Regular Paper

Evolutionary Robot with Competitive-Cooperative Neural Network

Masayoshi Tabuse,† Tatsuro Shinchi,†† Akinobu Todaka††† and Tetsuro Kitazoe†††

This paper describes a new approach to control systems for autonomous mobile robots, using sandwiches of two different kinds of neural networks. One is a neural network for recognizing sensor information with mechanisms of competition and cooperation ($CCNN$), where synaptic couplings are fixed. The other is a neural network controller with adaptive synaptic couplings corresponding to genotypes in creatures and used for self-learning of wheel controls (NNC). In a computer simulation model with both of neural network, we were successful in obtaining typical types of robot that performed well in following a curved wall. The role of $CCNN$ is to decide which sensor signals to select in a noisy environment, while NNC adjusts the synaptic couplings through genetic operations so that it can transfer the outputs from $CCNN$ to the rotation of robot's wheel. A test was performed to show the superiority of $CCNN$. A robot with $CCNN$ can pass through a narrow entrance to a concave space, and is very robust with respect to several kinds of noises. We also tested a real robot by using the synaptic couplings obtained from the simulation, and showed that the robot performs well in a real environment.

1. Introduction

Many attempts have been made to develop autonomous mobile robots inspired by animals or humans, which have robust adaptation and stable behavior in complicated and changing environments. One approach along these lines is to use neural networks to connect input from sensors and output to controllers, and to adapt synaptic couplings in the networks to the environment. Many researchers have proposed evolutionary robot control systems using evolutionary adaptations of neural networks [1)~3)], genetic programming [4)], and a classifier system [5)].

There are certainly some parts of our brain which do not include a training procedure, but which use processes inherited from our parents. Some early stages of image processing do not need a learning procedure when, for instance, we are learning driving techniques. Stereovision neural networks, for example, do not need a learning procedure, since we have an automatic focusing ability without training. Even if it is thought that early vision neural networks themselves have developed by means of genetic algorithms, we may assume that their synaptic couplings can be considered fixed when training the mobile control systems of robots.

The strategy of the present paper is to divide the neural networks into two parts. One, which we call a competitive-cooperative neural network ($CCNN$), is a sensor recognition processor in which sensor input data are processed with competition and cooperation, reducing noise from the environment and reaching definite decisions on the selection of sensor data. We consider the network parameters in this part to be fixed. The other is a processor with self-training ability, called a neural network controller (NNC), where the input data from $CCNN$ are processed. After the processor NNC has been trained, it outputs signals to control a robot's motion. The synaptic couplings between input and output are considered as genotypes and trained to adapt to the environment by means of genetic algorithms.

The idea of neural networks with competition-cooperation originated from stereovision pattern recognition in early vision processing. Marr and Poggio treated stereovision in their famous calculation theory [6)]. The neural network model for stereovision was studied by Amari and Arbib, who called it a primitive competitive model [7)]. Reimann and Haken proposed a neural network with cooperation and competition [8)]. Kitazoe, et al. presented a neural network model capable of stereovision recognition of moving objects [9)]. In neural networks with competition and cooperation, competition makes only one neuron active and cooperation

† Department of Environmental Information, Faculty of Human Environment, Kyoto Prefectural University
†† Division of Information Science, Faculty of Education and Culture, Miyazaki University
††† Department of Computer Science and Systems Engineering, Faculty of Engineering, Miyazaki University

maintains the active state. In a real robot, competition is performed among neural activities corresponding to different sensors, while cooperation takes place among time-delayed neural activities of the same sensor. Thus, the robot recognizes the nearest object in the surrounding environment, maintains this recognition in response to small fluctuations of sensor values, and behaves correctly in a dynamically changing environment.

After the $CCNN$ processing, we use evolutionary algorithms for NNC. In general, evolutionary processes require a large population and a number of generations. Thus, experiments for evolutionary robotics are usually carried out in computer simulations, which are helpful for training and testing robot control systems. In this case, the simulator must include appropriate noises similar to those a real robot will encounter.

The purpose of the present paper is to study

1. how well a combined use of $CCNN$ and NNC works in enabling a mobile robot to carry out tasks,
2. what types of robots are typically obtained after self-training, and
3. what roles $CCNN$ play in enabling a mobile robot to carry out tasks and to maintain robustness in noisy environment.

The task for the mobile robot described in the present paper is to follow walls of various shapes and in a noisy environment. Many authors have investigated this task, using various methods [10)~12)]. Our work differs from theirs in that we propose a new approach to control systems for a mobile robot, using $CCNN$ and NNC with evolutionary adaptation. Therefore, we chose this particular task not only to show how well our neural network system works in a noisy environment but also so that we can develop future realistic applications. For instance, we could consider an autonomous wheelchair traveling along a corridor with sensors installed in it or autonomous mobiles (with no humans inside) that travel along a highway, guided by sensory information reflected from guardrails or some other reflective devices.

2. The Robot and the Environment

In our experiments, we use a miniature mobile robot named Khepera [13)]. Khepera is 3.2 cm height and 5.5 cm in diameter, with eight infrared sensors and two wheels controlled by independent motors, as shown in **Fig. 1**.

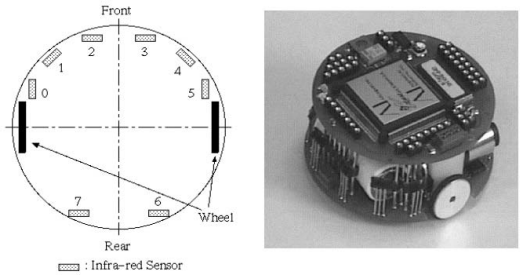


Fig. 1 Mobile robot Khepera.

