Advanced Robotics, Vol. 21, No. 10, pp. 1215–1229 (2007)
© VSP and Robotics Society of Japan 2007.
Also available online - www.brill.nl/ar

Full paper

Reinforcement learning of a continuous motor sequence with hidden states

HIROAKI ARIE [1,*], TETSUYA OGATA [2], JUN TANI [3] and SHIGEKI SUGANO [1]

[1] *Department of Mechanical Engineering, Waseda University, 3-4-1 Okubo Shinjuku-ku, Tokyo 169-8555, Japan*
[2] *Graduate School of Informatics, Kyoto University, Yoshida-honmachi Sakyo-ku, Kyoto 606-8501, Japan*
[3] *Brain Science Institute, RIKEN, 2-1 Hirosawa Wako-shi, Saitama 351-0198, Japan*

Received 3 October 2006; accepted 23 January 2007

Abstract—Reinforcement learning is the scheme for unsupervised learning in which robots are expected to acquire behavior skills through self-explorations based on reward signals. There are some difficulties, however, in applying conventional reinforcement learning algorithms to motion control tasks of a robot because most algorithms are concerned with discrete state space and based on the assumption of complete observability of the state. Real-world environments often have partial observablility; therefore, robots have to estimate the unobservable hidden states. This paper proposes a method to solve these two problems by combining the reinforcement learning algorithm and a learning algorithm for a continuous time recurrent neural network (CTRNN). The CTRNN can learn spatio-temporal structures in a continuous time and space domain, and can preserve the contextual flow by a self-organizing appropriate internal memory structure. This enables the robot to deal with the hidden state problem. We carried out an experiment on the pendulum swing-up task without rotational speed information. As a result, this task is accomplished in several hundred trials using the proposed algorithm. In addition, it is shown that the information about the rotational speed of the pendulum, which is considered as a hidden state, is estimated and encoded on the activation of a context neuron.

Keywords: Recurrent neural network; reinforcement learning; actor–critic method; perceptual aliasing problem; pendulum swing-up.

1. INTRODUCTION

Animals learn from the consequences of their actions. Actions that are followed by some reward are more likely to be generated again in the future; conversely,

*To whom correspondence should be addressed. E-mail: arie@sugano.mech.waseda.ac.jp actions that are followed by punishment are less likely to reoccur. This ability, to improve behavior through learning, is important to develop autonomous adaptive systems. Reinforcement learning is one such machine learning framework in which a robot takes a series of actions that changes the state of the robot in an environment and the environment provides feedback in the form of either reward or punishment as reinforcement signals. Robots change their control law according to these signals. This learning algorithm does not require a teacher who tells a robot about the target actions; instead, simply reinforcement signals are needed. Therefore, this learning algorithm can be a useful tool for the exploration of creating robot control systems that autonomously learn through experience. In some studies, reinforcement learning is used to create developmental robots [1–3]. There are some difficulties, however, in applying conventional reinforcement learning frameworks to continuous motor control tasks of robots.

First, most reinforcement learning frameworks are concerned with discrete actions. When the action space is discrete, the implementation of reinforcement learning is straightforward [4]. A robot selects the action that is expected to maximize the sum of total future rewards from a fixed set of actions.

For instance, in an object handling task, a robot learns to select an action such as reaching, grabbing, carrying or releasing an object for each step. However, this approach is hardly applicable to smooth behavior such as object grasping [5] because dividing such a behavior into a set of actions is difficult.

In contrast, when the action domain is continuous, quantizing the action space and applying the same methods or employing somewhat adapted methods without quantization [6, 7] is necessary.

Second, most reinforcement learning frameworks consider tasks where the state is completely observable. Robots that interact with an environment through their sensors and effectors encounter the perceptual limitation problem. In many cases, the actual state of the system cannot be identified explicitly just by looking at the current sensory cues. However, the state of the system can be identified through more context-dependent methods by utilizing the history of sensory-motor experiences. Accordingly, the robot needs to learn to use its internal state in combination with perception. It is shown that this problem, which is called the ‘perceptual aliasing problem’, can be solved by using a kind of memory that stores previous perceptions [8–10].

In this paper, we propose a reinforcement learning method that can deal with these problems simultaneously. The main idea of the proposed method is combining one of the reinforcement learning algorithms with the continuous time recurrent neural network (CTRNN) learning scheme. In Section 2, we describe the proposed algorithm. In Section 3, we describe the experiment. Results are shown in Section 4 and we summarize our findings in Section 5.

2. PROPOSED METHOD

2.1. Reinforcement learning algorithm

The actor–critic method [4] utilized in this study is one of the temporal difference [11] family of reinforcement learning methods.

As shown in Fig. 1, the controller consists of two parts called an actor and a critic. In this method, the actor plays the role of a controller whose output decides the behavior of a robot and the critic approximates the value function $V(t)$.

