論文 / 著書情報
Article / Book Information

Title	Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot
Author	Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, Gordon Cheng
Journal/Book name	International Journal of Robotics Research, Vol. 27, No. 2, pp. 213-228
Issue date	2008, 1
Note	このファイルは著者（最終）版です。 This file is author (final) version

Learning CPG-based Biped Locomotion with a Policy Gradient Method: Applied to a Humanoid Robot

Gen Endo*, Jun Morimoto†‡, Takamitsu Matsubara†§, Jun Nakanishi†‡ and Gordon Cheng†‡

*Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8550, Japan

†ATR Computational Neuroscience Laboratories, ‡Computational Brain Project, ICORP, Japan Science and Technology Agency
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0288, Japan

§Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma-shi, Nara, 630-0192, Japan
gendo@sms.titech.ac.jp, {xmorimo, takam-m, jun, gordon}@atr.jp

[ ]

Abstract

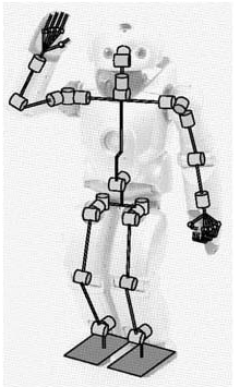
This paper describes a learning framework for a Central Pattern Generator (CPG) based biped locomotion controller using a policy gradient method. Our goals in this study are to achieve CPG-based biped walking with a 3D hardware humanoid, and to develop an efficient learning algorithm with CPG by reducing the dimensionality of the state space used for learning. We demonstrate that an appropriate feedback controller can be acquired within a few thousand trials by numerical simulations and the obtained controller in numerical simulation achieves stable walking with a physical robot in the real world. Numerical simulations and hardware experiments evaluated walking velocity and stability. The results suggest that the learning algorithm is capable of adapting to environmental changes. Furthermore, we present an online learning scheme with initial policy for a hardware robot to improve the controller within 200 iterations.

1 Introduction

Humanoid research and development has made remarkable progress over the past ten years (Hirai *et al.* 1998)(Nishiwaki *et al.* 2000) (Kuroki *et al.* 2001)(Hirukawa *et al.* 2004)(Park *et al.* 2005). Much of these successful humanoids utilize a pre-planned nominal trajectory designed in a typically known environment. Despite our best effort, it seems difficult to consider every possible situation in advance when designing such complex controllers. For broad range of applications working within unknown environments, equipping humanoid robots with learning capability provide a promising avenue in this undertaking. In this paper, we present a learning framework for bipedal locomotion for a humanoid robot.

Learning a biped walking pattern for a humanoid robot is a challenging task, owing to the large-scale problem involved in dealing with the real world. Unlike resolving simple tasks with robots with fewer degrees of freedom, we cannot directly apply conventional learning methods to humanoid robots, due to the dimensionality explosion that bound to incur. Our goals in this paper are to acquire a successful walking pattern through learning, and to achieve robust walking with a hardware 3D full-body humanoid robot (Kuroki *et al.* 2001)(Fig. 1).

While many attempts have been made to investigate learning algorithms for simulated biped walking, there are only a few successful implementation on real hardware, for example (Benbrahim & Franklin 1997)(Tedrake, Zhang, & Seung 2004)(Morimoto *et al.* 2005). To the best of our knowledge, Tedrake $et\,al.$ (Tedrake, Zhang, & Seung 2004) is the only example of an implementation of learning algorithm on a 3D hardware robot. They developed a simple physical 3D biped robot with specially designed round soles, possessing basic properties of a passive dynamic walker (McGeer 1990). They implemented a learning



Specification	
Height	580mm
Weight	7kg
Joint	38DOF
Neck	4DOF
Arms	5DOF(×2)
Fingers	5DOF(×2)
Waist	2DOF
Legs	6DOF(×2)

Figure 1: A hardware platform of full-body humanoid robot: joint configuration and its specification

algorithm on the hardware robot and successfully obtained an appropriate feedback controller for ankle roll joints via online learning. With the help of their specific mechanical design to embed an intrinsic walking pattern possessing passive dynamics, the state space for learning was drastically reduced from 18 to 2 in spite of the complexity of the 3D biped model, which usually suffers from dimensionality explosion.

The question is whether we can easily extend their approach to a general humanoid robot. They used the desired state on the return map taken from the gait of the robot walking down on a slope without actuation in order to define the reward function for reinforcement learning. Their learning algorithm owes much to the intrinsic passive dynamical characteristics of the robot that can walk down a slope without actuation. Although, humans also have passive dynamic property in their joints and muscles, it is extremely difficult with current hardware technology to design general humanoid robots, which has both passive dynamic property and high power joint actuation for various tasks. Therefore, their approach cannot be directly applied to general humanoid robots that are not mechanically designed with specific dynamical characteristics for only walking. Moreover, developing a specific humanoid hardware with uni-functionality, for example walking, may lose an important feature of humanoid robot such as versatility and capability of achieving various tasks.

Therefore, instead of gait implementation by mechanical design, we introduce the idea of using a Central Pattern Generator (CPG), which has been hypothesized to exist in the central nervous system of animals (McMahon 1984; Orlovsky, Deliagina, & Grillner 1999; Cohen 2003). It is known that during locomotion a feedforward excitation to the muscles exists that can be independent of sensory feedback and brain input (Grillner *et al.* 1995). The feedforward muscle activation is generated by a CPG within the spinal cord. The most interesting property of the CPG is that the basic pattern produced by intrinsic oscillation can interact with feedback signals. The intrinsic oscillation of CPG synchronizes the oscillation of feedback signals. This phenomenon is known as “entrainment”.

It can be demonstrated with numerical simulations that CPG can generate a robust biped walking pattern with appropriate feedback signals even in an unpredictable environment, due to the entrainment property of the CPG (Taga 1995; Miyakoshi *et*

al. 1998). However, designing appropriate feedback pathways of neural oscillators often requires much effort to manually tune the parameters of the oscillator. Thus, a genetic algorithm (Hase & Yamazaki 1998) and reinforcement learning (Mori *et al.* 2004) have been proposed to optimize the open parameters of the CPG for biped locomotion. However, these methods often require a large number of iteration to obtain a solution due to the large dimensionality of the state space used for optimization.

Our primary goals are to achieve biped walking with learning for a 3D full-body humanoid robot, which is not designed for a specific walking purpose, and to develop an efficient learning algorithm that can be implemented on a hardware robot with additional online learning capability to improve the controller. In a physical robot, we cannot accurately observe all states of the system due to limited number of equipped sensors and measurement noise in practice. Thus, we find it natural to postulate the learning problem as a Partially Observable Markov Decision Problem (POMDP).

In this paper, we use a policy gradient method which can be applied to POMDP (Kimura & Kobayashi 1998). In POMDP, it is generally known that a large number of iterations would be required for learning compared with learning in Markov Decision Problem (MDP), due to the lack of information, which yields large variance of the estimated gradient of expected reward with respect to the policy parameters (Sutton *et al.* 2000; Konda & Tsitsiklis 2003). However, in the proposed framework, when the CPG and the mechanical system of the robot converge to a periodic trajectory due to entrainment, the internal states of the CPG and the states of the robot will be synchronized. Thus, by using the state space only composed of the observable reduced the number of states, efficient learning can achieve steady periodic biped locomotion even in the POMDP.

In our previous work, we demonstrated a learning framework for a CPG-based biped locomotion with a policy gradient method on a two dimensional planar biped robot (Matsubara *et al.* 2006). The robot had four leg-joints and the control architecture of the robot consisted of a CPG-based controller for two hip joints and a state-machine controller for the two knee joints. Appropriate sensory feedback for the CPG to perform steady walking could be learned within a few hundred trials in simulations, and we applied the controllers acquired from numerical simulations to a physical 5-link biped robot. We empirically verified that the robot was able to successfully walk in a real environment.

In this paper, we extend our previous approach to a 3D full-body humanoid robot and present that the policy gradient method can acquire a steady walking pattern for a general 3D full-body humanoid. Since a three dimensional full-body humanoid robot has many degrees of freedom, the dynamics of the three dimensional humanoid is much more complicated than the two dimensional system. Thus, we propose the idea of allocating CPGs in a task space coordinate system, while exploiting symmetry to simplify the control architecture. In this paper, we demonstrate that an appropriate feedback controller for a 3D full-body humanoid can be acquired by using a policy gradient method and the obtained controller in numerical simulation can achieve stable walking with a physical robot in the real world. Moreover, we discuss a turning walk with a desired turning radius; and an online learning scheme with initial policy for a hardware robot to improve the controller.

2 CPG Control Architecture

This section describes the basic framework of our CPG control architecture. Our goals are to generate a steady straight walking pattern and a steady circular walking pattern with variable turning radius. Fig. 2 shows the basic framework of the CPG control architecture. We introduce a neural oscillator to model a CPG and the oscillator output q_j is transformed into leg position with respect to body-fixed Cartesian coordinate system. We also introduce, $R_{desired}$, a parameter to specify a turning radius defined on the ground to generate circular walking. $R_{desired}$ modulates leg position, $\boldsymbol{p}^{l,r}$, based on geometrical constraints to walk along a specified circular walking trajectory. Then, desired joint position, for joint PD servo is obtained through inverse kinematics and desired joint position is used to control an actuator.

The CPG feedback controller generates feedback signal to CPG, a_j, using sensory information from the robot. The CPG feedback controller consists of reinforcement learning for the oscillator allocated for the X (forward) direction and biologically-inspired feedbacks arranged for the Z (vertical) direction. Finally, a_j is fed back to the neural oscillator and the neural oscillator automatically adjusts its output due to entrainment property.

This framework provides us with an inherent rhythmic walking pattern modulated by the CPG feedback controller using sensory information from the environment. We discuss the above mentioned framework in detail in the following subsections.

2.1 Neural Oscillator Model

There are several ways to model a CPG and it can be mathematically modeled by a non-linear oscillator such as Van der Pol oscillator, phase oscillator and neural oscillator. One recent example is the work of Aoi *et al.*, they applied non-linear oscillators to the control of a small-sized walking humanoid robot, utilising the foot-contact to reset phase of the oscillators to increase the stability of the system (Aoi & Tsuchiya 2005). Their results demonstrated one successful application of phase oscillator to humanoid locomotion.

The learning framework we present in this paper, propose the modulation of the phase as well as the trajectories of the walking patterns of the robot to attain successful and robust locomotion. Our focus, in particular, is on a coupled neural oscillator model proposed by Matsuoka (Matsuoka 1985). The Matsuoka oscillator have a number of beneficial properties, noticeably, its allows modulations of sensory feedback for the adaptation to the environment, control of amplitude with a single scale factor. Those are some of the well-studied aspects of the Matsuoka oscillator, making it suitable for our investigation. The oscillator dynamics of j-th neural unit are:

$$\tau_{CPG}\dot{z}_j = -z_j - \sum_{k=1}^{n} w_{jk}q_k - \gamma z'_j + c + a_j, \tag{1}$$

$$\tau'_{CPG}\dot{z'}_j = -z_j + q_j, \tag{2}$$

$$q_j = \max(0, z_j), \tag{3}$$

where n is the number of neurons. The model represents the firing rate of a neuron by a continuous variable q_j with time. Z_j represents mean membrane potential and Z'_j is a variable which represents self inhibition effect. Adaptation is modeled by dynamics of Z'_j in Eqn. (2) and the degree of the adaptation influence on Z_j is represented by a constant parameter γ. τ_{CPG} and τ'_{CPG} are time constants of the mean membrane potential Z_j and adaptation effect of the j-th neuron, respectively. w_{jk} is an inhibitory synaptic weight from the k-th neuron to the j-th neuron. c is a tonic input with a constant rate and a_j is a feedback signal.

