Reinforcement Learning for a CPG-driven Biped Robot

Takeshi Mori[1], **Yutaka Nakamura**[3,1], **Masa-aki Sato**[2,3], **Shin Ishii**[1,3]
[1]Nara Institute of Science and Technology
Takayama 8916-5, Ikoma, Nara 630-0192
{tak-mori, yutak-na, ishii}@is.naist.jp
[2]ATR Computational Neuroscience Laboratories
Hikaridai 2-2-2, Seika, Soraku, Kyoto 619-0288
masa-aki@atr.jp
[3]CREST, JST

Abstract

Animal's rhythmic movements such as locomotion are considered to be controlled by neural circuits called central pattern generators (CPGs). This article presents a reinforcement learning (RL) method for a CPG controller, which is inspired by the control mechanism of animals. Because the CPG controller is an instance of recurrent neural networks, a naive application of RL involves difficulties. In addition, since state and action spaces of controlled systems are very large in real problems such as robot control, the learning of the value function is also difficult. In this study, we propose a learning scheme for a CPG controller called a CPG-actor-critic model, whose learning algorithm is based on a policy gradient method. We apply our RL method to autonomous acquisition of biped locomotion by a biped robot simulator. Computer simulations show our method is able to train a CPG controller such that the learning process is stable.

Introduction

There have been many studies of locomotion robots (Hirai *et al.* 1998) (Buehler *et al.* 1998), but few of them achieved dynamic walking on irregular terrain. Although these studies employed precise models of environments, it is necessary for robots to be adapted to unknown environments when applied to real world problems (Morimoto & Doya 2001).

On the other hand, animal's movement show rapid adaptability to environmental changes and robustness to disturbances. Such a mechanism has been studied both in biological science and in engineering. Existing neurobiological studies have revealed that rhythmic motor patterns are controlled by neural oscillators referred as central pattern generators (CPGs) (Grillner *et al.* 1991), and this CPG mechanism is good for both adaptability and stability of animals.

Recently, there are many studies on locomotion or swimming robots controlled by CPG controllers, motivated by the animal's locomotion mechanism (human (Taga, Yamaguchi, & Shimizu 1991), mammal (Fukuoka, Kimura, & Cohen 2003), hexapod (Barnes 1998), salamander (Ijspeert 2001), or lamprey (Ekeberg 1993)). In (Fukuoka, Kimura, & Cohen 2003), robots realized flexible and robust locomotion even in unknown environments.

A CPG controller is composed of a kind of recurrent neural network (RNN), called the neural oscillator network. This controller receives a sensory feedback signal from a physical system (controlled system) and outputs a control torque to a physical system. Although there are connection weight parameters in the neural oscillator network, there is no design principle to determine their values. This problem can be regarded as the exploration of the optimal solution over the space of weight parameters, hence various learning methods will be helpful for the exploration. Actually, there exist studies in which genetic algorithm (GA) is employed to determine the weight values (Ogihara & Yamazaki 2001) (Ijspeert 2001). However, GA is not a learning scheme by individual robots but a scheme over generations of robot ensembles. Then, we propose in this article a reinforcement learning (RL) method to determine autonomously the weight values. In contrast to GA, RL is the learning framework based on individual trial and error, and has some analogy to the developmental process of animal's motor controls. As an example, an RL for a biped robot is analogous to a baby's acquisition of biped locomotion along its growth. The relationship of RL with the brain's motor learning has also been suggested (Fiorillo, Tobler, & Schultz 2003).

RL methods have been successfully applied to various Markov decision processes (MDPs) (Sutton & Barto 1998). They can be straightforwardly applied to controllers without internal states, a controller composed by a feedforward neural network for example. Because the 'policy' in such a case can be defined by a mapping from a state of the target system to a control signal. However, such a straightforward approach needs to acquire a high-dimensional policy when applied to real world problems like the control problem for a biped robot. The approximation problem for a policy or a value function then becomes high-dimensional. This makes the exploration problem of the optimal policy very difficult. Furthermore, the training of nonlinear function approximators in a high-dimensional space is also difficult, which will be problematic in the value learning. On the other hand, a CPG controller is beneficial for the robustness of a biped robot to various environmental changes, because the robot can be entrained to the rhythm inherent in the CPG. In addition, a control signal produced by a CPG controller is effectively restricted within the space determined by the inherent rhythmic patterns of the CPG controller. Then, the searching for the optimal policy becomes much easier than that without any restriction. The output patterns of a CPG controller can be changed by mainly tuning the connection weights in the neural oscillator network. The number of these parameters is often much smaller than that of the high-dimensional policy for the original control problem.

When we consider a naive application of RL to a CPG controller, however, the policy becomes unstationary, because a control signal depends not only on a state of the target system but also on an internal state of neurons constituting the CPG controller. Furthermore, the large amount of computation is usually required to train an RNN. In order to overcome these problems, we propose a new RL method called the CPG-actor-critic model. In this model, the variable part in the CPG controller is represented as a simple feedforward neural network, then it is not required to train the RNN as itself.

An RL is primarily formulated as to find the policy that minimizes the cost on an Markov decision process (MDP) (Sutton & Barto 1998). In conventional value-based RL methods, such as the Q learning, the value function which represents cumulative cost toward the future for a state (and an action) is obtained. Then the policy is updated to minimize the value function over every state. In order to find the optimal policy, it is necessary to know the value function. In many real problems such as robot control, the computation of the correct value function or its function approximation (Sato & Ishii 1999) is difficult, analytically or numerically, because of the enormous size of the state and action spaces. Furthermore, it has been supposed that the convergence of these value-based RL methods is not guaranteed when employing function approximators, due to the effect of approximation errors (Bertsekas & Tsitsiklis 1996).

