta rule as a learning mechanism in collision avoidance simulations. In this method, the delta rule plays the role of online learning using the local evaluation of the reactive motions, while the GA plays the role of offline learning using the global evaluation based on the task-dependent constraints.

Next, we consider the relationship between structured intelligence and the evolutionary optimization of fuzzy controllers. The external evaluation corresponds to the error function in the collision avoidance behavior because the external evaluation is directly related to the environments. The internal evaluation corresponds to the evaluation against the structure of fuzzy rules and membership function based on the errors. The total evaluation corresponds to the evaluation function $\text{fit}_{\text{controller}}$. The collision avoidance behavior is acquired through evolution based on the total evaluation and the adaptation based on the local error function.

E. Simulation Results of Collision Avoidance

This section shows several simulation results of the mobile robot by a self-tuning fuzzy controller. First, we show the mobile robot and the work space including several obstacles (Fig. 17). The robot has eight range sensors for measuring the distance to obstacles (also see Fig. 6). In this figure, the dotted line means the initial attention range of each range sensor. The task is to reach the target point without colliding with obstacles. Let us assume that the mobile robot can recognize its own location and the location of the target point in the work space. Consequently, it can determine the direction toward the target point. The size of work space is 500×500, when the radius of the mobile robot depicted as a circle is seven. The starting and target points are (50, 50) and (400, 470), respectively. The

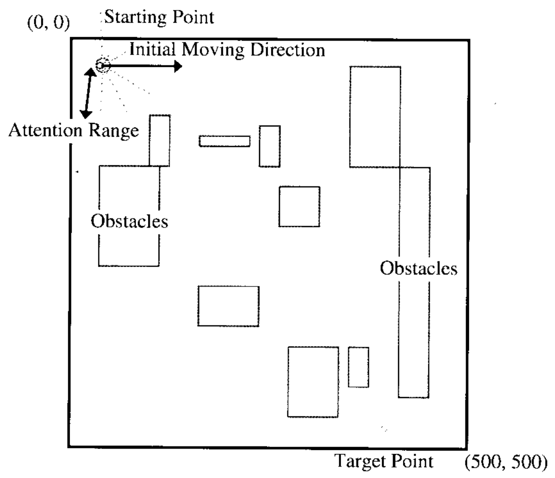


Fig. 17. The initial state of the mobile robot and the work space including several obstacles. The objective of the mobile robot is to reach the target point while avoiding collision with obstacles.

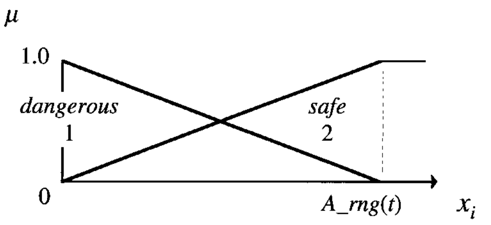


Fig. 18. Coding of membership functions corresponding to linguistic variables. The linguistic variables of dangerous and safe are encoded into 1 and 2, respectively. If neither of them is used in a fuzzy rule, 0 is used for representing the invalidity of the input variable x_i.

sensing range (maximal attention range) is 90. The steering angle is restricted between $-30°$ and $30°$. The maximal velocity is 14 (the same size as the mobile robot). These parameters are chosen according to our developing mobile robot and actual working space. The perception coefficient γ is 0.95. The initial value of $\mathrm{sprs}(0)$ and $\mathrm{sprs}_{\min}$ are 0.666 and 0.333, respectively. Consequently, the initial attention range and minimal attention range are about 60 and 30, respectively. The setting parameters of fuzzy controller and SSGA are as follows. The maximal number of fuzzy rules are 20. The numbers of inputs and outputs are eight and two, respectively. In this problem, we assume that the central values of membership functions for linguistic value dangerous and safe are zero and the value of the attention range $A\_\mathrm{rng}(t)$, respectively. Consequently, the combination problem of input variables can be considered as the combination of three genes: 0, 1, and 2 (Fig. 18). In this figure, 0 means the invalidity of the input x_i. The population size is 50. The crossover probability is 0.6. The simple mutation probabilities for r_{valid} and m_{valid} are 0.05 per gene. The exchanging mutation probabilities is 0.05 per gene. In the adaptive mutation, α_1 and β_1 for $w_{i,1}$ (steering angle) are 1.5 and 1.5 $*$ 10^{-3}, and α_2 and β_2 for $w_{i,2}$ (velocity) are 0.35 and 3.5 $*$ 10^{-4}. The number of iteration times is 500, i.e., the number of evaluation times including initialization is 1050. These values were chosen experimentally by a preliminary optimization phase for parameter setting.

First, we show simulation results of collision avoidance with/without the sensory network in the work space shown in Fig. 17 (case 0). In this simulation, we did not apply the SSGA and delta rule. Fig. 19(a)–(c) shows trajectories of the mobile robot without a sensory network where the sensing ranges (S_rng) are fixed at 30, 45, and 60, respectively. In this figure, the full line of the sensing range means that the mobile robot detects obstacles in the sensing direction. Also, the robot is depicted every ten discrete steps in a simulation. In Fig. 19(a), because the sensing range is very short, the mobile robot cannot find obstacles until it approaches them. As a result, it turns around near the obstacles at the beginning. In Fig. 19(b), the mobile robot smoothly reaches the target point [compare Fig. 19 (b) with (a) and (c)]. In Fig. 19(c), the mobile robot travels through a collision-free area surrounded by the obstacles because the sensing range is very long and the area before the target point is very narrow. Fig. 19(d) shows a trajectory of the mobile robot with the sensory network. In addition, Figs. 20 and 21 show the direction to the target point and the variation of its attention range, respectively. Its attention range is reduced gradually when it continually detects obstacles in its attention range. The mobile robot lengthens the attention range by the sensory network when it does not detect obstacles in its sensing directions. Furthermore, its velocity also decreases with decreasing attention range. As a result, the mobile robot can easily pass between obstacles. After passing through the obstacles before the target point with minimal attention range, the robot moves toward the target point and lengthens its attention range again. Table 1 shows the evaluation values concerning the time steps to reach the target point (P_{time}), moving length (P_{length}), the average of the degree of danger (D_{average}), and maximal degree of danger $(D_{\max})$. From the simulation results, the mobile robot with the sensory network obtains the best performance. Basically, it is difficult to determine the suitable sensing range beforehand, but the sensory network is capable of shortening and lengthening the attention range dynamically according to the density of obstacles in a work space. In addition, the number of the fuzzy rules designed by the human operators is eight. The mobile robot reaches the target point by using the sensory network, although the number of fuzzy rules is relatively small.

Next, we show simulation results of the evolution of the mobile robot by the SSGA. We conducted three different cases of collision avoidance where the numbers of obstacles were six, eight, and ten, respectively. Fig. 22(a) shows a trajectory of the mobile robot with the fuzzy controller before learning in case 1. The fuzzy controller is initialized by adding small random numbers to the fuzzy controller used in case 0. The mobile robot moves toward the target point while avoiding obstacles. Fig. 22(b) shows a trajectory of the mobile robot after learning by the SSGA and delta rule. The mobile robot takes a shorter path and moves toward the target point. Fig. 23 shows evaluation values of the SSGA for optimizing a fuzzy controller with delta rule in case 1.