Fig. 3 shows a schematic figure of a coupled oscillator where two neural units are connected by mutual inhibitions. The circles with number represent neural units whose dynamics are defined by Eqn.(1)-(3). Lines with a black circle and a white circle mean inhibitory/excitory neural connection, respectively.

Properties of the Matsuoka neural oscillator model has been numerically explored, signifying the relationship between the parameters and the oscillator output (Williamson 1998). For example, the two time constants τ_{CPG} and τ'_{CPG} determine the frequency and shape of the output, and if the ratio $\frac{\tau_{CPG}}{\tau'_{CPG}}$ is kept constant the natural frequency of the oscillator is proportional to $\frac{1}{\tau_{CPG}}$. The tonic input c controls the amplitude of the output of the oscillator. It is demonstrated that phase difference between the periodic input signal a_j and the output q_j is tightly locked through entrainment when the amplitude of a_j is large enough and its frequency is close to the oscillator's natural frequency.

Fig. 4 shows a time course of the oscillator output where a sinusoid input $a_1(= -a_2)$ is fed into the oscillator at $4.1\,sec$ for 10 seconds. The frequency of the oscillator output is immediately entrained to the frequency of the input sinusoid and phase difference between the input and the output becomes constant. Fig. 4 demonstrates that a neural oscillator has inherent dynamics which can be modulated by a input signal. The key issues to perform robust biped walking are how to allocate the neural oscillator to control biped walking and how to derive input signals a_j to exploit the entrainment property of the neural oscillator. We will discuss them in the following section.

2.2 CPG Arrangement and Leg Trajectory

In many of the previous applications of neural oscillator based locomotion studies, an oscillator is allocated at each joint and its output is used as a joint torque command to the robot (Taga 1995) (Hase & Yamazaki 1997) (Ishiguro, Fujii, & Hotz 2003). However, it is difficult to obtain appropriate feedback pathways for all the oscillators to achieve the desired behavior with the increase of the number of degrees of freedom of the robot because neural oscillators are intrinsically non-linear. Moreover, precise torque control of each joints is also difficult to realize for a hardware robot in practice. Thus, to simplify the problem, we have proposed a new oscillator arrangement with respect to the position of the tip of the leg in the Cartesian coordinate system, which can be reasonably considered as the task coordinates for walking (Endo *et al.* 2005b). We allocate only 6

neural units exploiting symmetry of the walking pattern between the legs. We decompose overall walking motion into stepping motion in place produced in the frontal plane and propulsive motion generated in the sagittal plane. The effectiveness of this decomposition have been empirically demonstrated in our previous studies (Endo *et al.* 2004) (Endo *et al.* 2005b).

Fig. 5 illustrates the proposed neural arrangement for the stepping motion in place in the frontal plane. We employ a coupled oscillator with mutual inhibitions ($w_{12} = w_{21} = 2.0$) and allocate it to control the position of both legs p_z^l, p_z^r along the Z (vertical) direction in a symmetrical manner with a π rad phase difference:

$$p_z^l = Z_0 - A_z\,(q_1 - q_2), \tag{4}$$

$$p_z^r = Z_0 + A_z\,(q_1 - q_2), \tag{5}$$

where Z_0 is a position offset and A_z is the amplitude scaling factor.

For a propulsive motion in the sagittal plane, we introduce a quad-element neural oscillator to produce coordinated leg movements with stepping motion based on the following observation: as illustrated in Fig. 6, when the robot is walking forward, the leg trajectory with respect to the body coordinates in the sagittal plane can be roughly approximated by the shape of an ellipsoid. Suppose the output trajectories of the oscillators can be approximated as $p_x^l = A_x \cos(\omega t + \alpha_x)$ and $p_z^l = A_z \cos(\omega t + \alpha_z)$, respectively. Then, to form the ellipsoidal trajectory on the X-Z plane, p_x^l and p_z^l need to satisfy the relationship $p_x^l = A_x \cos\phi$ and $p_z^l = A_z \sin\phi$, where ϕ is the angle defined in Fig. 6. Thus, the desired phase difference between vertical and horizontal oscillation should be $\alpha_x - \alpha_z = \pi/2$. To embed this phase difference as an intrinsic property, we install a quad-element neural oscillator with uni-directional circular inhibitions ($w_{34} = w_{43} = w_{56} = w_{65} = 2.0$, $w_{35} = w_{63} = w_{46} = w_{54} = 0.5$). It generates inherent phase difference of $\pi/2$ between two coupled oscillators, $(q_3 - q_4)$ and $(q_5 - q_6)$ (Matsuoka 1985). Therefore, if $(q_3 - q_4)$ is entrained to the vertical leg movements, then an appropriate horizontal oscillation with desired phase difference is achieved by $(q_5 - q_6)$. [1]

Similar to the Z direction, the neural output $(q_5 - q_6)$ is allocated to control the position of both legs p_x^l, p_x^r along the X (forward) direction in the sagittal plane:

$$p_x^l = X_0 - A_x\,(q_5 - q_6), \tag{6}$$

$$p_x^r = X_0 + A_x\,(q_5 - q_6), \tag{7}$$

where X_0 is an offset and A_x is the amplitude scaling factor.

This framework provides us with a basic walking pattern which can be modulated by feedback signals a_j.

[1] At the beginning of investigation, we directly used (q_1, q_2) for a quad-element oscillator. However, oscillator dynamics of stepping motion interfered with that of propulsive motion via uni-directional circular inhibitions. As a result, biologically inspired feedback signal, which was derived for a coupled oscillator, was not sufficiently effective to produce robust stepping motion in place. Thus, in order to clearly divide stepping motion and propulsive motion, we introduce duplicated neural units (q_3, q_4) for a quad-element oscillator to produce similar behavior of (q_1, q_2).

2.3 Turning Controller

We introduce an additional mechanism to control walking direction by modulating right-and-left step length as well as yaw rotation of both legs. In this controller, we focus on kinematic constraints to walk along a specified desired circular radius, $R_{desired}$, defined in the horizontal walking surface. $R_{desired}$ modulates the mapping from a CPG output q_j to leg position $\boldsymbol{p}^{l,r}$.

As illustrated in Fig. 7, we assume that the origin of body-fixed coordinates moves with constant velocity along a particular circular arc defined by $R_{desired}$ ($R_{desired} > 0$, when the robot makes a right turn), and left-and-right stance legs also move along the concentric circular arcs defined by $R_{desired} + Y_0^l$ and $R_{desired} - Y_0^r$, where $Y_0^{l,r}$ is leg position offset in the lateral direction. In order to satisfy kinematic constraints without slippage of the stance leg, inner step length should be decreased while the outer step length should be increased due to leg position offset $Y_0^{l,r}$ ($|Y_0^{l,r}| < |R_{desired}|$):

$$A_x^{l,r} = \frac{R_{desired} + Y_0^{l,r}}{R_{desired}} \cdot A_x, \tag{8}$$

where A_x is nominal step length in case of straight walking and $A_x^{l,r}$ are modulated step lengths of left and right legs. Thus, Eqn. (6)(7) are rewritten as follows:

$$p_x^l = X_0 - A_x^l \cdot (q_5 - q_6), \tag{9}$$

$$p_x^r = X_0 + A_x^r \cdot (q_5 - q_6). \tag{10}$$

The yaw rotation and Y position of the legs, $p_{yaw}^{l,r}$, $p_y^{l,r}$, are also controlled to satisfy kinematic constraints to keep constant angular velocity around the center of turning as follows:

$$p_{yaw}^{l,r} = -\frac{p_x^{l,r} - X_0}{R_{desired} + Y_0^{l,r}}, \tag{11}$$

$$p_y^{l,r} = Y_0^{l,r} - (1 - \cos(p_{yaw}^{l,r})) \cdot (R_{desired} + Y_0^{l,r}). \tag{12}$$

Although, considering only kinematic constraints is not sufficient to achieve an executable walking motion because it neglects the effect of dynamics. As demonstrated, the proposed framework possess entrainment property to some extent could better cope with the effect of dynamics. Additionally, the learning algorithm is capable of adjusting the basic walking pattern via CPG feedback signals. Therefore, in our case, we can conceive that kinematic constraints can be adequate to generate a turning motion. The advantage of this turning controller is that we can continuously vary the walking direction by only adjusting a single parameter, $R_{desired}$.

2.4 Sensory Feedback

In biological systems, several reflexes were found in order to generate recovery momentum according to the inclination of the body by adjusting the leg length. For example, a decerebrate cat stomps stronger when vertical perturbation force is applied to its planter during extensor muscle activation. This reflex is called an *Extensor Response* (Cohen & Boothe 1999). It is generally known that vestibular system measures body's inclination and activates contralateral muscles to keep upper body stabilized. This is one of the basic posture control in human and called *Vestibulo-spinal Reflex*. Effectiveness of these feedback pathways

was experimentally demonstrated with a hardware quadruped robot (Kimura, Fukuoka, & Cohen 2007) to maintain the balance of the body when walking over unknown terrain.

The first step to this study, we focus on acquiring the feedback controller (a_5) for the propulsive leg movement in the X direction (as illustrated in Fig.5). The explicitly designed sensory feedback pathways for stepping motion in place (a_1) is strongly motivated by biological observations (Fig.5).

Based on the assumption of symmetric leg movements generated by the coupled oscillator, we introduce symmetrical feedback signals to the oscillator as following:

$$a_{2m} = -a_{2m-1}, \ \ (m = 1, 2, 3). \tag{13}$$

We then introduce *Extensor Response* and *Vestibulo-spinal Reflex* by adjusting leg length according to the vertical reaction force or the inclination of the body as follows:

$$a_1 = h_{ER}\,(f_z^r - f_z^l)\,/\,mg \;+\; h_{VSR}\,\theta_{roll}, \tag{14}$$

where $(f_z^r - f_z^l)$ are right/left vertical reaction force differences normalized by total body weight mg. h_{ER}, h_{VSR} are scaling factors. *Extensor Response*, the first term of the right side of Eqn. (14), extends leg length when the vertical reaction force is applied. This motion moves the center of pressure of the robot to opposite side. Thus, repetitive stepping constructs a negative feedback loop to keep the center of pressure between two feet. *Vestibulo-spinal Reflex*, the second term of the right side of Eqn. (14), also extends leg length according to the inclination of the body. This reflex also constructs a negative feedback loop to maintain the upright posture. *Extensor Response* and *Vestibulo-spinal Reflex* construct redundant feedback pathways which are tolerant of failure of sensing in a hardware system and they can be nicely combined by adjusting scaling factors. A dominant feedback pathway to increase robustness against perturbation can change on a case to case situation, depending on the perturbations. We experimentally demonstrated the robustness of stepping motion against perturbation (Endo *et al.* 2005b). We empirically investigated scaling factors using the hardware robot by applying various perturbations and determined $h_{ER} = 0.4$, $h_{VSR} = 1.8$.

The same feedback signal is fed back to the quad-element neural oscillator for the propulsive motion, $a_3 = a_1$, to induce cooperative leg movements with $\pi/2$ phase difference between the Z and X direction (Fig.6).