In our method, the learning scheme is based on a policy gradient method (Konda 2002). Policy gradient RL methods are not primarily based on value functions. In these methods, the policy parameter is updated based on the gradient of average cost with respect to the policy parameter, and it is possible to learn without value functions (Williams 1992). However, learning methods without value function take much computation time, then they had been less attractive. Recently, a new actor-critic method, i.e., a policy gradient method incorporating a value function, has been proposed, and its convergence was proved (Konda 2002). In this actor-critic method, a lower-dimensional projection of the value function is approximated instead of the true value function, because the policy gradient depends only on the projection. The approximation of the projection is often easier than that of the original one.

The training of our CPG-actor-critic model is based on this actor-critic method. Weights of connections within the CPG controller are optimized so as to control the biped robot simulator proposed by (Taga, Yamaguchi, & Shimizu 1991). Simulation results show that stable locomotion can be achieved by the CPG controller trained by our RL approach.

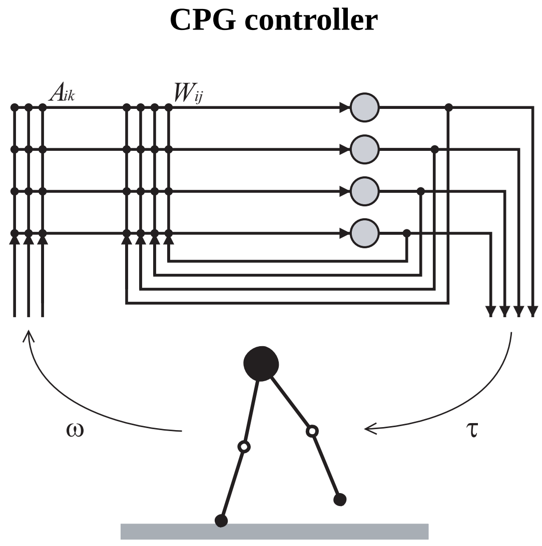


Figure 1: Control scheme using a CPG controller

The motion of a physical system like a biped robot is expressed as

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x}, \boldsymbol{\tau}), \tag{1}$$

where $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ denote the physical state and its time derivative, respectively. $\boldsymbol{\tau}$ denotes the control signal (torque) from the controller. $F(\boldsymbol{x}, \boldsymbol{\tau})$ represents the vector field of the dynamics.

The physical system is controlled by a CPG controller as depicted in Fig.1. The CPG controller is implemented as a neural oscillator network, and outputs a control signal $\boldsymbol{\tau}$ corresponding to neurons' states in the network. The CPG controller receives a sensory feedback signal $\boldsymbol{\omega}$ from the physical system.

The neural oscillator network is an instance of RNNs, and the dynamics of the i-th neuron is given by

$$\zeta_i \dot{\nu}_i = -\nu_i + I_i, \; y_i = G_i(\nu_i), \tag{2}$$

where ν_i, y_i, I_i and ζ_i denote the state, output, input, and time constant, respectively, of the i-th neuron. Output y_i is calculated from state ν_i through the transfer function G_i. Actual function form for G_i is described later.

Input I_i is given by

$$I_i = \sum_j W_{ij} y_j + I_i^{ext} + B_i, \tag{3}$$

where the first term is the feedback input from the other neurons, the second term is the external input denoting the sensory feedback signal, and the third term is a bias. W_{ij} represents the connection weight from the j-th neuron to the i-th neuron. The external input I_i^{ext} is defined by

$$I_i^{ext} = \sum_k A_{ik} \omega_k, \tag{4}$$

namely, a sum of the sensory feedback signal $\boldsymbol{\omega}$ weighted by connection weight $\boldsymbol{A}_i$.

The control signal to the physical system, $\boldsymbol{\tau}$, is given by a weighted sum of the outputs of the CPG neurons:

$$\tau_n = \sum_i T_{ni} y_i, \tag{5}$$

where T_{ni} represents the weight.

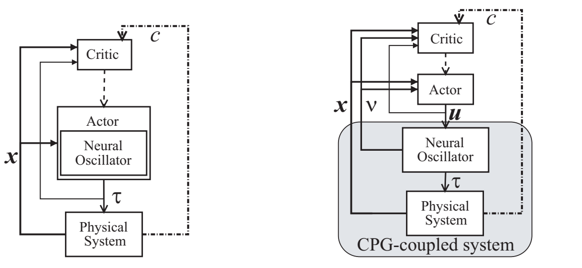
CPG-actor-critic model

An actor-critic method is one of the popular RL methods (Sutton & Barto 1998). In this method, the actor is a controller that transforms an observation of the target system into a control signal. The critic predicts the cumulative or average cost toward the future when the current actor is used as a controller. The actor's parameter is updated so that the cost predicted by the critic becomes small.

When we try to apply the actor-critic method to the CPG controller, several difficulties arise. A naive application of the actor-critic method (Fig.2(a)) is not suited for training RNNs, because the critic's and actor's learning is usually based on temporally instantaneous errors called temporal difference (TD) errors. Training of an RNN needs the "error-back-propagation through time" (Sato 1990), which is not suited for on-line learning and needs heavy computation. Moreover, because a control signal $\boldsymbol{\tau}$ is generated by the CPG controller which has the internal dynamics given by equation (2), the total system coupled with the CPG controller has an dynamics:

$$(\dot{\boldsymbol{x}}, \dot{\boldsymbol{\nu}}) = F_{CPG-coupled-system}(\boldsymbol{x}, \boldsymbol{\nu}, \boldsymbol{\tau}, \boldsymbol{I}). \tag{6}$$

This equation shows the existence of a hidden state variable $\boldsymbol{\nu}$. This makes the policy, which is a function from a state to an action, not unique for any state; namely, the policy becomes unstationary. Since the policy gradient method we employ is formulated under the condition that the policy is stationary, its application to the learning task of an unstationary policy suffers from a remained variance. To formulate the learning scheme of the CPG controller as an MDP, it is profitable that the physical system and the neural oscillator network are regarded as a unified system, a CPG-coupled system (Fig.2(b)).



