Cogn Neurodyn (2008) 2:39–48
DOI 10.1007/s11571-007-9029-6

RESEARCH ARTICLE

Novel tracking function of moving target using chaotic dynamics in a recurrent neural network model

Yongtao Li · Shigetoshi Nara

Received: 15 May 2007 / Revised: / Accepted: 14 September 2007 / Published online: 9 October 2007

Abstract Chaotic dynamics introduced in a recurrent neural network model is applied to controlling an object to track a moving target in two-dimensional space, which is set as an ill-posed problem. The motion increments of the object are determined by a group of motion functions calculated in real time with firing states of the neurons in the network. Several cyclic memory attractors that correspond to several simple motions of the object in two-dimensional space are embedded. Chaotic dynamics introduced in the network causes corresponding complex motions of the object in two-dimensional space. Adaptively real-time switching of control parameter results in constrained chaos (chaotic itinerancy) in the state space of the network and enables the object to track a moving target along a certain trajectory successfully. The performance of tracking is evaluated by calculating the success rate over 100 trials with respect to nine kinds of trajectories along which the target moves respectively. Computer experiments show that chaotic dynamics is useful to track a moving target. To understand the relations between these cases and chaotic dynamics, dynamical structure of chaotic dynamics is investigated from dynamical viewpoint.

Keywords Chaotic dynamics · Tracking · Moving target · Neural network

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

Associated with rapid development of science and technology, great attentions on biological systems have been paid because of excellent functions not only in information processing, but also in well-regulated functioning and controlling, which work quite adaptively in various environments. Despite many attempts to understand the mechanisms of biological systems, we have yet poor understanding them.

In biological systems, well-regulated functioning and controlling originate from the strongly nonlinear interaction between local systems and total system. Therefore, it is very difficult to understand and describe these systems using the conventional methodologies based on reductionism, which means that a system is decomposed into parts or elements. The conventional reductionism more or less falls into two difficulties due to enormous complexity originating from dynamics in systems with large but finite degrees of freedom. One is "combinatorial explosion" and the other is "divergence of algorithmic complexity". These difficulties are not yet solved in spite of many efforts. On the other hand, a novel idea based on functional viewpoint was introduced to understand the mechanisms. It is a new approach called "the methodology of complex dynamics", which has been constructed in various fields of science and engineering in the several decades associated with the remarkable development of computers and simulation methods. Especially, chaotic dynamics observed in biological systems including brains has attracted great interest (Babloyantz and Destexhe 1986; Skarda and Freeman 1987). It is considered that chaotic dynamics would play important roles in complex functioning and controlling of biological systems including brains. From this viewpoint, many dynamical models have been constructed for

approaching the mechanisms by means of large-scale simulation or heuristic methods. Artificial neural networks in which chaotic dynamics can be introduced has been attracting great interests.

Over decade years ago, chaotic itinerancy was observed in neural networks and was proposed as a universal dynamical concept in high-dimensional dynamical systems. Artificial neural networks for chaotic itinerancy were studied with great interests (Aihara et al. 1990; Tsuda 1991, 2001; Kaneko and Tsuda 2003; Fuji et al. 1996). As one of those works, by Nara and Davis, chaotic dynamics was introduced in a recurrent neural network model (RNNM) consisting of binary neurons, and for investigating the functional aspects of chaos, they have applied chaotic dynamics by means of numerical methods to solving, for instance, a memory search task which is set in an ill-posed context (Nara and Davis 1992, 1997; Nara et al. 1993, 1995; Kuroiwa et al. 1999; Suemitsu and Nara 2003). In their papers, they proposed that chaotic itinerancy could be potentially useful dynamics to solve complex problem, such as ill-posed problems. Standing on this viewpoint, auditory behaviour of the cricket shows a typical ill-posed problem in biological systems. Female cricket can track towards directions of male position leaded by calling song of male in dark fields with a large number of obstacles (Huber and Thorson 1985). This behaviour includes two ill-posed properties. One is that darkness and noisy environments prevent female from accurate deciding of directions of male positions, and the other is that a large number of big obstacles in fields force female to solve two-dimensional maze as one of ill-posed problems. Therefore, in order to investigate the brain from functional aspects, we try to construct a model to approach insect behaviours to solve ill-posed problems. As one of functional examples, chaotic dynamics introduced in a recurrent network model was applied to solving a two-dimensional maze, which is set as an ill-posed problem (Suemitsu and Nara 2004). A simple coding method translating the neural states into motion increments and a simple control algorithm adaptively switching a system parameter to produce chaotic itinerant behaviours are proposed. The conclusions show that chaotic itinerant behaviours can give better performance to solving a two-dimensional maze than that of random walk.

In order to further investigate functional aspects of chaotic dynamics, it was applied to tracking a moving target, which is set as another ill-posed problem. Generally speaking, as an object is tracking an target that is moving along a certain trajectory in two-dimensional space, it is an ill-posed problem because there are many tracking results with uncertainty. In conventional methods, the object is set to obtain more precise information from the target as possible, so as to successfully capture the moving target. However, in our study, the object obtain only rough information of the target and successfully capture the moving target using chaotic dynamics in RNNM.