Temporal difference learning provides an effective approach for the learning robot which is under the lack of direct teaching signals. However, applying traditional temporal difference learning to motion control of a robot is difficult due to the problem of dealing with continuous space and time. For this problem, Doya derived temporal difference learning for continuous space and time tasks [12].

In his method, a continuous-time deterministic system:

$$\dot{X}(t) = f\big(X(t), U(t)\big) \tag{1}$$

is considered, where X is the state and U is the action. The goal is to find a policy (control law) $U(t)$ that maximizes the expected cumulative rewards for a certain period in the future defined by:

$$V(t) = \int_t^{\infty} \frac{1}{\tau} \mathrm{e}^{-(s-t)/\tau} r(s)\, \mathrm{d}s, \tag{2}$$

where $r(t)$ is the immediate reward for the state at time t. $V(t)$ is called the value function of state $X(t)$ and τ is the time constant for discounting future rewards. This infinit-horizon formulation takes the long-run reward that the robot will receive from the environment. Using an approximation of this function, a robot can estimate the value of the state even if it cannot receive the reward signal at the time.

The basic idea in temporal difference learning is to predict future reinforcement. By differentiating (2) with respect to time t we have:

$$\tau \frac{\mathrm{d}}{\mathrm{d}t} V(t) = V(t) - r(t). \tag{3}$$

The temporal difference error $\hat{r}(t)$ is defined as a prediction error from (3):

$$\hat{r}(t) = r(t) - \frac{1}{\tau} V(t) + \frac{\mathrm{d}}{\mathrm{d}t} V(t). \tag{4}$$

If the current prediction for $V(t)$ is accurate, the temporal difference error should be equal to zero. In addition, if the prediction value for the current state is small, the temporal difference error becomes positive. Accordingly, in the actor–critic method, the temporal difference error is used as the reinforcement signal for policy improvement.

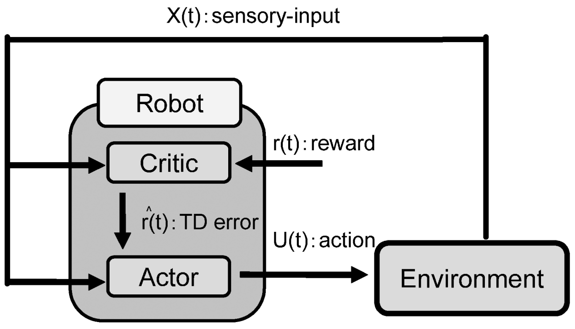


Figure 1. Actor–critic model.

To implement this algorithm, we use the backward Euler approximation of the time derivative $\frac{\mathrm{d}}{\mathrm{d}t}V(t)$:

$$\hat{r}(t) = \left(1 - \frac{\Delta t}{\tau}\right)V(t) + \frac{\Delta t}{\tau}r(t) - V(t - \Delta t). \tag{5}$$

2.2. CTRNN

The CTRNN [13] model utilized in this paper is a version of the Jordan-type RNN model [14] except for two points, i.e., the behavior of output and context neurons and the way output values are calculated. The network configuration of the CTRNN model is shown in Fig. 2, in which output and context neurons are dark gray and other neurons are light gray.

The CTRNN takes the input vector, $I(t)$, and the context vector, $C(t)$, to compute the output vector, $O(t)$, at every time step, t. The value of each hidden neuronal activation, $H_i(t)$, is computed from a given input vector and context vector in the same way as that of a feed-forward neural net:

$$H_i(t) = \sigma\left(\sum_j W_{ij}I_j(t) + \sum_k W_{ik}C_k(t) + \theta_i^H\right), \tag{6}$$

where j and k are numbers of input neurons and context neurons, respectively, and $I_n(t)$ and $C_n(t)$ are values of the n-th input and context neurons, respectively, at time step t. At the n-th neuron, the signal from the m-th presynaptic neuron, is weighted by weight W_{nm}. The conventional sigmoid function, σ, is defined as following:

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}}. \tag{7}$$

Neurons in the output and context layers used the following equations and parameters. In the following equations, u_i is the internal activation state of each i-th neuron and θ_i is a bias term. τ is a time constant of a neuron. It affects the response rate of neuronal activation, which is calculated by the hidden layer neuronal activation. If the parameter τ is large, the internal value of the neuron

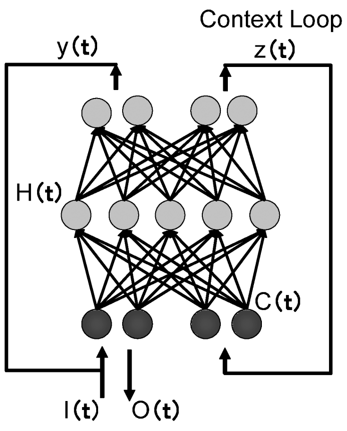


Figure 2. CTRNN model.