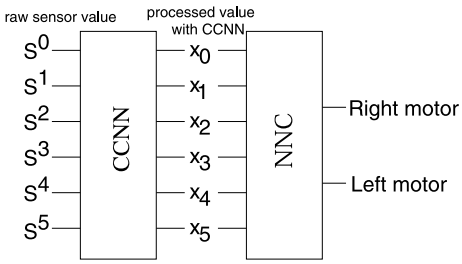


Fig. 2 Khepera's control system.

Khepera's sensors detect obstacles and return integer values from 0 to 1023 corresponding to the distances between Khepera and the obstacles. A low value means that there are no obstacles near the sensor, while a high value means that there is an obstacle close to the sensor. Khepera communicates with a computer by means of a serial line, so that the computer obtains the sensor values from Khepera and provides Khepera with the information necessary to control the wheel speed.

The control system of Khepera consists of two kinds of neural networks, $CCNN$ and NNC, shown in **Fig. 2**. $CCNN$ is a sensor recognition processor in which raw infrared sensor values S^a $(a = 0, 1, \ldots, 5)$ are input from Khepera and processed data x_a $(a = 0, 1, \ldots, 5)$ are output to NNC. NNC controls Khepera's motors according to data from $CCNN$. Synaptic couplings of NNC are trained to perform a given task by using genetic algorithms.

To train the synaptic couplings by means of genetic algorithms, we investigated robot movement in a computer simulation model. A difficult problem in computer simulations is how to simulate the real world with respect to noises. *Khepera Simulator Package ver.*2.0, by Olivier Michel [14)], is an excellent software tool that takes account of uncertainties assumed to exist in the real world. Noises of $\pm 10\%$, $\pm 10\%$ and $\pm 5\%$ are randomly added to

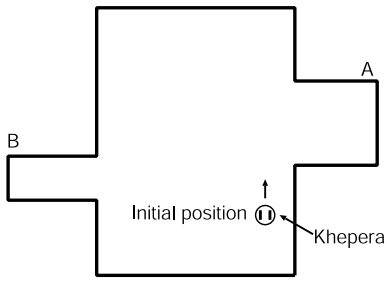


Fig. 3 Khepera's environment.

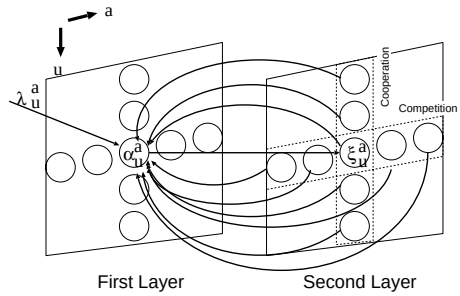


Fig. 4 Image of two-layered $CCNN$.

the amplitude of the sensor value, the amplitude of the motor speed, and the direction of Khepera resulting from the difference in speeds of the left and right motors, respectively.

As a typical example, we take a rectangular area with two concave spaces surrounded by a wall, shown in **Fig. 3**, in which Khepera is required to follow the wall counterclockwise. Khepera keeps a safe and short distance from the wall on the right, and enters a wide space A and also a narrow space B.

3. Two-Layered CCNN

We used a two-layered $CCNN$ [9] to apply a stereovision neural network to a robot sensory system that processes Khepera's sensor values. The two-layered $CCNN$ equations are as follows:

$$\tau_1 \frac{d}{dt}\alpha_u^a(t) = -\alpha_u^a(t) + A\lambda_u^a - B\sum_{a' \neq a} g(\xi_u^{a'}(t)) + D\sum_{u'=u-l}^{u} g(\xi_{u'}^a(t)), \quad (1)$$

$$\tau_2 \frac{d}{dt}\xi_u^a(t) = -\xi_u^a(t) + f(\alpha_u^a(t)). \quad (2)$$

In Eqs. (1) and (2), t denotes an internal processing time and u is an actual external time step to control a robot, where one step of u is carried out after dozens of internal steps of t carried out in numerical calculations. $\alpha_u^a(t)$ is a neural activity of the first layer to which λ_u^a is input from infrared sensor $\#a$ $(a = 0, 1, \ldots, 5)$. $\xi_u^a(t)$ is a neural activity of the second layer, which outputs x_a to CNN after a certain internal processing time. τ_1 and τ_2 are decay time constants for neural activities $\alpha_u^a(t)$ and $\xi_u^a(t)$, respectively [7]. The neural network for these equations has a two-layered structure, as shown in **Fig. 4**. $f(x)$ is a well-known sigmoid function, and $g(x)$ is a function given by Amari and Arbib [7]:

$$f(x) = \frac{1}{2}(\tanh(x - h) + 1), \quad (3)$$

$$g(x) = x^+ = \frac{1}{2}(x + |x|). \quad (4)$$

A, B, D, h, and l are positive constants, which are intended chosen appropriately.

The two-layered $CCNN$ was originally intended to realize stereovision recognition by fusing two $2D$ images from the left and right eyes. In stereovision, input similarities λ_u^a for the u-th coordinate at a certain disparity a are fed to the neural equations, which develop neural activities ξ_u^a to a stable point, enabling recognition of a definite disparity. For Khepera, we use input λ_u^a normalized by -1 to 1 as follows:

$$\lambda_u^a = 2.0 \cdot \frac{S_u^a}{1023.0} - 1.0, \quad (5)$$

where S_u^a is the value of sensor $\#a$ at an actual time step u.

In Eq. (1), the third term represents a competition with other neural activities $\xi_u^{a'}(a' \neq a)$, and the fourth term measures cooperation with past neural activities $\xi_{u'}^a$, which is executed at every step from the l-th past time $u' = u - l$ to the present $u' = u$.

The control of Khepera by means of $CCNN$ is based on the consideration that the neural activities related to Khepera's sensors compete with each other and that a sequence of time-delayed neural activities for each sensor cooperate with each other. We calculate the neural network equations competing with other neural activities and cooperating with past time activities $(a = 0, 1, \ldots, 5,\ u' = u - l, \ldots, u - 2, u - 1, u)$, and eventually obtain recognition of a particular sensor after arriving at a stable state.