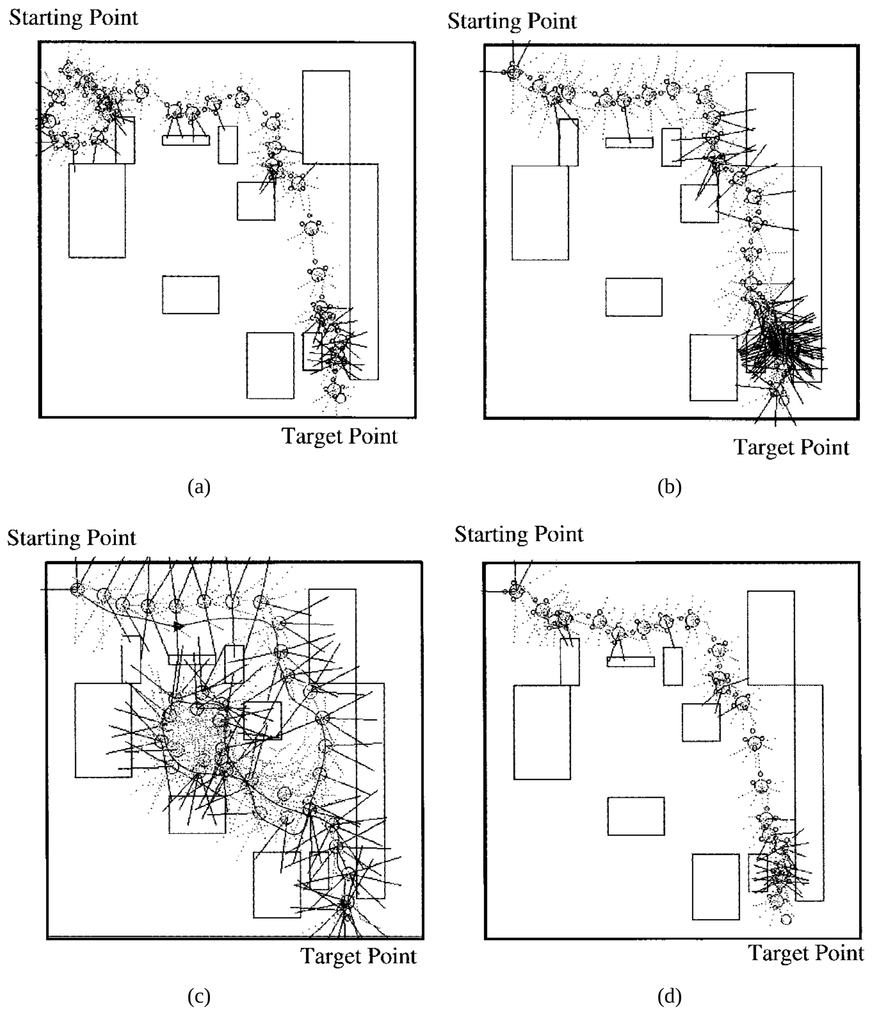


Fig. 19. Simulation results of collision avoidance with/without the sensory network (case 0). (a) A trajectory of the mobile robot without the sensory network where the sensing range (S_rng) is 30. It cannot find obstacles until it approaches them. (b) A trajectory of the mobile robot without the sensory network where S_rng is 45. It moves smoothly toward the target point. (c) A trajectory of the mobile robot without the sensory network where S_rng is 60. It moves around in the collision-free area. (d) A trajectory of the mobile robot with the sensory network. It reduces attention range and carefully moves toward the target point.

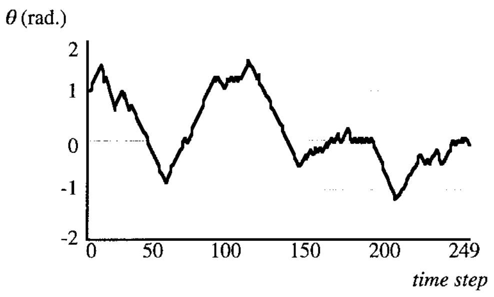


Fig. 20. Direction to the target point. The mobile robot moves toward the target point, while avoiding collision with obstacles.

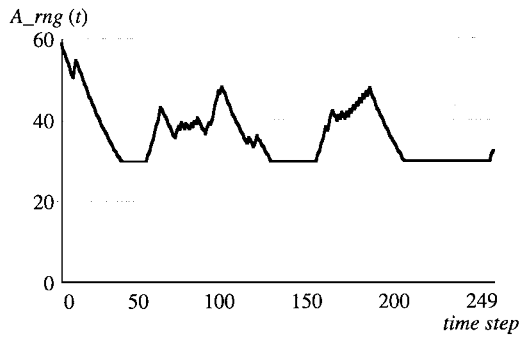
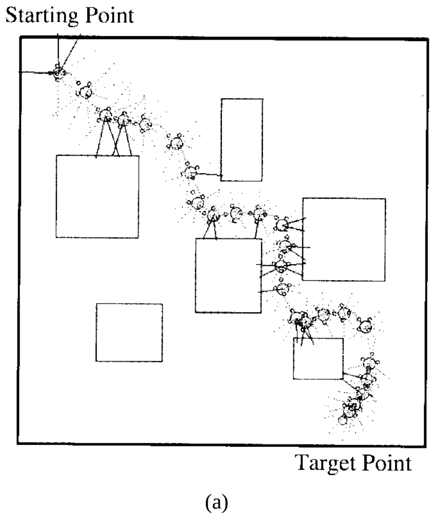


Fig. 21. The variation of the attention range of the mobile robot. The sensory network reduces the attention range when it detects obstacles in the attention range.

The SSGA reduces the evaluation value of the fuzzy controller for collision avoidance. Figs. 24 and 25 show simulation results of cases 2 and 3, respectively. Table 2 shows simulation results concerning evaluation values before and after learning in each case. Each value indicates the average of 50 trials. The performance of the fuzzy controller is improved by the SSGA and delta rule. In addition, the simulation case 1* means a simulation result of case 1 using the fuzzy controller learned in the case 3. In the case 1*, the fuzzy controller is not trained by the SSGA and delta rule, but the fuzzy controller of case 3 works well on the case 1* (Fig. 26). The results of time

Table 1
Simulation Results of the Mobile Robot with/Without Sensory Network Concerning the Performance Indexes in Fig. 19 (In the Column of the Sensing Range, SN Indicates a Simulation Result of the Mobile Robot with the Sensory Network)

	(a)	(b)	(c)	(d)
Sensing Range	30	45	60	SN
Time Step	332	302	562	249
Moving Length	820	631	1350	621
Average Danger	0.060	0.062	0.114	0.050
Max Danger	0.767	0.807	0.885	0.711



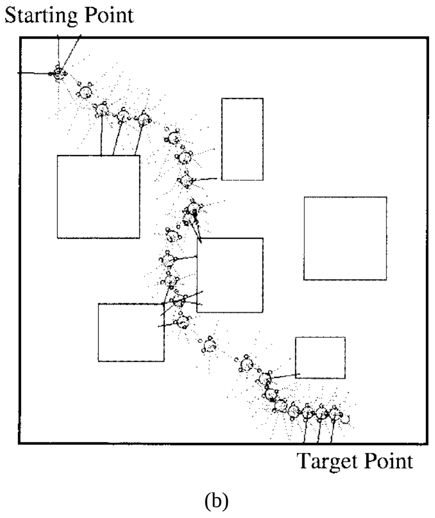


Fig. 22. Simulation results of collision avoidance of the mobile robot (case 1). (a) A trajectory before learning. The mobile robot moves toward the target point while avoiding collisions. (b) A trajectory after learning by SSGA and delta rule. The mobile robot takes a shorter path and moves toward the target point.

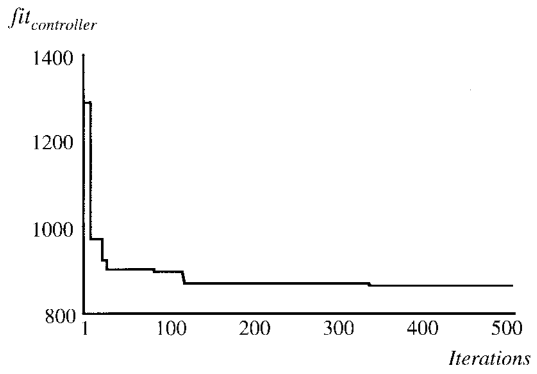
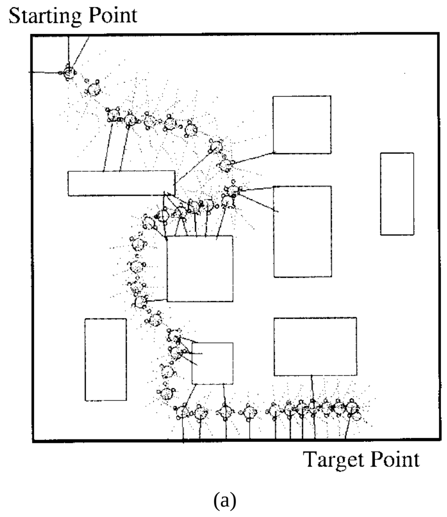


Fig. 23. Evaluation values of SSGA for optimizing fuzzy controller (case 1). SSGA reduces the evaluation values over iterations. Recall that "fitness" here is error and the task is to minimize.



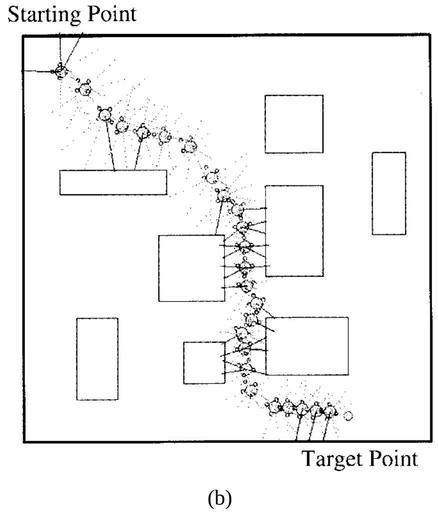
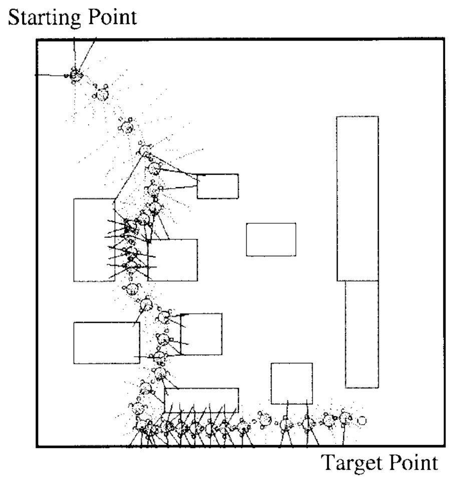


Fig. 24. Simulation results of collision avoidance of the mobile robot (case 2). (a) A trajectory before learning. (b) A trajectory after learning by SSGA and delta rule.