For the propulsive leg movements in the X direction, the feedback signal a_j are represented with a policy gradient.

$$a_j(t) = a_j^{\max} g(v_j(t)), \tag{15}$$

where the function g is a saturation function defined by $g(v_j(t)) = \frac{2}{\pi}\arctan\left(\frac{\pi}{2}v_j(t)\right)$, and $a_j^{\max}$ is the maximum value of the feedback signal. The output of the feedback controller v_j is sampled from a stochastic policy which locally maximizes expected total return (the accumulated reward, which will be defined in the following section). The stochastic policy is estimated by a probability distribution $\pi_{\mathbf{w}^a}(\mathbf{x}, v_j) = P(v_j|\mathbf{x}; \mathbf{w}^a)$:

$$\pi_{\mathbf{w}^a}(\mathbf{x}, v_j)$$

$$= \frac{1}{\sqrt{2\pi}\sigma_j(\mathbf{w}^\sigma)}\exp\left(\frac{-(v_j-\mu_j(\mathbf{x};\mathbf{w}^\mu))^2}{2\sigma_j^2(\mathbf{w}^\sigma)}\right), \tag{16}$$

where $\mathbf{x}$ denotes partial states of the robot, and $\mathbf{w}^a = [(\mathbf{w}^\mu)^T, (\mathbf{w}^\sigma)^T]$is the parameter vector of the policy. We can equivalently represent v_j by

$$v_j(t) = \mu_j(\mathbf{x}(t);\mathbf{w}^\mu) + \sigma_j(\mathbf{w}^\sigma)n_j(t), \tag{17}$$

where, $n_j(t) \sim N(0,1)$. $N(0,1)$ is a normal distribution which has a mean μ of 0 and a variance σ^2 of 1. In the next section, we discuss the learning algorithm to acquire a feedback controller $a_{j=5}$ for the propulsive motion.

3 Learning the sensory feedback controller

Promising results have been demonstrated in application of policy gradient technique to POMDP biped walking locomotion (Tedrake, Zhang, & Seung 2004; Endo *et al.* 2005a). In our previous work (Matsubara *et al.* 2006), we compared the policy gradient method with a conventional value function based reinforcement learning scheme on a two dimensional biped walking task. The policy gradient method could acquire steady walking with only a smaller number of trials, suggesting that the policy gradient method is suitable for POMDPs. In this paper, we use the same learning algorithm for the acquisition of a policy of the sensory feedback controller of the neural oscillator model in the X direction.

Humanoid robots have many degrees of freedom, and with great number of equipped sensors within a single hardware system, therefore, it is not feasible to use all states of the robot for learning. Thus, we have to select reasonable number of state variables for learning among all states variables of the robot. To describe the representative motion of the robot, we focus on the states of the pelvis of the robot. The position of the pelvis can roughly approximates the location of the center of mass (COM) of the system. We chose the pelvis angular velocity $\mathbf{x} = (\dot{\theta}_{roll}, \dot{\theta}_{pitch})^T$ as the input state to the learning algorithm, and $\dot{\theta}_{roll}, \dot{\theta}_{pitch}$ are measured by gyro sensors located on the pelvis of the hardware. Therefore, the rest of the states of the robot, such as inclinations of the pelvis and linear velocity with respect to world coordinates, position and velocity of each joint are considered hidden variables for the learning algorithm.

The learning framework of the policy gradient method for our CPG control architecture is illustrated Fig. 8. First, CPG controller generates leg trajectory in the X direction, allowing the robot interacts with the physical environment. The partial states of the robot and reward information are sent to the learning agent. A critic tries to estimate the value using Temporal Difference error (TD-error), represented in continuous time and space. Then, an actor generates CPG feedback signal for leg trajectory in the X direction based on a stochastic policy defined by a probability distribution with parameter vectors $\mathbf{w}$. Both the value function and the policy are updated using a TD-error and eligibility trace according to Kimura's update rule (see subsection 'Learning a policy of the sensory feedback controller' for more detail).

In the following subsections, we introduce the definition of the value function in continuous time and space to derive the Temporal Difference error (TD-error) (Doya 2000). Then, we discuss the learning method of a policy for the sensory feed-

back controller. We use a normalized Gaussian network (NGnet) to approximate both the value function and the policy (see Appendix). Finally, we design a reward function to generate steady walking.

3.1 Learning the value function

Consider the dynamics of the robot including the CPG defined in continuous-time and continuous-states,

$$\frac{d\mathbf{x}^{all}(t)}{dt} = f(\mathbf{x}^{all}(t), \mathbf{a}(t)), \tag{18}$$

where $\mathbf{x}^{all} \in X \subset \Re^l$ is all the states of the robot and the CPG, and $\mathbf{a} \in A \subset \Re^m$ is the output of the feedback controller to the CPG. We denote the immediate reward for the state and action as,

$$r(t) = r(\mathbf{x}^{all}(t), \mathbf{a}(t)). \tag{19}$$

The value function of state $\mathbf{x}^{all}(t)$ based on a policy, $\pi(\mathbf{x}^{all}, \mathbf{a})$ is defined as

$$V^{\pi}(\mathbf{x}^{all}(t)) = E\left\{ \int_t^{\infty} e^{-\frac{s-t}{\tau}} r(\mathbf{x}^{all}(s), \mathbf{a}(s))\, ds \middle| \pi \right\}, \tag{20}$$

where τ is a time constant for discounting future rewards. The consistency condition for the value function is given by the time derivative of (20) as,

$$\frac{dV^{\pi}(\mathbf{x}^{all}(t))}{dt} = \frac{1}{\tau} V^{\pi}(\mathbf{x}^{all}(t)) - r(t). \tag{21}$$

We denote the current estimate of the value function as $V(\mathbf{x}^{all}(t)) = V(\mathbf{x}^{all}(t); \mathbf{w}^c)$, where $\mathbf{w}^c$ is the parameter of the function approximator. If the current estimate of the value function V is perfect, it should satisfy the consistency condition of (21). If this condition is not satisfied, the prediction should be adjusted to decrease the inconsistency,

$$\delta(t) = r(t) - \frac{1}{\tau} V(t) + \dot{V}(t). \tag{22}$$

This is the continuous-time counterpart of TD-error (Doya 2000).

Because we consider a learning framework in POMDPs, i.e., we only observe partial states $\mathbf{x}$ from all states $\mathbf{x}^{all}$, the TD-error usually does not converge to zero. However, Kimura $et\ al.$ (Kimura & Kobayashi 1998) suggested that the approximated value function can be useful to reduce the variance of the gradient estimation in (25), even if the consistency condition in (21) is not satisfied.

The parameter vector of the value function $\mathbf{w}^c$ is updated with TD error and an eligibility trace. Eligibility trace is used to assign credit for the TD-error backward in time to previously visited states. The update laws for $\mathbf{w}^c$ and the eligibility trace vector $\mathbf{e}^c$ for $\mathbf{w}^c$ are defined respectively as,

$$\dot{\mathbf{e}}^c(t) = -\frac{1}{\kappa^c}\mathbf{e}^c(t) + \frac{\partial V_{\mathbf{w}^c}}{\partial \mathbf{w}^c}, \tag{23}$$

$$\dot{\mathbf{w}}^c(t) = \alpha\delta(t)\mathbf{e}^c(t), \tag{24}$$

where α is the learning rate and κ^c is the time constant of the eligibility trace.

We manually tune τ, α, κ^c based on following intuitions. For the learning rate, α, we try to maximize the learning rate that is small enough to estimate the expectation in Eqn. (20) by averaging sample data. The time constant τ corresponds to the discount rate in a discrete time learning system. Smaller τ means that future reward is considered to yield smaller value than actual value. We set sufficiently large value for this parameter to take into account the negative reward that is caused when the robot falls over. Time constant κ^c controls credit assignment of the reward to the previously visited state. Because we consider a partially observable environment, it is important to assign the credit from actual acquired reward, which does not depend on the estimated value function. However, a too large κ^c leads to large variance in the value function estimation. In this study, we select the learning parameters as $\tau = 1.0$, $\alpha = 78$, $\kappa^c = 0.5$.

3.2 Learning a policy of the sensory feedback controller

In Kimura *et al.* (Kimura & Kobayashi 1998), presented that by using TD error $\delta(t)$ and an eligibility trace vector $\mathbf{e}^a(t)$, it is possible to obtain an estimate of the gradient of the expected actual return V_t with respect to the parameter vector $\mathbf{w}^a$ in the limit of $\kappa^a = \tau$ as,

$$\frac{\partial}{\partial \mathbf{w}^a} E\{\, V_t \mid \pi_{\mathbf{w}^a}\} = E\{\delta(t)\mathbf{e}^a(t)\}, \tag{25}$$

where,

$$V_t = \int_t^\infty e^{-\frac{s-t}{\tau}} r(s)\, ds, \tag{26}$$

$\mathbf{w}^a$ is the parameter vector of the policy $\pi_{\mathbf{w}^a} = \pi(\mathbf{x}, \mathbf{a}; \mathbf{w}^a)$, and $\mathbf{e}^a(t)$ is the eligibility trace vector for the parameter vector $\mathbf{w}^a$. The parameter vector of the policy $\mathbf{w}^a$ is updated with TD-error and the eligibility trace. Eligibility trace is used to assign credit for the TD error backward in time to previously generated actions. The update laws for $\mathbf{w}^a$ and the eligibility trace vector $\mathbf{e}^a(t)$ can be derived respectively as,

$$\dot{\mathbf{e}}^a(t) = -\frac{1}{\kappa^a}\mathbf{e}^a(t) + \frac{\partial \ln \pi_{\mathbf{w}^a}}{\partial \mathbf{w}^a}, \tag{27}$$

$$\dot{\mathbf{w}}^a(t) = \beta\delta(t)\mathbf{e}^a(t), \tag{28}$$

where β is the learning rate and κ^a is the time constant of the eligibility trace.

In the actor-critic algorithm in (Sutton *et al.* 2000; Konda & Tsitsiklis 2003), they used a Q-function to update parameters of the actor. Whereas, in Kimura's approach, the TD-error is used to update the policy parameters. This is because, that if a target dynamics depends on a policy parameter, information for the proper gradient direction of the policy parameter can be acquired, since the TD-error depends on the dynamics of environment.

The basic intuition for updating policy with TD-error and the eligibility trace is the following: Larger TD-error indicates that generated action caused better result, i.e., an acquired larger reward and/or achieved a state has larger estimated value than the expected value. To increase the chance to acquire larger reward, we can increase policy parameters contributed to increase larger TD-error. Eligibility traces represent the contribution of each policy parameters.

We manually tune β, κ^a based on the following intuitions. For learning rate β, we try to maximize the learning rate that is small enough to estimate the expectation in Eqn. (25) by averaging sample data, (using small learning rate has the same effect as

sample data averaging). Time constant κ^a controls credit assignment of the reward to the previously generated action. Because we consider a partially observable environment, it is important to assign the credit from the actual acquired reward, which does not depend on the estimated value function. However, a too large κ^a leads to large variance in the gradient estimation. In this study, we set the learning parameters as $\beta^\mu = \beta^\sigma = 195$, $\kappa^\mu = 1.0$, $\kappa^\sigma = 0.1$.

3.3 Rewards

We design a reward function:

$$r(\mathbf{x}) = k_H\,(h_1 - h') \;+\; k_S\,v_x, \tag{29}$$

where h_1 is the pelvis height of the robot, h' is a threshold parameter for h_1 and v_x is forward velocity with respect to the ground. The reward function is designed to keep the height of the pelvis by the first term, while at the same time to achieve forward progress by the second term. In this study, the parameters were chosen as k_S in the range of 0.0 to 10.0, $k_H = 10.0$, $h' = 0.272$, where the unit for h_1, and h' are in meters, and for v_x is meter per second, respectively. The threshold parameter h' is determined by a position offset Z_0. The robot receives a punishment (negative reward) $r = -1$, if it falls over.