In the case of tracking a moving target, the first problem is to realize two-dimensional motion control of the object. In our model, we assume that the object moves with discrete time steps. The firing state in the neural network is transformed into two-dimensional motion increments by the coding of motion functions, which will be illustrated in the later section. In addition, several limit cycle attractors, which are corresponding to the prototypical simple motions of the object in two-dimensional space, are embedded in the neural network. At a certain time, if the firing pattern converges into an prototypical attractor, the object moves in a monotonic direction of several directions in two-dimensional space. If chaotic dynamics is introduced into the network, the firing pattern could not converge into an prototypical attractor, that is, attractors fall to ruin. At the same time, the corresponding motion of the object is chaotic in two-dimensional space. By adaptive switching of a certain system parameter, chaotic itinerancy generated in the neural network results in complex two-dimensional motions of the object in various environments. Considering this point, we have proposed a simple control algorithm of tracking a moving target, and quite good tracking performances have been obtained, as will be stated in the later section .

This paper is organized as follows. In the next section we describe the network model and chaotic dynamics in it, the control algorithm of tracking a moving target is illustrated in Sect. “Algorithm of tracking a moving target”. We discuss the results of computer simulation and evaluate the performance of tracking a moving target in Sect. “Experimental results”.

Chaotic dynamics in a recurrent neural network model

Our study works with a fully interconnected recurrent neural network consisting of N neurons , which is shown as Fig. 1. Its updating rule is defined by

$$S_i(t+1) = \mathrm{sgn}\left(\sum_{j\in G_i(r)} W_{ij}S_j(t)\right) \qquad (1)$$
$$sgn(u) = \begin{cases} +1 & \mathrm{u} \geq 0; \\ -1 & \mathrm{u} < 0. \end{cases}$$

where $S_i(t) = \pm 1$ ($i = 1 \sim N$) represents the firing state of a neuron specified by index i at time t. W_{ij} is an asymmetrical connection weight (synaptic weight) from the neuron S_j to the neuron S_i, where W_{ii} is taken to be 0. $G_i(r)$ means a connectivity configuration set of connectivity

r ($0 < r < N$) that is fan-in number for the neuron S_i. At a certain time t, the state of neurons in the network can be represented as a N-dimensional state vector $\boldsymbol{S}$(t), called as state pattern. Time development of state pattern $\boldsymbol{S}(t)$ depends on the connection weight matrix W_{ij} and connectivity r, therefore, in the case of full connectivity $r = N - 1$, if W_{ij} could be appropriately determined, arbitrarily chosen state pattern $\boldsymbol{\xi}(t)$ would be multiple stationary states in the development of $\boldsymbol{S}$(t), which is equivalent to storing memory states in the functional context. In our study, W_{ij} are determined by a kind of orthogonalized learning method and taken as follows.

$$W_{ij} = \sum_{\mu=1}^{L} \sum_{\lambda=1}^{K} (\xi_{\mu}^{\lambda+1})_i \cdot (\xi_{\mu}^{\lambda})_j^{\dagger} \tag{2}$$

where $\{\xi_{\mu}^{\lambda} | \lambda = 1 \ldots K, \mu = 1 \ldots L\}$ is an attractor pattern set, K is the number of memory patterns included in a cycle and L is the number of memory cycles. $\xi_{\mu}^{\lambda\dagger}$ is the conjugate vector of ξ_{μ}^{λ} which satisfies $\xi_{\mu}^{\lambda\dagger} \cdot \xi_{\mu'}^{\lambda'} = \delta_{\mu\mu'} \cdot \delta_{\lambda\lambda'}$, where δ is Kronecker's delta. This method was confirmed to be effective to avoid spurious attractors that affect L attractors with K-step maps embedded in the network when connectivity $r = N$ (Nara and Davis 1992, 1997; Nara et al. 1993, 1995; Nara 2003; Suemitsu and Nara 2003).

In the case of full connectivity $r = N - 1$, as time evolves, the state pattern $\boldsymbol{S}(t)$ converges into one of the cyclic memory patterns. Therefore, the network can function as a conventional associative memory. If the state pattern $\boldsymbol{S}(t)$ is one of the memory patterns, ξ_{μ}^{λ}, then the next output $\boldsymbol{S}(t + 1)$ will be the next memory pattern of the cycle, $\xi_{\mu}^{\lambda+1}$. Even if the state pattern $\boldsymbol{S}(t)$ is near one of the memory patterns, ξ_{μ}^{λ}. the output sequence $\boldsymbol{S}(t + kK)(k =$ 1,2,3...) will converge to the memory pattern ξ_{μ}^{λ}. In other words, for each memory pattern, there is a set of the state patterns, called as memory basin B_{μ}^{λ}. If $\boldsymbol{S}(t)$ is in the memory basin B_{μ}^{λ}, then the output sequence $\boldsymbol{S}(t + kK)$ (k = 1,2,3...) will converge to the memory pattern ξ_{μ}^{λ}.