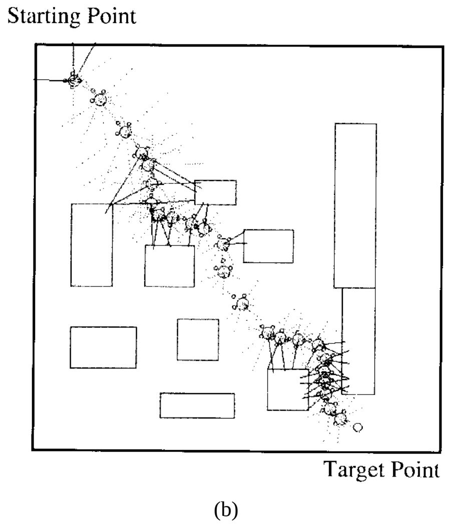
step and moving length in case 1* are similar to those of case 1, but the result of the maximal degree of danger is changed for the worse from 0.251 to 0.385. The reason is that the mobile robot passes near obstacles in the case 1* because the fuzzy controller is trained in the work space of case 3 which includes many obstacles. Next, we discuss the perception coefficient for fuzzy controller. Table 3 shows simulation results of the mobile robot with the fuzzy controller which is used in case 0. The optimal parameter setting depends on the posed work space, but we obtained good performance with the perception coefficients from 9.4 to 9.6. If the robot uses a small perception coefficient,

Table 2
Simulation Results of Learning by SSGA and Delta Rule in Cases 1, 2, and 3. Each Value Indicates the Average of 50 Trials. Case 1* Means a Simulation Result of Case 1 Using the Fuzzy Controller Learned in Case 3. Initialization (Init. in the Table) Is a Simulation Result of the Fuzzy Controller Designed by Human Operators. The Evaluation Value in Each Case Is Improved by SSGA (Offline Learning) and Delta Rule (Online Learning)

	Case 1		Case 2		Case 3		Case 1*
	Init.	GA	Init.	GA	Init.	GA	After Case 3
Evaluation Value	1458.5	882.9	2209.6	906.1	3592.2	1300.9	1198.4
Time Step	286.6	262.9	514.2	349.333	478.3	274	251
Moving Length	733.5	683.7	1224	736.3	906	621	658
Average Danger	0.0181	0.0083	0.0281	0.0077	0.053	0.016	0.014
Max Danger	0.462	0.26	0.564	0.284	0.634	0.394	0.385



(a)



(b)

Fig. 25. Simulation results of collision avoidance of the mobile robot (case 3). (a) A trajectory before learning. (b) A trajectory after learning by SSGA and delta rule.

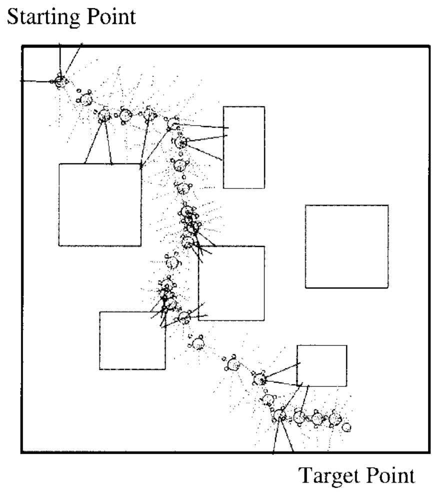


Fig. 26. Simulation result of case 1* by using the fuzzy controller learned in case 3. The mobile robot passes by near obstacles, bacause the fuzzy controller is learned in the work space of case 3, which includes many obstacles.

the attention range attains the maximal or minimal value by short time. Furthermore, if the perception coefficient is very large, the attention range hardly changes. We used the value of 9.5 as the perception coefficient in the following computer simulations.

We conducted two simulations where obstacles are randomly located (cases 4 and 5). Figs. 27 and 28 show simulation results of cases 4 and 5, respectively. Table 4 shows simulation results concerning evaluation values before and after learning in each case. Each value indicates the average of 50 trials. Since the obstacle size of cases 4 and 5 is smaller than that of cases 1, 2, and 3, the mobile robot detects not so many obstacles. As a result, the average degree of danger is small. In fact, the robot obtains collision avoidance behaviors that it can take a shorter and safer path through the learning of fuzzy controllers.

To summarize, a fuzzy controller that is suitable to a given work space can be trained by the SSGA and delta rule, but it is very difficult to generate universal fuzzy controllers that are capable of working well in various work spaces because we cannot give environmental conditions which are suitable to the learning of fuzzy controllers beforehand. However, if we design a complete fuzzy controller that satisfies the objectives of human operators without the sensory network, we need many input variables for the perception of the external states including the target direction, density of obstacles, distance to obstacles (eight parameters), and the internal states including the steering angle and velocity at least. As a result, the input dimension of the fuzzy controller without the sensory

Table 3
Simulation Results of Collision Avoidance Concerning the Perception Coefficient (τ): (a) Performance Indexes in Case 1 Where $\tau = \{0.5, 0.7, 0.8, 0.9, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97\}$; (b) Performance Indexes in Case 2 Where $\tau = \{0.93, 0.94, 0.95, 0.96, 0.97\}$; (c) Performance Indexes in Case 3 Where $\tau = \{0.93, 0.94, 0.95, 0.96, 0.97\}$

Perception Coefficient (τ)	0.5	0.7	0.8	0.9	0.92
Evaluation Value	2872.4	1649.9	1405.6	1151.7	1153.9
Time Step	424	297	260	243	236
Moving Length	888.23	789.08	741.1	705.2	703.2
Average Danger	0.044	0.022	0.018	0.0131	0.013
Max Danger	0.522	0.355	0.33	0.346	0.38

Perception Coefficient (τ)	0.93	0.94	0.95	0.96	0.97
Evaluation Value	1137.6	1112.9	1030.3	1032.9	1302.8
Time Step	234	231	236	239	250
Moving Length	700.5	693.1	691	696.4	710.8
Average Danger	0.0128	0.0122	0.0108	0.0109	0.0155
Max Danger	0.359	0.384	0.348	0.33	0.425

(a)

Perception Coefficient (τ)	0.93	0.94	0.95	0.96	0.97
Evaluation Value	1601	1610.2	1590.3	1632.2	2025.1
Time Step	258	260	260	335	355
Moving Length	724.2	726.8	728.5	927.9	909.3
Average Danger	0.019	0.0192	0.0192	0.0191	0.0263
Max Danger	0.728	0.723	0.675	0.599	0.736

(b)

Perception Coefficient (τ)	0.93	0.94	0.95	0.96	0.97
Evaluation Value	1519.5	1514.6	1517.5	1532.8	1701.9
Time Step	231	229	232	231	240
Moving Length	677.6	666.9	670.9	676.9	679.7
Average Danger	0.0203	0.0202	0.0203	0.0201	0.0238
Max Danger	0.41	0.42	0.408	0.475	0.42

(c)

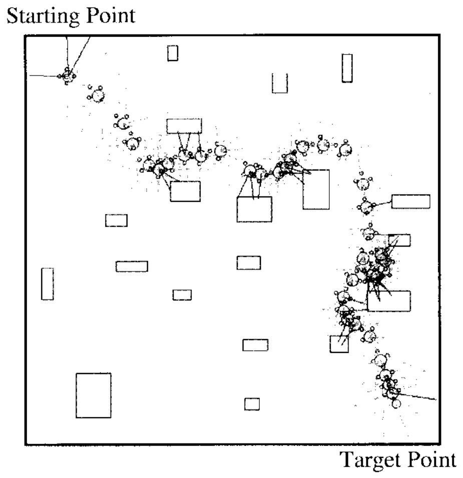
network becomes 12 and the mobile robot requires many more fuzzy rules than the fuzzy controller with the sensory network. The above simulation results show the mobile robot with the learned fuzzy controller can reach the target point without colliding with obstacles. The adjusting mechanism of the sensory network scales the output of the fuzzy controller according to the density of obstacles in the work space. Thus, the high perception capability is very important for robotic systems. Also, the closer linkage of perception, decision making, and action can realize the higher intelligence structured as a whole system.

V. Path Planning of the Mobile Robotic System

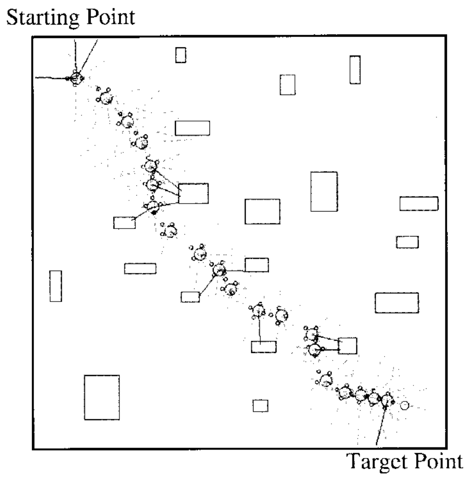
A. Path Planning Based on Reactive Motions

The robotic system requires a planning capability (see Fig. 3). Our previous works showed that collision-free paths and trajectories of robotic systems can be generated by EC-based planning [25], [38], [49], [58], but we have not used the already learned control rules of robotic systems in path planning. The robotic system should adapt to dynamic environmental conditions, and therefore it should perform path planning on the basis of the acquired knowledge and skill. Consequently, this section discusses path planning of the mobile robot in unknown work spaces without map building.