4 Experiments

4.1 Dynamics Simulator

We developed a dynamics simulator for our biped robot using SD/FAST (Symbolic Dynamics Inc.). We used detailed mass property data calculated from 3D CAD data and mass measurement of actual parts of the robot. We introduced an actuator model for all joints incorporating PD servo and gear head friction. The coefficients of PD gains, coulomb friction and viscous friction were experimentally determined using the hardware robot. We also introduced a sensor model with a digital filter whose coefficients were carefully tuned to match the property of the actual sensor. To calculate reaction forces from the environment, we assumed that each sole have four contact points and reaction forces are obtained by a simple spring-dumper model. The integration time step of the numerical simulator was set to 0.1 $msec$, and the learning was performed at 16 $msec$ interval. The total computation including learning algorithm required 3.8 times longer than real time.

4.2 Simulation Results and Hardware Verifications

Firstly, we carried out numerical experiments using dynamics simulator to acquire a feedback controller (policy) to the CPG for biped walking. Secondary, we implemented the acquired policies on the hardware robot.

We conducted a number of trials in a numerical experiment in the following sequence: at the beginning of each trial, we utilize a hand-designed feedback controller to initiate walking gait for several steps in order to start the learning process with the appropriate state of the robot. Then, we switch the feedback controller for the learning algorithm at random in order to generate various initial state inputs. Each trial is terminated if the immediate reward is -1 and below, or the robot walks for 20 sec. We repeated the numerical experiment trials and the policy was saved at every 50 trials. The learning process is considered a success when the robot does not fall over after 20 successive trials. We terminated the experiment in the case where additional 3000 trials were done after successful acquisition or 5000 trials even without successfully acquiring a policy.

As expected, at the beginning of a learning process, the robot immediately fell over within a few steps straight after switching to the learned feedback controller. The policy gradually increases the output amplitude of the feedback controller to improve walking motion as the learning proceeded. Based on 27 numerical experiments with various velocity reward, k_S, and walking motion was successfully acquired for 20 experiments (Table. 1). (We could not observe policy convergence for 7 failed experiments.) The required trials are averaged value of each successful experiment with the same k_S. Typically, it takes 20 hours to run one simulation for 1000 trials and the policy was acquired on average after 696 trials. Fig. 9 and Fig. 10 show a typical example of accumulated reward at each trial and an acquired policy, respectively. Fig. 10 shows, while $\dot{\theta}_{roll}$ dominates the policy output, $\dot{\theta}_{pitch}$ does not assert much influence. The reason is that $\dot{\theta}_{roll}$ is always being generated by the stepping motion regardless of the propulsive motion. Therefore, the policy tries to utilize $\dot{\theta}_{roll}$ to generate synchronized leg movements. We also observed that, $\dot{\theta}_{pitch}$ is suppressed by the reward function because $\dot{\theta}_{pitch}$ lowers the pelvis height by a pitching oscillation, which can cause falling.

We transferred the acquired 20 successful learning processes on to the robot by using series of policies with different trials with the same learning process. We then used these policies to determine the required additional iterations for the robot to walk. The walking experiment on the hardware was considered a success when the robot achieved steady walking on the carpet floor for 3 meters without falling over. We verified improvements of the policy is in according with numbers of trials in the numerical simulation. With the policy on the early stage of a learning process, the robot exhibited back and forth stepping then immediately fell over. With the policy on the intermediate stage, the robot performed unsteady forward walking and occasional stepping on the spot. With the policy after substantial trials, the robot finally achieved steady walking.

We confirmed that 14 of successful learning processes in the numerical experiments performed successful walking on the hardware (Table. 1). Fig. 11 shows snapshots of an acquired walking pattern. The 6 policies, which could not succeed on the hardware experiment, had similar profile to a typical policy shown in Fig. 10. However the output amplitude was slightly smaller or significantly larger than the policy that is applicable to the hardware. In particular, the large feedback signal to CPG led to instantaneous leg movement when $\dot{\theta}_{roll}$ was across zero. Consequently, the leg movement exceeded hardware current limitation, which was not modeled in the dynamics simulator and the robot fell over. The policy with slightly smaller output can be improved via on-line learning, which will be discussed in the following section.

In our experiments, additional 189 trials (on average) in numerical simulations were required for the policy to achieve walking in the physical environment. We confirmed that the output amplitude of the feedback signal progressively increased in accordance with the number of trials. This result suggests that the learning process gradually exploits the entrainment property of the CPG through iterations, and consequently the policy acquires adequate robustness against perturbation in the real environment. We also carried out walking experiments on a slope and the acquired policy achieved steady walking in the range of +3 to -4 deg inclination, suggesting sufficient walking stability.

k_S	Num. of Exp.	Num. of Achievements	
		Sim.(trials)	Hard.(trials)
0.0	4	1 (2385)	0 (-)
1.0	3	3 (528)	3 (1600)
2.0	3	3 (195)	1 (800)
3.0	4	4 (464)	2 (1500)
4.0	3	2 (192)	2 (350)
5.0	5	2 (1011)	1 (600)
10.0	5	5 (95)	5 (460)
Sum(Ave.)	27	20 (696)	14 (885)

Table 1: Achievements of acquisition of straight walking

4.3 Velocity Control

To control walking velocity, the relationship between the reward function and the acquired velocity was investigated. We set the parameters in Eqn.(1)(2) as $\tau_{CPG} = 0.105$, $\tau'_{CPG} = 0.132$, $c = 2.08$, $\gamma = 2.5$ to generate an inherent oscillation, where its amplitude is 1.0 and period is 0.8, respectively. Since we set $A_x = 0.015\,m$, the expected walking velocity with intrinsic oscillation, (step length)/(step cycle) becomes $(0.015 \times 1.0)/(0.8/2) = 0.075\,m/s$.

We measured average walking velocity both in numerical simulations and hardware experiments with various k_S in the range of 0.0 to 5.0 (Fig. 12). The resultant walking velocity in the simulation increased as we increased k_S and hardware experiments also demonstrated similar tendency.

This result shows the reward function works appropriately to obtain a desirable feedback policy, which is difficult for a hand-designed controller to accomplish. Thus, we believe that it would be possible to acquire different feedback controllers with some other criteria, such as energy efficiency by using the same scheme.

4.4 Stability Analysis

To quantify the stability of an acquired walking controller, we consider the periodic walking motion as discrete dynamics and analyze the local stability around a fix point using a return map. We perturbed the target trajectory on $(q_5 - q_6)$ to change step length at random timing during steady walking, and captured the states of the robot when left leg touched down. We measured two steps, right after the perturbation ($\mathbf{d}_n$) and the next step ($\mathbf{d}_{n+1}$). If acquired walking motion is locally stable, absolute eigenvalue of the return map should be less than 1.

Fig.13 shows the return map with 50 data points and a white dot indicates a fix point derived by averaging 100 steps without perturbation. The estimated eigenvalue is -0.0101 calculated by a least squares fit. The results suggests that even if step length was reduced to half of the nominal step length by perturbation, for example pushed forward, the feedback controller quickly converges to the steady walking pattern within one step.

4.5 Turning Walk

We carried out circular arc walking experiments using the turning controller. We set $R_{desired} = 0.3, 0.5, 1.0m$ and $A_x = 0.015m$, $Y_0^l = 0.04m$, $Y_0^r = -0.04m$. Since $p_{yaw}^{l,r}$ is small, we can abbreviate the second term in the right side of Eqn. (12). To calculate forward walking velocity v_x for the reward in Eqn. (29), we used the relative velocity of the stance leg with respect to the pelvis in the X direction. We performed 15 experiments on each $R_{desired}$ in numerical simulations, a total of 45 experiments were carried out. The numerical simulations was performed on a PC cluster using 45 nodes with AMD Opteron 248 CPU over 3 days. Note that the learning algorithm itself is exactly the same for the case of straight walking, because leg trajectory modulation by introducing the $R_{desired}$ parameter can be regarded as an environmental change for the learning algorithm.

$R_{desired}$	Num. of Exp.	Num. of Achievements			
		Sim.(trials)		Hard.(trials)	
0.3	15	11	(68)	5	(820)
0.5	15	12	(73)	7	(1529)
1.0	15	12	(293)	10	(2355)

Table 2: Achievements of acquisition

The learning algorithm successfully acquired a turning walk with 78% acquisition rate on average in numerical simulations (Table. 2). Fig. 14 shows trajectory of the center of mass during turning walk with respect to a ground-fixed world coordinate system. The simulated robot started walking at position $(0, 0)$ in the direction of X axis with $R_{desired} = 0.3, 0.5, 1.0m$. We can see that the robot nicely followed circular trajectories, which explicitly specified by $R_{desired}$ parameter.

We implemented all the acquired policies with different number of trials on the real robot (over 2 days of experimentation). We investigated numbers of achievements and the required numbers of trials in numerical simulations, same as in the case of straight walking experiments. In Table. 2, the required trials to acquire a turning walk were much less than the case of straight walking. A possible reason is that the walking velocity of a turning walk is relatively smaller than the straight walking due to slip between the stance leg and the ground, which is caused by modeling error of frictional forces in the horizontal direction. Since robustness for stepping motion in place is guaranteed by Eqn. (14), the robot does not fall over even when the robot does not have forward walking velocity. Thus, learning movement with lower walking velocity can be regarded as an easier task, than in the case of walking with higher walking velocity. Therefore, acquisition of a turning walk becomes easier in comparison with straight walking in numerical simulations.

Fig. 15 shows typical results of the acquired CPG feedback policies for all $R_{desired}$ considered. As in the case of straight walking, we could not observe policy convergence for failed experiments. The policies, which could not succeed on the hardware experiment, had similar profile to a typical policy. However the output amplitude was slightly smaller or significantly larger than the policy that is applicable to the hardware.

The smaller $R_{desired}$ utilizes $\dot{\theta}_{pitch}$ information more compared with the straight walking case. The reason would be that we

can not clearly decompose a turning walk into sagittal motion and a lateral motion due to a yaw rotational movement around the center of turning. And a smaller $R_{desired}$ would lead to a larger interference between the two motions. Thus, the learning algorithm tried to utilize both $\dot{\theta}_{roll}$ and $\dot{\theta}_{pitch}$ information to adapt to different $R_{desired}$, suggesting that the learning algorithm appropriately optimized the policy to negotiate with environmental changes.

We implemented the acquired 35 feedback controllers in Table. 2 on the hardware as in the case of straight walking and confirmed that additional 2062 trials on average in numerical simulations were required for the policy to perform circular walking in the physical environment. This result also suggests the learning process gradually improves robustness against perturbation in the physical environment. However, the rate of successful walking in the physical system decreases according with the decrease of $R_{desired}$. The main reason would be modeling error in numerical simulation, especially for ground reaction forces. Even in these cases, we can improve the policy through online learning with the hardware system, which is discussed in the following section.

4.6 Online learning based on obtained policy

As shown in Tables 1 and 2, several policies obtained in numerical simulations could not achieve walking with the physical robot in the real environment.

Although this being the case, even under these conditions, we could make use of an additional online learning on the real hardware. This is due to the fact that computational cost of the numerical simulation is largely due to the dynamics calculation, rather than the learning algorithm itself. Thus, online learning can be adapted. In this section, we attempt to improve the obtained policies of the numerical simulations, which could not originally produce steady walking in the hardware experiments to an online learning scheme.

In the case of the reward function, we are required to provide forward walking velocity and body height. Therefore, it is desirable for the robot to measure these information only by using equipped onboard sensors. We can derived walking velocity by the relative velocity of the stance leg with respect to the pelvis. This can be estimate with inverse kinematics and numerical differentiation of the measured joint angles of the stance leg. We introduced a first-order low pass filter with cutoff frequency of 1Hz to smooth out the velocity estimation. The body height was measured by the the joint angles of the stance leg and the absolute body inclination, which was derived from integration of an internal gyration sensor.

