Cogn Neurodyn (2010) 4:69–80
DOI 10.1007/s11571-009-9101-5

ORIGINAL RESEARCH

Chaotic neural network applied to two-dimensional motion control

Hiroyuki Yoshida · Shuhei Kurata · Yongtao Li · Shigetoshi Nara

Received: 22 April 2009 / Revised: 13 November 2009 / Accepted: 24 November 2009 / Published online: 11 December 2009

Abstract Chaotic dynamics generated in a chaotic neural network model are applied to 2-dimensional (2-D) motion control. The change of position of a moving object in each control time step is determined by a motion function which is calculated from the firing activity of the chaotic neural network. Prototype attractors which correspond to simple motions of the object toward four directions in 2-D space are embedded in the neural network model by designing synaptic connection strengths. Chaotic dynamics introduced by changing system parameters sample intermediate points in the high-dimensional state space between the embedded attractors, resulting in motion in various directions. By means of adaptive switching of the system parameters between a chaotic regime and an attractor regime, the object is able to reach a target in a 2-D maze. In computer experiments, the success rate of this method over many trials not only shows better performance than that of stochastic random pattern generators but also shows that chaotic dynamics can be useful for realizing robust, adaptive and complex control function with simple rules.

Keywords Constrained chaos · Chaotic neural network · Motion control · 2-Dimensional maze · Ill-posed problem

H. Yoshida · S. Kurata · S. Nara
Department of Electrical & Electronic Engineering,
Faculty of Engineering, Okayama University,
Okayama 700-8530, Japan

Y. Li (✉) · S. Nara
Graduate School of Natural Science and Technology,
Okayama University, 3-1-1 Tsushima-naka,
Okayama 700-8530, Japan
e-mail: li@chaos.elec.okayama-u.ac.jp

Introduction

The great progress of modern very large scale integrated circuit (VLSI) technologies has produced revolutionary devices and machines. However, algorithms on conventional computers run into problems with combinatorial explosion and program complexity in realizing flexible functions when there are too many degrees of freedom to control. On the other hand, recent progress in the study of biological information and control processing, particularly brain functions, suggests that they might be based on novel *dynamical mechanisms* that result in excellent functioning and control (Skarda and Freeman 1987; Haken 1988, 1996; Nara and Davis 1992; Aertsen et al. 1994; Tokuda et al. 1997; Fujii et al. 1996; Kay et al. 1996; Tsuda 2001). Our key idea is to somehow harness the onset of complex nonlinear dynamics in information processing or control systems. This idea arose from the observations of chaos in biological systems, which suggested that chaotic dynamics could have important potentiality for complex processing and control in situations where environments give systems simple evaluations or responses including "uncertainties" which result in complex dynamics that we call "constrained chaos" (Nara 2003) or constrained "chaotic itinerancy" (Kaneko and Tsuda 2000, 2001). Thus, it is our primary motivation for studying chaotic dynamics in neural networks from the functional point of view. As an example of such functionality, Nara and Davis proposed that chaotic dynamics which occur in a recurrent binary neuron network by changing a system parameter (connectivity) can be applied to solving ill-posed problems such as memory search or synthesis. In these tasks, certain dynamical structures play important roles, as described in previous works (Mori et al. 1989; Nara and Davis 1992, 1997; Nara et al. 1993, 1995; Kuroiwa et al. 1999; Nara 2003).

Furthermore, this approach was extended to an application of chaotic dynamics in adaptive motion control. Two novel problems, which were set as ill-posed problems, were investigated. One is the problem of controlling an object to reach a target in a 2-D maze, and another is the problem to controlling an object to capture a target moving in 2-D space, which were successfully executed in computer experiments and reported in Suemitsu and Nara 2004 and Li and Nara 2008, respectively. The neural network model used in these studies is based on a binary neuron model, and chaos occurs at some parameter values where the internal field applied to each neuron exhibits large dynamical fluctuations. In order to investigate the potential usefulness of chaos, we have extended the approach to two-dimensional cellular automata, which is one of many dynamical models. Chaotic dynamics in a two-dimensional cellular automata was successfully applied to solving 2-D mazes based on the idea of harnessing of chaos. From the viewpoint of functional aspects, both dynamical models show almost equivalent results (Takada et al. 2007). These studies not only indicate that the role of chaos is critically important to realize complex control via simple rules in particular situations, but also enables us to speculate about the usefulness of chaos in general, in the sense that the functional features do not depend strongly on the detailed structure of models.

In this paper, we extend these considerations to other neuron models. As well known, there are many neural network models which could cause chaotic dynamics. However, in order to make the study more general, we choose the "chaotic neuron model" proposed by Aihara et al. (Aihara et al. 1990; Adachi and Aihara 1997) as a typical example of a neuron model in which even a single neuron can generate chaotic behavior. Using a kind of pseudo-inverse method (orthogonalized learning method), multiple limit-cycle attractors are embedded into a recurrent neural network model consisting of chaotic neurons. By appropriately choosing values of parameters, either attractor dynamics or chaotic dynamics can be introduced into the network. A simple coding method is employed to transform the high dimensional dynamics in the network model into low dimensional complex motions with great robustness. Using the idea of adaptive switching of parameter values, chaotic dynamics are used to get an object to reach a target in a 2-D maze, which is a typical example of an ill-posed problem. The results of computer experiments indicate that, in the case of appropriate parameter values, chaotic dynamics has quite better performance in the control task than a stochastic random process. Comparing the dynamical structure of chaotic dynamics with that of stochastic random process from a statistical point of view we show why the chaotic dynamics are potentially useful for solving 2-D mazes.

Recurrent neural network model with chaotic neurons

In this paper we employ a recurrent neural network model consisting of chaotic neurons and with the network structure shown in Fig. 1. The dynamics of the ith neuron is described as follows (Aihara et al. 1990):

$$S_i(t+1) = f\left[\sum_{j=1}^{N} V_{ij} \sum_{d=0}^{t} k_e^d A_j(t-d) + \sum_{j=1}^{N} W_{ij} \sum_{d=0}^{t} k_f^d S_j(t-d) - \alpha \sum_{d=0}^{t} k_r^d g\{S_i(t-d)\} - \Theta_i\right] \quad (1)$$

where V_{ij} and W_{ij} are the synaptic weight of the connection to the ith neuron from the jth external input and from the jth neuron, respectively. ($W_{ii} = 0$ since each neuron has no feedback from itself). A_j is the jth external input, and k_e, k_f and k_r are the decay parameters for the external inputs, the feedback inputs, and the refractoriness, respectively. α is the refractory scaling parameter, and Θ_i is the threshold of the ith neuron. $f(\bullet)$ and $g(\bullet)$ are the activation function and the refractory function of the neuron, respectively.

When the external input A_j is assumed to be constant, the external input term can be included in the threshold. Therefore, we define a new threshold a_i that denotes the sum of threshold and the temporally constant external input term, and transform Eq. (1) into the following reduced and simultaneous form with only two internal states (Adachi and Aihara 1997):

$$S_i(t+1) = f(\zeta_i(t+1) + \eta_i(t+1)) \quad (2)$$

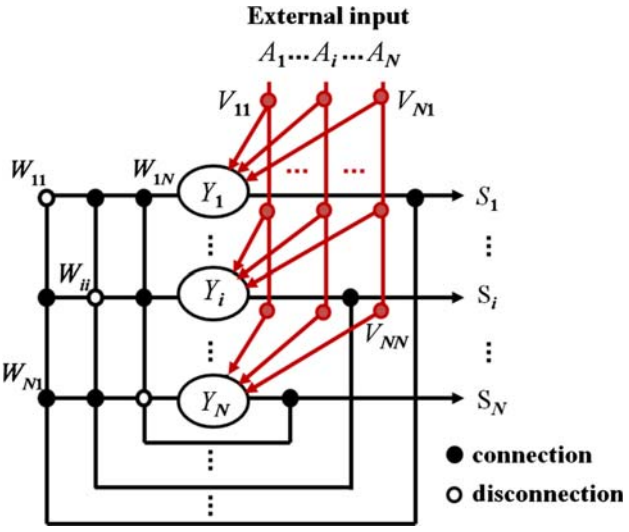


Fig. 1 A recurrent neural network model consisting of N chaotic neurons: in the connection grid, a *black dot* indicates that there is a connection, a *white dot* indicates that there is no connection

$$\zeta_i(t+1) = \sum_{j=1}^{N} W_{ij} S_j(t) + k_f \zeta_i(t) \tag{3}$$

$$\eta_i(t+1) = k_r \eta_i(t) - \alpha g\{S_i(t)\} + a_i \tag{4}$$

$$a_i = \sum_{j=1}^{N} V_{ij} \sum_{d=0}^{t} k_e^d A_j(t-d) - (1-k_r)\Theta_i \tag{5}$$