Assume that only a target point is given to the mobile robot. The mobile robot with structured intelligence takes the reactive motions of collision avoidance and target trace in a given work space. Its trajectory is generated as the result of these reactive motions. If the work space is complicated and has dead ends, the mobile robot should generate several intermediate points to the target point and it should trace these intermediate points in order. Furthermore, we do not need to consider collisions between

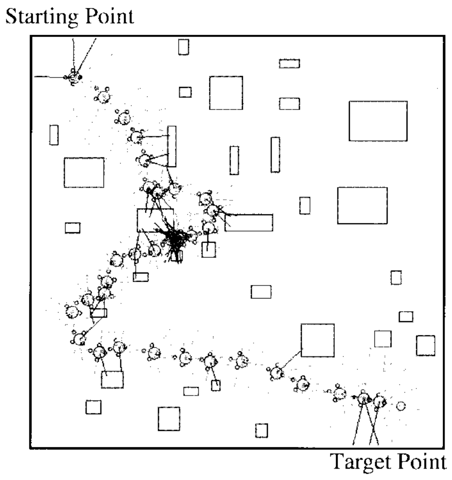


(a)

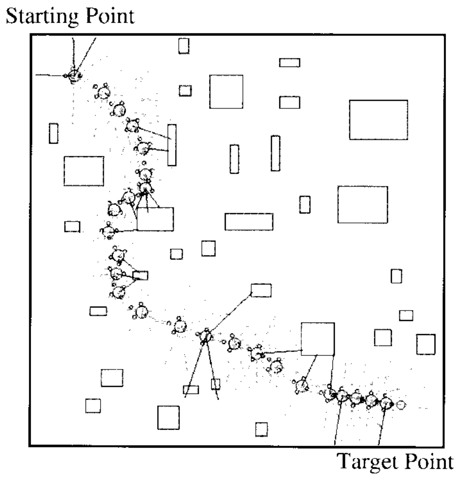


(b)

Fig. 27. Simulation results of collision avoidance of the mobile robot in the work space including obstacles randomly located (case 4). (a) A trajectory before learning. (b) A trajectory after learning by SSGA and delta rule.



(a)



(b)

Fig. 28. Simulation results of collision avoidance of the mobile robot (case 5). (a) A trajectory before learning. (b) A trajectory after learning by SSGA and delta rule.

the directly connected path and obstacles because the mobile robot can take on collision avoidance behaviors by reactive motions. Therefore, path planning results in a problem of generating intermediate target points.

Fig. 29 shows a path planning result of the mobile robot in an unknown work space. The number of the required intermediate points depends on the complexity of a given work space. A candidate solution which is composed of the validity $(\mathrm{IP}_{\mathrm{valid}\_i,j})$ and numerical values $(\mathrm{IP}_{x\_i,j}, \mathrm{IP}_{y\_i,j})$ of an intermediate point j of a candidate solution i is shown in Fig. 30. The aim of the mobile robot is to reach the target point, but it does not need to reach each intermediate point exactly. Consequently, the mobile robot passes by the intermediate point when it reaches within a predefined radius (IP_R) of each intermediate point. The set of these intermediate points is exactly evaluated only when the mobile robot reaches the target point. The evaluation function to be minimized is as follows:

$$\mathrm{fit}_{\mathrm{path}} = W_8 \cdot P_{\mathrm{distance}} + W_9 \cdot P_{\mathrm{length}} + W_{10} \cdot \mathrm{IP}_{\mathrm{NUM}} \tag{20}$$

Table 4
Simulation Results of the Collision Avoidance Where Obstacles Are Randomly Located in Work Spaces (Case 4 and 5); Each Value Indicates the Average of 50 Trials

	Case 4		Case 5	
	Init.	GA	Init.	GA
Evaluation Value	1455.7	629.9	2209.6	892.3
Time Step	368.5	198.5	514.2	266.2
Moving Length	876.5	609.4	1224	738.2
Average Danger	0.016	0.0041	0.0281	0.0075
Max Danger	0.545	0.259	0.564	0.365

where P_{distance} is the minimal distance to the target point in a simulation, P_{length} is the path length from the starting point to the target point, $\mathrm{IP}_{\mathrm{NUM}}$ is the number of intermediate points, and W_8, W_9, and W_{10} are weight coefficients. Consequently, when the mobile robot reaches the target point, the actual distance to the target point (P_{distance}) is zero. If it cannot avoid local dead ends, a large penalty

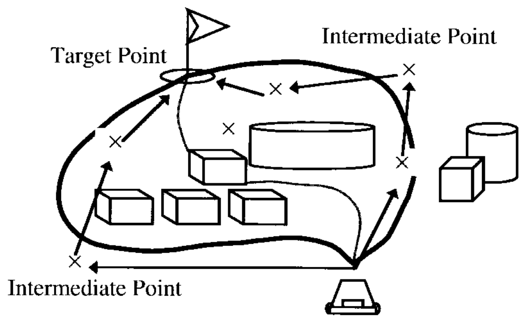


Fig. 29. Path planning of a mobile robot in an unknown work space. In the figure, × is an intermediate point. The mobile robot generates and traces several intermediate points.

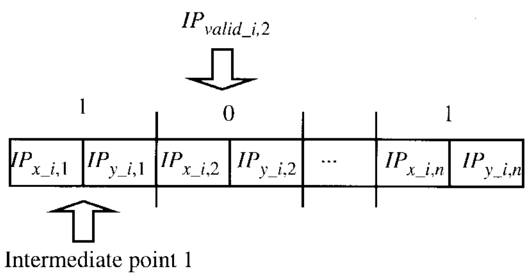


Fig. 30. Representation of an intermediate point for path planning. $\mathrm{IP}_{\mathrm{valid\_ij}}$ and $(\mathrm{IP}_{x\_i,j}, \mathrm{IP}_{y\_i,j})$ are the validity and the position of an intermediate point j of a candidate solution i, respectively.

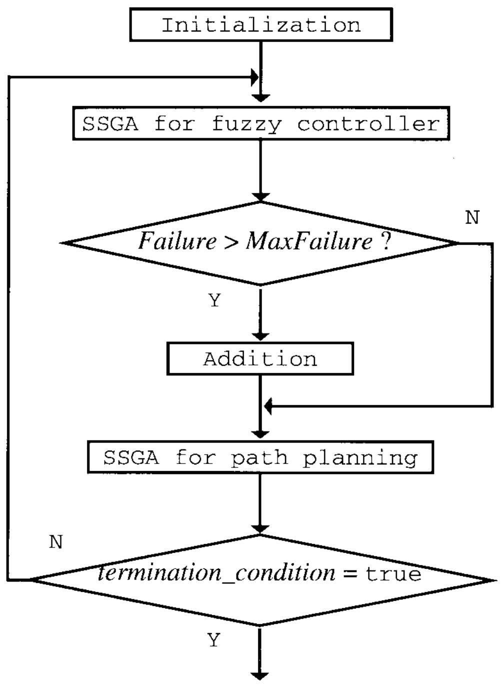


Fig. 31. Procedure of path planning in an unknown work space. *Failure* means the number of the failure to reach the target point in the simulation on `SSGA for fuzzy controller`. Initially, the mobile robot does not perform the path planning. After several simulations in `SSGA for fuzzy controller`, if it cannot reach the target point, the mobile robot gradually searches for a path in `SSGA for path planning` by adding intermediate points.

is added to P_{length}. We also apply the SSGA to the path planning problem.

We must take into account the evolution of fuzzy controllers in path planning. Fig. 31 shows the procedure of path planning including the evolution of fuzzy controllers in an unknown work space. Initially, the mobile robot has no intermediate points $(\mathrm{IP}_{\mathrm{valid}\_i,j} = 0)$ and does not perform path planning. In using SSGA for learning a fuzzy controller, several new candidate solutions of fuzzy controllers are generated by crossover and mutation, and evaluated through the collision avoidance simulations. In the simulation, if the mobile robot cannot reach the target point, the times of failure, *Failure*, is incremented. Next, if *Failure* is greater than the predefined threshold, *MaxFailure*, *Failure* is initialized at zero and *Addition* adds an intermediate point to a candidate solution of SSGA for path planning. The intermediate point is generated by: 1) random generation; 2) interpolated generation; and 3) neighborhood generation as follows.

1) Random generation [Fig. 32(a)]: A new intermediate point is generated within the work space by using a uniform random value. The random generation globally searches the work space for intermediate points.
2) Interpolated generation [Fig. 32(b)]: First, two points are randomly selected from the pool of the starting point, target point, and the already generated intermediate points. A new intermediate point is generated within the circle of which the center is defined as the middle point between the selected points. The interpolated generation of intermediate points is helpful to connect two existing points.
3) Neighborhood generation [Fig. 32(c)]: First, a point is randomly selected from the pool of the starting point, target point, and the already generated intermediate points. A new intermediate point is generated within the circle whose center is the selected point. The radius of the circle is generally predefined as half the work space size. The neighborhood generation of intermediate points mainly plays the role of avoiding dead ends of the street.