Despite delayed and inaccurate reward information, the online learning algorithm successfully improved the initial policy and performed steady walking within 200 trials (which took 2.5 hours to perform). Fig. 16 shows an example of online learning for straight walking where $k_S = 4.0$, $k_h = 10.0$ and $h' = 0.275$. (Note that the value of the accumulated reward differs from the simulated result in Fig. 9 due to a different time duration $16sec$ for one trial.) Fig. 17 shows the circular walking case where $k_S = 10.0$, $k_h = 10.0$, $h' = 0.275$ with a time duration of $20sec$ for one trial. The gray line indicates running average of accumulated reward for 20 trials. These results show the efficiency of learning speed, which is in practice applicable to the

physical system.

5 Conclusion

In this paper, we proposed an efficient learning framework for CPG-based biped locomotion control using the policy gradient method. We decomposed the walking motion into a stepping motion in place, with a propulsive motion, while the feedback pathways for the propulsive motion were acquired through the proposed policy gradient method. Despite considerable number of hidden variables, the proposed framework successfully obtained a steady walking pattern for straight walking within 1000 trials, on average in the simulation. Acquired feedback controllers were implemented on a 3D hardware robot, and demonstrated robust walking in the physical environment. We discussed velocity control and stability for straight steady walking, as well as an extension to circular walking. Finally, we demonstrated the possibility of online learning with the hardware robot.

To the best of our knowledge, our study is the first successful result to acquire biped locomotion, which can be applied to a full-body hardware humanoid robot.

In this paper, we only considered a policy gradient method based on Kimura's update rule. Comparison to other alternative policy search approaches such as value-function based reinforcement learning (Doya 2000), and GPOMDP developed by Baxter (Baxter & Bartlett 2001) form part of our future work.

Acknowledgments

We would like to thank Seiichi Miyakoshi of the Digital Human Research Center, AIST, Japan and Kenji Doya of Initial Research Project, Okinawa Institute of Science and Technology, for productive and invaluable discussions. We would like to thank all the persons concerned in hardware development of humanoid platform for supporting this research.

References

Aoi, S., and Tsuchiya, K. 2005. Locomotion control of a biped robot using nonlinear oscillators. *Autonomous Robots* 19(3):219–232.

Baxter, J., and Bartlett, P. L. 2001. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research* 15:319–350.

Benbrahim, H., and Franklin, J. A. 1997. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems* 22:283–302.

Cohen, A. H., and Boothe, D. L. 1999. Sensorimotor interactions during locomotion: Principles derived from biological systems. *Autonomous Robotics* 7(3):239–245.

Cohen, A. H. 2003. Control principle for locomotion – looking toward biology. In *Proceedings of the 2nd International Symposium on Adaptive Motion of Animals and Machines (AMAM'03)*. (CD-ROM, TuP-K-1).

Doya, K. 2000. Reinforcement learning in continuous time and space. *Neural Computation* 12:219–245.

Endo, G.; Nakanishi, J.; Morimoto, J.; and Cheng, G. 2004. An empirical exploration of a neural oscillator for biped locomotion control. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA'04)*, 3036–3042.

Endo, G.; Morimoto, J.; Matsubara, T.; Nakanishi, J.; and Cheng, G. 2005a. Learning CPG Sensory Feedback with Policy Gradient for Biped Locomotion for a Full-body Humanoid. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 1267–1273.

Endo, G.; Nakanishi, J.; Morimoto, J.; and Cheng, G. 2005b. Experimental studies of a neural oscillator for biped locomotion with QRIO. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA'05)*, 598–604.

Grillner, S.; Deliagina, T.; Ekeberg, O.; Manira, A. E.; Hill, R. H.; Lansner, A.; Orlovsky, G. N.; and Wallenén, P. 1995. Neural networks that co-ordinate locomotion and body orientation in lamprey. *Trends in NeuroSciences* 18(6):270–279.

Hase, K., and Yamazaki, N. 1997. A self-organizing model to imitate human development for autonomous bipedal walking. In *Proceedings of the 6th International Symposium on Computer Simulation in Biomechanics*, 9–12.

Hase, K., and Yamazaki, N. 1998. Computer simulation of the ontogeny of biped walking. *Anthropological Science* 106(4):327–347.

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 and Automation*, 1321–1326.

Hirukawa, H.; Kanehiro, F.; Kaneko, K.; Kajita, S.; Fujiwara, K.; Kawai, Y.; Tomita, F.; Hirai, S.; Tanie, K.; Isozumi, T.; Akachi, K.; Kawasaki, T.; Ota, S.; Yokoyama, K.; Handa, H.; Fukase, Y.; Maeda, J.; Nakamura, Y.; Tachi, S.; and Inoue, H. 2004. Humanoid robotics platforms developed in HRP. *Robotics and Autonomous Systems* 48(4):165–175.

Ishiguro, A.; Fujii, A.; and Hotz, P. E. 2003. Neuromodulated control of bipedal locomotion using a polymorphic cpg circuit. *Adaptive Behavior* 11(1):7–17.

Kimura, H., and Kobayashi, S. 1998. An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value function. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, 278–286.

Kimura, H.; Fukuoka, Y.; and Cohen, A. H. 2007. Biologically inspired adaptive dynamic walking of a quadruped robot. *Philosophical Transactions of The Royal Society A* 365(1850):153–170.

Konda, V., and Tsitsiklis, J. 2003. On actor-critic algorithms. *Society for Industrial and Applied Mathematics* 42(4):1143–1166.

Kuroki, Y.; Ishida, T.; Tamaguchi, J.; Fujita, M.; and Doi, T. T. 2001. A Small Biped Entertainment Robot. In *Proceedings of the 2001 IEEE-RAS International Conference on Humanoid Robots (Humanoids2001)*, 181–186.

Matsubara, T.; Morimoto, J.; Nakanishi, J.; Sato, M.; and Doya, K. 2006. Learning CPG-based biped locomotion with a policy gradient method. *Robotics and Autonomous Systems* 54:911–920.

Matsuoka, K. 1985. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics* 52:345–353.

McGeer, T. 1990. Passive dynamic walking. *International Journal of Robotics Research* 9(2):62–82.

McMahon, T. A. 1984. *Muscles, Reflexes, and Locomotion*. Princeton, USA: Princeton University Press.

Miyakoshi, S.; Taga, G.; Kuniyoshi, Y.; and Nagakubo, A. 1998. Three dimensional bipedal stepping motion using neural oscillators – towards humanoid motion in the real world –. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, 84–89.

Mori, T.; Nakamura, Y.; Sato, M.; and Ishii, S. 2004. Reinforcement learning for a CPG-driven biped robot. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, 623–630.

Morimoto, J.; Nakanishi, J.; Endo, G.; Cheng, G.; Atkeson, C. G.; and Zeglin, G. 2005. Poincare-map-based Reinforcement Learning for Biped Walking. In *Proceedings of 2005 IEEE International Conference on Robotics and Automation*, 2381–2386.

Nishiwaki, K.; Sugihara, T.; Kagami, S.; Kanehiro, F.; Inaba, M.; and Inoue, H. 2000. Design and Development of Research Platform for Perception-Action Integration in Humanoid Robot: H6. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00)*, 1559–1564.

Orlovsky, G. N.; Deliagina, T. G.; and Grillner, S. 1999. *Neuronal Control of Locomotion: From Mollusc to Man*. Oxford, UK: Oxford University Press.

Park, I.; Kim, J.; Lee, J.; and Oh, J. 2005. Mechanical Design of Humanoid Robot Platform KHR-3 (KAIST Humanoid Robot - 3: HUBO). In *Proceedings of the 2005 IEEE-RAS International Conference on Humanoid Robots (Humanoids2005)*, 321–326.

Peters, J.; Vijayakumar, S.; and Schaal, S. 2003. Reinforcement learning for humanoid robots – policy gradients and beyond. In *Proceedings of the third International Conference on Humanoid Robots (Humanoids2003)*. (CD-ROM).

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

Taga, G. 1995. A model of the neuro-musculo-skeletal system for human locomotion I. emergence of basic gait. *Biological Cybernetics* 73:97–111.

Tedrake, R.; Zhang, T. W.; and Seung, H. S. 2004. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04)*, 2849–2854.

Williamson, M. 1998. Neural control of rhythmic arm movements. *Neural Networks* 11(7–8):1379–1394.

Function Approximator for the Value Function and the Policy

We use a normalized Gaussian network (NGnet) (Doya 2000) to model the value function and the mean of the policy. The variance of the policy is modeled by a sigmoidal function (Kimura & Kobayashi 1998; Peters, Vijayakumar, & Schaal 2003). The value function is represented by the NGnet:

$$V(\mathbf{x}; \mathbf{w}^c) = \sum_{k=1}^{K} w_k^c b_k(\mathbf{x}), \tag{30}$$

where

$$b_k(\mathbf{x}) = \frac{\phi_k(\mathbf{x})}{\sum_{l=1}^{K} \phi_l(\mathbf{x})}, \phi_k(\mathbf{x}) = e^{-\|\mathbf{s}_k^T(\mathbf{x}-\mathbf{c}_k)\|}, \tag{31}$$

k is the number of the basis functions. The vectors $\mathbf{c}_k$ and $\mathbf{s}_k$ characterize the center and the size of the k-th basis function, respectively. The mean μ and the variance σ of the policy are represented by the NGnet and the sigmoidal function:

$$\mu_j = \sum_{i=1}^{K} w_{ij}^{\mu} b_i(\mathbf{x}), \tag{32}$$

and

$$\sigma_j = \frac{1}{1 + \exp(-w_j^{\sigma})}, \tag{33}$$

respectively. We assigned basis functions $\phi_k(\mathbf{x})$ at even intervals in each dimension of the input space ($-2.0 \leq \dot{\theta}_{roll}, \dot{\theta}_{pitch} \leq 2.0$). We used 225($= 15 \times 15$) basis functions for approximating the value function and the policy respectively.