In this paper, a new internal state $Y_i(t)(= \zeta_i(t) + \eta_i(t))$ is employed. In order to use the chaotic neural network for control, we use the following simplified form with only one internal state.

$$S_i(t+1) = f(Y_i(t+1)) \tag{6}$$

$$Y_i(t+1) = kY_i(t) + \sum_{j=1}^{N} W_{ij} S_j(t) - \alpha S_i(t) + a_i \tag{7}$$

$$f(Y) = \frac{1}{1+\exp(-Y/\varepsilon)} \tag{8}$$

where $g(u) = u$ is employed as the refractory function and k denotes $k_r = k_f$. The activation function is a sigmoid function, so each neuron is represented by continuous variables $0 < S_i < 1$. $i = 1 \sim N$ and N is the total number of neurons.

The dynamics of the network depends on the synaptic weight matrix $\{W_{ij}\}$ and the four parameters (α, k, ε, a_i). When W_{ij} is appropriately determined, choosing appropriate values of these parameters can introduce either attractor dynamics or chaotic dynamics. Next, let us introduce a method to determine W_{ij} so that multiple cycle attractors can be embedded into the network. The synaptic weight matrix is defined in terms of a set of patterns $\{\boldsymbol{\xi}_\mu^\lambda \mid \mu = 1 \sim M, \lambda = 1 \sim L\}$ as

$$\boldsymbol{W} = \sum_{\mu=1}^{M} \sum_{\lambda=1}^{L} \boldsymbol{\xi}_\mu^{\lambda+1} \otimes \boldsymbol{\xi}_\mu^{\lambda\dagger} \tag{9}$$

where $\boldsymbol{\xi}_\mu^{L+1} = \boldsymbol{\xi}_\mu^1$, M is the number of cycles and L (≥ 3) is the number of patterns per cycle, or the cycle period. Each neuron has no feedback from itself, so W_{ii} is set to zero. Each embedded pattern $\boldsymbol{\xi}_\mu^\lambda$ is a state vector with $\xi_{\mu,i}^\lambda = \pm 1$ ($i = 1 \sim N$). We assume that the total number of embedded patterns is much less than the number of neurons, K ($= M \times L$) $\ll N$. $\boldsymbol{\xi}_\mu^{\lambda\dagger}$ is the conjugate vector of $\boldsymbol{\xi}_\mu^\lambda$, which is obtained by the pseudo-inverse method, also known as the orthogonalized learning method (Amari 1977; Domany et al. 1991).

Let us give a brief description about the pseudo-inverse method. We take $\boldsymbol{\xi}_\nu (\nu = 1 \sim K)$ as the original set of patterns which we intend to embed as attractors, where we renumber $\{\boldsymbol{\xi}_\mu^\lambda \mid \mu = 1 \sim M, \lambda = 1 \sim L\}$ as $\{\boldsymbol{\xi}_\nu \mid \nu = 1 \sim K (= M \times L)\}$. Then the conjugate vectors $\boldsymbol{\xi}_\nu^\dagger$ are defined as

$$\boldsymbol{\xi}_\nu^\dagger = \sum_{\gamma=1}^{K} a_{\nu\gamma} \boldsymbol{\xi}_\gamma \tag{10}$$

where $\boldsymbol{a}$ is the $K \times K$ matrix obtained as the inverse of the overlap matrix, which is defined by

$$\boldsymbol{a} = \mathbf{o}^{-1}, o_{\nu\beta} = \boldsymbol{\xi}_\nu \bullet \boldsymbol{\xi}_\beta \tag{11}$$

Using $\boldsymbol{\xi}_\nu^\dagger$, the inner products with the original attractor patterns $\boldsymbol{\xi}_\nu (\nu = 1 \sim K)$ satisfy the relations

$$\boldsymbol{\xi}_\nu^\dagger \bullet \boldsymbol{\xi}_\beta = \delta_{\nu\beta} \tag{12}$$

$$\delta_{\nu\beta} = \begin{cases} 1 & (\nu = \beta) \\ 0 & (\nu \neq \beta) \end{cases} \tag{13}$$

It is almost trivial that $\boldsymbol{\xi}_\mu^{\lambda\dagger}$ and $\boldsymbol{\xi}_{\mu'}^{\lambda'}$ satisfy the orthogonal relation $\boldsymbol{\xi}_\mu^{\lambda\dagger} \cdot \boldsymbol{\xi}_{\mu'}^{\lambda'} = \delta_{\mu\mu'}\delta_{\lambda\lambda'}$, where the $K = M \times L$ numbering of the vectors $\{\boldsymbol{\xi}^{\nu(\dagger)} \mid \nu = 1 \ldots K\}$ is changed again to (M, L) numbering as $\{\boldsymbol{\xi}_\mu^{\lambda(\dagger)} \mid \mu = 1 \ldots M, \lambda = 1 \ldots L\}$. This pseudo-inverse (orthogonalized learning) method enables us to make patterns be attractors even if the patterns are similar. This method was confirmed to be effective to avoid spurious attractors (Nara and Davis 1992, 1997; Nara et al. 1993, 1995). If the parameters (k, α, a_i and ε) are appropriately chosen, the network acts as a conventional associative memory, that is, the cycles of patterns are limit cycle attractors in the state space.

Let us note that state vectors $\boldsymbol{\xi}$ and $\mathbf{S}$, where components of the former are ± 1, and the latter, 0 or 1, can be easily converted to each other by putting $\xi_i = 2\ S_i - 1$ or $S_i = (\xi_i + 1)/2$. So, if $\mathbf{S}(t)$ is initially near one of the attractor patterns $(\boldsymbol{\xi}_s^q + 1)/2$, then the sequence $\mathbf{S}(t + lL)$ ($l = 1, 2, 3, \ldots$) generated by the L-step map will converge to that pattern $\mathbf{S}_s^q = (\boldsymbol{\xi}_s^q + 1)/2$. For each attractor pattern $\mathbf{S}_\mu^\lambda = (\boldsymbol{\xi}_\mu^\lambda + 1)/2$, there is a set of states $B_\mu\lambda$, called attractor basins, such that if $\mathbf{S}(t)$ is in $B_\mu\lambda$ then $\mathbf{S}(t + lL)(l = 1, 2, 3, \ldots)$ will converge to $\mathbf{S}_\mu^\lambda = (\boldsymbol{\xi}_\mu^\lambda + 1)/2$. Let us show an example with $M = 4$, $L = 6$ and $N = 400$. The embedded pattern vectors $\boldsymbol{\xi}_\mu^\lambda$ are the face patterns shown in Fig. 2. Each pattern corresponds to an image of black ($+1$) or white (-1) states of $20 \times 20 = 400$ pixels. One can easily see that the patterns have strong similarities.

For some values of parameters (k, α, a_i and ε), in particular the parameter α that changes the effect of refractoriness, the internal field $\{Y_i(t)\}$ will fluctuate and then there may be deviation from the original attractor dynamics. From the results of the investigation of chaotic dynamics by Aihara et al., it was confirmed that the attractor dynamics become unstable and $\mathbf{S}(t)$ chaotically wanders around in pattern space (Adachi and Aihara 1997).