In *Addition*, one of them is randomly selected and an intermediate point is generated. In SSGA for path planning, the candidate solution of the intermediate points is evaluated by the simulation result of the collision avoidance behaviors. Basically, a candidate solution used in the next collision avoidance simulation is randomly selected, but the candidate solution that an intermediate point is added to is selected when *Addition* is performed. In SSGA for path planning, one-point crossover, adaptive mutation, and simple mutation are used as genetic operators. These are applied in the same manner as the learning of fuzzy controllers. Furthermore, simple mutation changes the validity of an intermediate point from zero to one; its location is updated by one of the above three intermediate point-generation methods. Adaptive mutation adds a small normal random value to each intermediate point. The termination condition is to satisfy the predefined aspiration level.

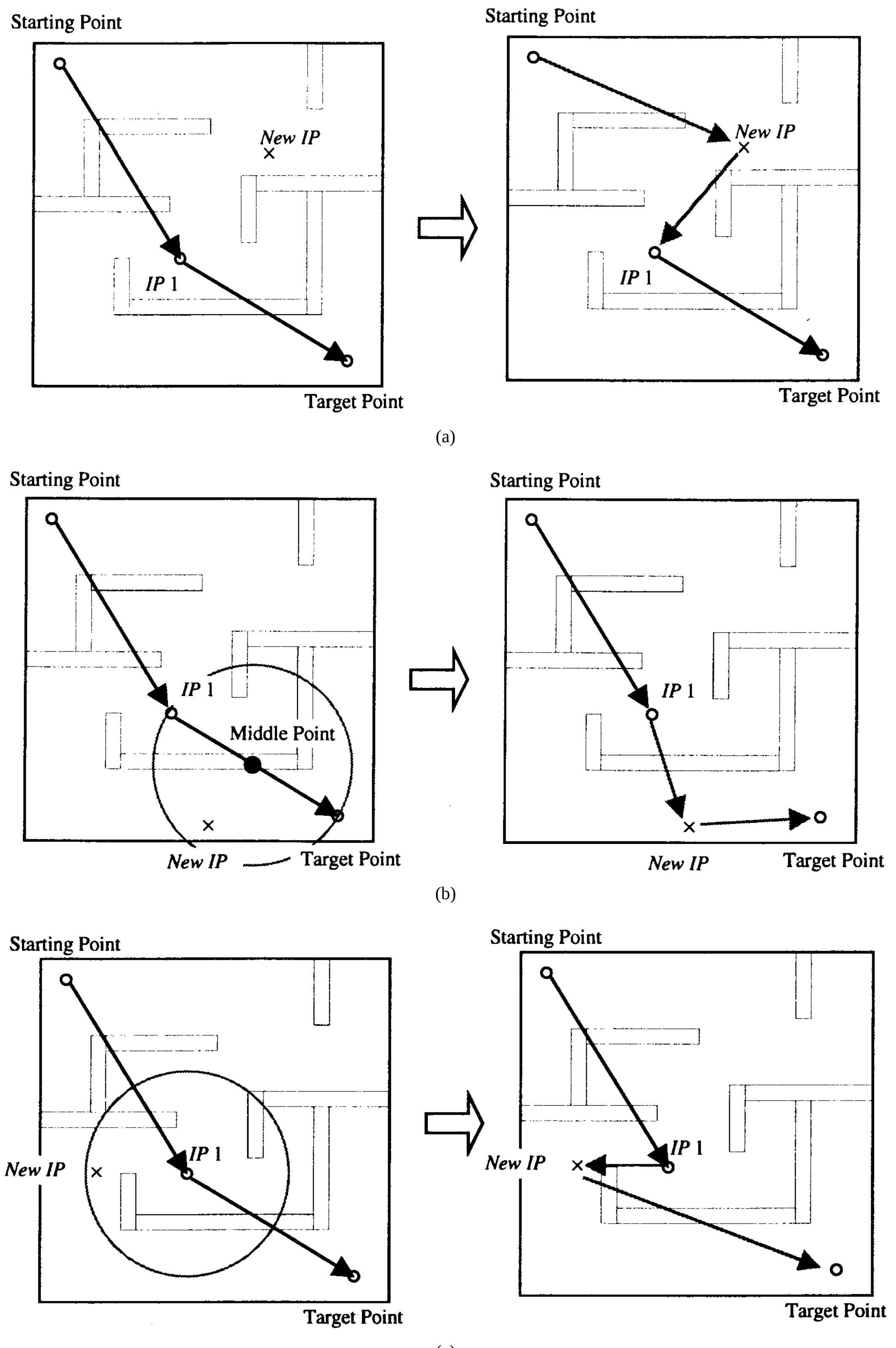


Fig. 32. (a) Random generation of an intermediate point. In the figure, a new intermediate point × is generated in the work space without considering other intermediate points. (b) Interpolated generation of an intermediate point. In the figure, a new intermediate point × is generated whithin the circle of which the center is defined as the middle point between two randomly selected points. (c) Neighborhood generation of an intermediate point. In the figure, a new intermediate point × is generated within the circle whose center is a randomly selected point.

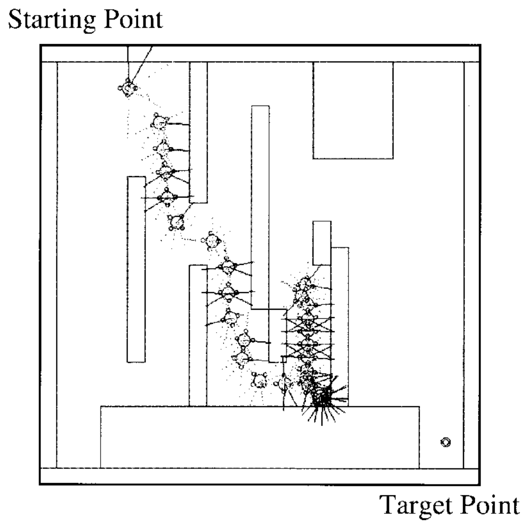


Fig. 33. A trajectory of mobile robot by the fuzzy controller without path planning (case 6). It cannot escape from a local area surrounded by obstacles.

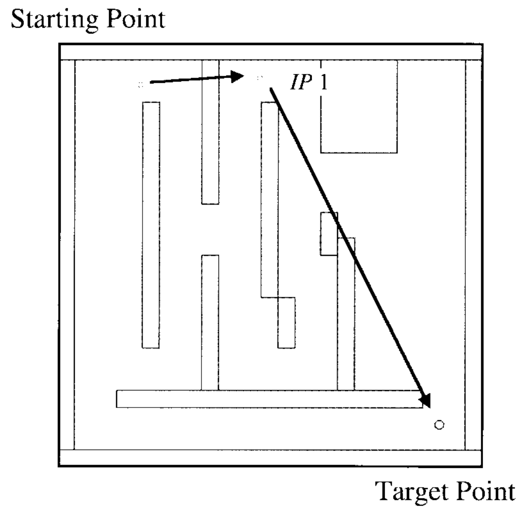


Fig. 34. Intermediate points generated by SSGA for path planning (case 6). One intermediate point is generated between the starting point and the target point. The directly connected path collides with obstacles, but we need not consider the collision, since the mobile robot avoids the collision with obstacles by using the fuzzy controller.

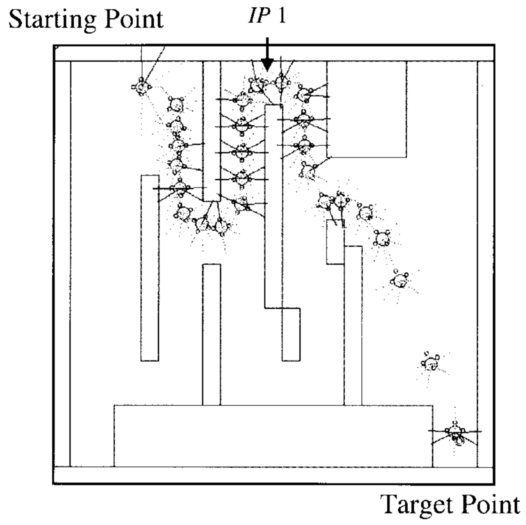


Fig. 35. A trajectory of the mobile robot traced by reactive motions (case 6). It reaches the target point without becoming trapping in local area.

The best combination of the fuzzy controller candidate and a set of intermediate points is required to reach the target point while avoiding obstacles and reducing moving time and length. Thus, the mobile robot gradually increases the number of intermediate points if it cannot reach the target point. In contrast, if it can easily reach the target point, the mobile robot does not perform the path planning based on intermediate points.