The qualitative features of this neural network can be described as follows:

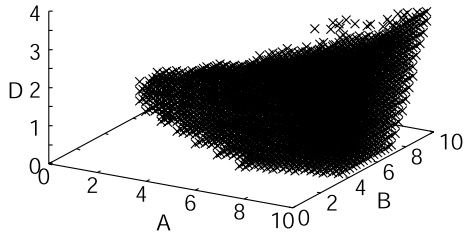


Fig. 5 Parameter ranges A, B, and D for $h = 1.0$ and $l = 6$ of $CCNN$.

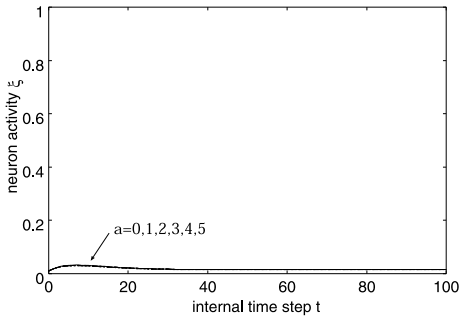


Fig. 6 Feature (2). All neural activities ξ become almost zero when all λ are equal to -0.1.

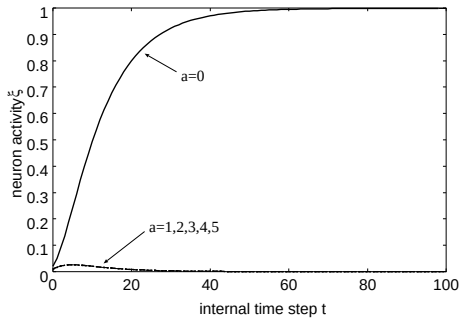


Fig. 7 Feature (3) with $\lambda = 0.3$ for $a = 0$ and $\lambda = -0.1$ for other values of a. ξ for $a = 0$ increases to almost 1, while other ξ become almost zero.

1) According to the values of λ, each ξ is active ($\xi \approx 1.0$) or inactive ($\xi \approx 0.0$).
2) When all λ have small values (≤ 0.0), all activities ξ are inactive, having values near zero.
3) When one λ has a large value ($0.0 < \lambda \leq 1.0$), it becomes active, having a value close to 1.
4) When two or more λ have large values, ξ of the largest λ is active and the others are suppressed on account of the competition term.
5) When two or more λ have very large values ($\lambda \approx 1.0$), ξ of these λ are all active.
6) Even if the value of λ for the active ξ becomes somewhat smaller, this ξ remains in an active state because of the cooperative term.

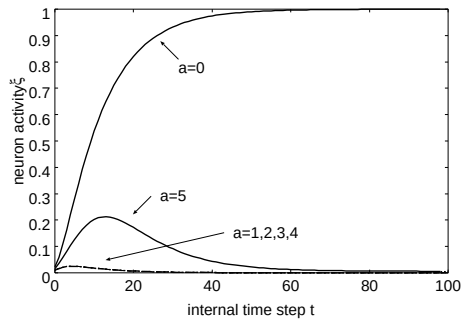


Fig. 8 Feature (4) with $\lambda = 0.4$ for $a = 0$, $\lambda = 0.3$ for $a = 5$, and $\lambda = -0.1$ for other values of a. Though both # 0 and # 5 ξ corresponding to $\lambda > 0$ increase initially, #0 ξ for the larger λ becomes almost one and others go to almost zero due to the competition term.

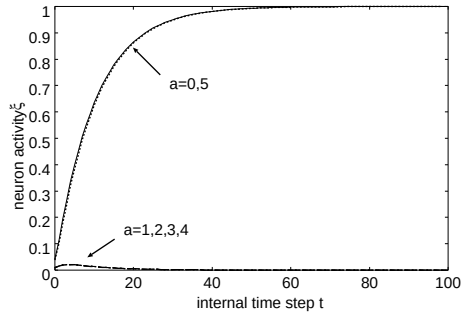


Fig. 9 Feature (5) with $\lambda = 1.0$ for $a = 0$, $\lambda = 0.9$ for $a = 5$, and $\lambda = -0.1$ for other values of a. In this case both ξ for $a = 0$ and $a = 5$ increase to almost one and others become almost zero.

These features of the neural network remain invariant in wide ranges of parameters A, B, D, h, and l, showing robustness with respect to a change of parameters. For example, **Fig. 5** shows parameter ranges of B and D for $0 < A \leq 10.0$, $h = 1.0$, and $l = 6$, where the above features (1)–(6) are satisfied. We show the features (2)–(6) in **Figs. 6**, **7**, **8**, **9**, **10**, **11** by using a typical set of parameters: $A = 8.0$, $B = 4.0$, $D = 2.0$, $h = 1.0$, and $l = 6$. Stable solutions of Eqs. (1) and (2) do not depend on the values of τ_1 and τ_2. We choose $\tau_1 = \tau_2 = 1$, and take 50 steps of t in numerical calculations hereafter so that neural activities take values close to the stable points, as shown in Figs. 6–11.

When Khepera with $CCNN$ is applied to a real environment, the above features (1)–(6) provide a clear-cut signal processing. While

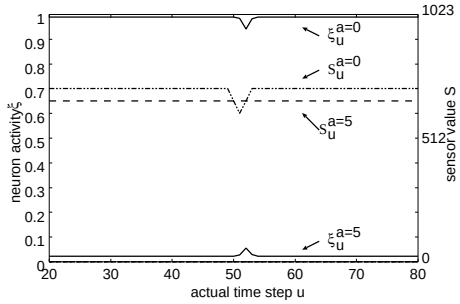


Fig. 10 Feature (6). From the beginning the sensor value for $a = 0$ is kept larger than for $a = 5$ ($S_u^{a=0} = 717$ and $S_u^{a=5} = 665$), so that ξ is active for $a = 0$ and inactive for other values of a. At actual time steps $u = 50$ to 52, $S_u^{a=0}$ is smaller than $S_u^{a=5}$. Nevertheless, $\xi_u^{a=0}$ remains active and $\xi_u^{a=5}$ does not become active due to the cooperative term, though the value of $\xi_u^{a=0}$ decreases slightly at the time interval.