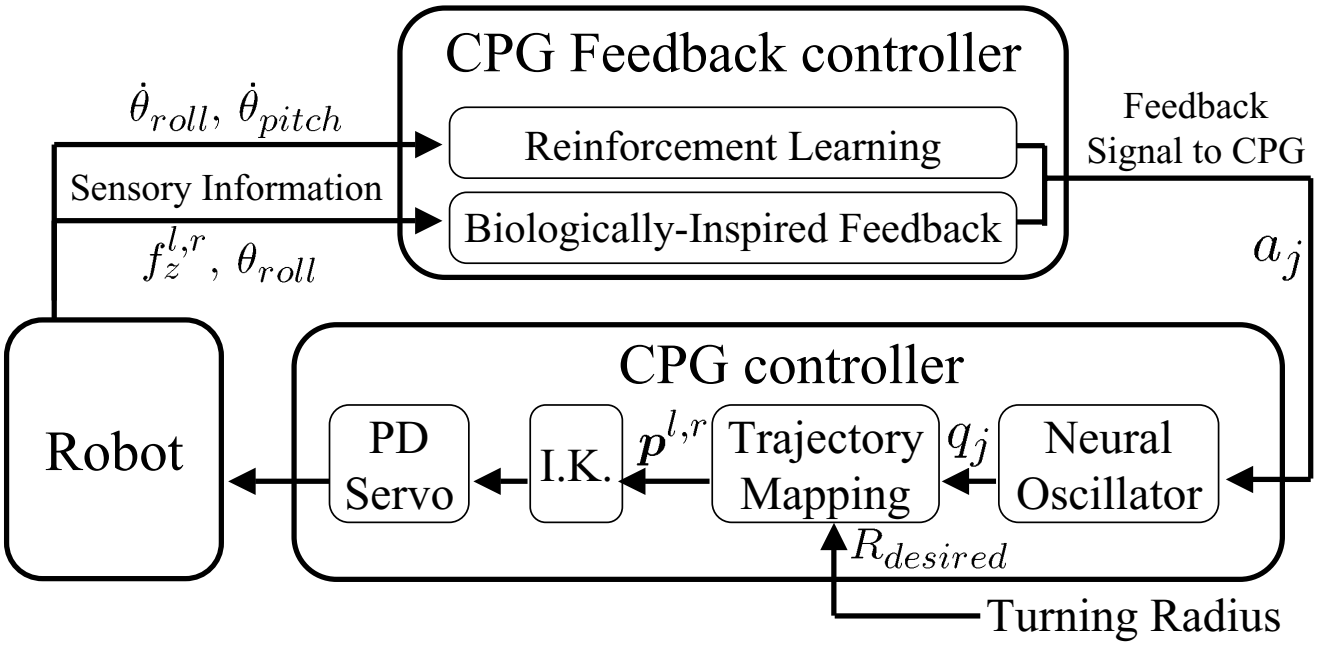


Figure 2: Basic framework of CPG control architecture. It consists of three parts: CPG controller, robot and CPG feedback controller. The CPG controller generates leg trajectory using a neural oscillator with a feedback signal. The leg trajectory is converted to joint torques via inverse kinematic calculation and PD servo. Then the robot interacts with an environment. The CPG feedback controller derives a feedback signal to CPG using incoming sensory information. a_j is a feedback signal to CPG and q_j is output of a neural oscillator. $R_{desired}$ is a parameter which specifies turning radius. $p^{l,r}$ indicates left/right leg position with respect to the body-fixed Cartesian coordinates. $\dot{\theta}_{roll}, \dot{\theta}_{pitch}$ are the angular velocity of the body in the roll and pitch direction, respectively. θ_{roll} is inclination angle in the direction of roll with respect to world coordinates. $f_z^{l,r}$ is the vertical reaction force of left/right leg.

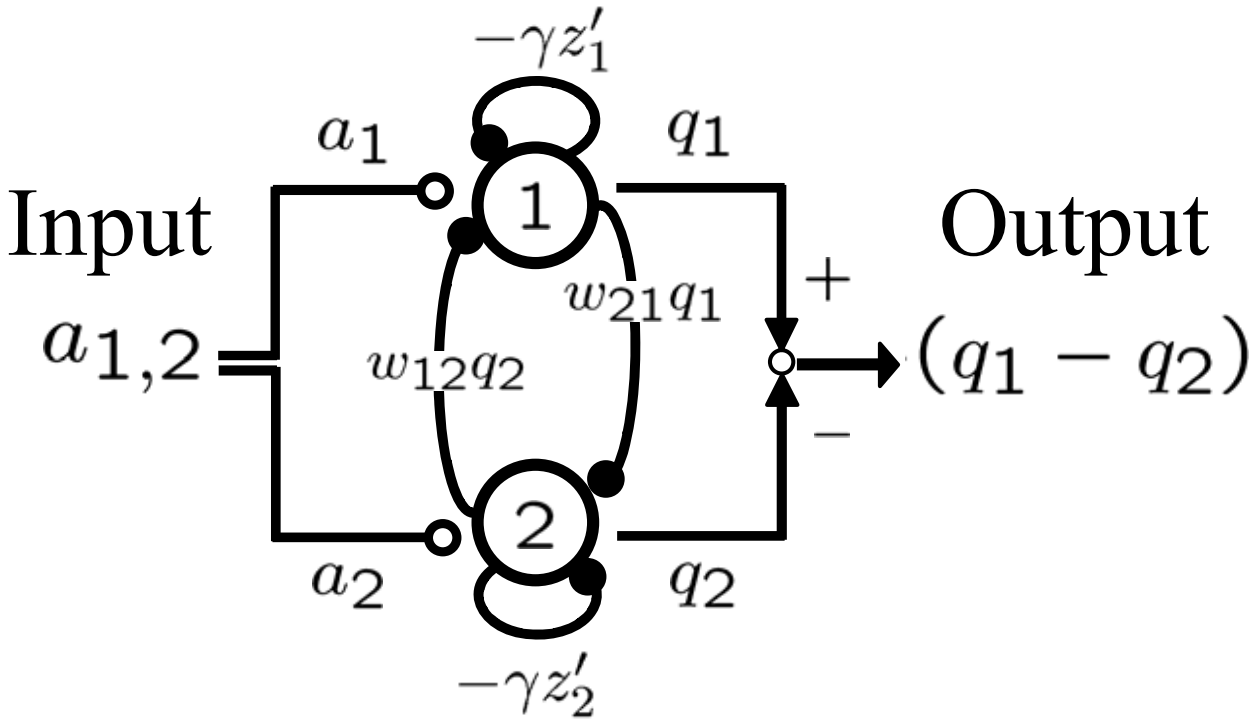


Figure 3: A schematic figure of a coupled neural oscillator: two neural units have mutual inhibition. $a_{1,2}$ and $q_{1,2}$ are input/output signals respectively. Line with a black circle and a white circle indicate inhibitory/excitory neural connection, respectively.

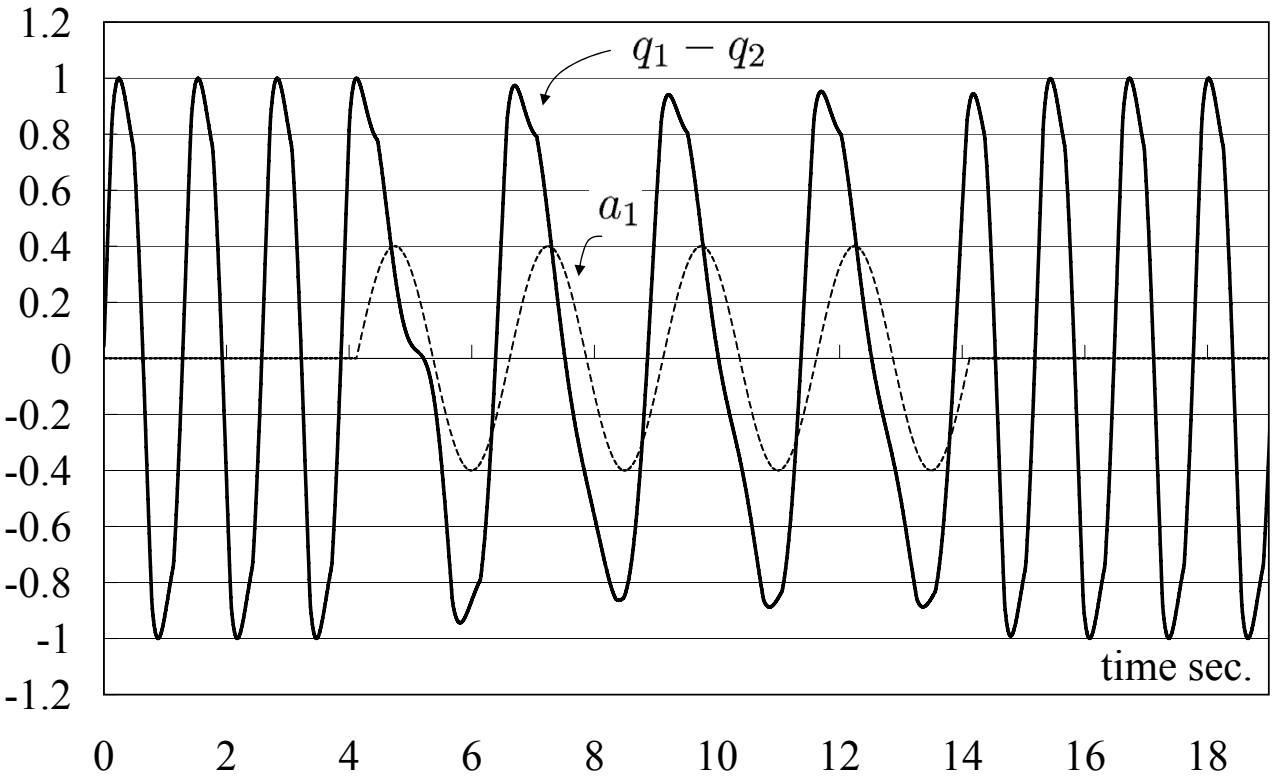


Figure 4: Entrainment property of a neural oscillator: a sinusoid input is fed into the oscillator at $4.1\,sec$ for 10 seconds. A dashed line, a_1 and a solid line, $(q_1 - q_2)$ are input/output signal of the neural oscillator, respectively. The parameters are $\tau_{CPG} = 0.224$, $\tau'_{CPG} = 0.280$, $\omega_{12} = \omega_{21} = 2.0$, $\gamma = 2.5$, $c = 2.36$ where the natural frequency of the oscillator and the amplitude are 0.775 and 1.0, respectively. The frequency of the input sinusoid is 0.4 whose amplitude is 0.4.

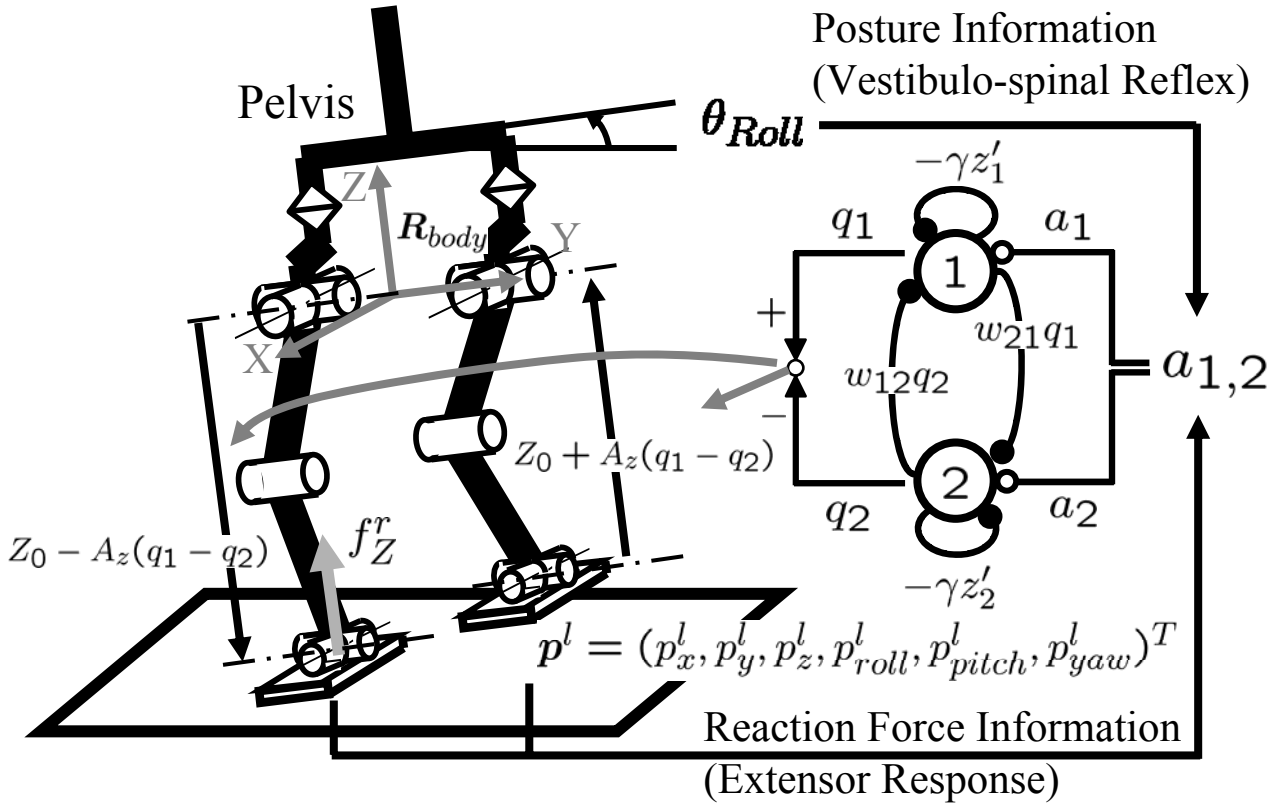


Figure 5: Neural oscillator allocation and biologically inspired feedback pathways for a stepping motion in place. The neural oscillator output, $(q_1 - q_2)$, symmetrically controls the left and right leg position in the vertical direction Z with respect to the body-fixed coordinates $\boldsymbol{R}_{body}$ where Z_0, A_x are an initial offset and a gain, respectively. The reaction force information in the Z direction, $f_Z^{l,r}$, is used as the extensor response and the posture inclination in the roll direction, θ_{Roll}, is used as the vestibulo-spinal reflex. $a_{1,2}$ are feedback signals derived by Eqn.(14)

