Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 71 (2008) 655–666
NEUROCOMPUTING
www.elsevier.com/locate/neucom

A behavior controller based on spiking neural networks for mobile robots

Xiuqing Wang[a,b], Zeng-Guang Hou[a,*], Anmin Zou[a], Min Tan[a], Long Cheng[a]

[a]*Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, P.O. Box 2728, Beijing 100080, China*
[b]*Vocational & Technical Institute, Hebei Normal University, Shijiazhuang 050031, China*

Abstract

Spiking neural networks (SNNs), as the third generation of artificial neural networks, have unique advantages and are good candidates for robot controllers. A behavior controller based on a spiking neural network is designed for mobile robots to avoid obstacles using ultrasonic sensory signals. Detailed structure and implementation of the controller are discussed. In the controller the integrated-and-firing model is used and the SNN is trained by the Hebbian learning algorithm. Under the framework of SNNs, fewer neurons are employed in the controller than those of the classical neural networks (NNs). Experimental results show that the proposed controller is effective and is easy to implement.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Spiking neural networks; Mobile robot; Obstacle avoidance; Hebbian learning; Ultrasonic data

1. Introduction

Nowadays autonomous navigation for robots has become one of the most active research areas. The autonomy of robots depends on the capability of the robots to explore unknown environments. The controller is a key issue for realization of the robot's autonomy. Soft computing methods have played important roles in the design of robot controllers. The commonly used soft computing methods include fuzzy logic methods, neural networks (NNs), neuro-fuzzy control and genetic algorithms (GA).

Fuzzy logic control has been widely used in robot navigations [5,35,36]. The evolutionary robots based on GA have strong adaptability and show great prospects [25,26]. Since NN controllers do not rely on the system model, they are suitable for uncertain and highly nonlinear situations. Moreover, NNs have strong parallel processing, adaptive and learning capabilities; thus they are popular in the design of robot controllers. Neuro-fuzzy controllers have the advantages of neural controllers as well as fuzzy logic's knowledge representation, and fuzzy deduction [12,31]. Reinforcement learning is also a good choice for the design of robot controllers [38].

Now artificial neural networks (ANNs) are in the third generation. The above mentioned neural controllers and the neuro-fuzzy controllers are all based on the first two generations of NNs. The first generation of ANNs consists of McCulloch–Pitts threshold neurons, and the second generation neurons use continuous activation functions to compute their output signals. The third generation of NNs is spiking neural networks (SNNs) [37]. The spiking neurons use pulse codings to incorporate spatial-temporal information in communication and computation, like real neurons do. SNNs represent more plausible models of real biological neurons than classical ones do.

SNNs, which convey information by individual spike times, have stronger computational power than other types of NNs. It has been proven that the networks of spiking neurons can simulate arbitrary feedforward sigmoidal neural nets and can thus approximate any continuous function [11,20].

Besides the computation capability, SNNs also show their capabilities in pattern recognition and classification. SNNs have been applied to solving high-dimensional

*Corresponding author. Tel.: +86 10 82614502; fax: +86 10 62650912.
E-mail address: zengguang.hou@ia.ac.cn (Z.-G. Hou).

0925-2312/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.08.025

cluster and nonlinear classifying problems successfully [4,15,24,30,32].

Though a lot of work have been done on SNNs, compared with the first two generations of ANNs, SNNs are still in their preliminary stage because of their relatively short history. The structure and the training rules are premature. There are few toolboxes for SNNs. But more and more people are paying attention to SNNs and trying to use them in different areas.

Researchers have tried to employ SNNs in the robotic area. GA is also applied in SNNs to find the optimal parameters. It has been proven that GA is an effective method for SNNs. Floreano et al. [8,9] have done a lot of work on evolution of spiking neural controllers for autonomous vision-based robots. They used the controller based on SNNs not only for wheeled robots but also for flying robots. Hagras et al. [14] presented an adaptive genetic algorithm to evolve the weights of the SNNs for the controller. Paolo [27] controlled a light-seeking robot using spike-timing dependent plasticity (STDP) in combination with an evolutionary strategy. Florian developed SNNs that implemented a seeking-push-release drive for a simple simulated agent to interact with objects [10].

Moreover, Soula et al. [34] used visual flow to train a recurrent SNN for a robot to avoid obstacles. Alamdari applied a model of self-organizing SNNs to environment representation and path planning for mobile robots [1]. Kubota and Nishida [19] proposed a prediction-based SNN controller using visual perception.

In recent years, a rigorous computational model, i.e. the liquid state machine (LSM), appeared, and neural micro-circuits for the implementation of LSMs have become a new trend for SNNs to be used in reality [13,22]. Burgsteiner et al. used an LSM to predict ball trajectories by a supervised learning algorithm, with input data from a video camera mounted on a robot participating in the robocup [6].

SNNs use temporal and spatial information, so they can be used for "real" dynamic environments. Spiking neurons can transmit and receive information through spikes' time. This leads to the possibility of fast and efficient implementations. Compared with the classical NNs, spikes are conveyed in SNNs, so SNNs have better robustness to noise. Moreover, spikes can be modeled relatively easily by digital circuits. As mentioned earlier, SNNs can compute any function that a second generation network can and usually with fewer neurons, so the neuro-chip based on SNNs has smaller size and consumes less power.

Since SNNs' output is pulses rather than continuous functions, the classical learning rules based on gradient decent methods cannot be applied to SNNs directly.

Because of the advantages of SNNs, they are suitable for robot controller design. In this study an obstacle avoidance controller based on an SNN is designed for mobile robots. The input information comes from ultrasonic sensors. In the proposed controller, the structure of SNN and training rules are not only effective but also easy to implement.

This paper is organized as follows: Section 2 introduces the basics of SNNs. Section 3 discusses the principle of the behavior controller in detail. Section 4 presents the experimental results. The paper is concluded in Section 5.

2. Spiking neural networks

2.1. Coding with spikes

Neuronal signals can be observed by placing a fine electrode close to the soma or axon of a neuron. By this way, the voltage trace in a typical recording shows a sequence of short pulses, called action potentials or spikes [21]. The variation of number and the time of spikes carries different information.

One can imagine at least three types of coding continuous signals with spikes [23]: (1) The frequency coding, neurons generate different frequency of spike trains as a response to different stimulus intensities. (2) Temporal coincidence coding, tighter coincidence of spikes recorded from different neurons represent higher stimulus intensity. (3) The delay coding, different delays of the spikes represent relative intensities of the different stimulus.

2.2. Spiking neuron and synapse models

Various spiking neuron models exist, such as the spike response model (SRM model), dynamic firing threshold model, and integrate-and-fire model (IAF model) [21]. There is also a new type of spiking synaptic model investigated recently, which is the dynamic synapse (DS) model [2]. IAF model is relatively simple and effective, so it is used for the proposed controller in this paper. Detailed descriptions of IAF model are presented in Section 3.6.

2.3. Training methods

The training algorithms of SNNs can be categorized as the supervised methods and the unsupervised methods as the classical NNs do. The unsupervised spike-based learning methods include long-term depression (LTD) learning, long-term potentiation (LTP) learning, STDP learning and spike-based Hebbian learning. Spike-based learning rules has been developed in [18,28,29,32,33]. In this study, unsupervised spike-based Hebbian learning [18] is used for the proposed controller. The supervised spike-based methods include spikeprop method [3], statistical learning methods, linear algebra methods, evolutionary methods, spike-based supervised-Hebbian learning and so on [17].

3. The principle of the behavior controller based on SNNs

3.1. The schematic diagram and the flow chart of the controller

The schematic diagram and the flow chart of the behavior controller are shown in Figs. 1 and 2 respectively.