(a) Actor-critic model (b) CPG-actor-critic model

Figure 2: CPG-actor-critic model

In our method, the CPG controller is divided into two modules: the basic CPG and the actor, as depicted in Fig.3. The basic CPG is a neural oscillator network whose connection weight is $\boldsymbol{W}^{fix}$. The actor receives an input signal, which is a pair of an output of the basic CPG and a sensory feedback signal, and outputs a control signal called an indirect control signal to the basic CPG. Corresponding to this separation, the input to the CPG neuron, I_i (equation (3)), is divided into two parts:

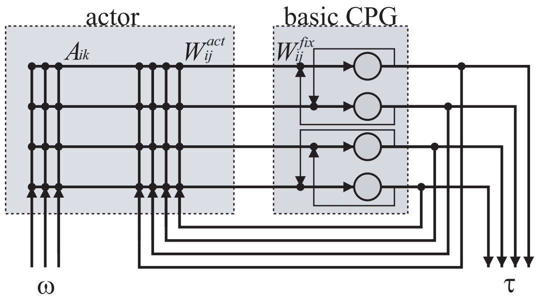


Figure 3: Actor and basic CPG

$$\begin{aligned} I_i &= I_i^{fix} + u_i & (7)\\ I_i^{fix} &= \sum_j W_{ij}^{fix} y_j + B_i & (8)\\ u_i &= \sum_j W_{ij}^{act} y_j + \sum_k A_{ik}\omega_k, & (9)\end{aligned}$$

where I_i^{fix} represents a feedback input from other neurons through the fixed mutual connection $\boldsymbol{W}^{fix}$ and the fixed bias input $\boldsymbol{B}$.

$\boldsymbol{u}$ is an indirect control signal and the output of the actor. The actor is a linear controller, and receives an output $\boldsymbol{y}$ of the basic CPG and a sensory feedback signal $\boldsymbol{\omega}$. The weight parameters W_{ij}^{act} and A_{ik} are adjustable, and are trained by RL. The control torque to the physical system is calculated by equation (5), and the weight parameter $\boldsymbol{T}$ is fixed. We call the architecture above the CPG-actor-critic model.

The CPG-actor-critic model has two aspects. From the control viewpoint, the CPG controller consists of the basic CPG and the actor (Fig.3), which cooperatively controls the physical system. From the RL viewpoint, the actor outputs an indirect control signal $\boldsymbol{u}$ to a CPG-coupled system which consists of the basic CPG and the physical system (Fig.2(b)). In the latter view, the actor is a linear controller without any mutual feedback connections, hence there is no need to train the RNN as itself. Another merit exists in this architecture, namely, the actor is trained as to modify the entrainment between the physical system and the basic CPG.

The critic observes the CPG-coupled system state, i.e., a basic CPG state $\boldsymbol{\nu}$ and a physical system state $\boldsymbol{x}$, and predicts the cumulative or average cost toward the future.

Learning Algorithm

The CPG-actor-critic model is trained according to the actor-critic method based on the policy gradient method (Konda 2002). In the conventional value-based actor-critic method, the critic approximates the value function on the state and action spaces, and the actor parameter is improved to minimize the cumulative cost in each state according to the evaluation by the critic. In order to acquire the optimal actor parameter, therefore, the critic is required to approximate the value function with a high accuracy. In this case, the computation amount necessary for training the critic will increase as the state and action spaces become large, and inaccurate critic badly affects the actor's learning and makes the learning process unstable. In the policy gradient method, on the other hand, the actor is trained based on the gradient (policy gradient) of the average cost with respect to the policy parameter under a stationary condition, and the critic is used for efficiently estimating the gradient through the approximation of the value function. Roughly speaking, the policy is updated as follows; when the state transition from $\boldsymbol{s}$ to $\boldsymbol{s}'$ occurs by an action $\boldsymbol{u}$, the action $\boldsymbol{u}$ is positively (or negatively) reinforced, if the evaluation of this transition is larger (or smaller) than the expected one. The policy gradient indicates the direction of the policy parameter to realize such a reinforcement manipulation of actions.

At time t, the actor receives a basic CPG's output $\boldsymbol{y}_t$ and a sensory feedback signal $\boldsymbol{\omega}_t$ which is a transformation of $\boldsymbol{x}_t$, and outputs an indirect control signal $\boldsymbol{u}_t$ calculated by equation (9). The CPG-coupled system receives the actor output $\boldsymbol{u}_t$, and changes its state $(\boldsymbol{\nu}_t, \boldsymbol{x}_t)$ to $(\boldsymbol{\nu}_{t+1}, \boldsymbol{x}_{t+1})$ according to the dynamics of the basic CPG and the physical system, (1)-(8). After that, it is assumed that the critic receives an immediate cost $c(\boldsymbol{\nu}_t, \boldsymbol{x}_t, \boldsymbol{u}_t)$.

Subsequently, a state of the CPG-coupled system, $(\boldsymbol{x}, \boldsymbol{\nu})$, is simply denoted by $\boldsymbol{s}$. We formulate an MDP on the state space $\mathbb{S}$ and action space $\mathbb{U}$ of the CPG-coupled system, assuming the cost function is given by a mapping $c : \mathbb{S} \times \mathbb{U} \to \mathbb{R}$. $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{u})$ represents the probability that the system changes its current state $\boldsymbol{s} \in \mathbb{S}$ to a next state $\boldsymbol{s}' \in \mathbb{S}$ when an action $\boldsymbol{u} \in \mathbb{U}$ is given.

A policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{u}|\boldsymbol{s})$ defines the probability of an action $\boldsymbol{u}$ at a state $\boldsymbol{s}$, where $\boldsymbol{\theta} \equiv \{\theta_i | i = 1, \cdots, M\}$ is a parameter vector. It is assumed that a stationary distribution $\eta_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u})$ with respect to the state-action pair $(\boldsymbol{s}, \boldsymbol{u})$ exists under a fixed policy $\pi_{\boldsymbol{\theta}}$. Under these conditions, the objective of the MDP here is to obtain the optimal parameter $\boldsymbol{\theta}^{opt}$ that minimizes the average cost:

$$\bar{\alpha}(\boldsymbol{\theta}) = \int_{\boldsymbol{s} \in \mathbb{S}, \boldsymbol{u} \in \mathbb{U}} d\boldsymbol{s} d\boldsymbol{u}\; c(\boldsymbol{s}, \boldsymbol{u}) \eta_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}). \tag{10}$$

The gradient of the average cost $\bar{\alpha}(\boldsymbol{\theta})$ with respect to the parameter θ_i is given (Konda 2002) (Sutton *et al.* 2000) (Marbach & Tsitsiklis 2001) by

$$\frac{\partial}{\partial \theta_i} \bar{\alpha}(\boldsymbol{\theta}) = \int_{\boldsymbol{s} \in \mathbb{S}, \boldsymbol{u} \in \mathbb{U}} d\boldsymbol{s} d\boldsymbol{u}\; \eta_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}) Q_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}) \psi^i_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}), \tag{11}$$

where

$$\psi^i_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}) \equiv \frac{\partial}{\partial \theta_i} \ln \pi_{\boldsymbol{\theta}}(\boldsymbol{u}|\boldsymbol{s}). \tag{12}$$

Equation (11) is called the policy gradient, and suggests that the gradient of the objective function with respect to the policy parameter θ_i is calculated by the expectation of inner product between the Q function and $\psi^i_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u})$ over the stationary distribution. Then, it is not necessary to calculate the true Q function, but its projection to the space spanned by $\{\psi^i_{\theta}(s, u) | i = 1, \cdots, M\}$ is sufficient. Although the true Q function may be a complex and high-dimensional mapping of $\mathbb{S} \times \mathbb{U} \to R$, the number of policy parameters is often smaller than the dimensionality of $\mathbb{S} \times \mathbb{U} \to R$, thus the approximation of the Q function becomes much easier (Konda 2002).

In this method, the critic approximates the projected Q function. A linear parametric model is employed as a function approximator:

$$Q^{\boldsymbol{r}}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}) = \sum_{i=1}^{M} \boldsymbol{r}_i \psi^i_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}), \tag{13}$$

where $\boldsymbol{r} \equiv \{r_i | i = 1, \cdots, M\}$ is a parameter vector of the critic. The critic is trained based on stochastic approximation with an eligibility trace, i.e., a TD(λ) learning:

$$r_i \leftarrow r_i + \gamma \delta Z_i, \quad i = 1, \cdots, M. \tag{14}$$

Here δ is a TD-error:

$$\begin{aligned} \delta \quad = \quad & c(\boldsymbol{s}_{t+1}, \boldsymbol{u}_{t+1}) + Q^{\boldsymbol{r}_{t+1}}_{\boldsymbol{\theta}_{t+1}}(\boldsymbol{s}_{t+1}, \boldsymbol{u}_{t+1}) \\ & - Q^{\boldsymbol{r}_t}_{\boldsymbol{\theta}_t}(\boldsymbol{s}_t, \boldsymbol{u}_t) - \alpha, \end{aligned} \tag{15}$$

where $\boldsymbol{Z} \equiv \{Z_i | i = 1, \cdots, M\}$ is the eligibility trace and performs responsibility assignment of the TD-error (Bradtke & Barto 1996), and α is the estimation of average cost.

Concurrently to this critic learning, actor parameter $\boldsymbol{\theta}$ is updated to minimize the objective function $\bar{\alpha}(\boldsymbol{\theta})$ according to the policy gradient. From equations (11) and (13), the following equation is derived:

$$\frac{\partial}{\partial \theta_i} \bar{\alpha}(\boldsymbol{\theta}) = \int_{\boldsymbol{s} \in \mathbb{S}, \boldsymbol{u} \in \mathbb{U}} d\boldsymbol{s} d\boldsymbol{u}\; \eta_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}) Q^{\boldsymbol{r}}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}) \psi^i_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}). \tag{16}$$

$Q^{\boldsymbol{r}}_{\boldsymbol{\theta}}$ is calculated by the current critic, and the actor is trained by the gradient method using equation (16). The expectation with respect to $\eta_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u})$ in equation (16) is approximated by using the empirical distribution $\{\boldsymbol{s}_t, \boldsymbol{u}_t\}, t = 1, 2, \cdots$. Concretely, the policy parameter θ_i is updated by

$$\theta_i \leftarrow \theta_i - \beta Q^{\boldsymbol{r}}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}) \psi^i_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{u}), \tag{17}$$

where β is the learning rate. The parameter update in the direction of the gradient of the average cost function, equation (16), is performed by repeating the individual update above. Namely, equation (17) is the stochastic gradient method for equation (16).

According to this method, the critic parameter may not converge, while the actor parameter converges because the actor's training is based on the long run average of the critic's output (Konda 2002). The behavior of the critic parameter will become more stable if the actor parameter is fixed for a certain period. Therefore, the actor's (critic's) parameter is fixed while the critic's (actor's) parameter is updated during a single learning episode in the experiments described below.