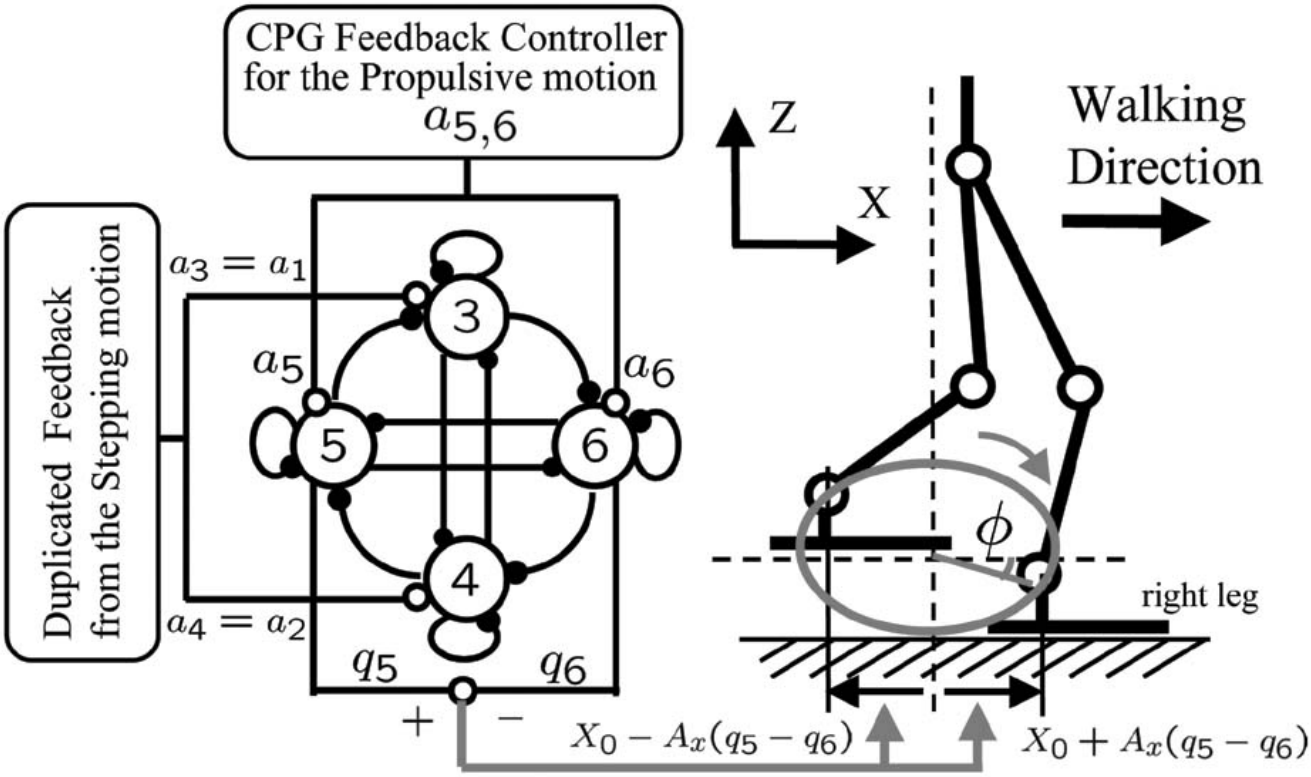


Figure 6: A quad-element neural oscillator for a propulsive motion in the sagittal plane. A schematic figure of a quad-element neural oscillator is shown on the left side and an ellipsoidal leg trajectory in the X-Z plane on the right side. The propulsive leg trajectory can be estimated by the shape of an ellipsoid. The ellipsoid can be express as $p_x^r = X_0 + A_x \cos\phi, p_z^r = Z_0 + A_z \sin\phi$, where ϕ is a parameter shown in this figure. Thus oscillatory movements in the X direction and Z direction needs a phase difference of $\pi/2$. A quad-element neural oscillator consists of two coupled oscillator with a uni-directional circular inhibitory neural connection. The oscillator output $(q_5 - q_6)$ has an inherent phase difference of $\pi/2$ with respect to the output $(q_3 - q_4)$. Since duplicated feedback from the stepping motion are fed into the neural units $(3, 4)$, the oscillator output $(q_5 - q_6)$ tends to keep the phase difference of $\pi/2$. The output $(q_5 - q_6)$ is used to move the left/right leg symmetrically in the X direction.

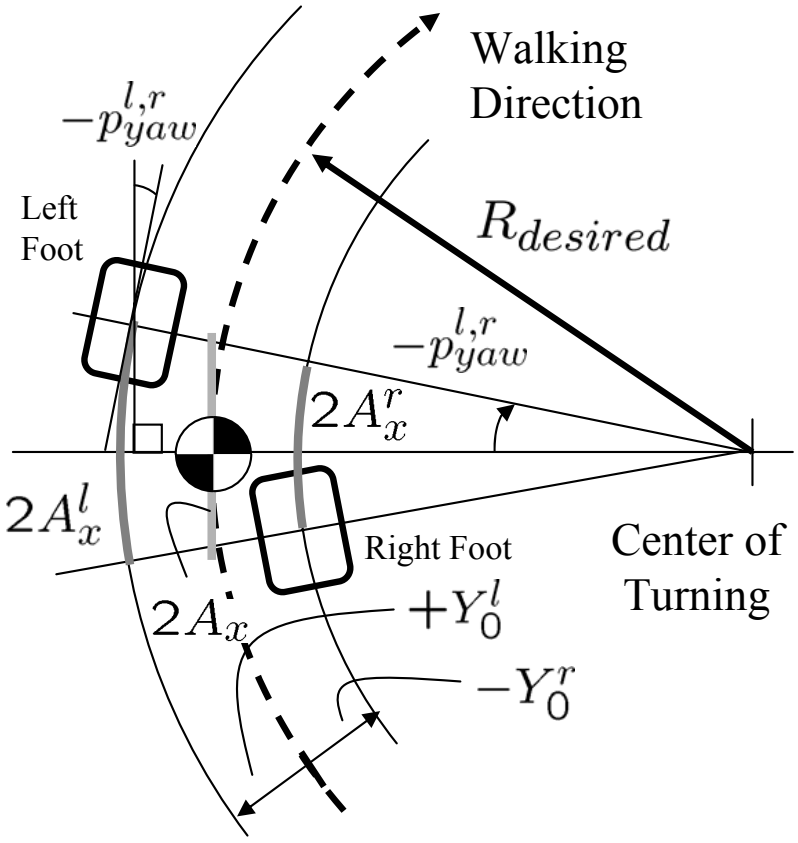


Figure 7: Foot prints captured in double support phase during a turning walk. Step length and leg yaw rotation are modulated with respect to a desired circular radius, $R_{desired}$. We assume that the origin of body-fixed coordinates moves along the desired circular arc (dashed arc). $2A_x$ is step length for a straight walk and $2A_x^l$, $2A_x^r$ are modulated step length in order to satisfy kinematic constraints without slippage due to lateral leg position offset Y_0^l, Y_0^r. $-p_{yaw}^{l,r}$ indicates the angle of necessary yaw rotation.

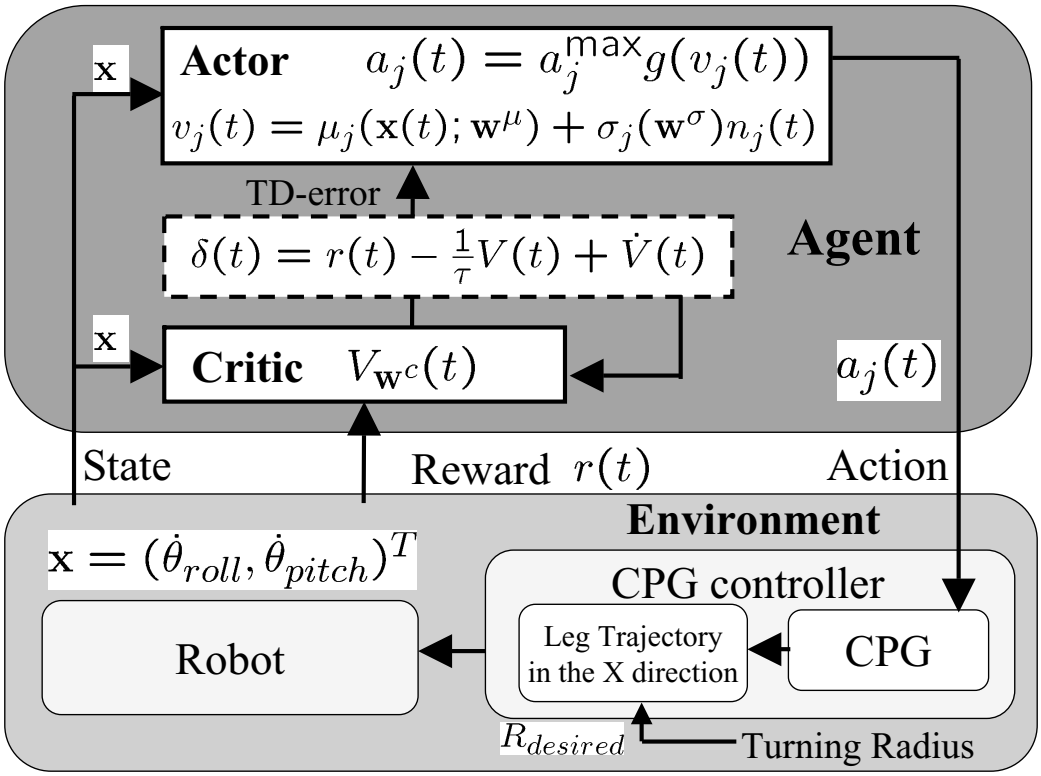


Figure 8: Schematic diagram of CPG feedback learning. $a_j(t)$ is the feedback signal derived with a policy gradient method. $\mathbf{x}$ is the input states used for learning, where $\dot{\theta}_{roll}$, $\dot{\theta}_{pitch}$ are the pelvis angular velocity. Critic estimates the value $V_{\mathbf{w}^c}(t)$, where $\mathbf{w}^c$ is the parameter of the function approximator. $\delta(t)$ is a TD-error in continuous time and space and $v_j(t)$ is a stochastic policy defined by a probability distribution, where $\mathbf{w}^\mu$, $\mathbf{w}^\sigma$ are parameter vectors of the policy and $n_j(t)$ is a normal distribution.