3.2. The sonar sensory model

Ultrasonic sensor has some attractive properties, e.g. cheapness, reliability and so on, which make it widely used in mobile robots. Ultrasonic sensor systems generally calculate distance by the time of flight (TOF) method. The mobile robot CASIA-I used in the simulation has a peripheral ring of 16 evenly distributed Polaroid ultrasonic sensors, which is shown in Fig. 3.

In the obstacle-avoidance simulation, the readings of the sonar sensors from N. 1 to N. 9 are considered. The sensors are divided into three groups: Group 1, including sensors from N. 1 to N. 3; Group 2, from N. 4 to N. 6; Group 3, from N. 7 to N. 9. In each group, the smallest reading will be the testing result and be used as the input to the respective sensory neuron.

In the proposed controller, discrete time is used. The interval of sampling for the sonar sensors is the same as the interval of updating the connection weights of SNN in the controller and the speed of the robot. The interval is denoted as T.

3.3. The structure of the SNN in the behavior controller

There are three groups of sonar sensory information. So in the input layer of the SNN of the controller, there are three sensory neurons. In the input layer, there are one approximate neuron N_n and two turning neurons T_R and T_L. In the hidden layer, there are two hidden neurons H_1 and H_2. The role of neuron N_n is to judge whether the opposite obstacle is too close that the robot need to turn around. The sonar sensors' reading from Group 2 is fed into the approximate neuron. When sampling, in the time window some pulses are put into the turning neurons T_R or T_L. To which turning neuron pulses are input depends on what direction the mobile robot should turn to. If the left direction is preferred when the controller is designed, then some pulses are put into T_L. Logically, the robot cannot turn left and turn right at the same time. So the pulses can not be put into T_R and T_L at the same time. The controller is designed for mobile robot CASIA-I which is a wheeled mobile robot with two driving wheels, so in the output layer there are two motor neurons. The output spikes of the 1st motor neuron control the rotating speed of the left motor and the 2nd motor neuron corresponding to the right motor. The structure of the SNN for the controller is shown in Fig. 4.

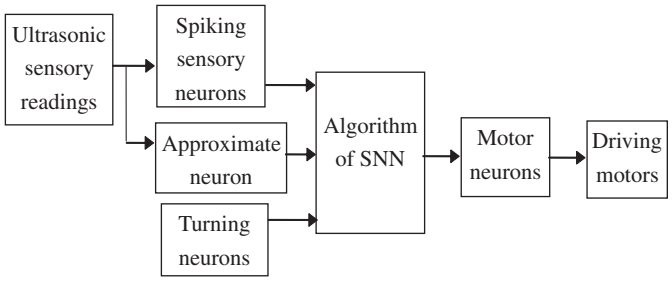

Fig. 1. Schematic diagram of the controller.

The 1st sensory neuron is an inhibiting neuron to the 2nd motor neuron, and the 3rd sensory neuron is an inhibiting neuron to the 1st motor neuron. The 2nd and the 3rd sensory neurons are all activating neurons to the 2nd motor neuron, and the 1st and the 2nd sensory neurons are all activating neurons to the 1st motor neuron. The signs of the connecting weights for the inhibiting neurons and the respective motor neurons are negative and that for the activating neurons and the respective motor neurons are positive. The initial connecting weight matrix for the sensory neurons and the motor neurons are set as $W = \begin{pmatrix} 1 & 1 & -1 \\ -1 & 1 & 1 \end{pmatrix} = [w(i,j)]_{2\times 3}$, where $i = 1, 2$, representing the ith motor neuron; $j = 1, 2, 3$, representing the jth sensory neuron.

The initial weights for the hidden neurons and the motor neurons equal to 2. The initial weights for the hidden neurons and the approximate neuron and the turning neurons are all constant 1.

3.4. Encoding the sonar sensory information into the frequency coding for the SNN's sensory neurons

In the controller, readings of the sonar sensors are encoded into spikes by frequency coding in time windows as the inputs of the sensory neurons. Smaller reading of the sonar sensor corresponds to fewer spikes in the time window, and the length of the time windows are the same for all the sensory neurons.

For example, if the reading of the sonar sensor is 400 mm, then there will be a spike in the time window; while if the reading of the sonar sensor is 4000 mm, then there will be 10 spikes in the time window. By this way, the readings of different groups are encoded into frequency coding. So the outputs of the sensory neurons are spikes of different frequencies in the time windows corresponding to the readings of three groups.

3.5. Coincidence detection for the approximate neuron and the turning neurons

The reading of the 2nd Group sonar is input to the approximate neuron N_n. The approximate neurons' coding is given by

$$I_{near} = 1 - \frac{x}{s_{near}}, \tag{1}$$

where x is the reading of the 2nd Group sonar sensor, and s_{near} is a constant. The firing threshold for the approximate spiking neuron N_n is ϑ_{N_n}. If $I_{near} \geqslant \vartheta_{N_n}$, the approximate neuron is fired and a spike is output.

Temporal coincidence coding is used for the approximate neuron N_n, the turning neurons T_R and T_L. That is, if T_R and N_n are fired at the same time, H_1 is fired and a spike is output, which is shown in Fig. 5(a); if T_R and N_n are fired at the same time, H_2 is fired and a spike is output from H_2, which is shown in Fig. 5(b).