However, in this current investigation we ask an important question, what type of chaotic dynamics arise when attractor patterns have similarities? Will similarity

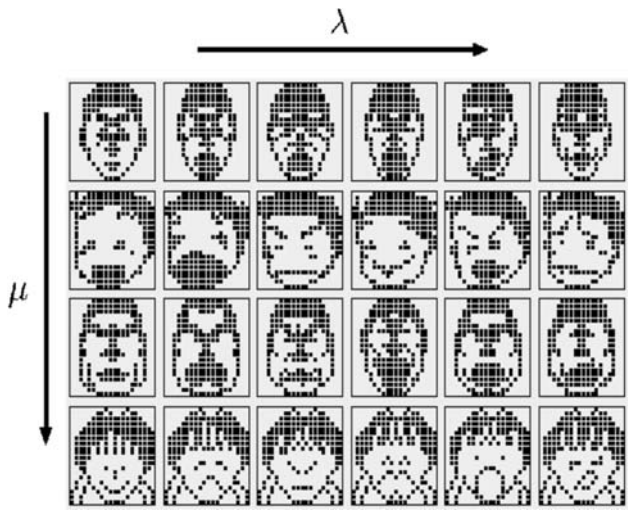


Fig. 2 An example of 24 face patterns embedded in a neural network as limit cycle attractors, $M(=4) \times L(=6) = 24$. Index numbers of $\xi^{\lambda}_{\mu}(\mu = 1 \sim M, \lambda = 1 \sim L)$ are shown. Let us note that these face patterns are shown only for an example of cycle attractor patterns. So, basin visiting measures shown in Fig. 4 and basin volumes shown in Fig. 5 are calculated with employing the cyclic attractor patterns shown in Fig. 6

between patterns be reflected in the nature of the onset of chaotic transitions between them?

Chaotic network dynamics and coding of motion function

Before applying the chaotic dynamics to control, let us first describe the characteristics of the chaotic dynamics. Figure 3 shows an example of iteration of internal field $\{Y_i\}$ according to Eq. (7). When the parameter values (k, α, a_i and ε) are changed, then the internal fields become unstable and the firing state vector determined by Eq. (6), $\mathbf{S}(t)$, chaotically wanders around in state space. Appropriately choosing parameter values results in either attractor dynamics or chaotic dynamics in the state space. The notation $P = \{k, \alpha, \varepsilon, a_i\}$ is employed to denote a set of parameter values. In this paper, a set denoted by P_A corresponds to "attractor regime" and a set denoted by P_C, corresponds to "chaotic regime".

We note the following observations:

(a) an chaotic attractor exists after the cycle attractors became unstable
(b) sequences in the chaotic attractor repeatedly visit all the regions which are basins in the attractor regime
(c) the distribution of chaotic sequences over the basins in the attractor regime is not uniform, so the basin visiting ratio is not simply related to the state-space volume of the basins in the attractor regime.

These properties were confirmed by observing statistics of basin visits. Specifically, the basin visiting measure was calculated as follows.

- Select an initial pattern
- Recursively update the network many times to generate a sequence with a particular parameter set $P = \{k, \alpha, a_i, \varepsilon\}$.
- At each step in the sequence, check which basin the present firing state (state vector) belongs to.
- Count the number of steps in each basin to obtain the basin visiting ratio for the sequence, corresponding to the particular initial pattern.

Several examples of basin visiting measure are shown in Fig. 4. When the parameter set corresponds to the regime of cycle attractors, such as in Fig. 4a, b, the sequence is localized in a single pattern cycle. On the other hand, in the case of $P_C = \{k = 0.90, \alpha = 7.3, \varepsilon = 0.0010, a_i = 3.12\}$, shown in Fig. 4c, d, the sequence does not converge to any cycle even after a long time, but wanders chaotically over a wide area in the N dimensional hypercube state space, so the measure is spread over the basins of many patterns. Moreover, the measure is the same even for different initial

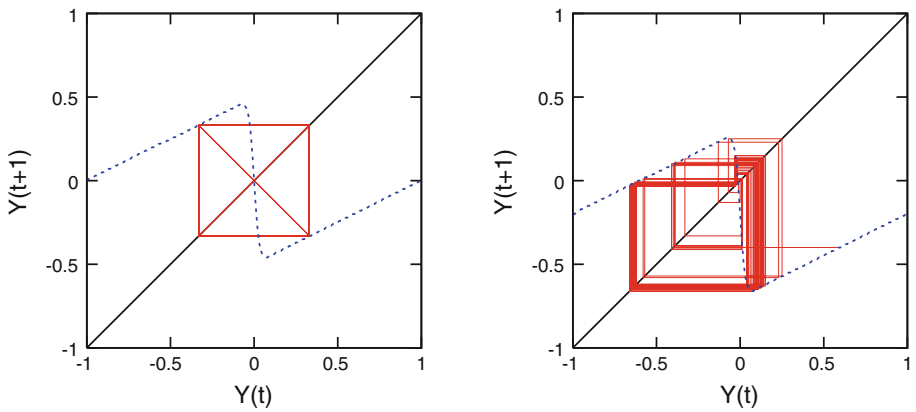


Fig. 3 Attractor dynamics (limit cycle with period 2) and chaotic dynamics in the recurrent mapping of the internal field $Y(n)$. The two different types of dynamics are obtained at different values of parameters

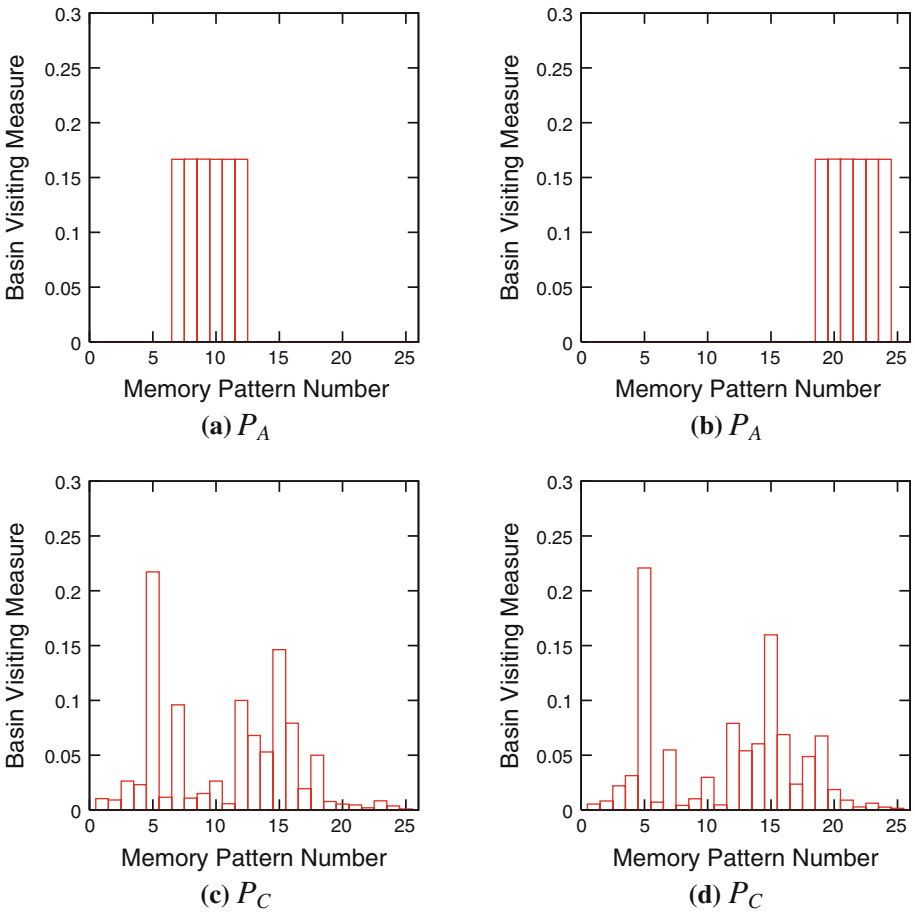


Fig. 4 Basin visiting measure (Fig. 6) for cycle attractor and chaotic sequences. The *horizontal axis* represents the pattern number and the *vertical axis* represents the frequency of visiting the basin of the corresponding pattern. Numbers 1 to 24 correspond to the embedded patterns ($1 \sim M \times L$). The data for *number 25* shows the ratio for convergence to spurious patterns and the data for *number 26* shows the ratio of patterns which initiated sequences which did not converge to any pattern. **a** and **c** memory pattern ${\xi^1}_2$ is taken as initial pattern; **b** and **d** memory pattern ${\xi^1}_4$ is taken as initial pattern. $P_A = \{k = 0.3, \alpha = 0.4, \varepsilon = 0.0010, a_i = 0.2\}$, $P_C = \{k = 0.90, \alpha = 7.3, \varepsilon = 0.0010, a_i = 3.12\}$