cannot change rapidly. In that case, the CTRNN cannot learn a rapidly change. On the contrary, the CTRNN has good performance in learning a long and gradually changing trajectory. In most cases, the values of sensory inputs and motor outputs do not change rapidly, so the CTRNN is more suitable than conventional RNN for learning such a continuous trajectory:

$$y_i(t) = \sigma\left(\sum_j W_{ij} H_j(t) + \theta_i^y\right) \tag{8}$$

$$\tau \frac{\mathrm{d}}{\mathrm{d}t} u_i^O(t) = -u_i^O(t) + \left\{y_i(t) - 0.5\right\} \times \psi_O. \tag{9}$$

The update rule of the internal activation state, $u_i^O(t)$, for each integration time step is given by discretizing (9) where the time step interval Δt is defined as 1:

$$u_i^O(t+1) = \left(1 - \frac{\Delta t}{\tau}\right) U_i^O(t) + \frac{\psi_O \Delta t}{\tau}\left(y_i(t) - 0.5\right) \tag{10}$$

$$O_i(t+1) = \sigma\left(u_i^O(t+1)\right). \tag{11}$$

The neuronal activity of context neurons is calculated in the same way as that of output neurons by substituting $y \to z$ and $O \to C$ into equations (8)–(11).

2.3. Reinforcement learning with CTRNN

We introduce the actor–critic method into a CTRNN learning algorithm. In our algorithm, an actor and a critic are implemented into one CTRNN, as shown in Fig. 3, to share the context loop which plays an important role in state recognition [9]. The weight values of context neurons are self-organized to store the contextual information through learning iterations.

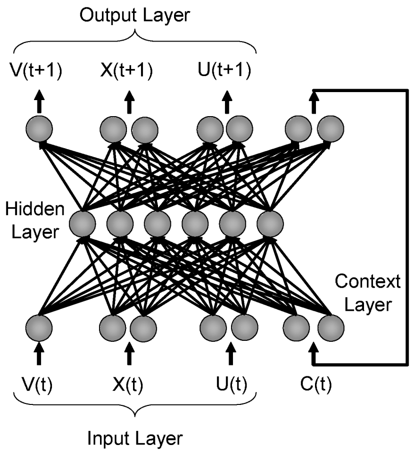


Figure 3. Actor–critic model implemented in CTRNN.

Input and output neurons represent three types of information described in Section 2.1, i.e., the value function $V(t)$, observation of the state $X(t)$ and action $U(t)$. The learning method of the CTRNN is based on the gradient descent optimization of connection weight values and biases of each neuron to minimize the learning error of a given set of teaching data. The Δ errors of the connection weight and biases are computed by using the conventional back-propagation through time (BPTT) algorithm [15]. In our method, the teaching data and propagating error are modulated by the temporal difference error to introduce reinforcement learning.

First, we consider the learning of the value function. The temporal difference error indicates the inconsistency between the ideal and predicted value functions; therefore, the teaching data for the neuron representing the value function $V(t)$ is updated using:

$$V^*(t) = V(t) + \hat{r}(t), \tag{12}$$

where $V^*(t)$ is the teaching data and $V(t)$ is the value predicted by the current CTRNN.

Next, we consider a way for improving the action using its associated value function $V(t)$. It is shown that, in the actor–critic method, the temporal difference error can be used as the reinforcement signal for the improvement of action [7]. A random noise signal is added to the output of the controller. Under such conditions, if the temporal difference error is positive, then the output was shifted to a good direction. From this cause, the output has to be learned substantially where the temporal difference error is positive. The error signal corresponding to the output neuron for action is modulated according to:

$$e(t) = \left\{\hat{U}(t) - U_{\mathrm{real}}(t)\right\} \times \sigma(\hat{r}(t) \times \phi), \tag{13}$$

where $\hat{U}(t)$ is the action calculated by the CTRNN and $U_{\mathrm{real}}(t)$ is the actual action including a noise signal. The sigmoid function $\sigma(x)$ is used to ensure that the magnification range is limited between 0 and 1.

2.4. Memory architecture for incremental learning

It is generally observed that if the RNN attempts to learn a new sequence, the content of the current memory is severely damaged. One way to avoid this problem is to save all the past teaching data in a database, add new data and use all the data to retrain the network. The problem with this method, however, is that the learning time of the RNN is increased by the increasing amount of stored data.

Tani proposed the consolidation-learning algorithm [16] to avoid this problem. In his method, the newly obtained sequence pattern is stored in the database. Then, RNN rehearses the memory patterns and these patterns are also saved in the database. The RNN is trained using both the rehearsed sequential patterns and the current sequence of the new experience.