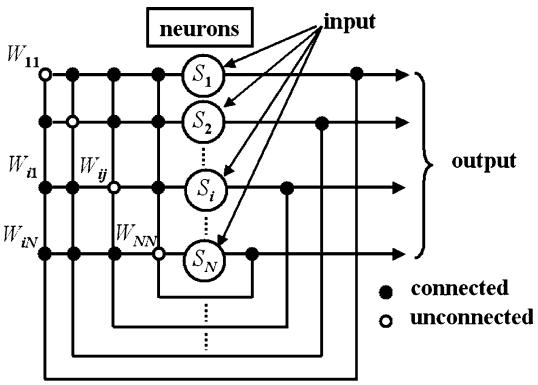

Fig. 1 Fully interconnected recurrent neural network model

It is quite difficult to estimate basin volume accurately because one must check the final state ($\lim_{k\to\infty} \boldsymbol{S}(kK)$) of all initial state patterns (the total number is 2^N), as requires an enormous amounts of time. Therefore, a statistical method is applied to estimating the approximate basin volume. First, random initial state patterns are generated with a sufficiently large amount so that they can cover the entire N-dimensional state space uniformly. As state updating develops, it is specified that the final state $\lim_{k\to\infty} \boldsymbol{S}(kK)$ of each initial pattern would converge into a certain memory attractor. The ratios between the number of initial state patterns that converge into a certain memory attractor and the total number of initial state patterns are taken. The rate of convergence to each memory attractor is proportional to the basin volume, and is regarded as the approximate basin volume for each memory attractor. An actual example of the basin volume is shown in Fig. 2. The basin volume shows that almost all initial state patterns converge into one of the memory attractors averagely, that is, there are mainly the memory attractors in the whole state space.

Next, we continue to decrease connectivity r. When r is large enough, $r \simeq N$, memory attractors are stable, the network can still function as a conventional associative memory. When r becomes smaller and smaller, the basin volume of all the memory attractors are becoming smaller and smaller, in other words, more and more state patterns gradually do not converge into a certain memory pattern despite the network is updated for a long time, that is, each basin vanishes and the attractor becomes unstable. So that if the number of connectivity r becomes quite small, state patterns do not converge into any memory pattern even if the network is updated for a long time.

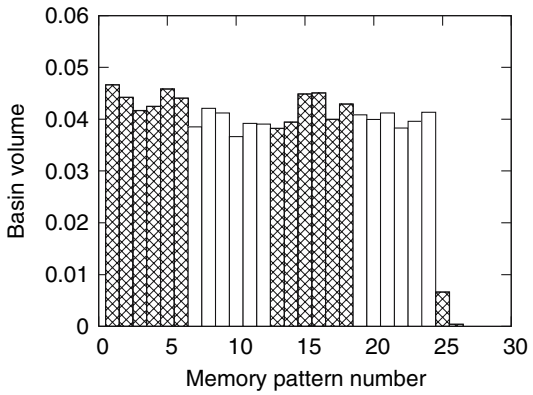

Fig. 2 Basin volume fraction ($r = N - 1 = 399$): The horizontal axis represents memory pattern number (1–24). The basin number 25 shows the volume fraction which corresponds to the initial patterns that converged into cyclic output states with a period of six steps but not any one of the memory attractors. The basin number 26 shows the volume fraction which corresponds to the initial patterns which did not converged, that means chaotically itinerant . The vertical axis represents the ratio of each sample to the total number of samples. Alternative hatching and nonhatching are used to show different cyclic attractors

Since chaotic dynamics in the network depends on system parameter—the connectivity r, in order to analyze the destabilizing process in our model, we have calculated a bifurcation diagram of overlap, where overlap means one-dimensional projection of state pattern $\boldsymbol{S}(t)$ to a certain reference pattern. Therefore, an overlap $m(t)$ is defined by

$$m(t) = \frac{1}{N}\boldsymbol{S}(0) \cdot \boldsymbol{S}(t) \tag{3}$$

where $\boldsymbol{S}(0)$ is an initial pattern(reference pattern) and $\boldsymbol{S}(t)$ is the state pattern at time step t. Because $m(t)$ is a normalized inner product, $-1 \le m(t) \le 1$. For connectivity from 1 to $N-1$, we have respectively calculated the corresponding overlap $m(t)$ as state pattern $\boldsymbol{S}(t)$ evolves for long time. Figure 3 shows the overlap $m(t)$ as a function of connectivity. In the case of large enough connectivity r, state pattern $\boldsymbol{S}(t)$ at each K time step is same with $\boldsymbol{S}(0)$. With the decrease of connectivity r, state pattern $\boldsymbol{S}(t)$ at each K time step gradually becomes different to $\boldsymbol{S}(0)$, that is, cyclic memory attractor becomes unstable. Finally, non-period dynamics occurs, that is, cyclic memory attractors ruin.

In our previous papers, we confirmed that the non-period dynamics in the network is chaotic wandering. In order to investigate the dynamical structure, we calculated basin visiting measures and it suggests that the trajectory can pass the whole N-dimensional state space, that is, cyclic memory attractors ruin due to a quite small connectivity (Nara and Davis 1992, 1997; Nara et al. 1993, 1995; Nara 2003; Suemitsu and Nara 2003).

Motion control and memory patterns

Biological data show that the number of neurons in the brain varies dramatically from species to species and the human brain has about 100 billion (10^{11}) neurons, but one human has only over 600 muscles that function to produce force and cause motion. These motions are controlled by the neuron system. That is, the motions of relatively few muscles are controlled by the activities of enormous neurons. Therefore, the neural network consisting of N neurons is used to realize two-dimensional motion control of an object.