B. Simulation Results of Path Planning

This section shows simulation results of path planning with the collision avoidance behavior. The setting parameters for SSGA for path planning are as follows. The maximal number of intermediate points is ten. The population size is 20. The crossover probability is 0.6. The simple mutation probabilities for IP_{valid} is 0.05 per gene. In adaptive mutation, α_j and β_j for $(\text{IP}_{x\_i,j}, \text{IP}_{y\_i,j})$ are 20.0 and 0.2, respectively. The number of iteration times is 1000, i.e., the number of evaluation times including initialization is 2000 because intermediate points are not generated in Initialization. *MaxFailure* is 5. The radius of the circle in the neighborhood generation is 125. The radius for passing by each intermediate point (IP_R) is 20. These values were chosen experimentally. In addition, we use the parameter setting of SSGA for fuzzy controller in the previous section, but the number of iterations is 1000. Next, we show two path planning simulations in work spaces including many obstacles.

Fig. 33 shows a simulation result of case 6 where the number of obstacles is nine and the size of the work space is 500 $\times$ 500. The starting and target points are (100, 50) and (450, 450), respectively. In the figure, the mobile robot is depicted every 30 discrete time steps. It cannot reach the target point, since it cannot escape from a local area surrounded by obstacles. Fig. 34 shows the intermediate point generated by SSGA for path planning. The directly connected path collides with obstacles, but the mobile robot avoids the collision with obstacles by using the fuzzy controller. Fig. 35 shows the actual trajectory of the mobile robot traced by reactive motions. The mobile robot reaches the target point without becoming trapped in local areas. In the figure, the first intermediate point plays the important role of reaching the target point because the intermediate point leads the mobile robot to the opposite side of a local area surrounded by obstacles.

We conducted a simulation in a more complicated work space where the number of obstacles is 26 and the size of the work space is 800 $\times$ 800 (case 7). The starting and target points are (100, 50) and (780, 780), respectively. Fig. 36 shows the simulation result of case 7 before path planning. The mobile robot cannot escape from a local area surrounded by obstacles near the starting point (first trap).

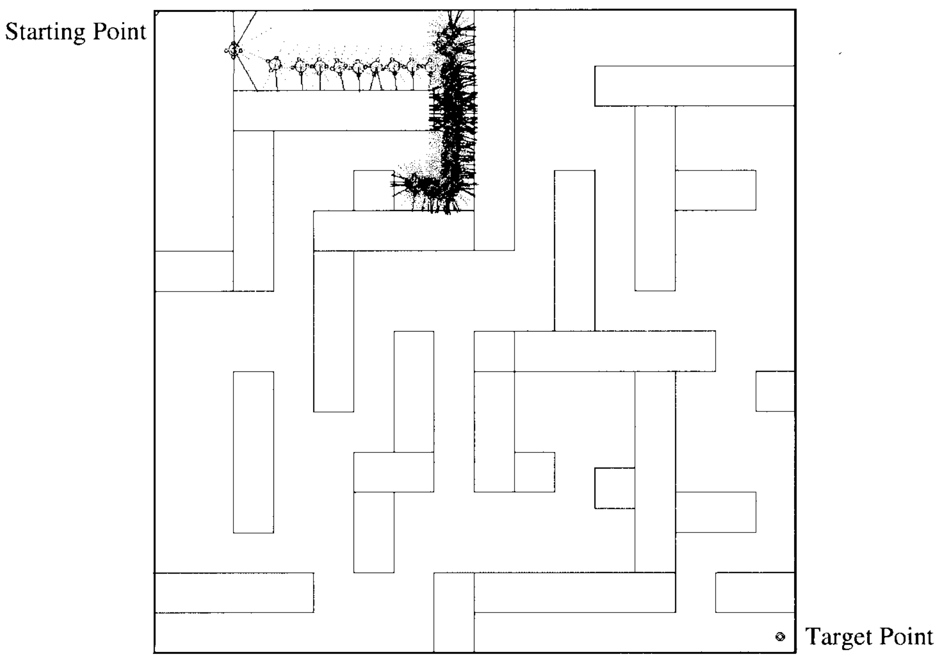


Fig. 36. A trajectory of mobile robot only by reactive motions in a work space including many obstacles and dead ends (case 7). The size of work space is 800 $\times$ 800 and the number of obstacles is 26. The mobile robot cannot escape from a local area surrounded by obstacles (first trap).

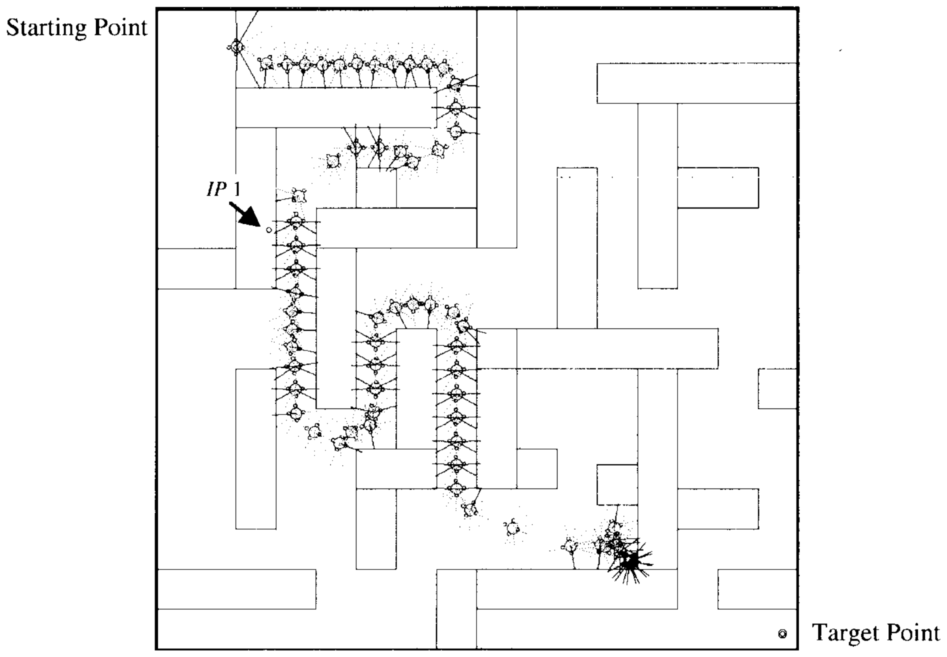


Fig. 37. A trajectory of mobile robot using the first intermediate point (case 7). The robot can avoid the first trap, but it cannot escape from the local area surrounded by obstacles near the target point (second trap).

Fig. 37 shows the trajectory of the mobile robot using the intermediate point which is generated by the neighborhood generation in *Addition* at iteration 18. The mobile robot avoids the first trap, but it cannot escape from the local area surrounded by obstacles near the target point (second trap). Basically, it is easy to generate intermediate points on the neighborhood of the line connected directly between the starting and target points in *Addition*, but the crossover can change the combination of intermediate points and the adaptive mutation can perform the local search. Fig. 38 shows the actual trajectory of the mobile robot traced by reactive motions through four intermediate points. Fig. 39(a)–(c) shows the evaluation values ($\mathrm{fit}_{\mathrm{path}}$), the number of intermediate points ($\mathrm{IP}_{\mathrm{NUM}}$), and the minimal distance to the target point (P_{distance}) of the best candidate solution in the path planning, respectively. The mobile robot

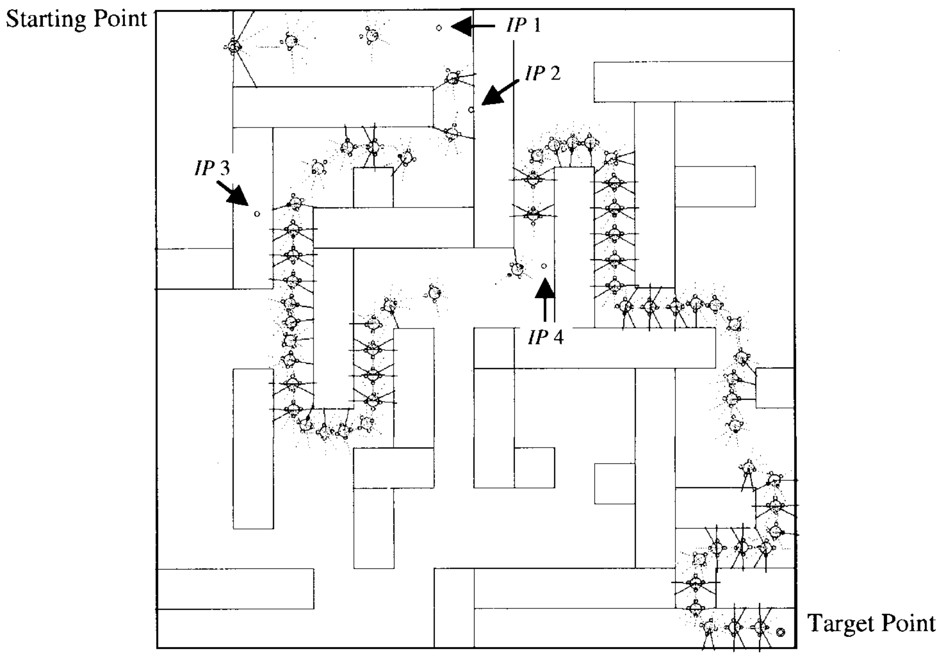
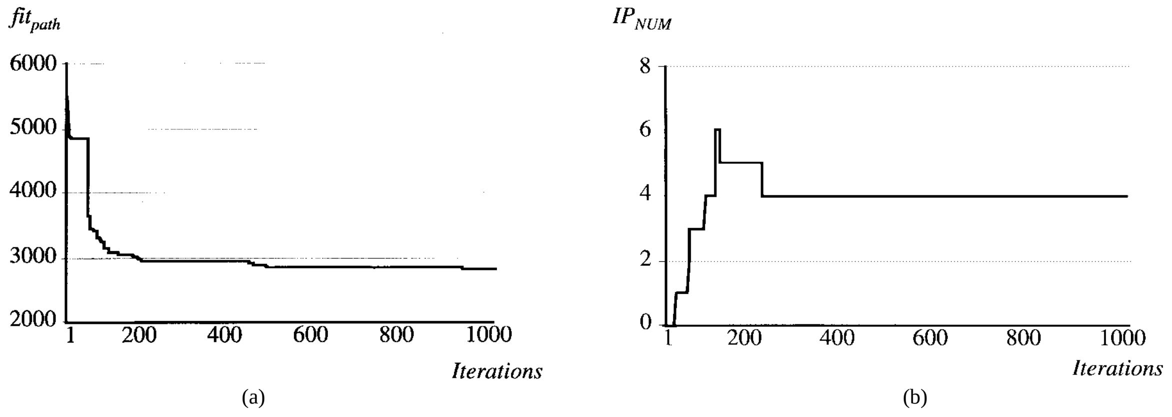