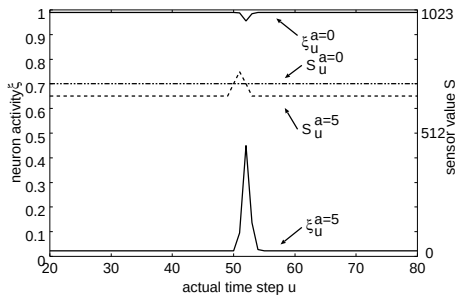


Fig. 11 Feature (6). From the beginning the sensor value for $a = 0$ is kept larger than for $a = 5$ ($S_u^{a=0} = 717$ and $S_u^{a=5} = 665$), so that ξ is active for $a = 0$ and inactive for other values of a. At actual time steps $u = 50$ to 52, $S_u^{a=5}$ becomes greater than $S_u^{a=0}$. Nevertheless, $\xi_u^{a=0}$ remains active and $\xi_u^{a=5}$ does not become active due to the cooperative term, though the value of $\xi_u^{a=0}$ decreases slightly.

each raw sensor returns a value simultaneously, as shown in **Fig. 12**, almost only one neural activity returns a high value after processing with $CCNN$, as shown in **Fig. 13**. Therefore, Khepera with $CCNN$ has the ability to decide or choose the sensor value to which Khepera should react.

4. Evolutionary Adaptation

To train robot control systems, we perform adaptation under a computer simulation model of a robot and its environment. As shown in **Fig. 14**, the data $x_0, x_1, x_2, \ldots, x_5$ output from $CCNN$ are fed to second neural networks with self-training ability. The synaptic couplings in NNC are then revised by genetic algorithms.

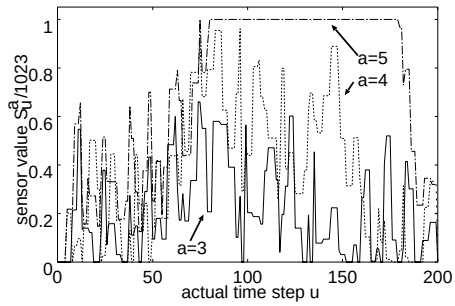


Fig. 12 Raw sensor values of Khepera in a real environment. Values of $S_u^a/1023$ for $a = 3, 4, 5$ are drawn as solid lines, dotted lines, and broken lines, respectively.

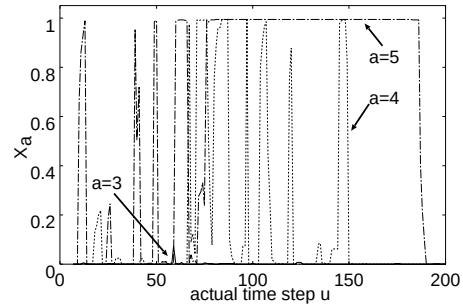


Fig. 13 Values after processing with $CCNN$ in a real environment. Values of x_a for $a = 3, 4, 5$ are drawn as solid lines, dotted lines, and broken lines, respectively.

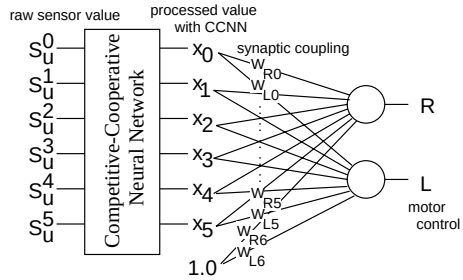


Fig. 14 $CCNN$ and NNC. The data x_0–x_5 output from $CCNN$ are fed to NNC.

The control signals to the right wheel and left wheel, R and L, and the speeds of the right motor and left motor, V_R and V_L, are given by

$$R = F\left(\sum_{i=0}^{5} W_{Ri} \cdot x_i + W_{R6}\right), \tag{6}$$

$$L = F\left(\sum_{i=0}^{5} W_{Li} \cdot x_i + W_{L6}\right), \tag{7}$$

$$F(x) = \tanh(x), \tag{8}$$

$$V_R = V_{max} \cdot R, \tag{9}$$
$$V_L = V_{max} \cdot L. \tag{10}$$

W_{Ri} and W_{Li} $(i = 0, 1, \ldots, 5)$ are synaptic couplings, as shown in Fig. 14, connecting data from $CCNN$ with the control of the right or left motor. W_{R6} and W_{L6} are thresholds, which adjust the values of sigmoid function (8). V_{max} is a maximum speed of the right and left motors.

We determine W_{Ri} and W_{Li} by using the genetic algorithms [15)~17)]. The algorithms for obtaining the best genes are as follows:

1) Make N_1 robots with randomly generated synaptic couplings and let them run in the area shown in Fig. 3 for a certain period.
2) Make new N_2 robots from the old N_1 robots by using genetic algorithms in which the synaptic couplings of new robots are generated by real-coded genetic algorithms [18)], in which α of BLX-α is set to 0.5. Then, let them run for the same period as in step 1.
3) Measure all the robots $N_1 + N_2$ by using a given evaluation function, and choose N_1 robots with the highest scores. If the total score of the N_1 robots exceeds a given threshold, stop the loop; otherwise go to step 2.