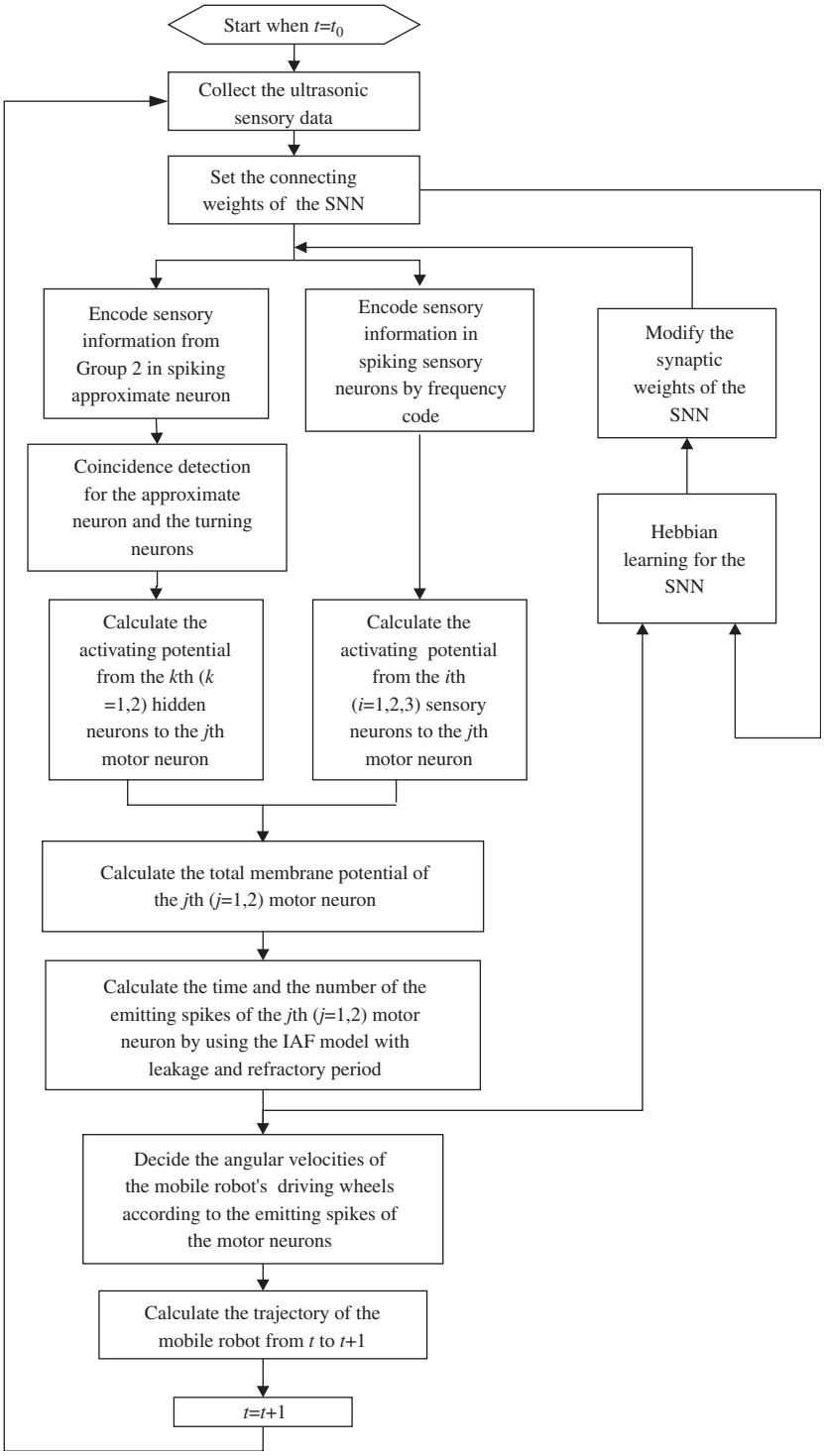

Fig. 2. Flow chart of the mobile robot behavior controller.

3.6. Integrated-and-firing model with refractory period in the SNN of the controller

The IAF model is an important example in the class of "threshold-fire models" [7,21]. In the SNN of the behavior controller, IAF model with refractory period as the spiking neuron model for the motor neurons is used. t in this section denotes time in the time window and the length of the time window is 100 ms in the designed controller. The details of the model used in the controller are in the following subsections:

3.6.1. The activating potential of incoming spikes

The activating potential of incoming spikes for the motor neurons includes two parts: $U_i^1(t)$ and $U_i^2(t)$. $U_i^1(t)$ represents the total activating potential of incoming spikes

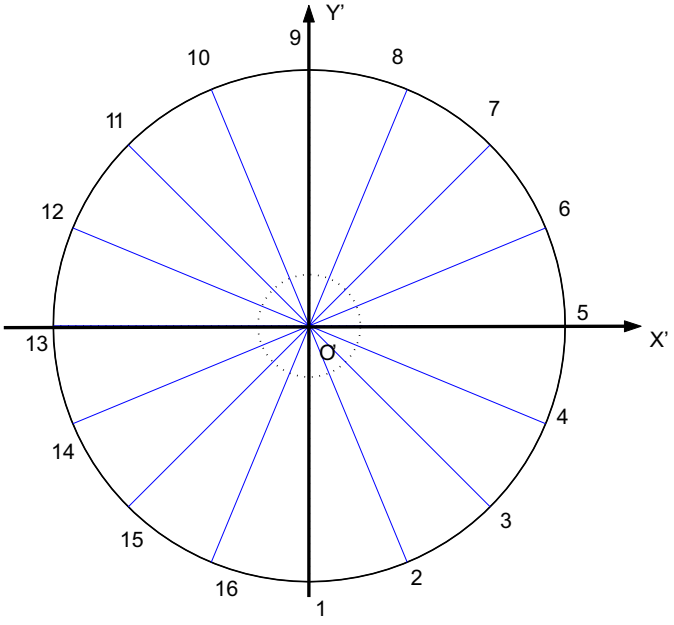

Fig. 3. Layout of the sonar sensors of CASIA-I.

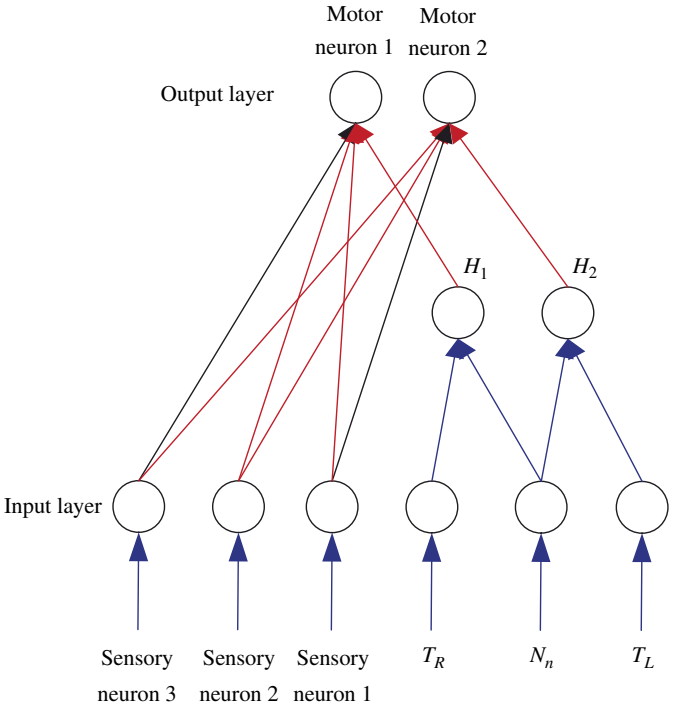

Fig. 4. Structure of SNN in the controller.

from the sensory neurons to the ith motor neuron at time t through the connecting weight w_{ij}, and $U_i^2(t)$ represents the activating potential of incoming spikes from the hidden neurons to the ith motor neuron at time t through the connecting weight w_{ik}^2:

$$\begin{cases} U_i^1(t) = \sum_j \sum_{t_j^{(f)}} w_{ij}\varepsilon(t - t_j^{(f)}), \\ U_i^2(t) = \sum_k \sum_{t_k^{(f)}} w_{ik}^2\varepsilon(t - t_k^{(f)}), \end{cases} \tag{2}$$

where $t_j^{(f)}$ represents the spikes' emitting time of the jth sensory neuron in the time window, and $t_k^{(f)}$ represents the spikes' emitting time of the kth hidden neuron in the time window.

$$\varepsilon(t) = \frac{t}{\tau_s}\exp\left(-\frac{t}{\tau_s}\right), \tag{3}$$

where τ_s is a time constant.

3.6.2. The membrane potential of the motor neuron

$V_i(t)$, the membrane potential of the ith motor neuron, is updated by the activating potential of incoming spikes as follows:

$$V_i(t) = \begin{cases} U_i^1(t) + U_i^2(t) & \text{if } U_i^1(t) + U_i^2(t) \geqslant V^{\text{rest}}, \\ V^{\text{rest}} & \text{otherwise.} \end{cases} \tag{4}$$

If the sum of $U_i^1(t)$ and $U_i^2(t)$ is lower than the resting potential V^{rest} (where $V^{\text{rest}} = 0$), the membrane potential is set to the resting potential V^{rest}.

3.6.3. Generation of spike for the motor neuron

$O_i(t)$ represents the output of the ith motor neuron. If the membrane potential of the motor neuron is larger than, or equal to, a threshold ϑ, the motor neuron is fired and emits a spike. At this time $O_i(t) = 1$ and the membrane potential is set to V^{rest}; otherwise $O_i(t)$ is set to 0 (there is no spike) and the membrane potential remains as same as the previous.

$$O_i(t) = \begin{cases} 1 & \text{if } V_i(t) \geqslant \vartheta, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Suppose n_1 and n_2 are the numbers of the emitting spikes of the motor neurons 1 and 2 in the time windows. n_1 and n_2 can determine the angular velocities of the driving wheels of the wheeled mobile robot respectively.

After the motor neuron emitting a spike, its membrane potential remains as V^{rest} and will not change during the refractory period δ^{abs}.