patterns, indicating the existence of a chaotic attractor. For comparison, the relative basin volume of each attractor in the N dimension state space is statistically estimated by counting the proportion of a large number of randomly generated initial patterns converging to each attractor, which is shown in Fig. 5. Obviously, the basin visiting measure for the chaotic attractor is different from the relative basin volumes, that is the basin visiting measure is not simply related to the state space volume of the basins in attractor regime. This shows that the chaotic sequences are not distributed uniformly in the state space. A large variety of chaotic dynamics can be observed in this model and comprehensive description is beyond the scope of this paper. More details, including evaluation of the instability properties of sequences in the chaotic attractor in terms of 'Lyapunov Spectra', and analysis of "haotic itinerancin terms of trapping in attractor ruins, will be left for future reports.

Next we describe the coding method which allows the network states to be used for motion control. A detailed description of motion control coding has been reported in a previous paper (Suemitsu and Nara 2004). An object is assumed to move in 2-D space, from the position $(q_x(t), q_y(t))$ to $(q_x(t+1), q_y(t+1))$ via a set of motion

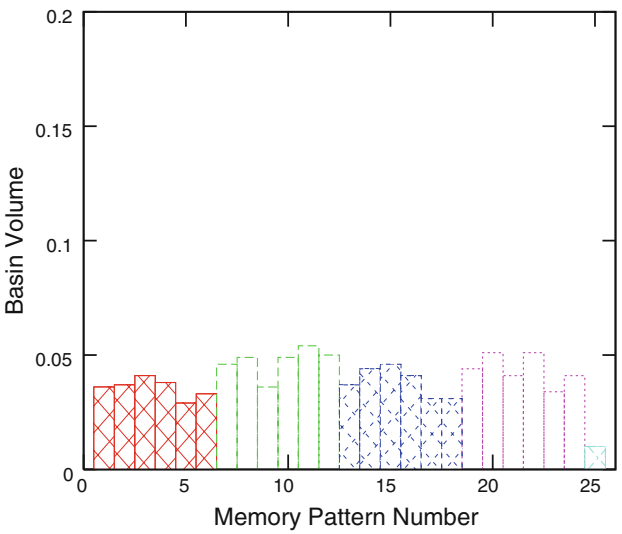


Fig. 5 Relative basin volume fraction(P_A) for the cycle attractor shown in Fig. 6: The *horizontal axis* represents the pattern number and the *vertical axis* represents the frequency of converging to the basin of the corresponding pattern. Numbers 1 to 24 correspond to the embedded patterns ($1 \sim M \times L$). The data for *number 25* shows the ratio for convergence to spurious patterns and the data for *number 26* shows the ratio of patterns which initiated sequences which did not converge to any pattern. *Alternate hatching* and *non-hatching* are used to indicate different cyclic attractors

functions. The motion function corresponding to a network state $\mathbf{S}(t)$, is defined, in terms of $\mathbf{X}(t) = 2\,\mathbf{S}(t) - 1$, by

$$f_x(\mathbf{X}(t)) = \frac{4}{N}\sum_{i=1}^{\frac{N}{4}} X_i(t)\cdot X_{i+\frac{N}{2}}(t) \tag{14}$$

$$f_y(\mathbf{X}(t)) = \frac{4}{N}\sum_{i=1}^{\frac{N}{4}} X_{i+\frac{N}{4}}(t)\cdot X_{i+\frac{3}{4}N}(t) \tag{15}$$

From the definition of $\mathbf{X} = \{X_i = 2\,S_i - 1 = \pm 1\}$, one can easily know that f_x and f_y range from -1 to $+1$ with the normalization by $4/N$. Note that each motion function is calculated by a self inner product between two parts of the network state $\mathbf{X}(t)$. In 2-D space, the actual motion of the object is given by

$$q_x(t+1) = q_x(t) + f_x(\mathbf{X}(t+1)) \tag{16}$$

$$q_y(t+1) = q_y(t) + f_y(\mathbf{X}(t+1)). \tag{17}$$

In the example presented below, we take $N = 400$, $M = 6$, and $K = 4$. Two dimensional space is discretized with the resolution 0.02, according to the definition of the state vector with dimension $N = 400$ and the definitions of f_x and f_y. The way to determine a set of attractor patterns $\xi_i^{\mu,\lambda}$ is as follows. Each attractor pattern is divided into two parts. One is a random pattern part, where each component has value $+1$ or -1 with probability 0.5 ($\xi_i^{\mu,\lambda} = \pm 1 : i = 1, ..., N/2$). The other part of the attractor pattern ($\xi_i^{\mu,\lambda} = \pm 1 : i = N/2 + 1, ..., N$) is determined so as the following relations are satisfied

$$(f_x(\boldsymbol{\xi}_1^{\lambda}), f_y(\boldsymbol{\xi}_1^{\lambda})) = (-1,-1)$$
$$(f_x(\boldsymbol{\xi}_2^{\lambda}), f_y(\boldsymbol{\xi}_2^{\lambda})) = (-1,+1)$$
$$(f_x(\boldsymbol{\xi}_3^{\lambda}), f_y(\boldsymbol{\xi}_3^{\lambda})) = (+1,-1)$$
$$(f_x(\boldsymbol{\xi}_4^{\lambda}), f_y(\boldsymbol{\xi}_4^{\lambda})) = (+1,+1).$$

Each limit cycle attractor corresponds to a constant motion of the object toward one of the four directions $(+1, +1)$, $(+1, -1)$, $(-1, +1)$, $(-1, -1)$. We call these "prototype patterns", as they drive monotonic motions. Figure 6 shows the patterns embedded in the connection matrix of the network. We choose $M = 6$, that is, $\lambda = 1, ..., 6$, so the $K = 4$ limit cycle attractors each have $M = 6$ patterns.

Figure 7 shows examples of two motions that are generated, respectively, by the second cycle attractor in the attractor regime (the parameter set P_A) and by chaotic dynamics in the chaotic regime (the parameter set P_C).