Experiment

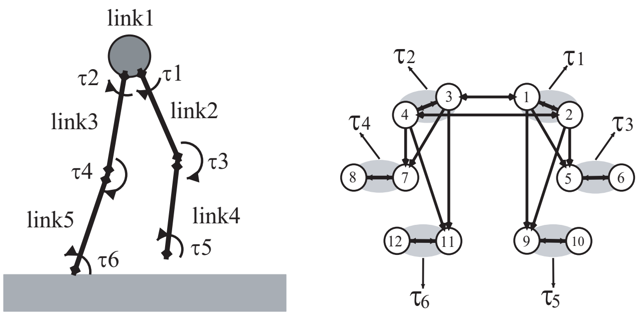
We apply our CPG-actor-critic model to a biped robot simulator (Taga, Yamaguchi, & Shimizu 1991). The objective of the simulation is to obtain a CPG controller that makes the robot simulator produce stable locomotion.

The biped robot is constructed by five links connected to each other, as depicted in Fig.4(a). The motions of these links are restricted in a sagittal plane. Link-1 is a point mass and representative of the upper body. Each leg consists of thigh (link-2,3) and shank (link-4,5), and has three joints, i.e., hip, knee and ankle. The length of thigh or shank is 0.5m or 0.6m, respectively. The robot is controlled by the torque $\tau_1 \cdots \tau_6$, each of which is applied to a single joint. When a shank is off the ground, the torque at the ankle joint is not generated. The action from the ground is modeled as a spring-damper model, and the reaction force occurs according to the displacement from the ground and the ankle velocity. (Taga, Yamaguchi, & Shimizu 1991).

A state of the biped robot is described by

$$\boldsymbol{x} = (x_1, h_1, a_2, a_3, a_4, a_5, \dot{x}_1, \dot{h}_1, \dot{a}_2, \dot{a}_3, \dot{a}_4, \dot{a}_5),$$

where x_1 and h_1 denote the horizontal and vertical coordinates of link-1, respectively, and a_i represents the angle of link-i from the vertical axis.



(a) Biped robot simulator (b) Neural oscillator network

Figure 4: Structure of CPG-driven biped robot

The structure of the neural oscillator network, which constitutes the CPG controller, is also adopted from (Taga, Yamaguchi, & Shimizu 1991), as depicted in Fig.4(b). There are 24 neurons; the i-th and the (i + 12)-th neurons are called a primary neuron and a supplementary neuron, respectively. Each supplementary neuron is solely connected to its primary neuron by excitation-inhibition mutual connections. An output of the i-th primary neuron is given by $G_i(\nu_i) = \max(0, y_i)$, and an output of the $(i + 12)$-th supplementary neuron is given by the identical function. The weight of the mutual connection $\boldsymbol{W}^{fix}$ (equation (8)) is also adopted from (Taga, Yamaguchi, & Shimizu 1991). Without any sensory feedback signal, each neuron in the basic CPG outputs rhythmic signals autonomously. A combination of two primary neurons and two supplementary neurons $(2i-1, 2i, 2i+11, 2i+12, i = 1, \cdots, 6)$ behaves as a neural oscillator, and each neural oscillator is responsible for controlling the corresponding joint.

Torque τ_i, which is applied to the i-th joint is calculated from the output of the CPG neurons:

$$\begin{aligned} \tau_i &= -T_i^F y_{2i-1} + T_i^E y_{2i} \quad (i = 1, \cdots, 4) \\ \tau_i &= (-T_i^F y_{2i-1} + T_i^E y_{2i})\Xi_{i-1} \quad (i = 5, 6), \end{aligned} \tag{18}$$

where Ξ_{i-1} represents an indicator function of shank link-i (i=4,5), i.e., $\Xi_i = 1$(or 0) when the link-i touches (or, is off) the ground. $\tau_{1,2}, \tau_{3,4}$ and $\tau_{5,6}$ represent the torques applied to hip, knee and ankle, respectively. T_i^F and T_i^E represent weights of the flexor and extensor, respectively, and their values are fixed. A sensory feedback signal from the biped robot is $\boldsymbol{\omega} = \{a_2, a_3, a_4\Xi_4, a_5, \Xi_5, \Xi_4, \Xi_5, \dot{a}_4\Xi_4, \dot{a}_5\Xi_5\}$.

Condition

The dynamics of the CPG-coupled system is calculated by the Runge-Kutta integration with time interval 0.0001 sec. The learning system observes the system state and outputs a control signal every 0.01 sec. The observation of each angular velocity is smoothed over the 0.01 sec. interval.

It is not desirable to make all parameters of the CPG controller variable, because the CPG controller uses the property that the motion of the physical system is entrained into the inherent rhythm of the CPG controller. In this experiment, we assume for simplicity that all mutual connection weights in the neural oscillator network are fixed, i.e., $W_{ij}^{act} \equiv 0$. We also assume that connection patterns from the sensory feedback signal to the CPG neurons have specific forms, as in (Taga, Yamaguchi, & Shimizu 1991):

$$\begin{aligned} I_1^{ext} &= \theta_1\omega_1 - \theta_2\omega_2 + \theta_3\omega_3 + \theta_4\omega_6, \\ I_3^{ext} &= \theta_1\omega_2 - \theta_2\omega_1 + \theta_3\omega_4 + \theta_4\omega_5, \\ I_5^{ext} &= \theta_5\omega_4, \;\; I_7^{ext} = \theta_5\omega_3, \\ I_9^{ext} &= -\theta_6\omega_3 - \theta_7\omega_4 - \theta_8\omega_7, \\ I_{11}^{ext} &= -\theta_6\omega_4 - \theta_7\omega_3 - \theta_8\omega_8, \\ I_{2i}^{ext} &= -I_{2i-1}^{ext} \;\; \text{for } i = 1, \cdots, 6, \end{aligned} \tag{19}$$

where $\{\theta_i | i = 1, \cdots, 8\}$ are elements of the connection weight $\boldsymbol{A}$ (equation (9)). The other elements of $\boldsymbol{A}$ are fixed at zero. The policy, which outputs an indirect control signal probabilistically to the CPG neurons, is given by a Gaussian distribution:

$$\mathcal{N}(\boldsymbol{\mu}, \theta_9), \tag{20}$$

where $\mu_i = I_{2i-1}^{ext} (i = 1, \cdots, 6)$ and θ_9 are the mean and variance of the distribution, respectively. Then, the policy parameter $\{\theta_i | i = 1, \cdots, 9\}$ is adjusted by our RL method.