How the IAF model with refractory period is used in the spiking motor neuron is explained next. The following data are obtained from the process of the robot's roam and obstacle-avoiding simulation: the connecting weights $W = \begin{pmatrix} 2.643 & 1.802 & -1.420 \\ -2.656 & 3.737 & 3.009 \end{pmatrix}$, the readings of the three groups of sonar sensors are 3752.9, 1758.2, 1729.8 mm and these data are encoded into pulse frequency codings for the spiking sensory neurons. The incoming activating potential for the motor neuron 2 can be calculated according to Eqs. (2), (4), and the results are shown in Fig. 6(a). The membrane potential and the output pulse of the motor neuron 2 are illustrated in Fig. 6(b), (c). In Fig. 6(b), when $t = 29$ ms the incoming activating potential reached the

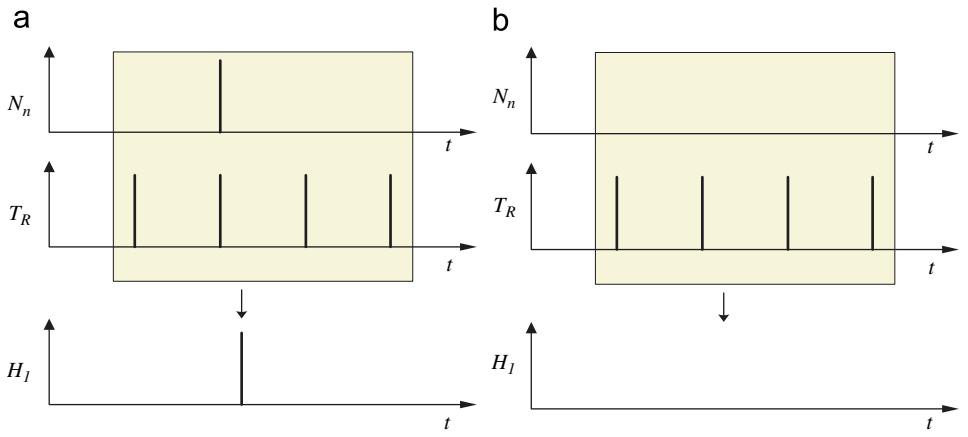

Fig. 5. Coincidence detection coding in the controller.

firing threshold, the motor neuron output one spike and entered into the refractory period. In the simulation the refractory period is 20 ms, so the period during $t = 29\sim49$ ms is the refractory period, and the membrane potential of the motor neuron 2 is resting potential V^{rest} (here $V^{\text{rest}} = 0$). After the refractory period, the membrane potential of the motor neuron is also calculated by Eq. (4). By calculation, when $t = 50, 72, 95$ ms the membrane potential of the motor neuron reached the firing threshold, and one spike was output, later the membrane potential of the motor neuron entered into the refractory period, all these are shown in Fig. 6.

3.7. Tuning the synaptic weights of the proposed controller by the unsupervised spike-based Hebbian rule

The Hebbian learning algorithm, which is correlation-based, is an important mechanism for the tuning of neural connections during development and thereafter. The Hebbian learning rule has been widely used in many NNs. In the behavior controller the connecting weights for the sensory neurons and motor neurons are updated by the spike-based Hebbian learning rules. Here the formula that Richard Kempter used in SNNs [18] is adopted. The formula is simplified as

$$\Delta w_{ij}(t) = l \times \sum_{t_j^f, t_i^n} W(s,t), \tag{6}$$

where l is the learning rate; t is the sampling time of the sonar sensory information and is also the time when synaptic weights are updated; s is the delay between presynaptic spike arrival and postsynaptic firing, and

$$s = t_j^f(k_j) - t_i^n(k_i), \tag{7}$$

Where $t_j^f(k_j)$ is the time of the k_jth spike of the jth sensory neuron in the given time window; $t_i^n(k_i)$ is the time of the k_ith spike of the ith motor neuron in the given time window.

$W(s,t)$ is used as a learning window function [18], and

$$W(s,t) = \begin{cases} \exp\left(\frac{s}{\tau^{\text{syn}}}\right)\left[A_+\left(1-\frac{s}{\tilde{\tau}_+}\right) + A_-\left(1-\frac{s}{\tilde{\tau}_-}\right)\right] \\ \quad \text{for } s \geqslant 0, \\ A_+ \exp\left(-\frac{s}{\tau_+}\right) + A_- \exp\left(\frac{-s}{\tau_-}\right) \\ \quad \text{for } s < 0 \end{cases} \tag{8}$$

where $\tau_+, \tau_-, \tau^{\text{syn}}$ are time constants. $\tilde{\tau}_+$ and $\tilde{\tau}_-$ are also time constants, and $\tilde{\tau}_+ = \tau^{\text{syn}}\tau_+/(\tau^{\text{syn}} + \tau_+)$, $\tilde{\tau}_- = \tau^{\text{syn}}\tau_-/(\tau^{\text{syn}} + \tau_-)$. The dimensionless constants A_+ and A_- determine the strength of synaptic potentiation and depression respectively. A detailed explanation of the choice of the learning window of the Hebbian learning algorithm can be found in [18] and its related references.

At last the connecting weights are updated by

$$w_{ij} = r^{\text{dis}} \times w_{ij}(t) + \Delta w_{ij}(t), \tag{9}$$

where r^{dis} ($r^{\text{dis}} \in (0,1)$) is a discount rate for the connecting weights.

3.8. The kinematic model of the mobile robot

In this study, the mobile robot shown in Fig. 7 is a system subject to nonholonomic constraints. In a 2-dimensional Cartesian space, its pose is represented by

$$q = (x, y, \theta)^{\text{T}}, \tag{10}$$

where $(x,y)^{\text{T}}$ is the position of the robot in the reference coordinate system XOY, and the heading direction θ is taken counterclockwise from the positive direction of X-axis. $X'O'Y'$ is the coordinate for the robot system. The rectangle in solid line represents the camera, and the rectangles in dashed line represents the wheels.

If Δt is small enough, the mobile robot's trajectory can be approximated by the following equation from

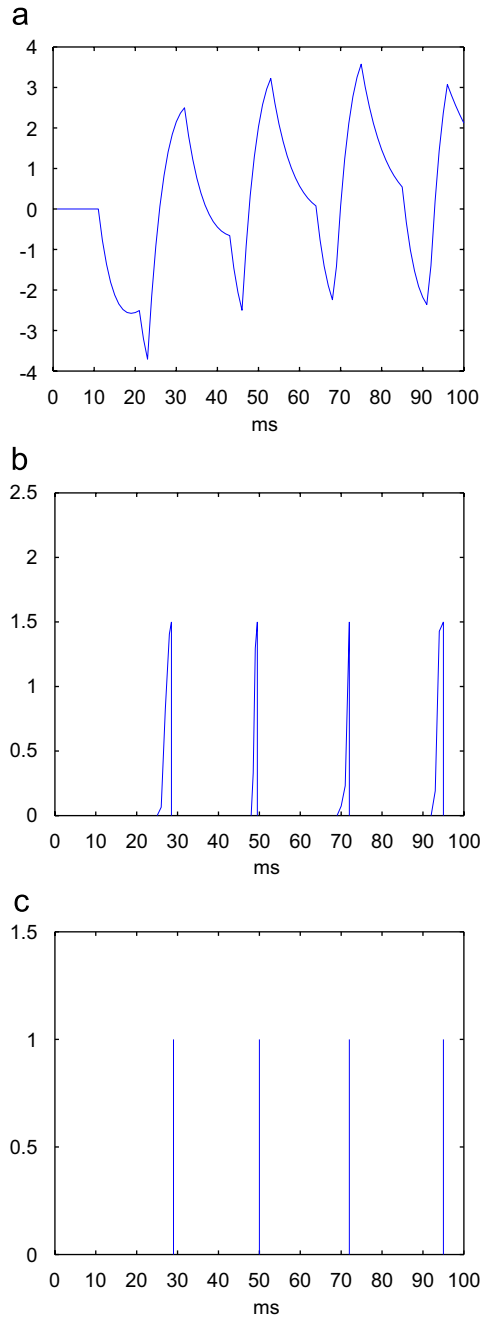

Fig. 6. Membrane potential and outputting pulses for the motor neuron. (a) Activating potential of incoming spikes. (b) Membrane potential of the motor neuron 2. (c) Outputting pulses of the motor neuron 2.

t to $t+k\Delta t$:

$$\begin{cases} x(k+1) = x(k) + v\cos(\theta(k))\Delta t, \\ y(k+1) = y(k) + v\sin(\theta(k))\Delta t, \\ \theta(k+1) = \theta(k) + \omega(k)\Delta t, \end{cases} \tag{11}$$

where k is an integer and $k = 1, 2, \ldots, [1/\Delta t]$.