Motion control using adaptive switching between attractor regime and chaotic regime

Generally speaking, from the viewpoint of control, chaos has been considered to spoil the control of systems. A large number of methods have been proposed in order to avoid emergence of chaos, as the OGY-method was proposed to stabilize an arbitrary periodic orbit in chaos (Ott et al. 1990; Grebogi and Yorke 1997). Now, as stated in the introduction, we consider chaos to be useful not only in solving ill-posed problems but also in controlling of systems with large but finite degrees of freedom. In order to show potential capabilities of chaotic dynamics generated by CNN, let us try to apply it to a control task, specifically, as an example, a maze in 2-D space. An object is assumed to move in a 2-D maze and approaches the target using chaotic dynamics. One of the reasons why we consider the maze task is that the process of solving a maze can be

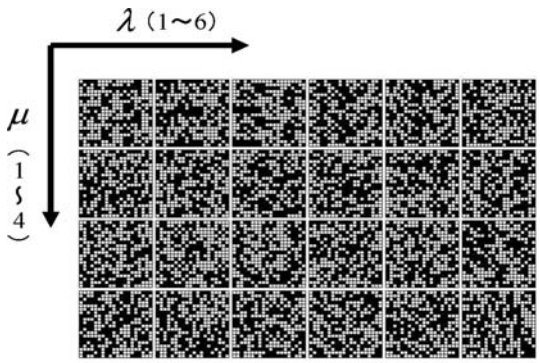


Fig. 6 Cycle attractors embedded for motion control. Each cycle consists of six patterns. The first cycle $\{\xi_1^{\lambda}|\lambda = 1...6\}$ is chosen so as $(f_x(\xi_1^{\lambda}), f_y(\xi_1^{\lambda})) = (-1,-1)$ is satisfied. The second, third and fourth cycles similarly correspond to $(-1, +1)$, $(+1, -1)$, and $(+1, +1)$, respectively

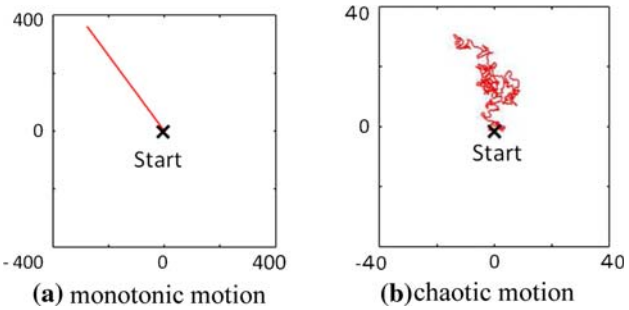


Fig. 7 Free running trajectories from a starting point which is the origin (0, 0) (the center of *square*) during the same updating time steps. (**a**) An example of the object orbit from start point in 2D space in a attractor regime. As the parameter set is P_A, attractor dynamics result in monotonic motions with long journey because pattern dynamics converges into one of embedded pattern attractors. So the scale is 800×800. One of the monotonic motions which are embedded in synaptic connection matrix W_{ij} is shown. (**b**) An example of the object orbit from start point in 2D space in chaotic regime. As the parameter set is P_C, chaotic dynamics correspond to complex but rather localized motion, which is called chaotic motion. So the scale is 80×80. Note that the chaotic motions strongly depend on the values of parameters

easily visualized. Hence, we can understand how the dynamical structures are effectively utilized in controlling.

In this section, a method of controlling the object by switching the parameters $P = \{k, \alpha, a_i, \varepsilon\}$ according to a simple evaluation is proposed. A target is assumed to be set in a 2-D space, at a location with coordinates (Q_{x0}, Q_{y0}). Note that location coordinates of the target are not used in the control method. Only the rough direction of the target $D_1(t)$ is used to control the object. This is an example of an external response with ambiguity. For example, $D_1(t)$ becomes 1 if the direction from the object to the target is observed between angles 0 and $\pi/2$ (the first quadrant), where angles are defined with respect to the x-axis in 2-D space. Similarly, $D_1(t)$ becomes n (= 1, 2, 3, 4) if the direction is between angles $(n-1)\pi/2$ and $n\pi/2$, respectively. The direction of the object motion $D_2(t)$ from time $t-1$ to t is defined as

$$D_2(t) = \begin{cases} 1 & (c_x(t) = +1 \text{ and } c_y(t) = +1) \\ 2 & (c_x(t) = -1 \text{ and } c_y(t) = +1) \\ 3 & (c_x(t) = -1 \text{ and } c_y(t) = -1) \\ 4 & (c_x(t) = +1 \text{ and } c_y(t) = -1) \end{cases},$$

where $c_x(t)$ and $c_y(t)$ are given as

$$c_x(t) = \frac{q_x(t) - q_x(t-1)}{|q_x(t) - q_x(t-1)|} \tag{18}$$

$$c_y(t) = \frac{q_y(t) - q_y(t-1)}{|q_y(t) - q_y(t-1)|}. \tag{19}$$

Finally, using $D_1(t)$ and $D_2(t)$, a time dependent set of network parameters $P(t)$ is determined

$$P(t) = \begin{cases} P_A & (\text{if } D_1(t-1) = D_2(t-1)) \\ P_C & (\text{otherwise}) \end{cases}, \tag{20}$$

where P_A is the parameter set that gives the "attractor regime" and P_C is the parameter set that gives the "chaotic regime". After the determination of the parameter set $P(t)$, the motion of the object is calculated from the network state updated using the parameter values $P(t)$. After the motion new values of $D_1(t)$ and $D_2(t)$ are calculated. Repeating this process, the connectivity is switched between P_A and P_C, and the object moves about in the 2-D space. In this control method, the parameter set is kept at $P(t) = P_A$ if the object moves toward the target with a tolerance of $\pi/2$. This provides stable and monotonic motion toward the target. We assume that the object can know $D_1(t)$ at each time step even though there are walls in the maze. In our experiments, both $f_x(\mathbf{X}(t))$ and $f_y(\mathbf{X}(t))$ are taken to be zero if the moving object hits a wall in the next step of motion. This means that the object cannot penetrate the wall when it hits the wall. It is necessary to propose a control algorithm which gets the object to approach the target in the 2-D maze. The control algorithm using chaos is shown in Fig. 8.

Figure 9 shows two examples of solving a 2 dimensional maze by switching the parameter set between chaotic regime and attractor regime using the proposed simple algorithm.

In order to investigate the performance of the method proposed above for solving two-dimensional mazes, we employ a quantitative evaluation function $\rho(P)$, the success rate. The evaluation method is as follows.

- Randomly generate a set of I_{trial} firing patterns;
- Choose two parameter sets P_C and P_A;
- Using the algorithm proposed above, solve a 2-D maze with an initial firing pattern chosen from one of the I_{trial}

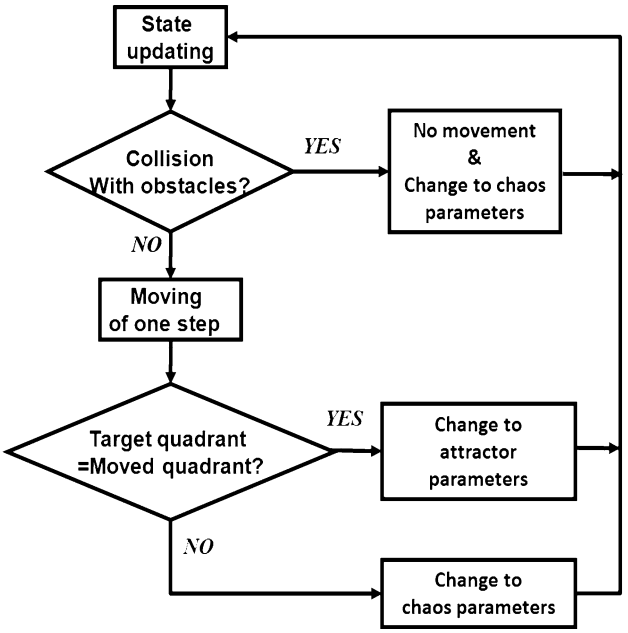


Fig. 8 The block diagram of the control method using the chaotic neural network (CNN) model

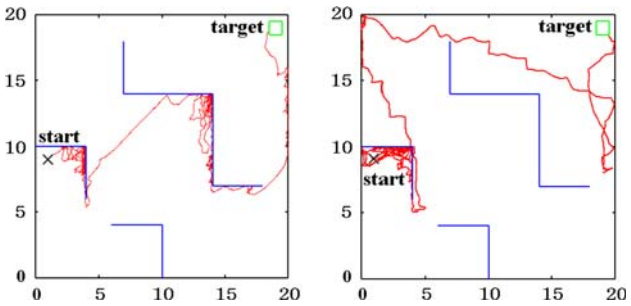


Fig. 9 Two examples of the object trajectory in a 2D maze with obstacle walls. The ×-mark is the starting position and the *open square* is the target position. This maze type is called Ω-type

random patterns. If the object can avoid obstacles and reach the target within $T_{\max}$ steps, it is regarded as a successful trial.