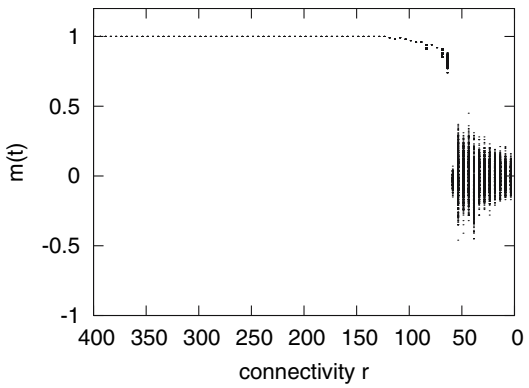

Fig. 3 Bifurcation diagram with respect connectivity r: The horizontal axis represents connectivity r (0–399). The vertical axis represents the long-time behaviours of overlap $m(t)$ at K-step mappings

We confirmed that chaotic dynamics introduced in the network does not so sensitively depend on the size of the neuron number (Nara 2003). However, if N is too small, chaotic dynamics can not occur; whereas if N is oversized, it results in excessive computing time. Therefore, the number of neurons is $N = 400$ in our actual computer simulation. At a certain time, the state pattern in the network is represented by 400-dimensional state vectors, while the motion in two-dimensional space is only two-dimensional vectors. Suppose that a moving object moves from the position $(p_x(t),p_y(t))$ to $(p_x(t+1), p_y(t+1))$ with a set of motion increments $(\Delta f_x(t), \Delta f_y(t))$. The state pattern $\boldsymbol{S}(t)$ at time t is a 400-dimensional vector, so we must transform it to two-dimensional motion increments by coding. The coding relations are implemented by replacing motion increments with a group of motion functions $(f_x(\boldsymbol{S}(t)),\ f_y(\boldsymbol{S}(t)))$. In 2-dimensional space, the actual motion of the object is given by

$$p_x(t+1) = p_x(t) + f_x(\boldsymbol{S}(t)) \tag{4}$$

$$p_y(t+1) = p_y(t) + f_y(\boldsymbol{S}(t)) \tag{5}$$

where $f_x(\boldsymbol{S}(t))$, $f_y(\boldsymbol{S}(t))$ are the x-axis increment and y-axis increment respectively, and they are calculated from firing states of the neural network model and defined by

$$f_x(\boldsymbol{S}(t)) = \frac{4}{N}\boldsymbol{A} \cdot \boldsymbol{C} \qquad f_y(\boldsymbol{S}(t)) = \frac{4}{N}\boldsymbol{B} \cdot \boldsymbol{D} \tag{6}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ are four independent $N/4$ dimensional sub-space vectors of state pattern $\boldsymbol{S}(t)$. Therefore, after the inner product between two independent sub-space vectors is normalized by $4/N$, motion functions range from -1 to $+1$, that is,

$$-1 \le f_x(\boldsymbol{S}(t)) \le +1 \tag{7}$$

$$-1 \le f_y(\boldsymbol{S}(t)) \le +1 \tag{8}$$

Referring to Eq. (4) and (5) and the definition of motion functions, in our actual simulations, two-dimensional space is digitized with a resolution $8/N = 0.02$ due to the binary neuron state ± 1 and $N = 400$.

Next, let us consider the construction of memory attractors corresponding to prototypical simple motions. It is considerable that two-dimensional motion consists of several prototypical simple motions. We take four types of

motion that one object moves toward (+ 1, + 1), (–1, + 1), (–1, –1), (+ 1, –1) in two-dimensional space, as prototypical simple motions. Under these situations, four groups of attractor patterns, which are corresponding to the prototypical simple motions of the object in two-dimensional space, are embedded in the neural network by means of embedding of associative memory introduced in the previous section. Each group of attractor patterns includes six patterns that are corresponding to one prototypical simple motion. Each group is a cyclic memory, or a limit cycle attractor in 400-dimensional state space. We take ξ_μ^λ (μ = 1, 2, 3, 4 and λ = 1, 2, …, 6) as the attractor pattern that is λ pattern in μ group (see Fig. 4). Therefore, in our actual simulation, $L = 4$, $K = 6$. In the present simulation, we directly employed $K = 6$ because it is an optimized selection after one of the authors and his collaborators had done a number of simulations for various K (3, 5, 6, 10, for instance). All of the results show that , if K is too small, it is difficult to avoid spurious attractors, on the other hand, quite large K can not also give stronger attraction. The corresponding relations between attractor patterns and prototypical simple motions are shown as follows.

$$(f_x(\xi_1^\lambda),\ f_y(\xi_1^\lambda)) = (+1,+1)$$
$$(f_x(\xi_2^\lambda),\ f_y(\xi_2^\lambda)) = (-1,+1)$$
$$(f_x(\xi_3^\lambda),\ f_y(\xi_3^\lambda)) = (-1,-1)$$
$$(f_x(\xi_4^\lambda),\ f_y(\xi_4^\lambda)) = (+1,-1)$$

When connectivity r is sufficiently large, one random initial pattern converges into one of four limit cycle attractors as time evolves. The corresponding motion of the object in 2-dimensional space becomes monotonic, and a simulation example is shown in Fig. 5. On the other hand, when connectivity r is quite small, chaotic dynamics is observed in the network with the development of time. At the same time, the corresponding motion of the object is chaotic, and Fig. 6 shows an simulation example of chaotic motion in two-dimensional space. Therefore, when the network evolves, monotonic motion and chaotic motion can be switched by switching the connectivity r.