In reinforcement learning, CTRNN has to learn a new sequence generated in each trial, which is inherently incremental learning. A trial, however, is performed in the close-loop manner including an environment and a CTRNN in which a small noise signal is added to the action output of the CTRNN. Consequently, a new sequence partly represents the structure of the past memory in the dynamical system sense, but not completely. Therefore, we use a database that stores sequences like in consolidation-learning and the CTRNN is trained using these sequences. A sequence, stored in the database, is selected by the total reward of the trial. The maximal total reward in passed trials is stored and the total reward of a new trial is compared to it. If the total reward of a new trial is greater than a certain rate of a passed maximal one, then the new sequence is stored in the database.

3. EXPERIMENT

To test the effectiveness of the proposed method, we applied it to the task of swinging-up a pendulum with limited torque (Fig. 4) [7, 12]. The experiment was carried out using a physical simulation.

In this task, a robot has to bring a pendulum to an inherently unstable upright equilibrium by controlling the torque $U(t)$ around the rotational axis. Furthermore, the maximal torque is smaller than mgl, so the robot has to swing the pendulum several times to build up sufficient momentum to bring it upright. The state of the pendulum is defined by the joint angle $\theta(t)$ and the pendulum's rotational speed $\dot{\theta}(t)$. The equation describing the physics of the pendulum is:

$$I\ddot{\theta}(t) = -\frac{1}{2}mgl\sin(\theta(t)) - \gamma\theta\dot{(t)} + U(t), \tag{14}$$

where $U(t)$ is the torque controlled by the CTRNN. Parameters, used in the following result, were $m = 0.5$ kg, $l = 1.4$ m, $g = 9.8$ m/s^2, $\gamma = 0.1$ and I is

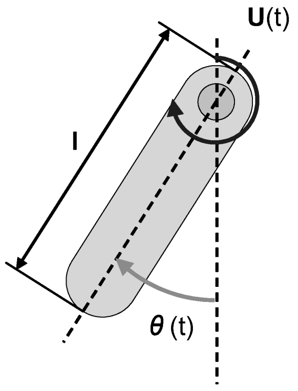


Figure 4. Pendulum with limited torque.

the inertial moment of the pendulum. Driving torque $U(t)$ is constrained within $[-2.0, 2.0]$, which is smaller than mgl, so the pendulum has to be swung back and forth at the bottom to build up sufficient momentum for a successful swing up. Equation (14), calculated in a physical simulation, is discretized using the time step of 0.00005 s. A CTRNN can change $U(t)$ and observe the state $X(t)$ in every 1000 time steps or 50 ms. Here, a partially observable environment is considered, in which the robot cannot observe the state information corresponding to the rotational speed $\dot{\theta}(t)$. The robot has to learn to approximate this information using its internal state to solve the task.

A CTRNN model with 10 hidden neurons and three context neurons is used. The time constant τ is defined as 5. There is one observation unit for the joint angle $\theta(t)$, which is normalized to the range [0, 1]. Furthermore, as in the previous section, there are additional output and input neurons that code for the action output and value function. Consequently, there are three neurons in input and output layer, respectively.

The reward signal is given by the height of the tip of the pendulum, i.e. $r(t) = \{(1 - \cos(\theta(t))/2\}^2$. Each trial is started from an initial state $\theta(0) = 0$, $\dot{\theta}(0) = 0$. Trials lasted for 120 000 time steps unless the pendulum was over-rotated ($|\theta(t)| > 2\pi$). Upon such a failure, the trial was terminated with a reward $r(t) = -1$ for the last five time steps.

A trial in which the total reward is greater than 80% of the maximal value of passed trials is stored in the database and used in the learning phase, as described in Section 2.4.

4. RESULT AND ANALYSIS OF A TRAINING RUN

The time-course of the total reward through learning is shown in Fig. 5. In this section, only a trial which is used in the learning phase is counted and others are discarded. The initial performance is about 2.5, but quickly increased to above 20

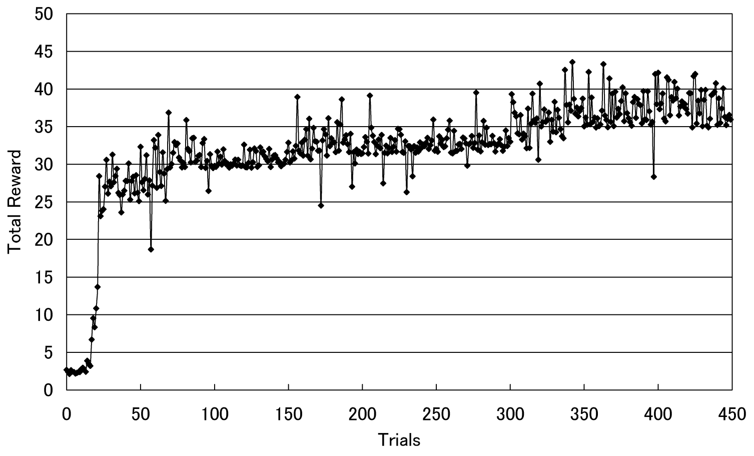


Figure 5. Learning curve.

within the first 30 trials. The performance of the CTRNN increase gradually when there are more learning iterations and the performance of the CTRNN reached its peak in trial 342. The trajectory of the pendulum in some trials is shown in Fig. 6.