- Count the number of successful trials over I_{trial} trials. If O_{suc} denotes the number of successful trials, the success rate $\rho(P)$ is defined by

$$\rho(P) = \frac{O_{\text{suc}}}{I_{\text{trial}}} \tag{21}$$

We present the performance result for the Ω-type maze as an example. The success rate for solving the Ω-type maze is shown in Fig. 10. The total number of trials is $I_{\text{trial}} = 300$. A successful trial means that the robot can reach the target within $T_{\max} = 5000$ steps. The results of the computer experiments indicate that, once appropriate P_C is found, the dynamics of the network of chaos-neurons is useful for solving a specified configuration of maze. Now, a question arises, that is, are these chaotic dynamics useful in other configurations, that is, in general? To answer this question, we prepared the three kinds of maze shown in Fig. 11, and furthermore, variations of these configurations consisting of four mirror symmetries of them. The results indicate that, in all cases, chaos caused by appropriate parameter values gives satisfactory results with high success rates. Here, let us show only the successful cases of the mirror symmetries in the first type maze. The results are shown in Fig. 12.

.

In the case of the functional application of chaos, it is always asked whether chaos has advantages over a stochastic random process or not. Therefore, we replace P_C in Eq. (20) with two kinds of stochastic random process, and try to solve the two-dimensional mazes. Corresponding to two kinds of stochastic random process, we employed two

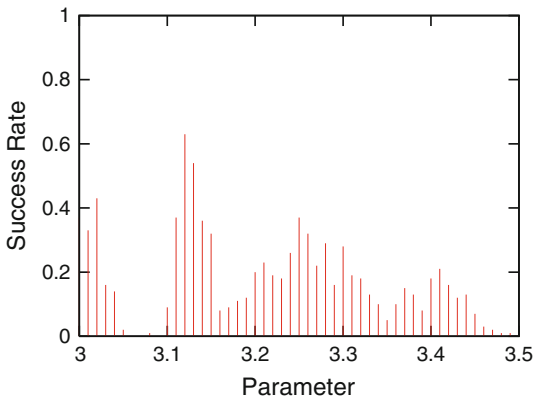


Fig. 10 Success rate: The *horizontal axis* is a_i (3.0 ∼3.5) corresponding to 50 instances of $P_C = \{ k = 0.9, \alpha = 7.3, \varepsilon = 0.001, a_i \}$

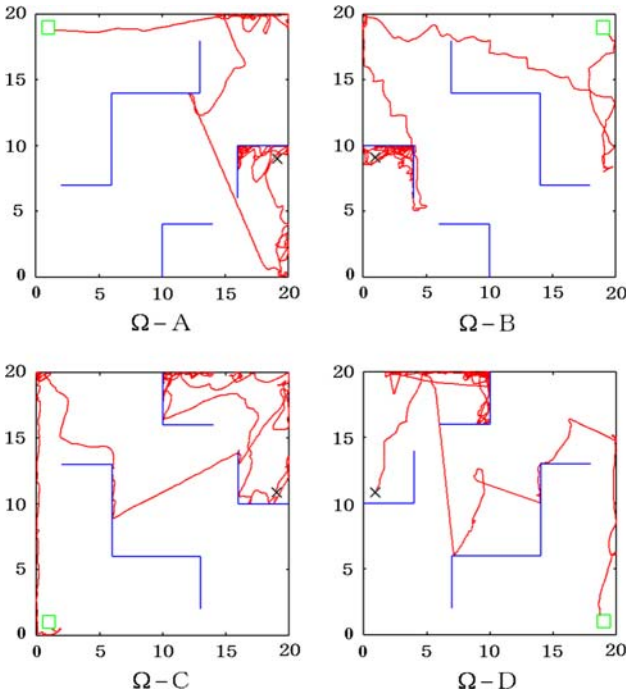
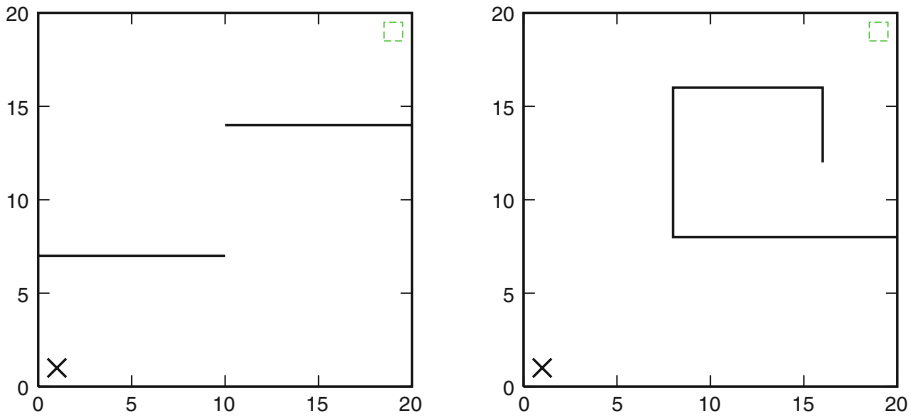
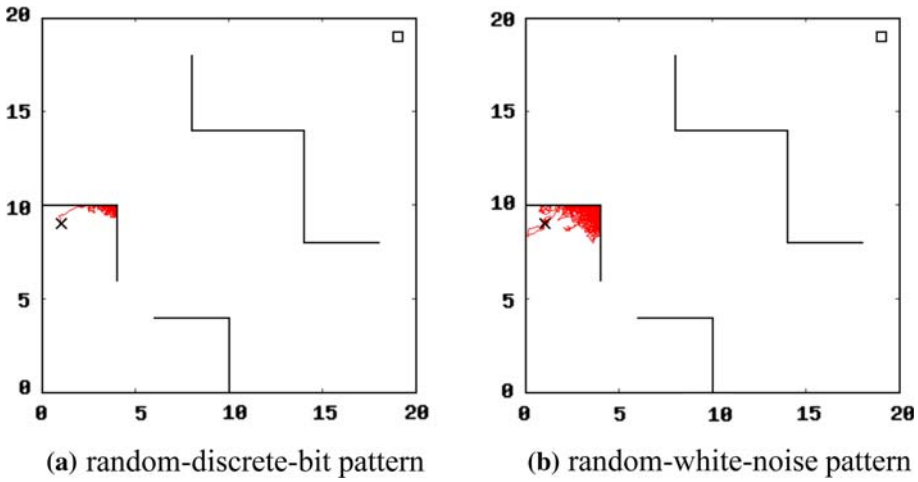


Fig. 12 The four types of mazes taken in the mirror symmetries and the successful trajectory in each case, obtained using CNN. The starting positions and the target positions are the same as those of Fig. 9, and the scale of each maze is also the same

Fig. 11 The other two types of mazes (S-type, and V-type) used in our simulations. The ×-mark is the starting position and the *open square* is the target position





(a) random-discrete-bit pattern (b) random-white-noise pattern

Fig. 13 Examples of solving two-dimensional mazes using stochastic random pattern generators: the robot is seldom able to find a detour to avoid obstacles

kinds of stochastic random pattern generator: an N dimensional random-discrete-bit pattern generator, and an N dimensional random-white-noise pattern generator. An N dimensional random-discrete-bit pattern generator is utilized to generate a random pattern with N bits only including 1 or -1. However, an N dimensional random-white-noise pattern generator is utilized to generate a random pattern with N bits, but each bit of the pattern is not a binary value but a random real number that ranges from -1.0 to 1.0. Figure 13a, b show examples of solving 2-D mazes using the two kinds of random pattern generator, respectively. The results of the computer experiments indicate the robot cannot avoid obstacles using either of the random generators. According to the method of performance evaluation defined above, the success rate of each of the two kinds of generator is zero.