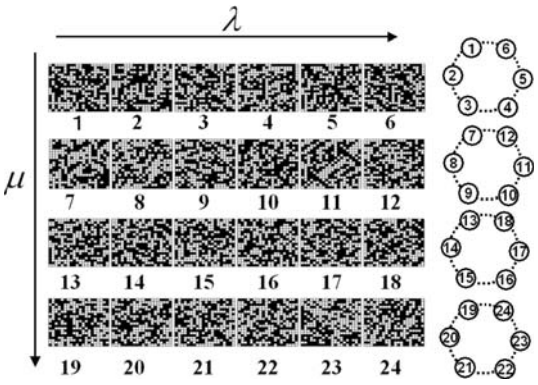

Fig. 4 Memory attractor patterns: Pattern (1–24) includes $\mu = 4$ groups of cyclic memory consisting $\lambda = 6$ patterns. Each cyclic memory corresponds to a prototypical simple motion

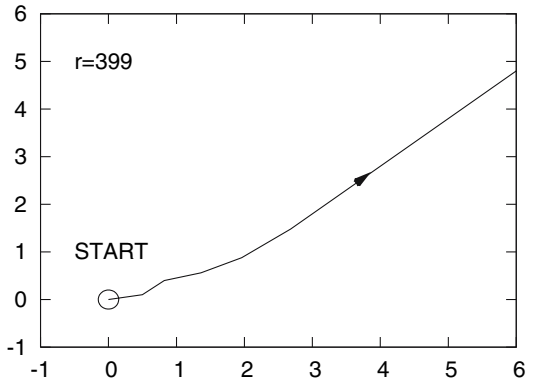

Fig. 5 An example of monotonic motion: When associative network state ($r = 399$) occurs in the state space, the object performs monotonic motion (+ 1, + 1) after some updating steps, from start point (0, 0) in two-dimensional space

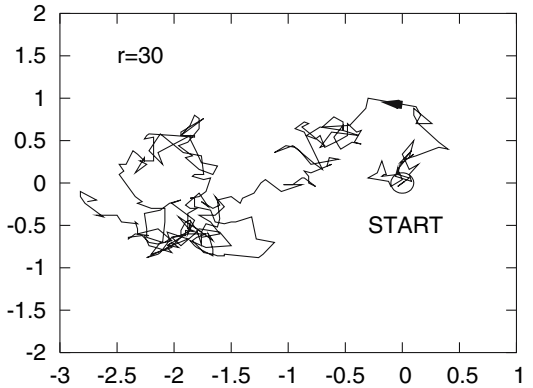

Fig. 6 An example of chaotic motion: When chaotic network state ($r = 40$) occurs in the network, correspondingly, the object moves chaotically from start point (0,0) in two-dimensional space

Algorithm of tracking a moving target

Now we want to discuss how to realize motion control so as to track a moving target. In our study, we suppose that an object is tracking a target that is moving along a certain trajectory in two-dimensional space, and the object can obtain the rough directional information $D_1(t)$ of the moving target. At a certain time t, the present position of the object is assumed at the point $(p_x(t), p_y(t))$. This point is taken as the origin point and two-dimensional space can be divided into four quadrants. If the target is moving in the first quadrant, $D_1(t) = 1$. Therefore, if the target is moving in the nth quadrant, $D_1(t) = n$ (n = 1,2,3,4), which is called *global target direction* for it is a rough directional information.

Next, we also suppose that the object can also know another directional information $D_2(t)$, which means which quadrant the moving object has moved toward from time $t-1$ to t, that is, in the previous step. The direction $D_2(t)$ is called *global motion direction*, and defined as

$$D_2(t) = \begin{cases} 1 & (c_x(t) = +1 \text{ and } c_y(t) = +1) \\ 2 & (c_x(t) = -1 \text{ and } c_y(t) = +1) \\ 3 & (c_x(t) = -1 \text{ and } c_y(t) = -1) \\ 4 & (c_x(t) = +1 \text{ and } c_y(t) = -1) \end{cases} \quad (9)$$

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

$$c_x(t) = \frac{p_x(t) - p_x(t-1)}{|p_x(t) - p_x(t-1)|} \quad (10)$$

$$c_y(t) = \frac{p_y(t) - p_y(t-1)}{|p_y(t) - p_y(t-1)|} \quad (11)$$

Now we know that global target direction $D_1(t)$ and global motion direction $D_2(t)$ are time-dependent variables. If the network can get feedback signals from these two directions in real time, the connectivity r also becomes a time-dependent variable $r(t)$ and is determined by global target direction $D_1(t)$ and global motion direction $D_2(t)$. Therefore, a simple control algorithm of tracking a moving target is proposed and shown in Fig. 7, where R_L is a sufficiently large connectivity and R_S is a quite small connectivity that can lead to chaotic dynamics in the neural network. Adaptive switching of connectivity is the core idea of the algorithm. If global motion direction and global target direction is coincident, that is, $D_2(t) = D_1(t)$, the network is updated with sufficiently large connectivity $r(t) = R_L$; otherwise, if global motion direction and global target direction is not coincident, the network is updated with quite small connectivity $r(t) = R_S$. When the synaptic connectivity $r(t)$ is determined by comparing two directions, $D_1(t-1)$ and $D_2(t-1)$, the motion increments of the object are calculated from the state pattern of the network updated with $r(t)$. The new motion causes the next $D_1(t)$ and $D_2(t)$, and produces the next synaptic connectivity $r(t+1)$. By repeating this process, the synaptic connectivity $r(t)$ is adaptively switching between R_L and R_S, the object is alternatively implementing monotonic motion and chaotic motion in two-dimensional space.

In closing this section, one point we must mention is that we have to use an engineering approach to switch the parameter r in computer simulation experiments because we started from heuristic approach using a simple model to apply chaotic dynamics to complex problems which includes ill-posed property. However, as the future scope, we will develop it to investigate biological mechanism of advanced functions or controls in real biological systems.