The trajectory starts from the center of the phase space ($\theta = 0, \dot{\theta} = 0$), which corresponds to the pendulum hanging down-ward. A dot is painted according to the value function, which is indicated on the right color map, predicted by the CTRNN.

Recall that this task consists of two parts. The first part is swinging the pendulum several times to build up sufficient momentum to bring it upright and the second step is maintaining the pendulum in an upright position. As can be seen in Fig. 6a, in the initial stage, the CTRNN cannot generate appropriate torque sequence to swing the pendulum, so that the pendulum remains in a lower position and the value function predicted by CTRNN is almost flat. The CTRNN goes through several trials and learns to swing the pendulum. The pendulum comes to an upright position, but the CTRNN cannot maintain this state, as shown in Fig. 6b. In Fig. 6b, the value function takes a maximal value before the pendulum arrives at the upright position; therefore, the prediction of the value function is reasonable because the goal of the critic is to provide a prediction, for each time t in the trial, of the total future reward that will be gained in the trial from time t to the end of the trial. On the other hand, the actor part of the CTRNN cannot generate the appropriate torque sequence so the pendulum is only swung. Therefore, the action policy is immature.

In trial 260 (Fig. 6c), the swing-up trajectory is not much different from that of a successful trial, but the CTRNN cannot maintain the pendulum in the upright position. Learning the control law of this 1-d.o.f. system is difficult near the upright equilibrium because the pendulum is quite unstable and sensitive to torque, so the CTRNN needs many trials to learn the control law.

Finally, in trial 342, the CTRNN can swing-up the pendulum and maintain the pendulum in the upright position successfully. The trajectory of this trial is shown in Fig. 6d and the time course of the system dynamics is shown in Fig. 7. In this

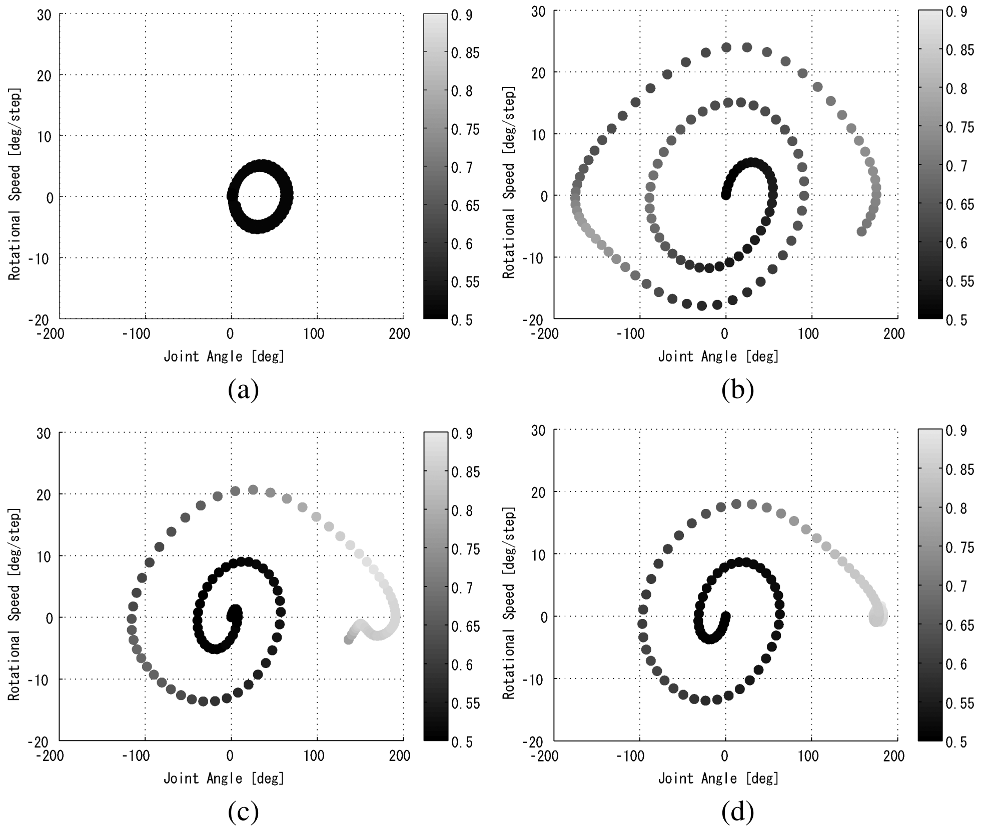


Figure 6. Trajectory of pendulum and predicted value function. (a) Trial 5. (b) Trial 30. (c) Trial 260. (d) Trial 342.