Discussion

The above cases indicate two important points, and result in two questions immediately. One is that, appropriate parameter values can produce chaotic dynamics that are effective to solve the 2-D mazes, but not all. Here, a question arises. What factors cause the difference of the performance in the same task? The other is that, stochastic random processes cannot solve the 2-D mazes at all. Then another question becomes clearly. What is the difference between chaotic dynamics and a stochastic random process? As we said above, we think that chaotic dynamics has some dynamical structure, which might give us some heuristic clues or a partial answer. Therefore, in order to show the relations among these cases, from a dynamical viewpoint, we have evaluated the dynamical structures of these dynamical behaviours using a statistical method. First, let us introduce the method of evaluation of chaotic dynamics. The network is updated for many steps with a particular parameter set P_C, so that chaotic wandering occurs in N dimensional hypercube state space. During this wandering, we have taken statistics of the residence time, the time during which the system continuously stays in a certain basin (Suemitsu and Nara 2004; Li and Nara 2008) and evaluated the distribution $p(l, \mu)$ which is defined by

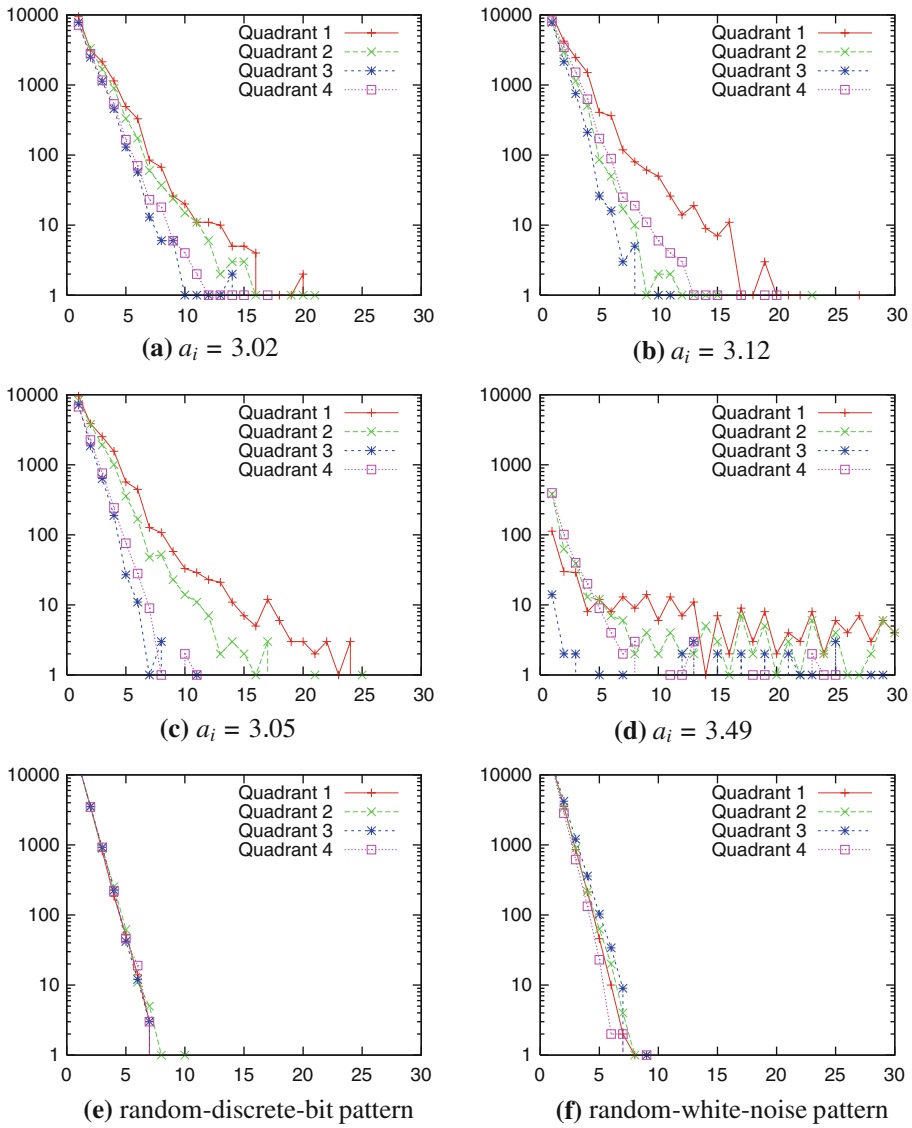
$$\begin{aligned} p(l,\mu) \quad &= \{\text{the number of } l | \mathbf{S}(t) \in \beta_\mu \text{ in } \tau \le t \le \tau + l \\ &\qquad \text{and } \mathbf{S}(\tau - 1) \notin \beta_\mu \\ &\qquad \text{and } \mathbf{S}(\tau + l + 1) \notin \beta_\mu, \mu | \mu \in [1, M]\} \end{aligned} \tag{22}$$

$$\beta_\mu = \sum_{\lambda=1}^{L} B_\mu^\lambda \tag{23}$$

$$T = \sum_{l} l p(l,\mu) \tag{24}$$

where l is the residence time in each attractor basin, and T is the total number of steps. So $p(l, \mu)$ denotes a distribution of residence time l in attractor basin $L = \mu$ within T steps. In our actual simulation, $T = 10^5$. Figure 14a–d shows the distributions $p(l, \mu)$ for four parameter sets $P_{C1} = (k, \alpha, a_i$ and $\varepsilon)$ and $P_{C2} = (k, \alpha, a_i$ and $\varepsilon)$, $P_{C3} = (k, \alpha, a_i$and $\varepsilon)$ and $P_{C4} = (k, \alpha, a_i$ and $\varepsilon)$. Next, stochastic random processes are evaluated with a similar statistical method. Two kinds of random pattern generator are employed. One is an N dimensional random-discrete-bit pattern generator, and the other is an N dimensional random-white-noise pattern generator. With the pattern generator, random wandering occurs in N dimensional hypercube state space. Applying the method defined in Eq. (24) to the stochastic random process, the distribution of residence time in basins is evaluated. The distributions corresponding to the two kinds of random pattern generator are shown in Fig. 14e, f, respectively.

Comparing these figures, the obvious differences between chaotic dynamics and the stochastic random processes has two points. One point is the maximum l_{max} of



(a) $a_i = 3.02$ **(b)** $a_i = 3.12$ **(c)** $a_i = 3.05$ **(d)** $a_i = 3.49$ **(e)** random-discrete-bit pattern **(f)** random-white-noise pattern

Fig. 14 The log plot of the frequency distribution of residence time l: The *horizontal axis* represents residence time steps l in a certain basin μ during long time chaotic wandering, and the *vertical axis* represents the accumulative number $p(l, \mu)$ of the residence time steps l in a certain basin μ

residence times in a certain basin. For the stochastic random processes, l_{max} is not over 10. But for the chaotic dynamics, almost all l_{max} are over 10. The other point is the similarity of the distribution $p(l, \mu)(\mu = 1M)$ in M basins. For the stochastic random processes, the distribution $p(l, \mu)$ in M basin are almost the same. But for the chaotic dynamics, each $p(l, \mu)$ is quite different, or unequal. The above two points could be reasons why the stochastic processes cannot solve the two-dimensional mazes but chaotic dynamics can. Longer residence times and unequal residence time distributions $p(l, \mu)$ could enable the robot to find a detour to avoid obstacles and finally reach the target. Are longer residence times and unequal residence time distributions $p(l, \mu)$ better for solving the two-dimensional mazes? We find the answer is not "Yes". Referring to the success rate indicated in Fig. 10, by contrast with P_{C3} and P_{C4}, P_{C1} and P_{C2} show better performance, but their residence times l are shorter and their distributions $p(l, \mu)$ are not as unequal. In particular, comparing these figures, we can find an interesting phenomenon. In Fig. 14c, d, residence time l is always much longer than that in the other three basins, and l_{max} is longer than 25 steps. In other words, chaotic wandering could often visit a certain basin and show localized dynamical behaviors. So, it is reasonable for us to consider that localized chaotic dynamics is not effective for solving the 2-D mazes. According to the above discussion, we can conclude that somewhat longer residence times l and non-localized dynamical distribution $p(l, \mu)$ could enable the robot to find a detour to avoid obstacles and finally reach the target.