Fig. 38. An actual trajectory of the mobile robot traced by reactive motions through four intermediate points (case 7). It reaches the target point avoiding in the first and second traps.



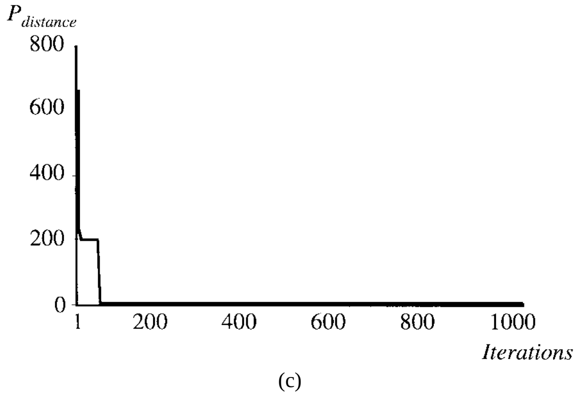


Fig. 39. (a) Evaluation values of the best candidate solution in the path planning (case 7). (b) The number of intermediate points of the best candidate solution (case 7). The number of intermediate points increases in the beginning, but finally it is four. (c) The minimal distance to the target point of the best candidate solution (case 7). The mobile robot succeeds to reach the target point at the interation 71.

reaches the target point at iteration 71. In the beginning, the mobile robot moves far away from obstacles by tracing the first and second intermediate points which play the role of reducing the time step and degree of danger. The fourth intermediate point is very important because it leads the mobile robot in the right direction. Furthermore, intermediate points may be located within obstacles because the mobile robot can pass by when it reaches within the

Table 5
Simulation Results of the Path Planning (Case 6 and 7); Each Value Indicates the Average of 30 Trials

	Case 6	Case 7
Evaluation Value	1000.8	2904.1
Time Step	532.3	2401.5
Moving Length	969.3	2446.7
Number of IP	2.5	4.8

predefined radius. The location of intermediate points within obstacles is infeasible in standard path-planning problems but the collision check of intermediate points against obstacles is not required, since the proposed method includes collision avoidance behavior. This behavior can provide a high feasibility for intermediate points. Table 5 shows simulation results of the path planning (case 6 and 7). Each value indicates the average of 30 trials. The mobile robot reaches the target point in all trials. The optimal number of intermediate points depends on the posed work space. The further addition of intermediate points can be improved the collision avoidance performance.

These simulation results show that the mobile robot can reach the target point by generating several intermediate points without the consideration of collisions between obstacles and a path connecting intermediate points directly. Standard path-planning problems have a problem that collision-free paths must be generated beforehand. Path planning based on reactive motions overcomes this problem. We believe this approach is sufficient to generate an efficient path, although we agree that global map building is very useful to reduce the number of trials in path planning.

The architecture of structured intelligence for a mobile robot proposed in this paper is qualitatively similar to the work of Donnart and Meyer [45]. The MonaLysa architecture relies on a hierarchical classifier system including a reactive module and a planning module [45]. In the reactive module, a simulated robot has many possible rules and decides what action to perform next. Furthermore, a task is logically decomposed into a series of subtasks in the planning module. The MonaLysa architecture effectively combines the reactive module and the planning module, but the optimization of rules has not yet been performed [45]. Compared with this work, structured intelligence not only tunes the fuzzy controller but also optimizes the structure of fuzzy rules in path planning. Furthermore, in [58], we have shown the effectiveness of primitive motion planning based on the acquired skills and motions. Thus, the robotic system with structured intelligence makes decisions and takes actions while learning according to a given environmental state.

VI. Conclusion

This paper proposes an entire integrated structure of intelligent robotic systems based on a fuzzy controller. The close linkage of perception, decision making, and action plays a very important role in achieving high intelligence for robotic systems. Furthermore, we apply structured intelligence to a mobile robotic system. Based on the sensory network, perception adjusts the attention range to the environment recursively and also adjusts the output of the fuzzy controller according to time series of sensed information. As tuning mechanisms of fuzzy controllers, we apply a genetic algorithm and the delta rule. The simulations show that a mobile robot can reach a target point without colliding with obstacles, even though the number of fuzzy rules is relatively small. Furthermore, we apply a genetic algorithm to path-planning problems of the mobile robot. The simulation results show that the mobile robot can reach a target point by generating several intermediate points. Also, the simulations show the importance of reactive motions in path planning.

This paper does not, however, discuss how to determine optimal parameters of the proposed method in detail because these parameters depend on a given environment. We can optimize these parameters when the environmental conditions are fixed, but then the robotic system should self tune these parameters. Therefore, the robotic system requires internal evaluation criteria for the self tuning in a dynamic environment. If we can describe the internal evaluation criteria exactly, the robotic system can evolve through interaction with the dynamic environment. However, it may not adapt to big changes in a dynamic environment. We intend to discuss how to update the internal criteria against a dynamic environment in future work.

Navigation and path-planning problems for mobile robotic systems have been widely discussed in various fields [37]–[49]. We intend to compare the proposed robotic system with other methodologies. In this paper, we focus mainly on the perception capability based on the sensory network. Perception is a very important topic and a very difficult task for intelligent robotics. The integration of already acquired knowledge and skill depend on the perception capability that produces the relationship between the external information and action. We intend to further address the problem of the fusion and integration of already acquired knowledge and skills for intelligent robotic systems in a dynamic environment. Finally, we must address that the attention, intuition, and consciousness proposed in this paper are different from those of psychology. As future subjects, we intend to discuss the concepts of perception and consciousness of robotic systems.

Acknowledgment

The authors would like to thank D. B. Fogel, L. Guan, G. Vachkov, F. Kojima, and anonymous reviewers for their valuable comments that improved the readability of the paper.

References

[1] L. A. Zadeh, “Fuzzy sets,” *Inform. Control*, vol. 8, pp. 338–353, 1965.
[2] J. M. Zurada, R. J. Marks II, and C. J. Robinson, Eds., *Computational Intelligence—Imitating Life*. Piscataway, NJ: IEEE Press, 1994.

[3] M. Palaniswami, Y. Attikiouzel, R. J. Marks II, D. Fogel, and T. Fukuda, Eds., *Computational Intelligence—A Dynamic System Perspective*. Piscataway, NJ: IEEE Press, 1995.

[4] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[5] S. V. Kartalopoulos, *Understanding Neural Networks and Fuzzy Logic*. Englewood Cliffs, NJ: IEEE Press, 1996.

[6] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—Part I & part II," *IEEE Trans. Systems, Man, Cybern.*, vol. 20, pp. 404–435, Feb. 1990.

[7] Y. Shi, M. Mizumoto, N. Yubazaki, and M. Otani, "A learning algorithm for tuning fuzzy rules based on the gradient method," in *Proc. 5th IEEE Int. Conf. Fuzzy Systems*, New Orleans, LA, 1996, pp. 55–61.

[8] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm," *IEEE Trans. Neural networks*, vol. 3, pp. 801–806, Sept. 1992.

[9] C. M. Higgins and R. M. Goodman, "Learning fuzzy rule-based neural networks for function approximation," in *Proc. Int. Joint Conf. Neural Network*, vol. 1, 1992, pp. 251–256.

[10] R. Katayama, Y. Kajitani, K. Kuwata, and Y. Nishida, "Self generating radial basis function as neuro-fuzzy model and its application to nonlinear prediction of chaotic time series," in *Proc. IEEE Int. Conf. Fuzzy Systems*, 1993, pp. 407–414.