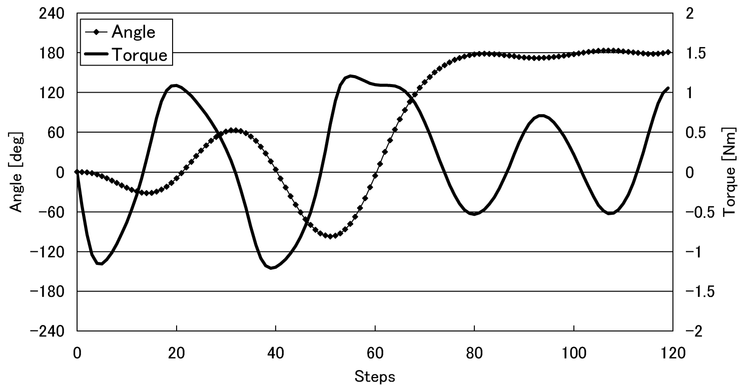


Figure 7. Time course of system dynamics in trial 342.

trial, the maximal value of the predicted value function is 0.89 in the state where $\theta = 180$ and $\dot{\theta} = 0$. In trial 30, the pendulum approaches the near state, but the predicted value is 0.74. This is attributed to the definition of the value function.

The value function is defined as the expected total reward for a certain period in the future with the current actor. Therefore, the actor learns to improve the policy, so the value function has to be changed simultaneously.

The neuronal activations of context neurons are shown in Figs 8–11. The initial

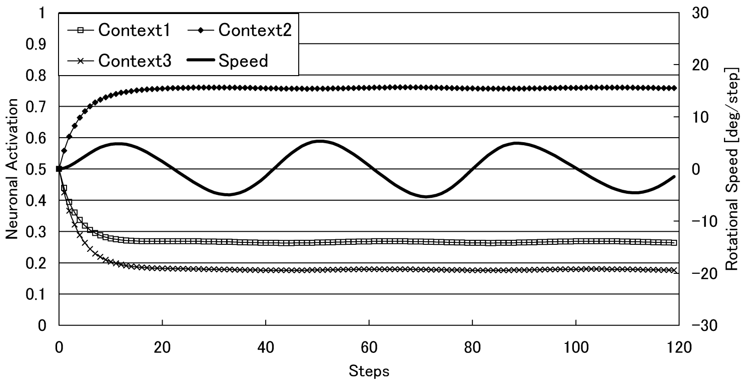


Figure 8. Time-course of context activation and rotational speed in trial 5.

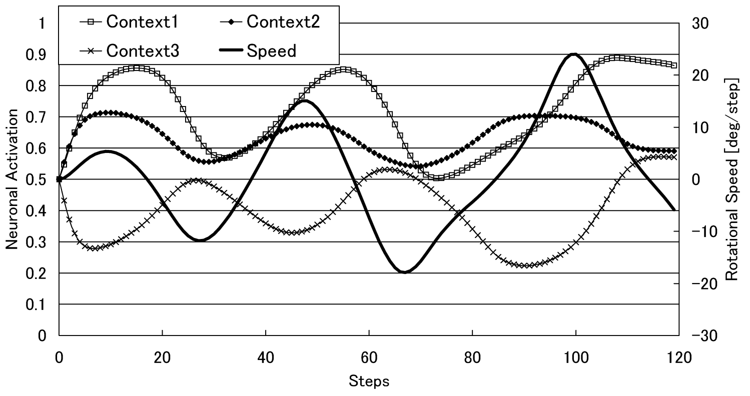


Figure 9. Time-course of context activation and rotational speed in trial 30.

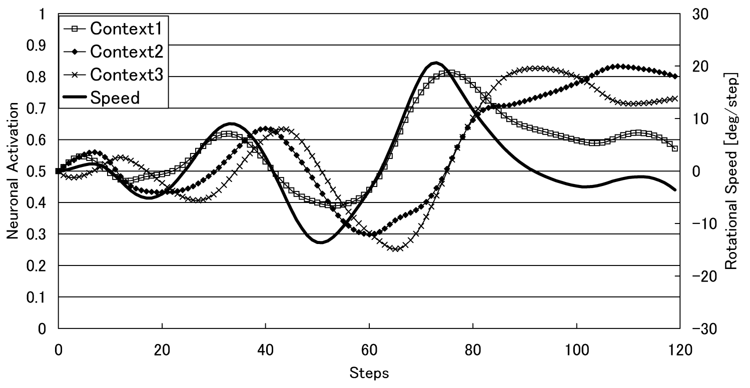


Figure 10. Time-course of context activation and rotational speed in trial 260.