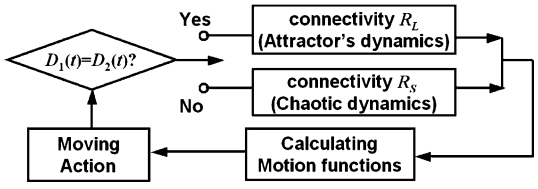

Fig. 7 Control algorithm of tracking a moving target: By judging whether global target direction $D_1(t)$ coincides with global motion direction $D_2(t)$ or not, adaptive switching of connectivity r between R_S and R_L results in chaotic dynamics or attractor's dynamics in state space. Correspondingly, the object is adaptively tracking a moving target in two-dimensional space

Experimental results

Generally speaking, it is difficult to give mathematical proof that our method always produces correct solutions in tracking of arbitrarily moving target. Necessarily, we must rely on computer experiments and showing typical properties connected to universal effectiveness of chaos in biological systems. Therefore, at the start point, some simple trajectories along which the target moves should be set. In future, more complex orbits of moving target will be investigated. In order to simplify our investigation, we have taken nine kinds of trajectories that include one circular trajectory and eight linear trajectories, shown in Fig. 8.

Suppose that the initial position of the object is the origin (0,0) of two-dimensional space. The distance d between initial position of the object and that of the target is a constant value. Therefore, at the beginning of tracking, the object is at the circular center of the circular trajectory and the other eight linear trajectories are tangential to the circular trajectory along a certain angle α, where the angle is defined by the x axis. The tangential angle $\alpha = n\pi/4 (n = 1,2,\ldots,8)$, so we number the eight linear trajectories as LT_n.

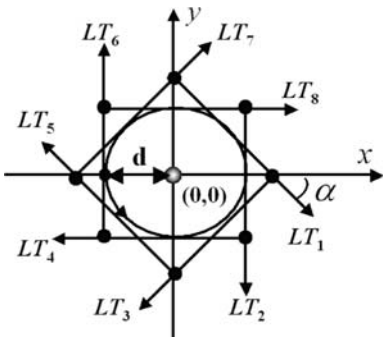

Fig. 8 Trajectories of moving target: one is circular and eight are linear($LT_1 - LT_8$)

Next, let us consider the velocity of the target. In computer simulation, the object moves one step per discrete time step, at the same time, the target also moves one step with a certain step length *SL* that represents the velocity of the target. The motion increments of the object ranges from –1 to 1 (see Eq. (7) and (8)), so the step length *SL* is taken with an interval 0.01 from 0.01 to 1 up to 100 different velocities. Because velocity is a relative quantity, so $SL = 0.01$ is a slower target velocity and $SL = 1$ is a faster target velocity relative to the object.

Now, let us look at a simulation of tracking a moving target using the algorithm proposed above, shown in Fig. 9. When an target is moving along a circular trajectory at a certain velocity, the object captured the target at a certain point of the circular trajectory, which is a successful capture to a circular trajectory. Another simulation of tracking a target that moves along a linear trajectory is shown in Fig. 10, which is a successful capture to a linear trajectory.

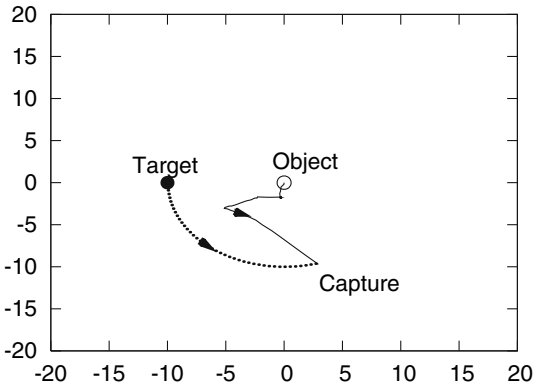

Fig. 9 An example of tracking a target that is moving along a circular trajectory with the simple algorithm. The object captured the moving target at the intersection point

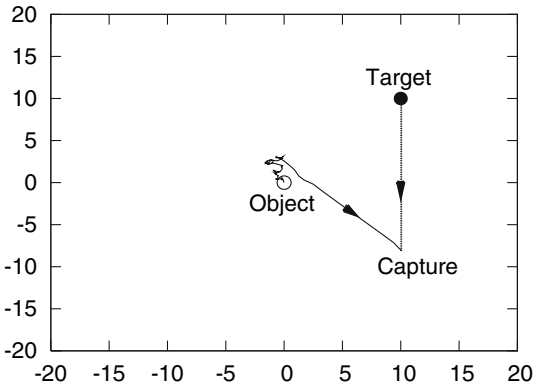

Fig. 10 An example of tracking a target that is moving along a linear trajectory with the simple algorithm. The object captured the moving target at the intersection point

Performance evaluation

To show the performance of tracking a moving target, we have evaluated the success rate of tracking a moving target that moves along one of nine trajectories. However, even though tracking a same target trajectory, the performance of tracking depends not only on synaptic connectivity *r*, but also on target velocity or target step length *SL*. Therefore, when we evaluate the success rate of tracking, a pair of parameters, that is, one of connectivity r ($1 \le r \le 60$) and one of target velocity *SL* ($0.01 \le T \le 1.0$), is taken. Because we take 100 different target velocity with a same interval 0.01, we have C_{60}^{100} pairs of parameters. We have evaluated the success rate of tracking a circle trajectory, shown as Fig. 11. From the simulation results, we can know that the success rate of tracking a circle trajectory with chaotic dynamics is significantly high, and that, the success rate highly depends on synaptic connectivity *r* and the velocity of the target.