5. Experiments

We investigate movements along a wall in order to estimate the possibility of controlling Khepera by means of combined use of $CCNN$ and NNC with genetic algorithms. Sensor #5 has a crucial role in counterclockwise movement along a wall. If x_5 is active, it is near a wall, so it should follow the wall. Since we do not need independent information from each sensor to perform the task of following a wall, we can degenerate variables x_i to X_i, which are defined as follows:

$$X_0 = x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4, \tag{11}$$
$$X_1 = x_5, \tag{12}$$

where $\oplus$ means that $X_0 = x_0 + x_1 + x_2 + x_3 + x_4$ for $x_0 + x_1 + x_2 + x_3 + x_4 \leq 1.0$ and $X_0 = 1.0$ for $x_0 + x_1 + x_2 + x_3 + x_4 > 1.0$. Let us substitute Eqs. (11) and (12) into Eqs. (6) and (7), as follows:

$$R = F\left(\sum_{i=0}^{1} W_{Ri} X_i + W_{R2}\right), \tag{13}$$
$$L = F\left(\sum_{i=0}^{1} W_{Li} X_i + W_{L2}\right). \tag{14}$$

Table 1 Coupling values of the best 10 robots after 100 generations.

No.	W_{R0}	W_{R1}	W_{R2}	W_{L0}	W_{L1}	W_{L2}	fitness
1	1.06	1.18	−0.05	−2.54	0.09	1.50	7685.05
2	0.96	1.13	0.02	−2.64	0.15	1.52	7679.29
3	1.02	1.18	−0.04	−2.46	0.01	1.47	7636.54
4	1.11	1.20	−0.04	−2.85	0.56	1.48	7623.72
5	4.80	1.11	−0.03	−3.11	0.65	1.51	7620.40
6	0.91	1.12	−0.06	−2.51	0.09	1.48	7612.27
7	4.47	1.08	−0.08	−2.95	0.41	1.55	7602.17
8	4.70	1.04	0.05	−2.94	0.64	1.48	7598.13
9	1.26	0.99	0.01	−3.38	0.99	1.64	7585.33
10	1.19	0.98	0.05	−3.50	0.91	1.64	7581.13

Table 2 Typical behaviors of the best 10 robots in Table 1.

$X_0 \setminus X_1$	0	1
0	Sharp right	Wide right
1	Sharp left	Sharp left

The evaluation function in genetic algorithms is given as

$$g = \frac{|V_R + V_L|}{2.0} \cdot \left(1 - \frac{|(V_R + 1) - V_L|}{2V_{max} + 1.0}\right) \cdot \left(1.0 - \frac{1}{5}\sum_{a=0}^{4} \frac{S_u^a}{1023.0}\right) \cdot \left(\frac{V_R + V_L + 2V_{max}}{4V_{max}}\right) \cdot S_u^5/1023.0. \tag{15}$$

Each term in Eq. (15) evaluates the robot performance from different points of view, measuring Khepera's speed, counterclockwise rotation, movements without obstacles on the X_0 side, going forward and following a wall to its right side. The evaluation function g has a high value if a robot follows a wall without colliding with anything and if it moves forward as fast as possible. We set $A = 8.0$, $B = 4.0$, $D = 2.0$, and $l = 6$ in Eq. (1) and $h = 1.0$ in Eq. (3). We take $N_1 = N_2 = 20$ and $V_{max} = 8$. The evaluation function is calculated for each 2000 step run of a robot.

5.1 Behavior Type by Evolutionary Adaptation

As a result of evolutionary adaptation, the coupling values of the best 10 robots are shown in **Table 1** together with the fitness value, which is the sum of g for 2000 steps. It is interesting to see that these best 10 robots have almost the same behaviors, which are shown in **Table 2**. The data x_i $(i = 0, 1, \ldots, 5)$ output from $CCNN$ show that all values of x_i are almost zero or that one value of x_i is almost 1

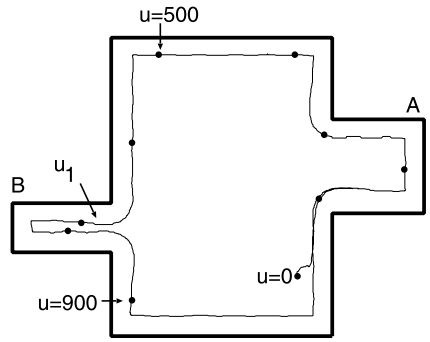


Fig. 15 Trail of a robot with $CCNN$, which takes about 1200 simulation steps to complete a circuit. Dots on the line indicate intervals of 100 steps, and u_1 corresponds to Fig. 16.

and the others are almost zero. Therefore, we can assume that X_0 and X_1 have values of 0 or 1. The robot reacts to obstacles on two sides of Khepera: the right side (X_1 Side) and the left or front side (X_0 Side). A robot moves with sharp right-handed rotation if it does not encounter any obstacle $(X_0, X_1) = (0, 0)$. It moves with sharp left-handed rotation if it encounters obstacles on both sides $(1, 1)$ or only on the X_0 Side $(1, 0)$. When a robot encounters obstacles only on the X_1 Side $(0, 1)$, it moves with wide right-handed rotation.

5.2 Effects of *CCNN*

In order to see the effects of $CCNN$, we investigate the behaviors of Khepera with $CCNN$. **Figure 15** shows the trail of a robot in environment with some concave spaces of varying sizes that are wide enough for a robot to enter. The widths of the entrances to space A and space B are four body lengths and two body lengths of Khepera. The values in Fig. 15 show the step number after the commencement of a simulation. The behavior shown in Fig. 15 occurs with two neural networks, $CCNN$ and NNC. In NNC, the values x_a are processed with the self-adapted coupling values. The movement in Fig. 15 is presented with the best set of coupling values in Table 1, which is the typical one with the coupling values shown in Table 1. A robot can follow a wall and enter both the wide space A and the narrow space B. Though information from many sensors tends to confuse a robot moving in general, this robot moves smoothly alongside a curved wall. Note the movements in the narrow concave space B, where the robot reaches the inner part of the space and exits from the space.