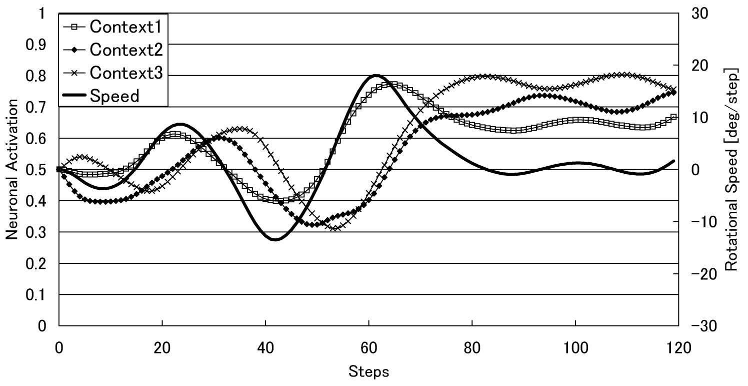


Figure 11. Time course of context activation and rotational speed in trial 342.

value of all context neuronal activation is set to 0.5 in each trial. It is notable that weights and biases corresponding to context neurons are randomly initialized and self-organized during the learning phase. These parameters determine neuronal activation of context neurons, so neuronal activation of context neurons remains flat in the early phase of learning, as shown in Fig. 8. Then, after going through several learning iterations, the neuronal activation of the first context neuron gradually comes to represent the rotational speed of the pendulum accurately (Figs 9–11). It is shown that the information about the rotational speed of the pendulum, which is considered as a hidden state, is estimated and encoded on the activation of a context neuron.

5. CONCLUSIONS

In this paper, we proposed a reinforcement learning method with CTRNN that is applicable to continuous motor command learning with the perceptual aliasing problem. We tested the effectiveness of this method in a nonlinear and partially observable task. In simulations of the pendulum swing-up task, the rotational speed, which was not directly observable but needed for solving the task, was estimated. The learning algorithm proposed in this paper allowed the CTRNN to organize its internal state space successfully and to use that information.

The pole-balancing or inverted pendulum task is a classical control problem defined in a continuous space and time domain which has been well studied [4, 7, 8, 17]. However, these prior studies did not deal with the perceptual aliasing problem and continuous space and time problem simultaneously. In Doya's study, the problem with continuous state and action was considered, but all state variables of a system were completely observable [7]. Lin and Mitchell used an Elman-type RNN, which approximates the value function. They also applied their algorithm to a pole-balancing problem, but their studies employed a discrete action formulation [8]. It is difficult to apply their algorithm to continuous motor command learning because of the limitation of the conventional RNN in learning long-step sequences. When the action space is continuous, the trajectories that a RNN has to learn tend to become long-step sequences numerically, as in our experimental cases. A conventional RNN, which is used in their study, might be inadequate to learn such long-step sequences using gradient descent learning algorithms as have been shown in Ref. [18]. For this reason, we investigate the ability of a CTRNN to learn such long-step sequences and used it in our proposed method.

Two further studies should be carried out. The first is a study of the problem with the sparseness and delay of the reward signal. In our current study, the reward is provided in each time step. However, there are many cases in which the reward signal is not given constantly, but only when a robot arrives at a goal.

The second is a more detailed analysis of the characteristics of the CTRNN utilized in this paper and a development of the learning algorithm in which time constant parameter τ is self-determined. In our current method, the time constant parameter τ is manually defined. The learning performance is dependent on whether τ is a pertinent value for a trajectory used in training a CTRNN. Therefore, if the parameter were self-determined, the proposed method could be more adaptive.

Acknowledgments

This research was partially supported by the Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research on Priorirty Areas (454, 2005-2010).

REFERENCES

1. X. Huang and J. Weng, Novelty and reinforcement learning in the value system of developmental robots, in: *Proc. 2nd Int. Workshop on Epigenetic Robotics* (2002).
2. M. Asada, E. Uchibe and K. Hosoda, Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development, *Artif. Intell.* **110**, 275–292 (1999).
3. L.-J. Lin, Self-improving reactive agents based on reinforcement learning, *Machine Learn.* **8**, 293–321 (1992).
4. A. G. Barto, R. S. Sutton and C. W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, *IEEE Trans. Syst. Man Cybernet.* **13**, 834–846 (1983).
5. R. Bianco and S. Nolfi, Evolving the neural controller for a robotic arm able to grasp objects on the basis of tactile sensors, *Adapt. Behav.* **12**, 37–45 (2004).
6. V. Gullapalli, A stochastic reinforcement learning algorithm for learning real-valued functions, *Neural Networks* **3**, 671–692 (1990).
7. K. Doya, Reinforcement learning in continuous time and space. *Neural Comput.* **12**, 219–245 (2000).
8. L.-J. Lin and T. M. Mitchell, Reinforcement learning with hidden state, in: *Proc. 2nd Int. Conf. on Simulation of Adaptive Behavior* (1993).
9. J. Tani, Model-based learning for mobile robot navigation from the dynamical system perspective, *IEEE Trans. Syst. Man Cybernet. B* **26**, 421–436 (1996).