We assume that an immediate cost $c(\boldsymbol{\nu}_t, \boldsymbol{x}_t, \boldsymbol{u}_t)$ is determined only by the robot state at the next time step, $\boldsymbol{x}_{t+1}$, and defined as $\tilde{c}_{t+1}$:

$$\begin{aligned} &\tilde{c}(\boldsymbol{x}) = 0.1c_h(\boldsymbol{x}) + c_h'(\boldsymbol{x}) + 0.0002c_s(\boldsymbol{x}) \qquad (21) \\ &c_h(\boldsymbol{x}) = -(h_1 - 0.9 - \min(h_4, h_5)) \\ &c_s(\boldsymbol{x}) = \begin{cases} -\dot{x}_1 & \text{if } |\dot{x}_1| < 1 \\ -\dot{x}_1/|\dot{x}_1| & \text{otherwise} \end{cases} \\ &c_h'(\boldsymbol{x}) = \begin{cases} (0.9 + c_h)^2 & \text{if } 0.9 + c_h > 0 \\ 0 & \text{otherwise} \end{cases}, \end{aligned}$$

where $h_i (i = 4, 5)$ represents the height of link-i. $c_h(\boldsymbol{x})$ encourages the robot not to fall down, and $c_s(\boldsymbol{x})$ encourages the robot to proceed to the forward direction. $c_h'(\boldsymbol{x})$ incurs a large penalty when the robot falls down.

The maximum period in one learning episode was 5 sec. (500 time steps). If the robot tumbled before 5 sec., the learning episode was terminated at that time. At the beginning of each learning episode, the state of the robot was initialized as a motion-less posture such that the two legs slightly opened, the states of whole neurons were initialized as $\mathbf{0}$, and the angles of the legs were selected randomly within a small range. At the beginning of the whole learning procedure, the critic parameter $\boldsymbol{r}$ was initialized simply as $\mathbf{0}$.

Result

Learning from the random initial value of the actor parameter

First, we examined whether the actor parameter that generates stable walking can be acquired by our learning scheme from a random initial value of the actor parameter with which the robot hardly walks.

Learning from the actor parameter which hardly generates a walk At the beginning of the learning procedure, the actor parameter $\boldsymbol{\theta}$ was randomly initialized around $\mathbf{0}$.

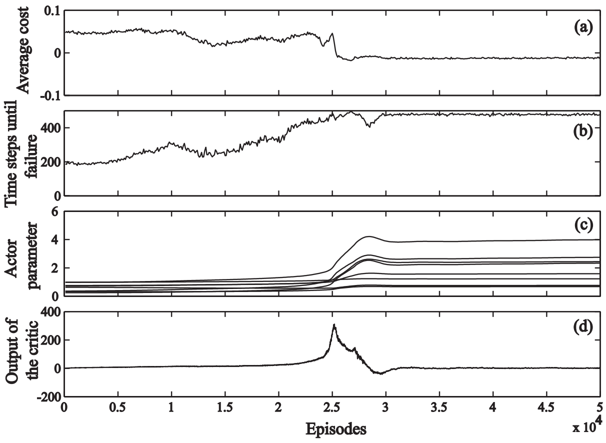


Figure 5: Learning curve

Fig.5 shows the learning curve until 50,000 learning episodes, where the horizontal axis denotes the number of learning episodes. Fig.5(a) shows the average cost in one time step. The average cost decreased almost monotonously, implying that the learning successfully proceeded. After about 25,000 learning episodes, a large drop off of the average cost was observed, suggesting a significant progress of the learning occurred. Fig.5(b) shows the number of time steps until failure, averaged over every 500 learning episodes. After about 30,000 learning episodes, a good CPG controller was acquired such that the robot seldom fell down. Fig.5(c) shows the actor parameter. After about 25,000 learning episodes, the parameter significantly changed, while after 30,000 learning episodes, it almost converged. Fig.5(d) shows the average output of the critic. Large values at around 25,000 learning episodes reflected the large drop off of the cost. When the critic's output was large, it caused large change of the actor parameter (equation (17)).

According to our cost design (equation (21)), a large penalty c'_h is incurred when the robot falls down. Therefore, there is large difference in the cost per step between

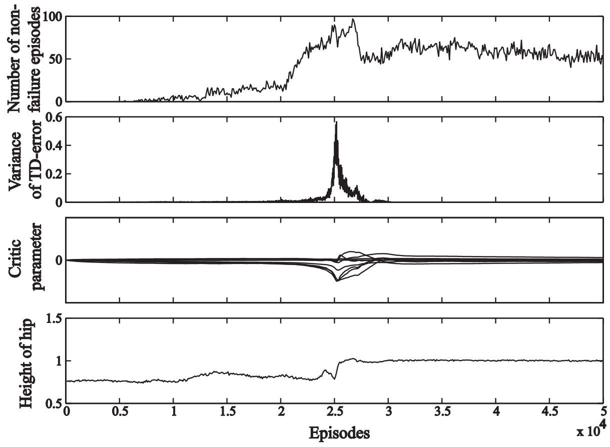
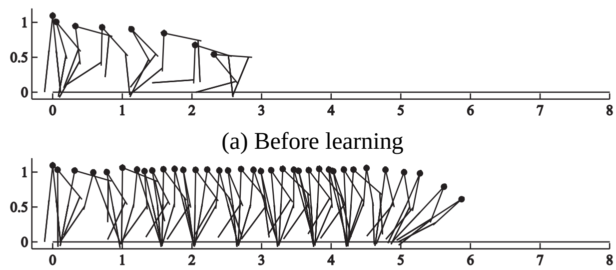


Figure 6: Analysis

learning episodes in which the robot falls down and does not fall down. According to the policy gradient method, the large difference between the actual and the expected value function causes the large gradient. Therefore, a large variance among episodes produces a large gradient which accelerates the learning. Fig.6 shows the detailed analysis of the learning curve in Fig.5. Fig.6(a) shows the number of learning episodes in which the robot does not tumble within every 100 episodes. The number of successful episodes increased after about 20,000 learning episodes. Concurrently to this increase, the variance of the TD-error became large (Fig.6(b)) and the critic's learning was accelerated (Fig.6(c)). Depending on this critic's learning, the actor learning is considered to also be accelerated (Fig.5(c)).