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{pmatrix}^{-1} \times \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix}, \tag{12}$$

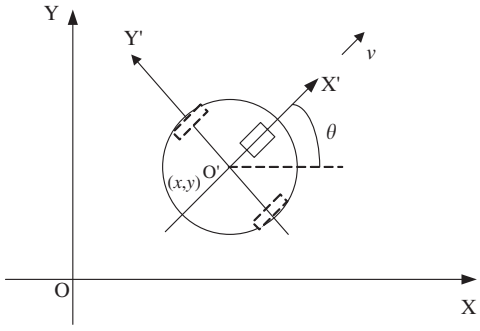

Fig. 7. Illustration of the mobile robot's pose.

where v and ω are the linear and angular velocities of the mobile robot. ω_r and ω_l are the angular velocities of the right wheel and the left wheel respectively. r is the radius of the wheel, and b is the half distance between the two driving wheels.

3.9. Discussion about the robot's velocity

From Eq. (12) the following equation can be derived:

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2b} & -\frac{r}{2b} \end{pmatrix} \times \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix},$$

where $\omega_r = n_2 \times p$, $\omega_l = n_1 \times p$, n_1, n_2 are the number of the outputting pulses from the motor neuron 1 and 2. p represents the angular velocity per outputting pulse and its unit is rad/s.

T is the time interval for updating the robot's velocities according to the motor neurons' outputting pulse. During each T, the maximum linear velocity of the robot should satisfy the following inequality:

$$v_{\max} = \frac{r}{2}\left(\omega_{r_{\max}} + \omega_{l_{\max}}\right) < \frac{d_{\text{obst}_{\min}} - d_{\text{safe}}}{T}, \tag{13}$$

where $v_{\max}$ is the maximum linear velocity of the robot during the time interval T, $d_{\text{obst}_{\min}}$ is the minimum distance between the robot and obstacles, and d_{safe} is the safe threshold for the distance between the robot and the obstacle.

$\omega_{r_{\max}}$ and $\omega_{l_{\max}}$ are as follows:

$$\begin{cases} \omega_{r_{\max}} = n_{2_{\max}} \times p, \\ \omega_{l_{\max}} = n_{1_{\max}} \times p. \end{cases} \tag{14}$$

$n_{1_{\max}}$, $n_{2_{\max}}$ satisfy the following inequalities:

$$\begin{cases} n_{1_{\max}} \leqslant \dfrac{t_w}{\delta^{\text{abs}}} \\ n_{2_{\max}} \leqslant \dfrac{t_w}{\delta^{\text{abs}}}. \end{cases} \tag{15}$$

According to Eqs. (13)–(15), the following inequality can be obtained:

$$r\frac{t_w}{\delta^{\text{abs}}}p < \frac{d_{\text{obst}_{\min}} - d_{\text{safe}}}{T}, \tag{16}$$

where t_w is the length of the time window and $t_w = 100$ ms in the designed controller. In each T, Eq. (16) and $v = r/2(n_1 + n_2)p < v_{\max}$ should be met to ensure that the robot roam with obstacle free.

In the experiment, $p = 0.5$ rad/s, $\omega_r = n_2 \times 0.5$ rad/s, $\omega_l = n_1 \times 0.5$ rad/s.

4. Experiments

4.1. The parameters of the robot and the controller in the simulation

In the simulation, the parameters of the robot are: $b = 225$ mm, $r = 50$ mm.

The length of the time windows for pulse encoding of the sensory neurons and the motor neurons is 100 ms. The initial connecting weights $W = \begin{pmatrix} 1 & 1 & -1 \\ -1 & 1 & 1 \end{pmatrix}$. T, the interval time of the sensors' sampling, the update of the weights and the speed of the simulated robot is 1 s. The resting potential $V^{\text{rest}} = 0$. The learning rate $l = 0.1$, the discount rate $r^{\text{dis}} = 0.8$, Δt for calculating the trajectory of the robot is 10 ms. The constants for the Hebbian learning window functions $W(s)$ are: $A_+ = 1$, $A_- = -1$, $\tau_+ = 1$ ms, $\tau_- = 20$ ms, $\tau^{\text{syn}} = 5$ ms. The time constant τ_s for ε function is 8 ms.

4.2. Experimental results

4.2.1. Experimental results for different initial locations and poses

The obstacle avoidance and roam simulations are performed for the mobile robot. By using the designed controller described in Section 3, 20 different starting positions are selected for the robot with various initial