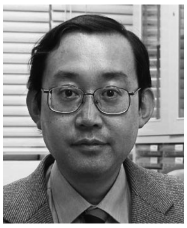
10. A. Kachites McCallum, Reinforcement learning with selective perception and hidden state, *PhD Thesis*, Univertsity of Rochester (1995).
11. R. S. Sutton, Learning to predict by the methods of temporal difference, *Machine Learn.* **3**, 9–44 (1988).
12. K. Doya, *Temporal Difference Learning in Continuous Time and Space*, volume 8 of *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA (1996).
13. R. J. Williams and D. Zipser, A learning algorithm for continually running fully recurrent neural networks, *Neural Comput.* **1**, 270–280 (1989).
14. M. I. Jordan and D. E. Rumelhart, Forward models: supervised learning with a distal teacher, *Cognitive Sci.* **16**, 307–354 (1992).
15. D. Rumelhart, G. Hinton and R. Williams, *Learning Internal Representations by Error Propagation*, volume 1 of *Parallel Distributed Processing*. MIT Press, Cambridge, MA (1986).
16. J. Tani, An interpretation of the "self" from the dynamical system perspective: a constructivist approach, *Consciousness Studies* **5** (1998).
17. C. W. Anderson, Strategy learning with multilayer connectionist representations, in: *Proc. 4th Int. Workshop on Machine Learning*, pp. 103–114 (1987).
18. Y. Bengio, P. Simard and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. Neural Networks* **5**, 157–166 (1994).

ABOUT THE AUTHORS

Hiroaki Arie received the BS and MS degrees from Waseda University, Tokyo, Japan, in 2003 and 2005, respectively. His interests include intelligence model for self-developing autonomous robots and embodied artificial intelligence. He is currently a PhD student at Waseda University.

Tetsuya Ogata received the BS, MS and DE degrees in Mechanical Engineering in 1993, 1995, and 2000, respectively, from Waseda University. From 1999 to 2001, he was a Research Associate in Waseda University. From 2001 to 2003, he was a Research Scientist in the Brain Science Institute, RIKEN. Since 2003, he has been a Faculty Member in the Graduate School of Informatics, Kyoto University, where he is currently an Associate Professor. Since 2005, he has been a Visiting Associate Professor of the Humanoid Robotics Institute of Waseda University. His research interests include human–robot vocal-sound interaction, dynamics of human–robot mutual adaptation and active sensing with robot systems. He received the JSME Outstanding Paper Medal from the Japan Society of Mechanical Engineers in 2000.

Jun Tani received the BS degree in Mechanical Engineering from Waseda University, dual MS degree in Electrical Engineering and Mechanical Engineering from the University of Michigan, and DE degree from Sophia University. He started his research career in Sony Computer Science Laboratory in 1990. He has been appointed as a Team Leader in the Laboratory for Behavior and Dynamic Cognition, Brain Science Institute, RIKEN in Tokyo since 2000 where he is currently leading seven Posdocs. He was also appointed as a Visiting Associate Professor in the University of Tokyo from 1997 to 2002. He has studied the problems of robot learning with theoretical modeling of complex adaptive systems and neuronal networks for more than 15 years. He has been also interested in phenomenological problems of 'self-consciousness' and his studies have addressed those problems from the view of embodied cognition. Five years ago, he started neuroscience studies on behavior learning processes in real brains utilizing both schemes of human brain imaging and animal electrophysiology. His vision is to establish 'brain-inspired robotics' by integrating these approaches.



Shigeki Sugano received the BS, MS and DE degrees in Mechanical Engineering in 1981, 1983, and 1989, respectively, from Waseda University. From 1987 to 1991, he was a Research Associate in Waseda University. Since 1991, he has been a Faculty Member in the Department of Mechanical Engineering, Waseda University, where he is currently a Professor. From 1993 to 1994, he was a Visiting Scholar in the Mechanical Engineering Department, Stanford University. Since 2001, he has been a Director of the Waseda WABOT-HOUSE Laboratory. He is a Member of the Humanoid Robotics Institute of Waseda University. Since 2001, he has been the President of the Japan Association for Automation Advancement. His research interests include anthropomorphic robots, dexterous manipulators and human–robot communication. He received the Technical Innovation Award from the Robotics Society Japan for the development of the Waseda Piano-Playing Robot: WABOT-2 in 1991. He received the JSME Outstanding Paper Medal from the Japan Society of Mechanical Engineers in 2000. He received the JSME Fellow Award in 2006. He received the IEEE Fellow Award in 2007. He serves as the Secretary of the IEEE Robotics and Automation Society (RAS). He serves as a Co-Chair of the IEEE RAS Technical Committee on Humanoid Robotics. He served as the IEEE RAS Conference Board, Meetings Chair from 1997 to 2005. He is an Associate Editor of *Advanced Robotics*. He served as the General Chair of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003) and he served as the General Co-Chair of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006).