Fig.7(a) and Fig.7(b) show an example gait pattern before learning and that after learning, respectively. In order to evaluate the actor's performance, we repeated a test task in which the robot was initialized at a motion-less posture and was controlled until the robot tumbled or 1000 sec. elapsed. By using the CPG controller before learning, the robot walked for 1.9 sec. on average. By using the CPG controller after learning, on the other hand, the robot could walk for 30.9 sec. on average. The performance of the CPG controller was thus improved by our RL method. Among these 100 times tasks, however, stable walking for 1000 seconds was achieved only once, because the learning converged to a locally optimum solution.



(a) Before learning

(b) After learning

Figure 7: Gait patterns

Learning from the actor parameter acquired by the above learning To escape from the local optimum, we then re-initialized the critic parameter, and trained the actor again. Fig.8 shows the learning curve until 150,000 learning episodes during this re-training. The vertical axis and the horizontal axis are the same as those in Fig.5. The average cost became temporally large after 25,000 learning episodes, but became eventually smaller than the initial value, as shown in Fig.8(a).

After this re-training, Fig.9 shows examples of gait patterns on a flat ground, an up-slope and a down-slope. Although the learning process proceeded on a flat ground, the robot was adapted flexibly to unknown environments, slopes. Before re-training, the number of successful test tasks in which the robot could walk stably for 1000 seconds was only one out of 100, while it became 84 after re-training. This result shows that a better CPG controller was acquired through this re-training process. The connection weights from the sensory feedback signal were $\boldsymbol{\theta}_{RL} = \{0.72, 0.73, 4.5, 1.3, 0.58, 3.7, 5.3, 2.3\}$, which were quite different from the hand-tuned parameter found in (Taga, Yamaguchi, & Shimizu 1991), $\boldsymbol{\theta}_{HT} = \{1.5, 1.0, 1.5, 1.5, 3.0, 1.5, 3.0, 1.5\}$.

The actor parameter converged to a local optimum in the previous experiment, and moreover there may be a lot of local optima or plateaus in the parameter space. By introducing perturbation to the value approximation due to the re-initialization of the critic parameter, the actor could find a better policy in the current experiment. Because the learning process is stable according to our learning scheme based on the policy gradient method, such a re-training is applicable in order to improve the policy.

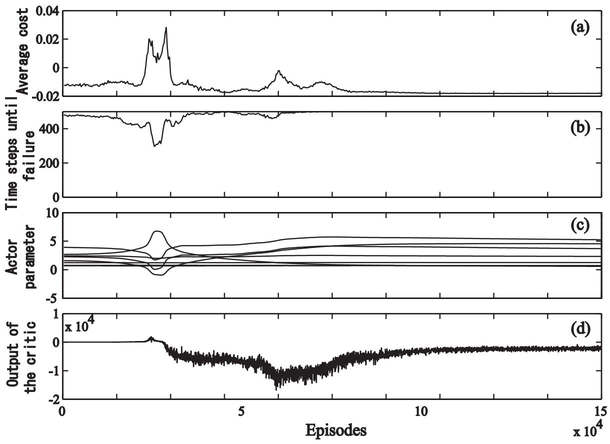
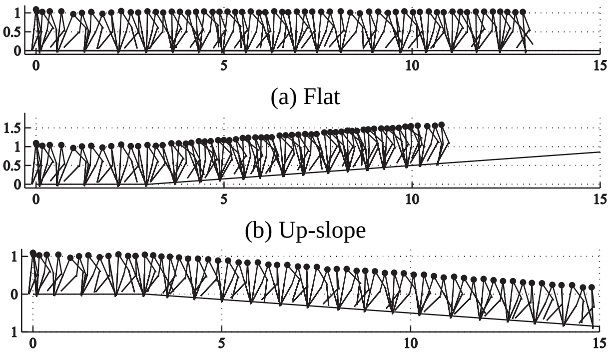


Figure 8: Re-training curve

The comparison of RL and hand-tuned parameter

Next, we trained the CPG controller on a flat or a slope ground starting from the hand-tuned parameter $\boldsymbol{\theta}_{HT}$.

Learning on a flat ground After 50,000 learning episodes on a flat ground starting from the hand-tuned parameter $\boldsymbol{\theta}_{HT}$, the actor parameter was changed to $\boldsymbol{\theta}_{RL}^{flat} = \{1.46, 0.96, 1.39, 1.49, 2.73, 1.33, 2.72, 1.23\}$. In order to



(a) Flat

(b) Up-slope

(c) Down-slope

Figure 9: Gait patterns after re-training

evaluate this parameter, we repeated a test task in which the robot was initialized to a motion-less posture and was controlled on a flat ground, and an up- or down-slope of 2.9 degrees until the robot tumbled or 20 sec. elapsed.