The four graphs each in **Fig. 16** (a) and 4

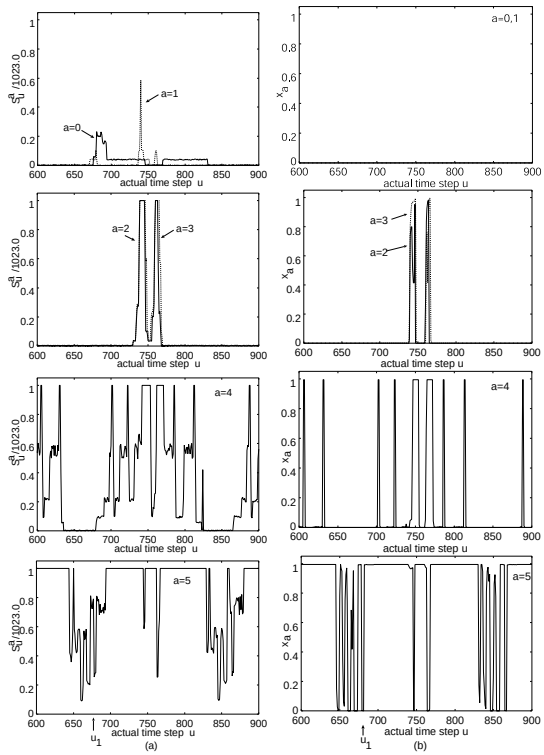


Fig. 16 Time-series value of a sensor. The four graphs in (a) and (b) are obtained from the robot movement with $CCNN$, as shown in Fig. 15. In (a) the values are directly returned from the robot's raw sensor data. In (b), x_a are values of S_u^a processed with $CCNN$. u_1 corresponds to the time in Fig. 15.

graphs in Fig. 16 (b) show the sensor values S_u^a of the robot and x_a after S_u^a have been processed with $CCNN$, respectively, corresponding to Fig. 15. As Fig. 16 (a) indicates, the robot has sensor values for $a = 0$ at $u_1 \approx 680$ when the robot reaches the entrance of space B and detects the wall on the left-hand side of the robot. Therefore, if the sensor values S_u^a are output to NNC directly, the robot turns sharp left and cannot enter space B. On the other hand, the processed values with $CCNN$ become zero for $a = 0$ at $u_1 \approx 680$, because of the competitive and cooperative terms of $CCNN$, and the robot turns right. In other words, the robot with $CCNN$ can enter concave space B and keep following a wall effectively.

5.3 Robustness of a Robot's Behavior in a Noisy Environment

We examine the robustness of a robot's behavior with $CCNN$ in a noisy environment. A random noise is added to the sensor value of a robot, where it takes a value ranging be-

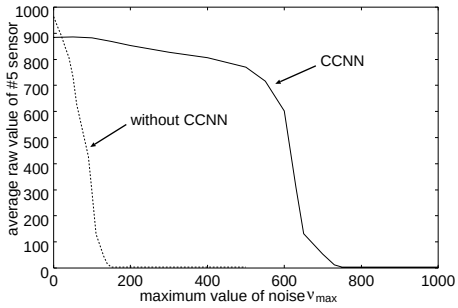


Fig. 17 Average raw value of sensor #5 corresponding to the maximum value of noise ν_{max}. The solid line, which remains high for large amounts of noise, represents a value of the robot with $CCNN$, while the dotted line, which decreases rapidly, represents the value of a robot without $CCNN$.

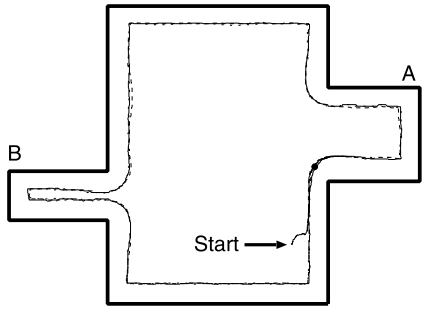


Fig. 18 Trails of a robot with $CCNN$, which takes about 1200 simulation steps to complete the circuit. The sold, dotted, and broken lines denote the trails for $\nu_{max} = 0, 50$, and 100, respectively.

tween 0 and the maximum value ν_{max}. If a sensor value is over 1023, it is set to 1023. Since the amount of noise is represented by ν_{max}, we measure the average raw value of sensor #5 by changing ν_{max}. For the robot with $CCNN$, we use the same coupling values as in the previous subsection. For the robot without $CCNN$, we use the best set of adaptive coupling values in NNC, which are $W_{R0} = 4.74$, $W_{R1} = 0.65$, $W_{R2} = -0.28$, $W_{L0} = -3.90$, $W_{L1} = 1.45$, and $W_{L2} = 1.80$, obtained in a computer simulation of the robot without $CCNN$. In **Fig. 17**, we see that the average value of sensor #5 of a robot without $CCNN$ decreases rapidly, while that of one with $CCNN$ maintains a high value and decreases slowly. Since the value of sensor #5 represents the distance between the right-hand side of a robot and a wall, a high value of sensor #5 means that a robot behaves very well in following a wall. The actual trails of a robot with and without $CCNN$ in a noisy environment are also shown in **Figs. 18** and **19**, respectively. A robot with $CCNN$ has good behavior in following a wall for noises $\nu_{max} = 0, 50, 100$. On the other hand, a robot without $CCNN$ fails in the movement task of following a wall for $\nu_{max} = 50, 100$. These results show that a robot with $CCNN$ has much more robust behavior in a noisy environment than one without $CCNN$.

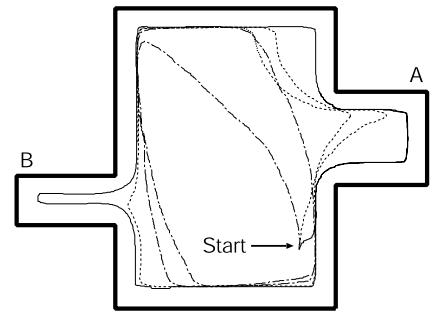


Fig. 19 Trails of a robot without $CCNN$, which takes about 1200 simulation steps to complete the circuit. The solid, dotted, and broken lines denote the trails for $\nu_{max} = 0, 50$, and 100, respectively.