[11] D. A. Linkens and J. Nie, "Fuzzified RBF network-based learning control: Structure and self-construction," in *Proc. IEEE Int. Conf. Neural Networks*, 1993, pp. 1016–1021.

[12] D. Whitley, T. Starkweather, and T. Bogart, "Genetic algorithms and neural networks: Optimizing connection and connectivity," *Parallel Computing*, vol. 14, pp. 347–361, 1990.

[13] H. Nomura, I. Hayashi, and N. Wakami, "A self-tuning methods of fuzzy control by decent method," in *Proc. 4th IFSA Congr. Engineering*, 1991, pp. 155–158.

[14] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proc. 2nd IEEE Int. Conf. Fuzzy System*, 1993, pp. 612–617.

[15] K. Shimojima, T. Fukuda, and Y. Hasegawa, "RBF-fuzzy system with GA based unsupervised/supervised learning method," in *Proc. Int. Joint Conf. 4th Fuzz-IEEE/2nd IFES*, 1995, pp. 253–258.

[16] K. Shimojima, N. Kubota, and T. Fukuda, "Virus-evolutionary genetic algorithm for fuzzy controller optimization," in *Studies in Fuzziness, Genetic Algorithms and Soft Computing*, vol. 8. Berlin, Germany: Physica-Verlag, pp. 369–388, 1997.

[17] S. J. Russell and P. Norvig, *Artificial Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[18] J. A. Anderson and E. Rosenfeld, *Neurocomputing—Foundations of Research*. Cambridge, MA: MIT Press, 1988.

[19] C. G.Langton, Ed., *Artificial Life—An Overview*. Cambridge, MA: MIT Press, 1995.

[20] J. C. Bezdek, "What is computational intelligence?" in *Computational Intelligence—Imitating Life* J. M. Zurada, R. J. Marks II, C. J. Robinson, Eds. Piscataway, NJ: IEEE Press, 1994, pp. 1–12.

[21] T. Fukuda and N. Kubota, "Computational intelligence in robotics and mechatronics," in *Proc. 23rd Int. Conf. Industrial Electronics, Control, and Instrumentation*, vol. 11, 1997, pp. 1517–1528.

[22] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge, MA: MIT Press, 1981.

[23] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. New York: McGraw-Hill, 1987.

[24] T. Fukuda and T. Ueyama, *Cellular Robotics and Micro Robotic Systems, World Scientific Series in Robotic and Automated Systems*, vol. 10. New York: World Scientific, 1994.

[25] T. Fukuda, N. Kubota, and T. Arakawa, "GA algorithms in intelligent robots," in *Fuzzy Evolutionary Computation*. Norwell, MA: Kluwer, 1997, pp. 81–105.

[26] W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control*. Cambridge, MA: MIT Press, 1990.

[27] K. S. Narendra, "Associative learning in random environments using neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 20–31, Jan. 1991.

[28] Y. Davidor, "Analogue crossover," in *Proc. 3rd Int. Conf. Genetic Algorithm*, 1989, pp. 98–103.

[29] Y. Davidor, "A genetic algorithm applied to robot trajectory generation," in *Handbook of Genetic Algorithms*. New York: Van Nostrand, 1991, pp. 144–165.

[30] H. S. Lin, J. Xiao, and Z. Michaleswicz, "Evolutionary algorithm for path planning in mobile robot environments," in *Proc. 1st IEEE Conf. Evolutionary Computation*, 1994, pp. 211–216.

[31] T. Shibata, T. Fukuda, K. Kosuge, and F. Arai, "Selfish and coordinative planning for multiple mobile robots by genetic algorithm," in *Proc. 31st Conf. Decision and Control*, 1992, pp. 2686–2691.

[32] Y. K. Hwang, P. C. Chen, and P. A. Watterberg, "Interactive task planning through natural language," in *Proc. 1996 IEEE Int. Conf. Robotic and Automation*, 1996, pp. 24–29.

[33] S. Ahson, N. Sharkey, and B. Nicolas, "Avoiding joint limits and obstacles for kinematically redundant manipulators: A fuzzy-logic based approach," in *Proc. 5th IEEE Int. Conf. Fuzzy Systems*, 1996, pp. 1777–1781.

[34] T. Horiuchi, A. Fujino, O. Katai, and T. Sawaragi, "Fuzzy interpolation-based Q-learning with continuous state and actions," in *Proc. 5th IEEE Int. Conf. Fuzzy Systems*, 1996, pp. 595–600.

[35] T. Fukuda and T. Shibata, "Theory and applications for neural networks for industrial control systems," *IEEE Trans. Ind. Electron.*, vol. 39, pp. 472–489, Dec. 1992.

[36] H. Bersini, M. Saerens, and G. Sotelino, "Hopfield net generation, encodings and classification of temporal trajectories," *IEEE Trans. Neural Networks*, vol. 5, pp. 945–953, Nov. 1994.

[37] T. Arakawa and T. Fukuda, "Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, 1996, pp. 1495–1500.

[38] N. Baba and N. Kubota, "Collision avoidance planning of a robot manipulator by using genetic algorithm—A consideration for the problem in which moving obstacles and/or several robots are in the workspace," in *Proc. 1st IEEE Conf. Evolutionary Computation*, 1994, pp. 714–719.

[39] T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst., Man, Cybern.*, vol. 11, pp. 681–698, 1981.

[40] R. A. Brooks, "Planning collision-free motions for pick-and-place operation," *Int. J. Robotics Res.* , vol. 2, no. 4, pp. 19–44, 1983.

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

[42] D. Jung and K. K. Gupta, "Octree-based hierarchical distance maps for collision detection," in *Proc. 1996 IEEE Int. Conf. Robotic and Automation*, 1996, pp. 454–459.

[43] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Trans. Robot. Automat.*, vol. 2, pp. 14–23, Jan. 1986.

[44] M. Colombetti, M. Dorigo, and G. Borghi, "Behavior analysis and training—A methodology for behavior engineering," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 365–380, Mar. 1996.

[45] J. Y. Donnart and J. A. Meyer, "Learning reactive and planning rules in a motivationally autonomous animat," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 381–395, Mar. 1996.

[46] J. Tani, "Model-based learning for mobile robot navigation from the dynamical systems perspective," *IEEE Trans. Syst., Man, Cybern. B* , vol. 26, pp. 421–436, Mar. 1996.

[47] J. R. Millan, "Rapid, safe, and incremental learning of navigation strategies," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 408–420, Mar. 1996.

[48] T. Tsuji, S. Nakayama, and K. Ito, "Parallel and distributed trajectory generation of redundant manipulators through cooperation and competition among subsystems," *IEEE Trans. Systems, Man, Cybern. B*, vol. 27, pp. 498–509, Mar. 1997.

[49] N. Kubota, T. Arakawa, T. Fukuda, and K. Shimojima, "Motion planning for a robotic system with structured intelligence," in *Proc. 1997 IEEE Int. Symp. Computational Intelligence in Robotics and Automation*, 1997, pp. 60–65.

[50] R. A. Brooks, "Intelligence without representation," *Artificial Intell. J.*, vol. 47, no. 1–3, pp. 139–159, 1991.

[51] R. C. Eberhart, "Computational intelligence: A snapshot," in *Computational Intelligence—A Dynamic System Perspective*. Piscataway, NJ: IEEE Press, pp. 9–15, 1995.

[52] T. Fukuda, S. Shiotani, and F. Arai, "A new neuron model for additional learning," in *Proc. 1992 Int. Joint Conf. Neural

Network, pp. 938–943.
[53] D. B. Fogel, *Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1995.
[54] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[55] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison Welsey, 1989.
[56] J. Koza, *Genetic Programming*. Cambridge, MA: MIT Press, 1992.
[57] ——, *Genetic Programming II*. Cambridge, MA: MIT Press, 1994.
[58] N. Kubota, T. Arakawa, and T. Fukuda, "Trajectory planning and learning of a redundant manipulator with structured intelligence," *J. Brazilian Comput. Soc.*, vol. 4, no. 3, pp. 14–26, 1998.
[59] G. Syswerda, "A study of reproduction in generational and steady-state genetic algorithms," in *Foundations of Genetic Algorithms*, G. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 94–101.
[60] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Trans. Evolutionary Computation*, vol. 1, pp. 18–28, 1998.

Toshio Fukuda (Fellow, IEEE), for a photograph and biography, see this issue, p. 1422.

Naoyuki Kubota (Associate Member, IEEE) graduated from Osaka Kyoiku University, Osaka, Japan, in 1992 and received the M.E. degree from Hokkaido University, Sapporo, Japan, in 1994 and the Dr. Eng. degree from Nagoya University, Nagoya, Japan, in 1997.

He is currently an Assistant Professor in the Department of Mechanical Engineering, Osaka Institute of Technology, Osaka, Japan. His main research interests are computational intelligence, intelligent robotic systems, and intelligent manufacturing systems.

Dr. Kubota received the IECON'96 Best Paper Award and the CIRA'97 Best Paper Award.