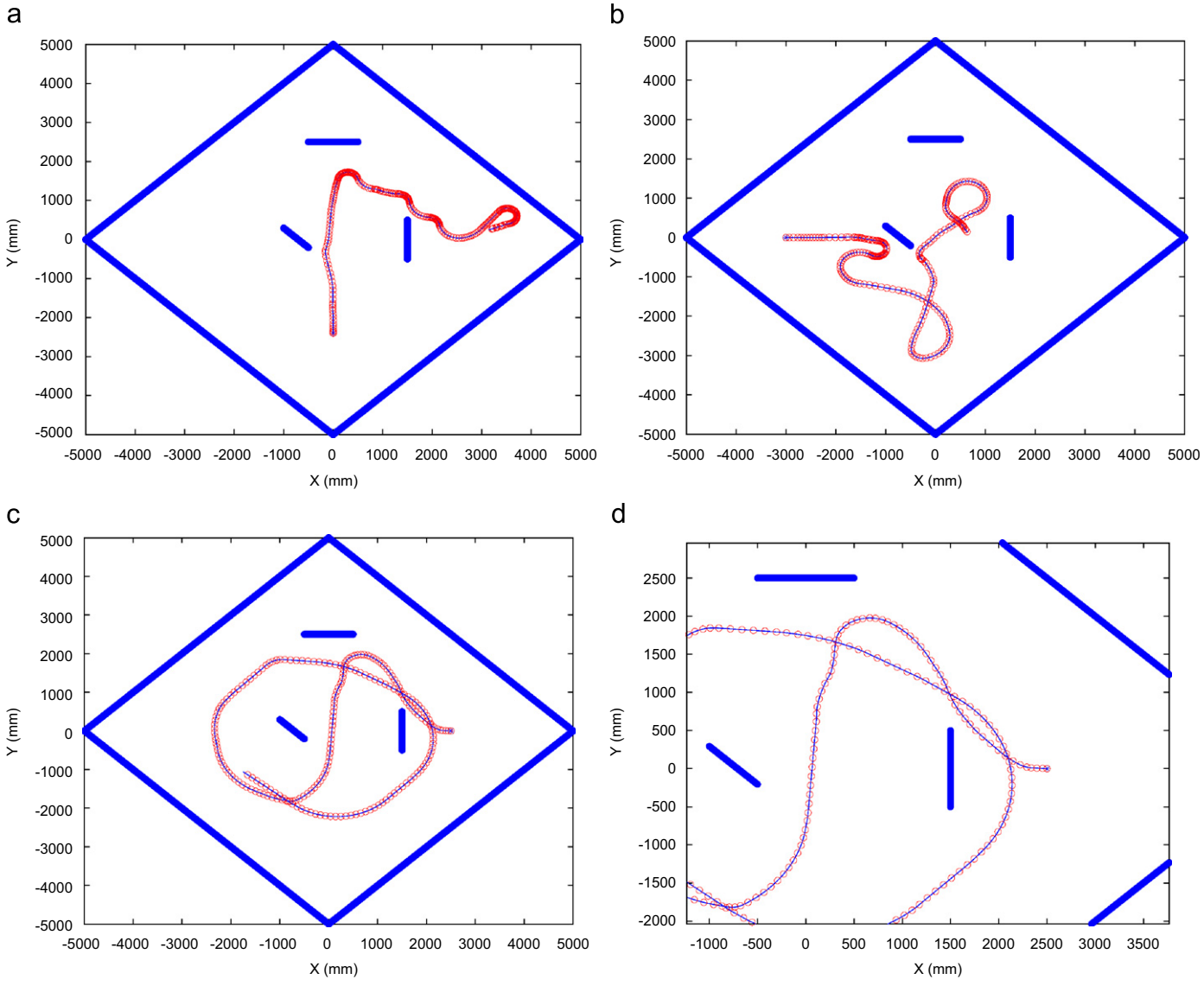

Fig. 8. Experimental results for various initial poses and locations of the robot. (a) Initial q is $(0, -2400, 90^\circ)$. (b) Initial q is $(-3000, 0, 0^\circ)$. (c) Initial q is (2500, 0, 180°). (d) Zooming in for (c).

poses. The robot can avoid the obstacle successfully for all the starting situations. The simulation environment is a square area with walls, and the size of the square area is $7070 \times 7070\,\mathrm{mm}^2$. δ^{abs} represents the refractory period, ϑ represents the firing threshold, and l is the learning rate for tuning the connecting weights. Fig. 8 shows some of those trajectories when the robot has different initial poses and $l = 0.1, \delta^{\mathrm{abs}} = 20\,\mathrm{ms}$, $\vartheta = 1.5$. In Fig. 8 the small circles represent the locations of the robot at the sampling time and the sampling interval is 1 ms. The thin lines represent the trajectories of the robot and "*" stands for the initial position of the robot. In the square area, the thick lines stand for the obstacles and the walls around. From the simulation it can be seen that the obstacle-avoiding controller based on SNNs is effectively.

4.2.2. Experimental results for the controller with and without Hebbian learning

NNs have learning ability and can improve their properties by learning. In the proposed controller, the unsupervised spike-based Hebbian learning algorithm is used and the controller can be trained on line. To illustrate the role of the Hebbian learning algorithm in the controller, the simulations are carried out under the two conditions: (1) The connecting weights are tuned by the Hebbian learning algorithm. (2) The connecting weights are not tuned and just remain the same.

In this simulation the initial pose of the robot is $(0, -2400, 90^\circ)$, the refractory period of the spiking motor neuron is $\delta^{\mathrm{abs}} = 10\,\mathrm{ms}$, $r^{\mathrm{dis}} = 0.9$. The other parameters are the same as the previous sections. The robot's

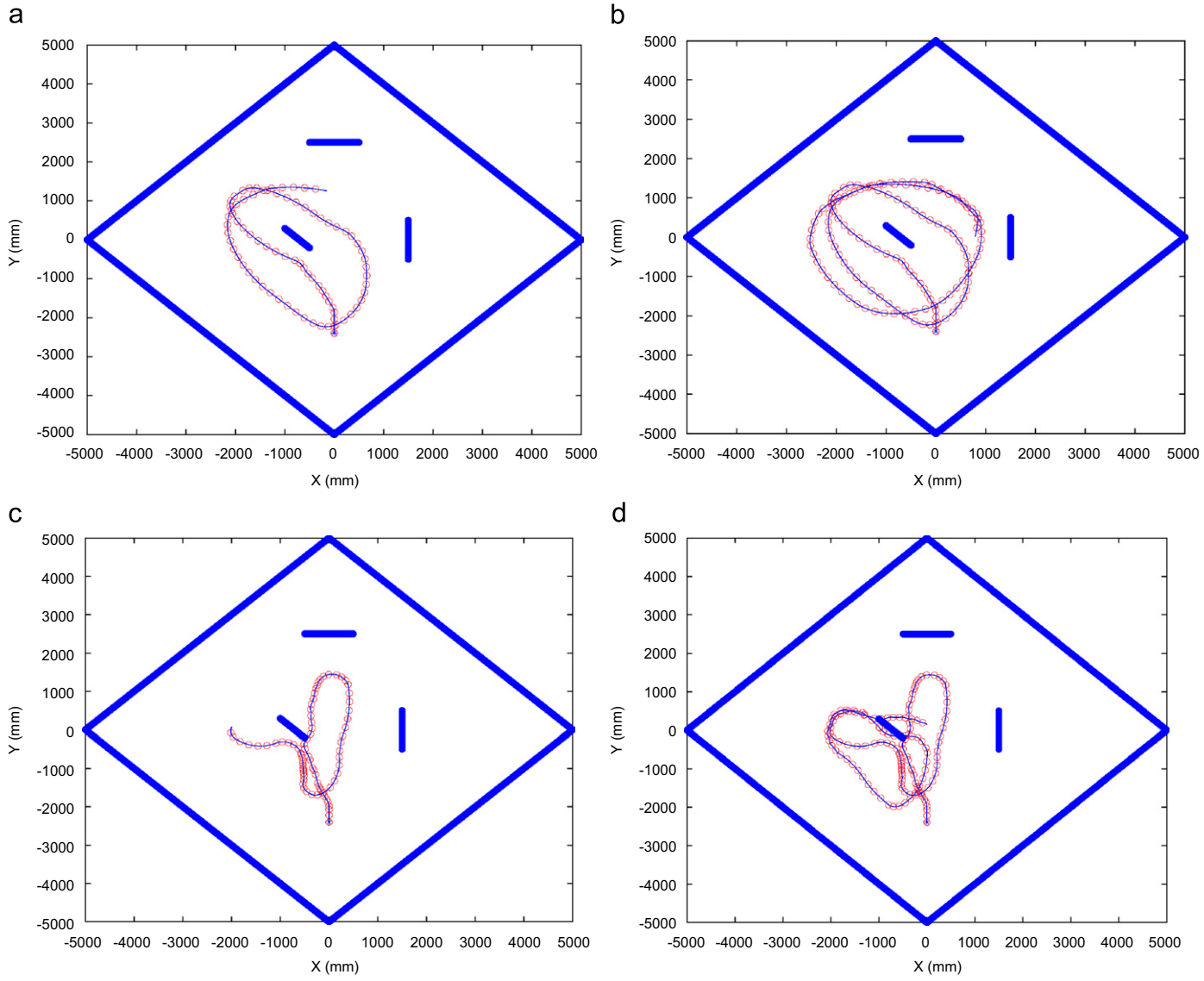

Fig. 9. Experimental results for the controller with and without Hebbian learning. (a) Experimental result after 80 steps with Hebbian learning. (b) Experimental result after 140 steps with Hebbian learning. (c) Experimental result after 80 steps without Hebbian learning. (d) Experimental result after 140 steps without Hebbian learning.

trajectories for the controller with Hebbian learning are shown in Fig. 9(a) and (b), and those for the controller without Hebbian learning are shown in Fig. 9(c) and (d). When the connecting weights and the robot's velocity are updated, a learning step is finished. In this simulation the interval between the two learning steps is 1 s. Fig. 9(c), (d) are the obstacle avoidance behavior when the controller without learning. Through Hebbian learning the controller can make the robot find an obstacle-free path by learning from the environment and at last reached into an equilibrium state. When in the equilibrium state the robot always follows the same obstacle-free path. The controller without learning can avoid the obstacle in some extent, but its obstacle-avoiding effect is worse than that of the controller with Hebbian learning.

4.2.3. Experimental results for different firing thresholds and refractory periods

For different firing threshold and refractory period, different trajectories for the robot are obtained when its pose q at the starting point is (2500, −2000, 90°).