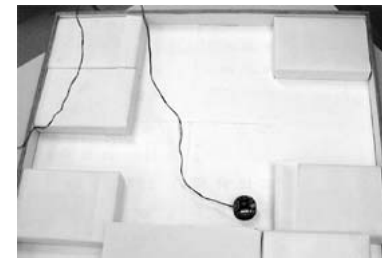


Fig. 20 Real environment.

5.4 Experiments in a Real Environment

We investigate Khepera's behaviors in a real environment, using synaptic couplings obtained from the simulation. All the parameters of $CCNN$ and NNC are set to the same values as in the simulation. **Figure 20** shows a real environment, in which the color of all walls is white, because white enables Khepera's sensors to detect obstacles more easily. The widths of the entrances to spaces A and B are 27 cm and 12 cm, respectively. Khepera is controlled by a workstation (Sun UltraSparc 450 MHz) through a serial line. **Figures 21** and **22** show the trails of Khepera with and without $CCNN$, respectively, in a real environment. The solid and dotted lines indicate the trails under natural light from windows and the light of fluorescent lamps

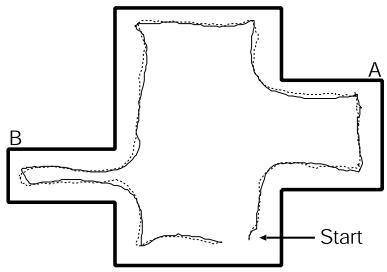


Fig. 21 Trails of Khepera with $CCNN$ in the real environment, where it takes about 6000 actual time steps (60 seconds) to complete a circuit. The solid and dotted lines show the trails under natural light and the light of fluorescent lamps, respectively.

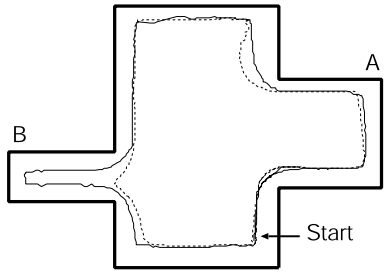


Fig. 22 Trails of Khepera without $CCNN$ in the real environment, where it takes about 6000 actual time steps (60 seconds) to complete a circuit. The solid and dotted lines show the trails under natural light and the light of fluorescent lamps, respectively.

in the ceiling, respectively. The light of fluorescent lamps fluctuates temporally, so that it adds to the noise detected by Khepera's sensors. Figure 12 shows the sensor values of Khepera, whose fluctuation is the effect of the light of fluorescent lamps and reaction to the walls.

Khepera with $CCNN$ follows along a wall and enters both wide and narrow spaces, A and B, very well in both natural light and the light of fluorescent lamps. On the other hand, Khepera without $CCNN$ cannot enter a narrow space B and does not follow a wall correctly under the light of fluorescent lamps. These results are similar to those in the simulation, and we think Khepera with $CCNN$ performs better both in the simulation and in the real environment.

6. Discussion on *CCNN* Parameters

Finally, we discuss the parameters of $CCNN$. In biological development of multistage neural networks with different functions, it seems natural to assume that, each has been trained sequentially rather than that all were trained in parallel. In the present paper, the parameters A, B, D, and h for $CCNN$ are set to typical values showing the features (1)–(6) in Section 3, because we think that the sensor recognition neural network already had competitive and cooperative features before the NNC network was trained. However, in order to optimize from the computational point of view, it seems meaningful to train $CCNN$ and NNC in parallel by using evolutionary adaptation, though it takes a lot of CPU time. Thus, we may determine these parameters by using genetic algorithms.

We tested an approach in which the parameters of $CCNN$ and the synaptic couplings of NNC are determined in parallel by using genetic algorithms in the same environment as the previous computer simulation described in Section 5. We set $N_1 = N_2 = 100$ and $l = 6$. The parameters and coupling values of the best 10 robots after evolutionary adaptation are shown in **Table 3** together with the fitness value, which is the sum of g for 2000 steps. Typical behaviors of these robots are shown in **Table 4**. We find that Khepera was adapted to the environment, having the same features as and similar effectiveness to the robots listed in Tables 1 and 2, and that it performed a wall-following

Table 3 Parameters and coupling values of the best 10 robots after 100 generations.

No.	A	B	D	h			
	W_{R0}	W_{R1}	W_{R2}	W_{L0}	W_{L1}	W_{L2}	fitness
1	17.32	5.33	0.91	3.93			
	1.95	1.19	0.09	−2.90	0.53	1.47	7881.31
2	12.13	3.64	0.85	4.41			
	2.25	1.32	−0.08	−2.98	0.51	1.54	7796.91
3	21.28	4.26	1.29	5.18			
	2.12	0.83	0.21	−3.04	0.59	1.57	7746.14
4	12.84	4.03	0.12	6.00			
	2.50	0.96	0.10	−2.83	0.68	1.43	7741.23
5	28.50	6.73	1.31	5.75			
	1.51	1.24	−0.07	−2.86	0.62	1.51	7709.41
6	11.31	4.61	0.06	2.96			
	2.47	0.94	0.12	−3.28	0.87	1.44	7697.68
7	9.16	4.12	0.59	3.03			
	2.40	1.01	0.07	−3.16	0.94	1.44	7683.49
8	6.05	2.75	0.09	1.77			
	0.63	0.92	0.18	−2.83	0.81	1.38	7664.65
9	10.21	4.33	1.43	1.00			
	1.46	1.01	−0.00	−3.07	1.16	1.166	7657.10
10	9.60	4.20	0.51	3.17			
	2.33	1.06	0.10	−3.36	0.82	1.56	7649.83

Table 4 Typical behaviors of the best 10 robots in Table 3.