In order to observe the performance clearly, we have taken the data of certain connectivities, and plot them in two-dimensional coordinates, shown as Fig. 12. Comparing these figures, we can see a novel performance, when the target velocity becomes faster, the success rate has a upward tendency, such as $r = 51$. In other words, when the chaotic dynamics is not too strong, it seems useful to tracking a faster target.

Certainly, the performance of tracking a moving target also depends on the target trajectory. In this paper, because linear target trajectories is too much, we only show two of them in Fig. 13. From the results of computer experiment, we know the following two points. First, the success rate decreases rapidly as the target velocity increases. Second, comparing these success rate of the linear trajectories with that of the circular trajectory, we are sure that tracking a moving target of circular trajectory has better performance than that of linear trajectory. However, to some linear trajectories, quite excellent performance was observed, such as Fig. 13b.

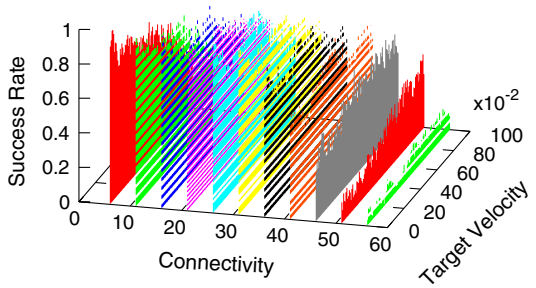

Fig. 11 Success rate of tracking a moving target along circle trajectory: Over 100 random initial patterns, the rate of successfully capturing the moving target within 600 steps is estimated as the success rate. The positive orientation obeys the right-hand rule. The vertical axis represents success rate, and two axes in the horizontal plane represents connectivity *r* and target velocity *SL*, respectively

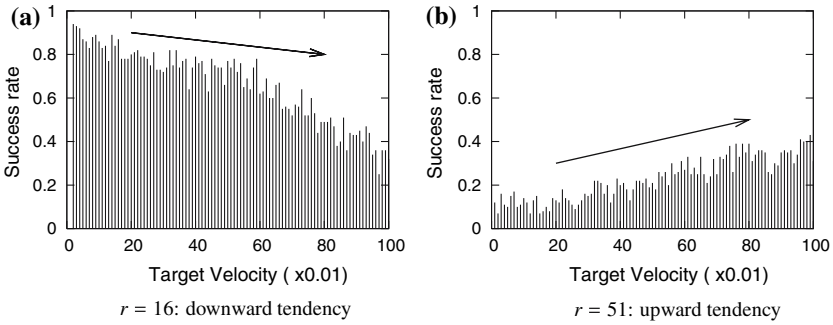

Fig. 12 Success rates drawn from Fig. 11. We take the data of a certain connectivity and show them in two dimension diagram. The horizontal axis represents target velocity from 0.01 to 1.0, and the vertical axis represents success rate. With the increase of target velocity, (**a**) r = 16: downward tendency; (**b**) r = 51: upward tendency

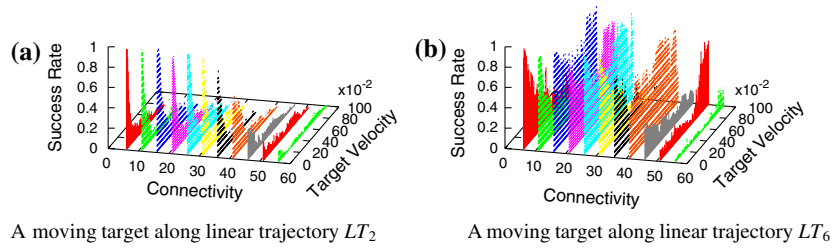

Fig. 13 Success rates of tracking a moving target along different linear trajectories. (**a**) A moving target along linear trajectory LT_2; (**b**) A moving target along linear trajectory LT_6

Discussion

In order to show the relations between the above cases and chaotic dynamics, from dynamical viewpoint, we have investigated dynamical structure of chaotic dynamics. For small connectivities from 1 to 60, the network takes chaotic wandering. During this wandering, we have taken a statistics of continuously staying time in a certain basin (Suemitsu and Nara 2004) and evaluated the distribution $p(l,\mu)$ which is defined by

$$\begin{aligned} p(l,\mu) = \{&\text{the number of } l \mid \boldsymbol{S}(t) \in \beta_\mu \text{ in } \tau \le t \le \tau + l \\ &\text{and } \boldsymbol{S}(\tau - 1) \notin \beta_\mu \\ &\text{and } \boldsymbol{S}(\tau + l + 1) \notin \beta_\mu, \mu | \mu \in [1, L]\} \end{aligned} \tag{12}$$

$$\beta_\mu = \sum_{\lambda=1}^{K} B_\mu^\lambda \tag{13}$$

$$T = \sum_{l} lp(l,\mu) \tag{14}$$

where l is the length of continuously staying time steps in each attractor basin, and $p(l,\mu)$ represents a distribution of continuously staying l steps in attractor basin $L = \mu$ within T steps. In our actual simulation, $T = 10^5$. To different connectivity r = 15 and r = 50, the distribution $p(l,\mu)$ are shown in Fig. 14a and b. In these figures, different basins are marked with different colors and symbols. From the results, we know, with increase of the connectivity, continuously staying time l becomes longer and longer.