In Fig. 10(a) and (b), the refractory period are 10 ms, and that for Fig. 10(c), (d) are 25 ms. The firing threshold 1.5 for Fig. 10(a) and (d), and 1 for Fig. 10(b), (c).

The refractory period and the firing threshold of the spiking neuron affect the output spikes of the SNNs directly, while the number of emitting spikes decides the angular velocities of the wheels of the mobile robot. Using the same sensory information, if the threshold is smaller, there are more emitting spikes in the time window, and the ratio $|(n_1 - n_2)/(n_1 + n_2)|$ is smaller than that of the larger threshold. So the curvature of the turning around

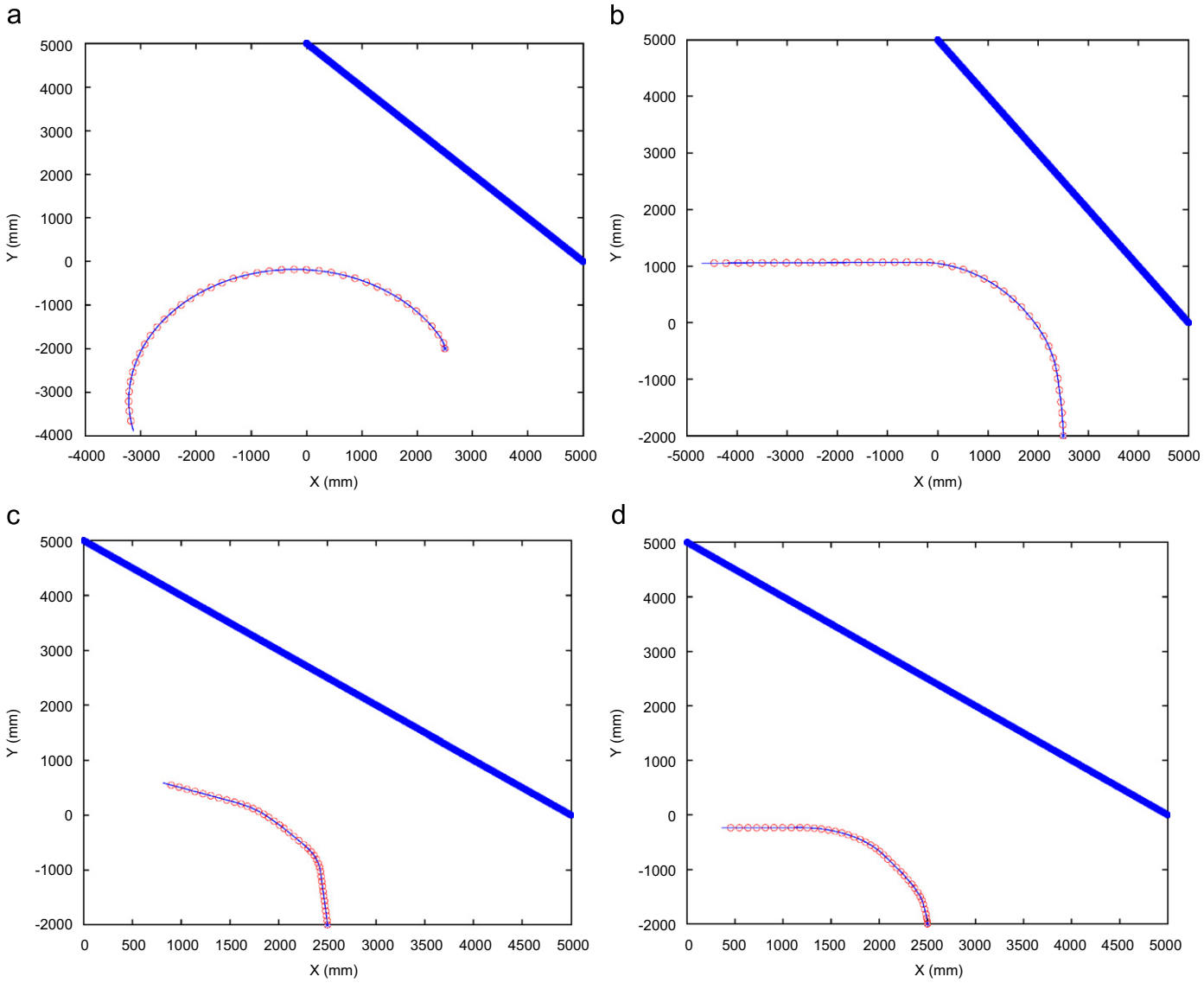

Fig. 10. Experimental results for various δ^{abs} and ϑ. (a) $\delta^{abs} = 10$ ms, $\vartheta = 1.5$. (b) $\delta^{abs} = 10$ ms, $\vartheta = 1$. (c) $\delta^{abs} = 25$ ms, $\vartheta = 1$. (d) $\delta^{abs} = 25$ ms, $\vartheta = 1.5$.

trajectory will be smaller, as shown in Fig. 10. Similarly, the smaller refractory period, there will be more emitting spikes in the time window, and the smaller $|(n_1 - n_2)/(n_1 + n_2)|$ is. So the curvature of the trajectory will also be smaller, as shown in Fig. 10.

5. Conclusions

A novel behavior controller for mobile robots based on SNNs using ultrasonic sensory information is designed. Detailed descriptions for the controller are given in this paper. The ultrasonic information is encoded into frequency codings for the sensory neurons. In the controller, the spiking neurons adopt the IAF model with refractory period, and the SNN is tuned by the unsupervised-spike-based Hebbian learning algorithm. Compared with the classical NNs, fewer spiking neurons are used in the controller and the training method is relatively simple. The simulation results show that the controller can be used in obstacle avoidance successfully. The angular velocities of the mobile robot's driving wheels can be controlled by the emitting spikes of the motor neurons directly. The controller has simple structure and can be implemented easily. Efforts are underway on how to set optimal parameters and select the proper training methods for the SNNs.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grants 60635010 and 60775043) and the Natural Science Foundation of Hebei Province (Grant F2004000180).