$X_0 \setminus X_1$	0	1
0	Sharp right	Wide right
1	Sharp left	Sharp left

task very well in this approach. We conclude that a parallel adaptation of $CCNN$ and NNC does not lead to a drastic improvement over a sequential adaptation of $CCNN$ and NNC.

7. Conclusion

We have presented a control system for a mobile robot using a competitive and cooperative two-layered neural network ($CCNN$) and also a self-adaptive neural network (NNC). The competitive term makes only one neuron ξ active for large input values and the cooperative term makes ξ maintain the active state in response to a small fluctuation of the input value. As a result, a self-learning neural network created wall following behavior by using ξ output from the two-layered network. We found that Khepera, controlled by $CCNN$, showed robust behaviors in response to many kinds of noises while following a curved wall. Moreover, we have found the robot with $CCNN$ follows a wall smoothly and does not fail to pass through a small entrance. We also tested Khepera in a real environment, using the synaptic couplings obtained from the simulation. Khepera with $CCNN$ followed a wall and entered a wide space A and a narrow space B, while without $CCNN$ it failed to perform well a the real environment. We conclude that a robot with a two-layered competitive-cooperative neural network and self-adaptive neural network is capable of very effective movements in a complex environment. The success of the present approach suggests future applications in various kinds of environment.

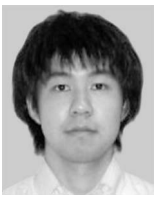
References

1) Floreano, D. and Mondada, F.: Evolutionary neurocontrollers for autonomous mobile robots, *Neural Networks*, Vol.11, pp.1461–1478 (1998).
2) Nolfi, S. and Floreano, D.: *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, MIT Press (2000).
3) Harvey, I., Husbands, P., Cliff, D., Thompson, A. and Jakobi, N.: Evolutionary Robotics: the Sussex Approach, *Robotics and Autonomous Systems*, Vol.20, pp.205–224 (1997).
4) Brooks, R.A.: Artificial Life and Real Robots, *Proc. First European Conference on Artificial Life*, pp.3–10 (1992).
5) Dorigo, M. and Colombetti, M.: *Robot Shaping: An Experiment in Behaviour Engineering*, MIT Press (1998).
6) Marr, D. and Poggio, T.: Cooperative Computation of Stereo Disparity, *Science*, Vol.194, pp.283–287 (1976).
7) Amari, S. and Arbib, M.A.: Competition and Cooperation in Neural Nets, *Systems Neuroscience*, pp.119–165 (1977).
8) Reimann, D. and Haken, H.: Stereo Vision by Self-Organization, *Biol. Cybern.*, Vol.71, pp.17–26 (1994).
9) Kitazoe, T., Tomiyama, J., Yoshitomi, Y. and Shii, T.: Sequential Stereoscopic Vision and Hysteresis, *Proc. Fifth International Conference on Neural Information Processing*, pp.391–396 (1998).
10) Braunstingl, R., Sanz, P. and Ezkerra, J.M.: Fuzzy Logic Wall Following of a Mobile Robot Based on the Concept of General Perception, *ICAR '95*, pp.367–376 (1995).
11) Yata, T, Kleeman, L. and Yuta, S.: Wall Following Using Angle Information Measured by a Single Ultrasonic Transducer, *Proc. 1998 IEEE International Conference on Robotics and Automation*, pp.1590–1596 (1998).
12) Ebner, M. and Zell, A.: Evolving a Behavior-Based Control Architecture—From Simulations to the Real World, *Proc. Genetic and Evolutionary Computation Conference*, pp.1009–1014 (1999).
13) K-Team SA: Khepera User Manual Version 4.06. (1995).
14) Michel, O.: *Khepera Simulator*, Package Version 2.0. Freeware Mobile Robot Simulator. Downloadable from the World Wide Web at http://diwww.epfl.ch/lami/team/michel/khep-sim/
15) Holland, J.H.: *Adaptation in Natural and Artificial Systems*, Univ.of Michigan Press (1975).
16) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley (1989).
17) Mitchell, M.: *An Introduction to Genetic Algorithms*, MIT Press (1996).
18) Eshelman, L.J. and Schaffer, J.D.: Real-Coded Genetic Algorithms and Interval-Schemata, *Foundations of Genetic Algorithms 2*, pp.187–202 (1993).

(Received September 24, 2002)
(Accepted July 3, 2003)

Masayoshi Tabuse was born in 1959. He received his M.S. and Ph.D. degree from Kobe University in 1985 and 1988 respectively. He had been working in Miyazaki University since 1992 and has been working in Kyoto Prefectural University as an associate professor since 2003. His current research interests are evolutionary robotics, evolutionary computation and multiagent systems. He is a member of IPSJ, IEICE and PSJ.

Tatsuro Shinchi was born in 1963. He received Master of School Education degree from Hyogo Univ. of Teacher Education in 1994, and Dr. of Eng. degree from Miyazaki Univ. in 2003. He had worked as a teacher of public senior high school in Kagoshima pref. since 1986. He had been working in Faculty of Education, Miyazaki Univ. as a lecturer since 1994, and has been working in Faculty of Education and Culture, Miyazaki Univ. as an associate professor since 1999. His current research interests are science of complexity, analyses and simulations of complex behavior seen in nature, multiagent systems and educational technology. He is a member of IPSJ, IEICE, Japan Society for Educational Technology (JET).



Akinobu Todaka was born in 1977. He received his M.E. degree from Miyazaki University in 2002. He has been working in Nissin Software Corp. since 2002.

Tetsuro Kitazoe was born in 1937. He received Dr. Sci. degree from Dept. of Physics, Osaka University in 1966. He was an assistant professor from 1966 to 1972 and an associate professor from 1972 to 1991 at Dept. of Physics, Kobe University. He moved to Dept. of Computer Science and Systems Engineering, Miyazaki University as a professor. His current research interests are visual neural network, neural network and speech recognition, robotics and neural networks, and cosmology. He is a member of IPSJ, RSJ and PSJ.