Table.1 shows the number of successful tasks in such 100 tasks controlled by $\boldsymbol{\theta}_{HT}$ and $\boldsymbol{\theta}_{RL}^{flat}$. They exhibited comparable performance because $\boldsymbol{\theta}_{HT}$ had already been a good controller, while $\boldsymbol{\theta}_{RL}^{flat}$ outperformed $\boldsymbol{\theta}_{HT}$ on up-slopes. $\boldsymbol{\theta}_{RL}^{flat}$ was robust to environmental disturbance such as up-slopes. Since the control signal is produced probabilistically (equation (20)) according to our RL scheme, the stochastic nature works similarly to the disturbance induced by the unknown environments. Then, it is considered that the robust policy against actual disturbance has been acquired.

Table 1: The performance comparison of $\boldsymbol{\theta}_{HT}$ and $\boldsymbol{\theta}_{RL}^{flat}$

	$\boldsymbol{\theta}_{HT}$	$\boldsymbol{\theta}_{RL}^{flat}$
Flat	94	96
Up-slope	17	44
Down-slope	95	95

Learning on a slope To evaluate the effect of environmental change in learning, the terrain was prepared as an up-slope of 5.7 degrees. After learning of 15,000 learning episodes, the actor parameter became $\boldsymbol{\theta}_{RL}^{slope} = \{1.51, 1.00, 2.55, 1.51, 1.66, 1.39, 2.87, 1.47\}$. In order to evaluate this parameter, we repeated a test task in which the robot was initialized to a motion-less posture and was controlled on a flat ground, and an up- or down-slope of 5.7 degrees until the robot tumbled or 20 sec elapsed (Table.2).

Table 2: The performance comparison of $\boldsymbol{\theta}_{HT}$ and $\boldsymbol{\theta}_{RL}^{slope}$

	$\boldsymbol{\theta}_{HT}$	$\boldsymbol{\theta}_{RL}^{slope}$
Flat	96	96
Up-slope	13	41
Down-slope	41	98

Although the performance on a flat ground did not show any difference, the performance on slopes was much improved by $\boldsymbol{\theta}_{RL}^{slope}$ on $\boldsymbol{\theta}_{HT}$. The policy parameter which is suitable for walking on slopes has thus been acquired by our learning scheme. Moreover, although the learning episodes are carried out on up-slopes, the performance on down-slopes is also improved. The reason is probably similar to that of the improvement by $\boldsymbol{\theta}_{RL}^{flat}$.

These two experiments suggest that a policy can be improved such to adapt to new environments according to our RL method.

Discussion

In this article, we proposed an RL method for a CPG controller, called the CPG-actor-critic model, which is based on a policy gradient method. We applied our method to an automatic acquisition problem of biped locomotion. Simulation results showed the CPG controller was able to generate stable biped locomotion.

In our method, the policy parameter is supposed to converge to one of the local optima. In order to improve the policy by escaping from such a local optimum, we re-trained the actor parameter by re-initializing the critic's parameter, and then could obtain better one. Acquisition of the locomotion by human being may have such a process. The locomotion is unstable when a baby obtains it first, but is improved as it grows. It may be important to forget past successful experience to obtain a better control, when the learner is caught in a local optimum. We expect that the RL research provides some intuitions on the animal's developmental processes.

Although the simulation was successful, a lot of training episodes were still required. Therefore, it is difficult to apply the method directly to real robots; it is necessary to develop a more efficient algorithm which enables the robot to learn fast. Moreover, the CPG's internal weights were fixed and only sensory feedback connections were adjusted in the current study. It also remains a future study to adjust the weights of our CPG's internal connections by our CPG-actor-critic RL method.

Acknowledgements

One of the authors (MS) is supported in part by the National Institute of Information and Communications Technology.

References

Barnes, D. P. 1998. Hexapodal robot locomotion over uneven terrain. In *Proceedings of IEEE Conference on Control Applications*, 441–445.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Bradtke, S., and Barto, A. 1996. Linear least-squares algorithms for temporal difference learning. *Machine Learning* 22.

Buehler, M.; Battaglia, R.; Cocosco, A.; Hawker, G.; Sarkis, J.; and Yamazaki, K. 1998. Scout: A simple quadruped that walks, climbs and runs. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*.

Ekeberg, Ö. 1993. A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics* 69:363–374.

Fiorillo, C. D.; Tobler, P. N.; and Schultz, W. 2003. Discrete coding of reward probability and uncertainty by dopamine neurons. *SCIENCE* 299:1898–1902.

Fukuoka, Y.; Kimura, H.; and Cohen, A. H. 2003. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *the International Journal of Robotics Research* 22:187–202.

Grillner, S.; Wallen, P.; Brodin, L.; and Lansner, A. 1991. Neuronal network generating locomotor behavior in lamprey: circuitry, transmitters, membrane properties and simulations. *Annual Review of Neuroscience* 14:169–199.

Hirai, K.; Hirose, M.; Haikawa, Y.; and Takenaka, T. 1998. The development of honda humanoid robot. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*.

Ijspeert, A. J. 2001. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics* 84:331–348.

Konda, V. R. 2002. Actor-critic algorithms, phd thesis. *Department of Electrical Engineering and Computer Science Massachusetts Institute of Technology*.

Marbach, P., and Tsitsiklis, J. N. 2001. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control* 46:191–209.

Morimoto, J., and Doya, K. 2001. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems* 36:37–51.

Ogihara, N., and Yamazaki, N. 2001. Generation of human bipedal locomotion by a bio-mimetic neuro-musculo-skeletal model. *Biological Cybernetics* 84:1–11.

Sato, M., and Ishii, S. 1999. Reinforcement learning based on on-line em algorithm. In *Advances in Neural Information Processing Systems*, volume 11, 1052–1058.

Sato, M. 1990. A real time learning algorithm for recurrent analog neural networks. *Biological Cybernetics* 62:237–241.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1057–1063.

Taga, G.; Yamaguchi, Y.; and Shimizu, H. 1991. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics* 65:147–159.

Williams, R. 1992. Simple statistical gradient following algorithms for connectionist reinforcement learning. *Machine Learning* 8:279–292.