References

[1] A.R.S.A. Alamdari, Unknown environment representation for mobile robot using spiking neural networks, Trans. Eng. Comput. Technol. 6 (2005) 49–52.
[2] A. Belatreche, P. Liam, M. McGinnity, Advances in design and application of spiking neural networks, Soft Comput. 11 (3) (2007) 239–248.
[3] S.M. Bohte, J.N. Koka, H.L. Poutré, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (2002) 17–37.
[4] S.M. Bohte, H.L. Poutre, J.N. Kok, Unsupervised clustering with spiking neurons by sparse temporal coding and multi-layer spike Neural Network, IEEE Trans. Neural Networks 13 (2) (2002) 426–435.
[5] A. Bonarini, G. Invernizzi, T.H. Labella, An architecture to coordinate fuzzy behaviors to control an autonomous robot, Fuzzy sets Syst. 134 (2003) 101–115.
[6] H. Burgsteiner, M. Kröll, A. Leopold, G. Steinbauer, Movement prediction from real-world images using a liquid state machine, Applied Intelligence, in: Lecture Notes in Computer Science, vol. 3533, Springer, Berlin, 2005, pp. 121–130.
[7] D. Floreano, Y. Epars, J.C. Zufferey, C. Mattiussi, Evolution of spiking neural circuits in autonomous mobile robots, Int. J. Intelligent Syst. 21 (2006) 1005–1024.
[8] D. Floreano, C. Mattiussi, Evolution of spiking neural controllers for autonomous vision-based robots, in: ER 2001, Lecture Notes in Computer Science, vol. 2217, Springer, Berlin, 2001, pp. 38–61.
[9] D. Floreano, J.C. Zufferey, J.D. Nicoud, From wheels to wings with evolutionary spiking neurons, Artif. Life 11 (2005) 121–138.
[10] R.V. Florian, Spiking neural controllers for pushing objects around, in: Lecture Notes in Computer Science, vol. 4095, Springer, Berlin, 2006, pp. 570–581.
[11] W. Gerstner, Time structure of the activity in neural network models, Phys. Rev. E 51 (1995) 738–758.
[12] J. Godjavec, N. Steele, Neuro-fuzzy control for basic mobile robot behaviors, in: Fuzzy Logic Techniques for Autonomous Vehicle Navigation, Springer, 2000, pp. 97–117.
[13] A. Gupta, Y. Wang, H. Markram, Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex, Science 287 (2000) 273–278.
[14] H. Hagras, A. Pounds-Cornish, M. Colley, Evolving spiking neural network controllers for autonomous robots, in: The Proceeding of IEEE International Conference on Robotics & Automation 2004, New Orleans, LA, USA, 2004, pp. 4620–4626.
[15] J.J. Hopfield, Pattern recognition computation using action potential timing for stimulus representation, Nature 376 (1995) 33–36.
[17] A. Kasiski, F. Ponulak, Comparison of supervised learning methods for spike time coding in spiking neural networks, Int. J. Appl. Math. Comput. Sci. 16 (2006) 101–113.
[18] R. Kempter, W. Gerstner, J.L. van Hemmen, Hebbian learning and spiking neurons, Phys. Rev. E 59 (1999) 4498–4514.
[19] N. Kubota, K. Nishida, The role of spiking neurons for visual perception of a partner robot, in: The Proceeding of IEEE International Conference on Fuzzy Systems 2006, Vancouver, BC, Canada, 2006, pp. 530–537.
[20] W. Maass, Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons, in: Advances in Neural Information Processing Systems, vol. 9, The MIT Press, Cambridge, MA, 1997.
[21] W. Maass, C.M. Bishop (Eds.), in: Pulsed Neural Networks, MIT-Press, Cambridge, MA, 1999.
[22] W. Maass, T. Natschläger, T. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations, Neural Comput. 14 (2004) 2531–2560.
[23] Z. Nadasdy, Spatio-temporal patterns in the extracellular recording of hippocampal pyramidal cells: from single spikes to spike sequences, Ph.D. Thesis, Rutgers University, 1998.
[24] T. Natschläer, B. Ruf, Spatial and temporal pattern analysis via spiking neurons, Network: Comput. Neural Syst. 9 (3) (1998) 319–332.
[25] A.L. Nelson, Competitive relative performance and fitness selection for evolutionary robotics, Ph.D. Thesis, North Carolina State University, USA, May, 2003.
[26] S. Nolfi, D. Floreano, O. Miglino, F. Mondada, How to evolve autonomous robots: different approaches in evolutionary robotics, in: Proceedings of the IV International Workshop on Artificial Life, MIT Press, Cambridge, MA, 1994.
[27] E.A.D. Paolo, Spike timing dependent plasticity for evolved robots, Adaptive Behav. 10 (3–4) (2002) 243–263.
[28] W.R. Ritz, J.L. van Hemmen, Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns, Biol. Cybern. 69 (1993) 503–515.
[29] P.D. Roberts, Computational consequences of temporally asymmetric learning rules: I, Differential Hebbian learning, J. Comput. Neurosci. 7 (1999) 235–246.
[30] B. Ruf, Computing and learning with spiking neurons—Theory and simulations, Ph.D. Thesis, Institute for Theoretical Computer Science Technische University at Graz, Austria, May, 1998.
[31] P. Rusu, E.M. Petriu, T.E. Whalen, Behavior-based neuro-fuzzy controller for mobile robot navigation, IEEE Trans. Instrum. Meas. 52 (4) (2003) 1335–1340.
[32] B.M. Schmitt, Unsupervised learning in networks of spiking neurons using temporal coding, in: Proceedings of the 7th International

Conference on Artificial Neural Networks (ICANN97), Springer, Heidelberg, 1997, pp. 361–366.
[33] W. Senn, M. Tsodyks, H. Markram, An algorithm for synaptic modification based on exact timing of pre- and postsynaptic action potentials, in: Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN97), Springer, Heidelberg, 1997, pp. 121–126.
[34] H. Soula, A. Alwan, G. Beslon, Learning at the edge of chaos: temporal coupling of spiking neurons controller for autonomous robotic, in: Proceedings of American Association for Artificial Intelligence (AAAI) Spring Symposia on Developmental Robotic 2005, Stanford, USA, 2005, p. 6.
[35] S. Thongchai, Behavior-based learning fuzzy rules for mobile robots, in: Proceeding of American Control Conference 2002, Anchorage, Alaska, USA, 2002, pp. 995–1000.
[36] S. Thongchai, S. Suksakulchai, D.M. Wilkes, Sonar behavior-based fuzzy control for a mobile robot, in: Proceeding of IEEE International Conference on Systems, Man, and Cybernetics 2000, Nashville, Tennessee, USA, 2000, pp. 425–430.
[37] J. Vreeken, Spiking neural networks, an introduction, Technical Report UU-CS-2003-008, Institute for Information and Computing Sciences, Utrecht University, 2002. ⟨http://ai-lab.cs.uu.nl/pubs/SNN_Vreeken_Introduction.pdf⟩.
[38] C. Ye, N.H.C. Yung, D.W. Wang, A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance, IEEE Trans. Syst. Man Cybern Part B 33 (1) (2003) 17–27.

Xiuqing Wang received the B.S. degree (with honors) in precision instrument from Tianjin University, Tianjin, China, in 1992, and the M.S. degree (with honors) in control theory and control engineering from Hebei Technology University, Tianjin, China, in 2000. She is a Ph.D. Candidate at the Institute of Automation, Chinese Academy of Sciences, Beijing, China. Her research interests are neural networks, multisensor-fusion, and their applications to robotics.

Zeng-Guang Hou received the B.E. and M.E. degrees in electrical engineering from Yanshan University, Qinhuangdao, China, in 1991 and 1993, respectively. He received the Ph.D. degree in electrical engineering from Beijing Institute of Technology, Beijing, China, in 1997. He is a Full Professor at the Institute of Automation, Chinese Academy of Sciences, Beijing, China. Dr. Hou is currently serving as an Associate Editor for *IEEE Computational Intelligence Magazine*, and an Editorial Board Member of *International Journal of Intelligent Systems Technologies and Applications*. He served as a Guest Editor of special issues of *International Journal of Vehicle Autonomous Systems* on "Computational Intelligence and Its Applications to Mobile Robots and Autonomous Systems," and of *Soft Computing* on "Fuzzy-Neural Computation and Robotics". His research interests include neural networks, optimization algorithms, robotics, and intelligent control systems.

Anmin Zou received the B.S. degree (with honors) in control engineering from Northeast University, Shenyang, China, in 2001. He is a Ph.D. Candidate at the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests are robot control, neural networks, and NNs' application to robotics.

Min Tan received the B.S. degree in control engineering from Tsinghua University, Beijing, China, in 1986 and the Ph.D. degree in control theory and control engineering from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1990. He is a Full professor at the Institute of Automation, Chinese Academy of Sciences. Dr. Tan is an Acting Committee Member of Robotics Technology of the National Hi-Tech Development (863) Program. He is also a Director of several academic societies, such as the Artificial Intelligence Society, and Youth Automation Society of China, etc. Dr. Tan is currently serving as an editorial board member of several journals, such as *Acta Automatica Sinica*, *Robot*, etc. He also got several academic awards from Chinese Academy of Sciences and the Beijing Science and Technical Committee. His research interests include advanced robot control, multirobot system, biomimetic robot, and manufacturing system.

Long Cheng received the B.S. degree (with honors) in control engineering from Nankai University, Tianjin, China, in 2004. He is a Ph.D. Candidate at the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include neural networks, optimization, nonlinear control, and their applications to robotics.