Finally, let us talk about our navigation strategy. As we stated in Section "Introduction", our research is motivated by the discovery of chaotic dynamics in brains. Moreover,

in our opinion, chaotic dynamics in biological systems play important roles in adaptive controlling via simple rules. Therefore our navigation strategy emphasizes the features of robustness, simple rules and autonomous control. Recent works about brain-machine interface and the parietal lobe suggested that, in cortical area, the "message" defining a given hand movement is widely disseminated (Wessberg et al. 2000; Nicolelis 2001). So, a relatively large neural network using a large number N of neurons compared with other navigation strategies (but actually quite small compared with the number of neurons in brain), can provide a huge reservoir of redundancy and can result in great robustness. Generally speaking, many navigation strategies often fall into enormous algorithm complexity. However, our navigation strategy is based on a simple adaptive algorithm responding to external stimulus from the environment. Without any pre-knowledge of the configuration of obstacles, the robot, which is driven by chaotic dynamics corresponding to appropriate parameter values, can autonomously avoid obstacles and successfully reach the target.

Summary and concluding remarks

Our investigation of the dynamics in a chaotic neural network (CNN), where multiple cyclic attractors are embedded by setting synaptic connection strengths using the pseudo-inverse method (a kind of orthogonalized learning method),has shown that chaotic dynamics occur at some values of neuron parameters. For particular parameter sets, $P = \{\ k,\ \alpha, \varepsilon, a\ \}$, there are chaotic trajectories which wander among all the regions which are basins of attractions in the attractor regime. This is advantageous for use of bifurcation to chaos for search in state space, as shown in earlier work (Mikami and Nara 2003; Suemitsu and Nara 2003). In particular, chaotic dynamics in a CNN with multiple cyclic attractors has some features additional features mentioned in Section "Chaotic network dynamics and coding of motion function", which are significant in a functional sense.

Based on the novel idea of harnessing of chaos, chaotic dynamics generated in a chaotic neural network model are applied to 2-D motion control. With just a few simple motions embedded in the network, using a simple coding method called motion functions, chaotic dynamics in high-dimensional state space result in various motions in 2-D space. Using a simple control algorithm to control adaptive switching of the system parameters between a chaotic regime and an attractor regime, the object is able to reach a target in a 2-D maze successfully. The success rate of this method over many trials is good in comparison with that of stochastic random pattern generators. In order to clarify the difference between motion control using chaotic dynamics and random pattern generation, we have statistically analyzed their dynamical structures. It was found that the maximum value $l_{\max}$ of basin residence times and the similarity of the distributions of residence times $p(l, \mu)$ in all the basins indicate why the stochastic processes cannot solve the two-dimensional mazes. In contrast, it was found that chaotic dynamics with somewhat longer basin residence times l and non-localized distributions $p(l, \mu)$ resulted in a higher probability of the object finding a detour around obstacles and finally reaching the target ie. these dynamical features are a necessary (though not sufficient) condition for solving the 2-D mazes. These conclusions are consistent with the conclusions presented in our earlier work (Suemitsu and Nara 2004; Takada et al. 2007; Li and Nara 2008; Li et al. 2008). As the functional features do not depend strongly on the detailed structure of models, we can speculate about the usefulness of chaos in general Ethat chaotic dynamics are potentially useful in systems containing large numbers of degrees of freedom to realize robust, adaptive and complex control functions with simple rules. Moreover, our results support the idea that chaos observed in biological systems could play a very important role in realizing adaptive functions and control with simple rules.

Acknowledgments Authors would like to appreciate Dr. Peter Davis for his valuable discussions over almost all of this work. Many thanks to the anonymous reviewers for their good suggestions and critical comments. This work has been supported by Grant-in-Aid for Promotion of Science #16500131 in Japan Society for the Promotion of Science.

References

Adachi M, Aihara K (1997) Associative dynamics in a chaotic neural network. Neural Netw 10(1):83–98

Aertsen A, Erb M, Palm G (1994), Dynamics of functional coupling in the cerebral cortex: an attempt at a model based interpretation. Physica D 75:103–128

Aihara K, Takabe T, Toyoda M (1990) Chaotic neural networks. Phys Lett A 144(6–7):333–340

Amari S (1977) Neural theory of association and concept-formation. Biol Cybern 26:175–185

Domany E, van Hemmen JL, Schluten K (eds) (1991) Physics of neural networks. Springer-Verlag, Berlin

Fujii H, Ito H, Aihara K, Ichinose N, Tsukada M (1996) Dynamical cell assembly hypothesis—theoretical possibility of spatio-temporal coding in the cortex. Neural Netw 9(8):1303–1350

Grebogi C, Yorke JA (eds) (1997) The impact of chaos on science and society. United Nations University Press, Tokyo

Haken H (1988) Information and self-organization. Springer Verlag, Berlin (The second edition was published in 2000)

Haken H (1996) Principles of brain functioning. Springer-Verlag, Berlin

Kaneko K, Tsuda I (2000) Complex systems: chaos and beyond. Springer-Verlag, Berlin (The first edition was published in 2000)

Kay LM, Lancaster LR, Freeman WJ (1996) Reafference and attractors in the olfactory system during odor recognition. Int J Neural Syst 4:489–495

Kuroiwa J, Nara S, Aihara K (1999) Functional possibility of chaotic behaviour in a single chaotic neuron model for dynamical signal processing elements. In: 1999 IEEE international conference on systems, man, and cybernetics (SMC'99), Tokyo, October, 1999, vol 1. p 290

Li Y, Nara S (2008) Novel tracking function of moving target using chaotic dynamics in a recurrent neural network model. Cogn Neurodyn 2:39–48

Li Y, Kurata S, Morita S, Shimizu S, Munetaka D, Nara S (2008) Application of chaotic dynamics in a recurrent neural network to control: hardware implementation into a novel autonomous roving robot. Biol Cybern 99:185–196

Mikami S, Nara S (2003) Dynamical responses of chaotic memory dynamics to weak input in a recurrent neural network model. Neural Comput Appl 11(3–4):129–136

Mori Y, Davis P, Nara S (1989) Pattern retrieval in an asymmetric neural network with embedded limit cycles. J Phys A 22:525–532

Nara S (2003) Can potentially useful dynamics to solve complex problems emerge from constrained chaos and/or chaotic itinerancy? Chaos 13(3):1110–1121

Nara S, Davis P (1992) Chaotic wandering and search in a cycle memory neural network. Prog Theor Phys 88:845–855

Nara S, Davis P (1997) Learning feature constraints in a chaotic neural memory. Phys Rev E 55:826–830

Nara S, Davis P, Kawachi M, Totuji H (1993) Memory search using complex dynamics in a recurrent neural network model. Neural Netw 6:963–973

Nara S, Davis P, Kawachi M, Totuji H (1995) Chaotic memory dynamics in a recurrent neural network with cycle memories embedded by pseudo-inverse method. Int J Bifurcation and Chaos Appl Sci Eng 5:1205–1212

Ott E, Grebogi C, Yorke JA (1990) Controlling chaos. Phys Rev Lett 64:1196–1199

Skarda CA, Freeman WJ (1987) How brains make chaos in order to make sense of the world. Behav Brain Sci 10:161–195

Suemitsu Y, Nara S (2003) A note on time delayed effect in a recurrent neural network model. Neural Comput Appl 11(3–4):137–143

Suemitsu Y, Nara S (2004) A solution for two-dimensional mazes with use of chaotic dynamics in a recurrent neural network model. Neural Comput 16(9):1943–1957

Takada R, Munetaka D, Kobayashi S, Suemitsu Y, Nara S (2007) Numerically evaluated functional equivalence between chaotic dynamics in neural networks and cellular automata under totalistic rules. Cogn Neurodyn 1(3):189–202

Tokuda I, Nagashima T, Aihara K (1997) Global bifurcation structure of chaotic neural networks and its application to traveling salesman problems. Neural Networks 10(9):1673–1690

Tsuda I (2001) Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems. Behav Brain Sci 24(5):793–847