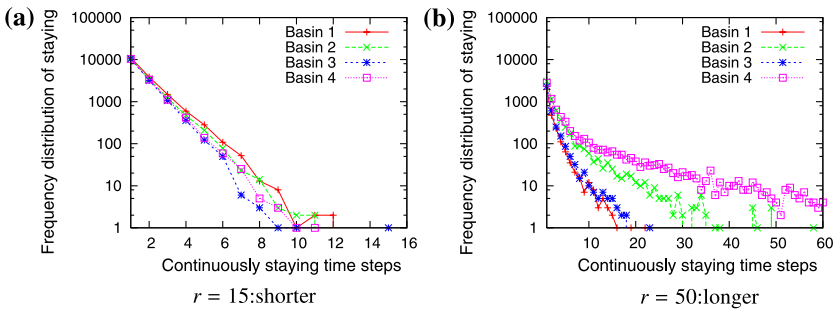

Fig. 14 The log plot of the frequency distribution of continuously staying time l: The horizontal axis represents continuously staying time steps l in a certain basin μ during long time chaotic wandering, and the vertical axis represents the accumulative number $p(l,\mu)$ of the same staying time steps l in a certain basin μ. continuously staying time steps l becomes long with the increase of connectivity r. (**a**) r = 15: l is shorter; (**b**) r = 50: l is longer

Referring to those novel performances talked in previous section, let us try to consider the reason. First, in the case of slower target velocity, a decreasing success rate with the increase of connectivity r is observed from both circular target trajectory and linear ones. This point shows that chaotic dynamics localized in a certain basin for too much time is not better to track a slower target.

Second, in the case of faster target velocity, it seems useful to track a faster target when chaotic dynamics is not too strong. Computer experiments show that, when the target moves quickly, the action of the object is always chaotic so as to track the target. In past experiments, we know that motion increments of chaotic motion is very short. Therefore, shorter motion increments and faster target velocity result in not good tracking performance. However, when continuously staying time l in a certain basin becomes longer, the object can move toward a certain direction for l steps. This is useful to track the faster target for the object. Therefore, when connectivity becomes a little large ($r = 50$ or so), success rate arises following the increase of target velocity, such as the case shown in Fig. 12.

Third, we try to explain the reason why success rate of tracking a moving target along a linear trajectory decreases rapidly when the target velocity increases a little. In this case, faster target velocity results in more chaotic motions of the object. At the same time, the target has moved too far away from the object along a linear trajectory. Therefore, the success rates become worse. Generally speaking, chaotic dynamics is not always useful to solve an ill-posed problem. However, better performance can be often observed using chaotic dynamics to solve an ill-posed problem. As an issue for future study, a functional aspect of chaotic dynamics still has context dependence.

Finally, let us consider the approach in robot navigation. There are many approaches in robot navigation. As an approach using dynamical neural network, a simple mechanism—dynamical neural Smitt trigger was applied to a small neural network controlling the behaviour of a autonomous miniatur robot (Hülse and Pasemann 2002). On the other hand, our model is a recurrent neural network with N neurons, that is, a large neural network. 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). Therefore, the difference between our novel approach and the Smitt trigger approach in robot navigation are quite big. Our approach emphasizes the whole state of neurons, but the Smitt trigger pays attention to the interaction between a few of neurons. Furthermore, our approach has a huge reservoir of redundancy and results in great robustness. Generally speaking, methods in robot navigation often fall into enormous computing complexity. However, our approach proposed a simple adaptive control algorithm.

Summary

We proposed a simple method to tracking a moving target using chaotic dynamics in a recurrent neural network model. Although chaotic dynamics could not always solve all complex problems with better performance, better results often were often observed on using chaotic dynamics to solve certain ill-posed problems, such as tracking a moving target and solving mazes (Suemitsu and Nara 2004). From results of the computer simulation, we can state the following several points.

- A simple method to tracking a moving target was proposed
- Chaotic dynamics is quite efficient to track a target that is moving along a circular trajectory.
- Performance of tracking a moving target of a linear trajectory is not better than that of a circular trajectory, however, to some linear trajectories, excellent performance was observed.
- The length of continuously staying time steps becomes long with the increase of synaptic connectivity r that can lead chaotic dynamics in the network.
- Continuously longer staying time in a certain basin seems useful to track a faster target.

References

Aihara K, Takabe T, Toyoda M (1990) Chaotic neural networks. Phys Lett A 114:333–340

Babloyantz A, Destexhe A (1986) Low-dimensional chaos in an instance of epilepsy. Proc Natl Acad Sci USA 83:3513–3517

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

Huber F, Thorson H (1985) Cricket auditory communication. Sci Amer 253:60–68

Hülse M, Pasemann F (2002) Dynamical neural Schmitt trigger for robot control. In: Dorronsoro J (ed) ICANN 2002: topics in artificial neural networks. International Conference On Artificial Neural Networks, Madrid, Spain, August 28–30, 2002. Lecture notes in computer science, vol 2415. Springer Verlag, Berlin, pp 783–788

Kaneko K, Tsuda I (2003) Chaotic itinerancy. Chaos 13(3):926–936

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

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

Nicolelis M (2001) Actions from thoughts. Nature 409:403–407

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

Tsuda I (1991) Chaotic itinerancy as a dynamical basis of hermeneutics in brain and mind. World Futures 32:167–184

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

Wessberg J, Stambaugh C, Kralik J, Beck P, Laubach M, Chapin J, Kim J, Biggs S, Srinivasan M, Nicolelis M (2000) Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. Nature 408:361–365