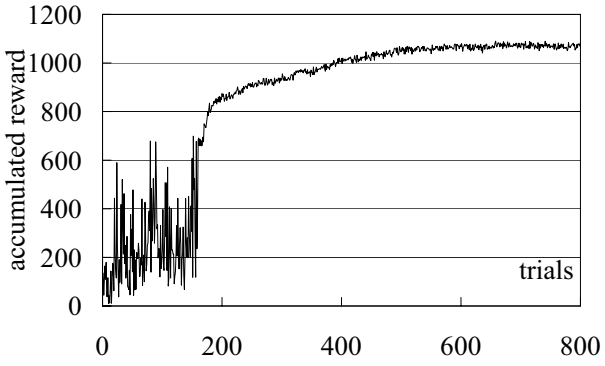


Figure 9: Typical learning curve. A steady walking without falling over is acquired after 180 trials in this example.

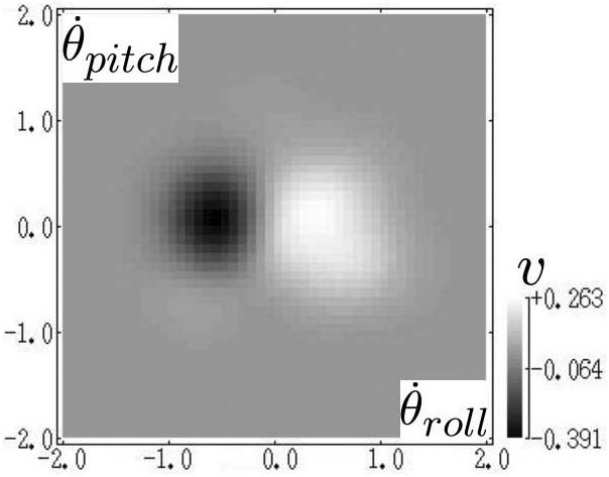


Figure 10: Typical learned policy. Gray scale shows acquired stochastic policy where the horizontal and vertical axes are input states used for learning $\dot{\theta}_{roll}$, $\dot{\theta}_{pitch}$, respectively.

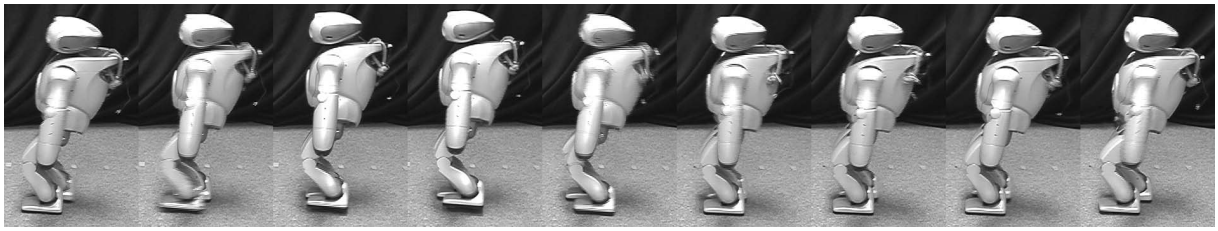


Figure 11: Snapshots of straight steady walking with acquired feedback controller ($A_x = 0.015\,m$, $A_z = 0.005\,m$, $v_x = 0.077\,m/s$. Photos were captured every 0.1 $sec.$)

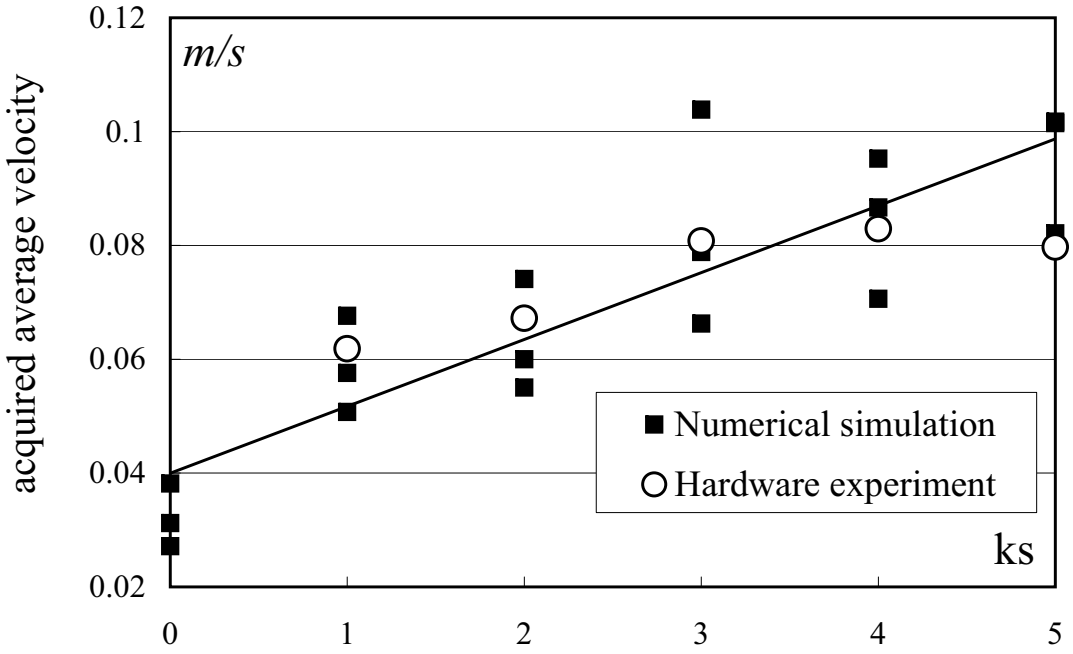


Figure 12: The relationship between acquired average velocity and velocity reward k_S. Each data point used different policy and average velocity was derived by averaging steady walking velocity for ten seconds. A solid line shows linear approximation for all data points.

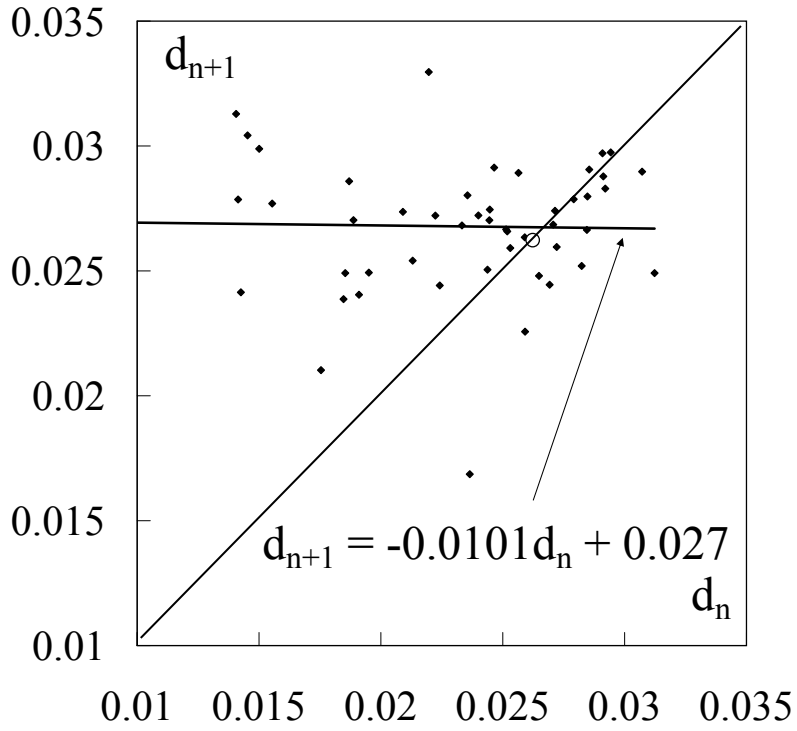


Figure 13: Return map of step length. We captured step length when left leg touched down. $\mathbf{d}_n$, $\mathbf{d}_{n+1}$ are step length right after the perturbation and the next step, respectively. A solid line indicated by an arrow is linear approximation for all data points and a diagonal linear line represents the identity map.

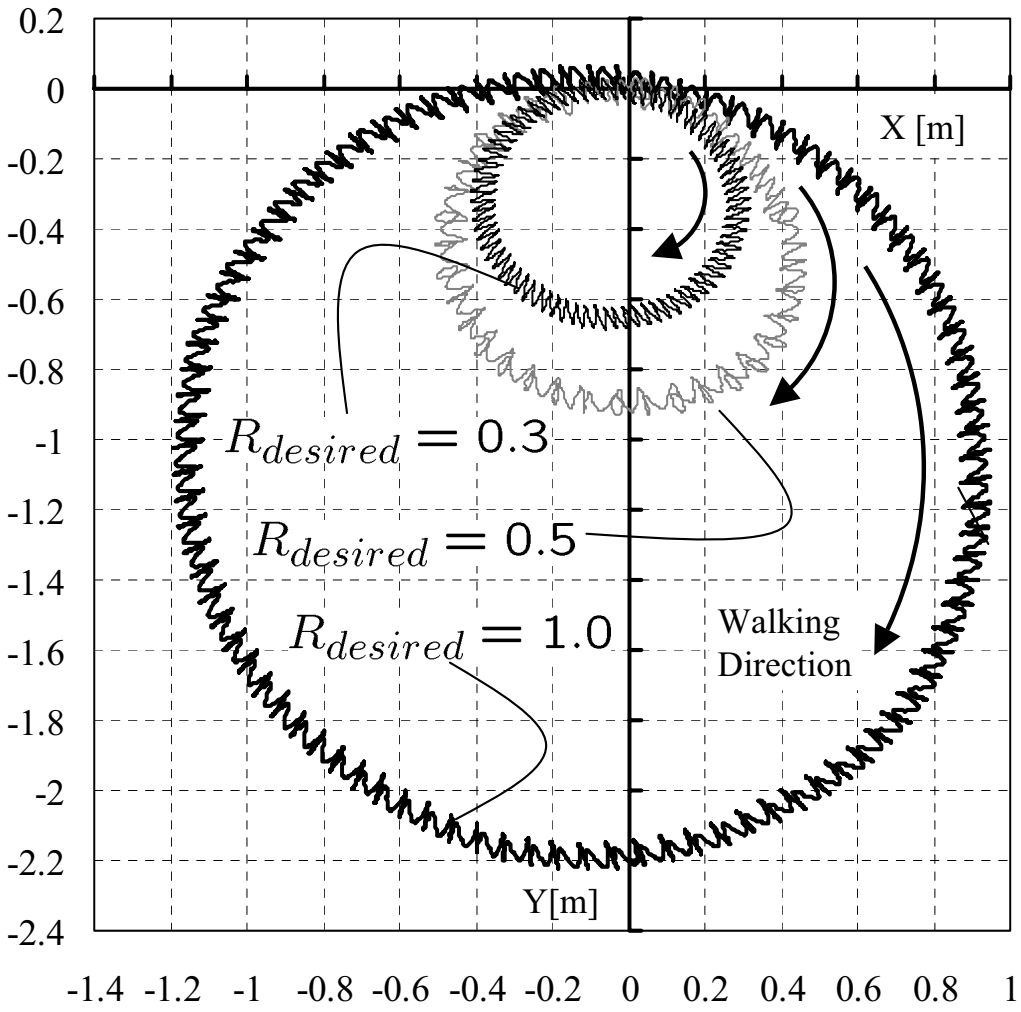


Figure 14: Trajectories of center of mass with different $R_{desired}$. The robot started walking at $(X, Y) = (0, 0)$ in the direction of positive X axis.

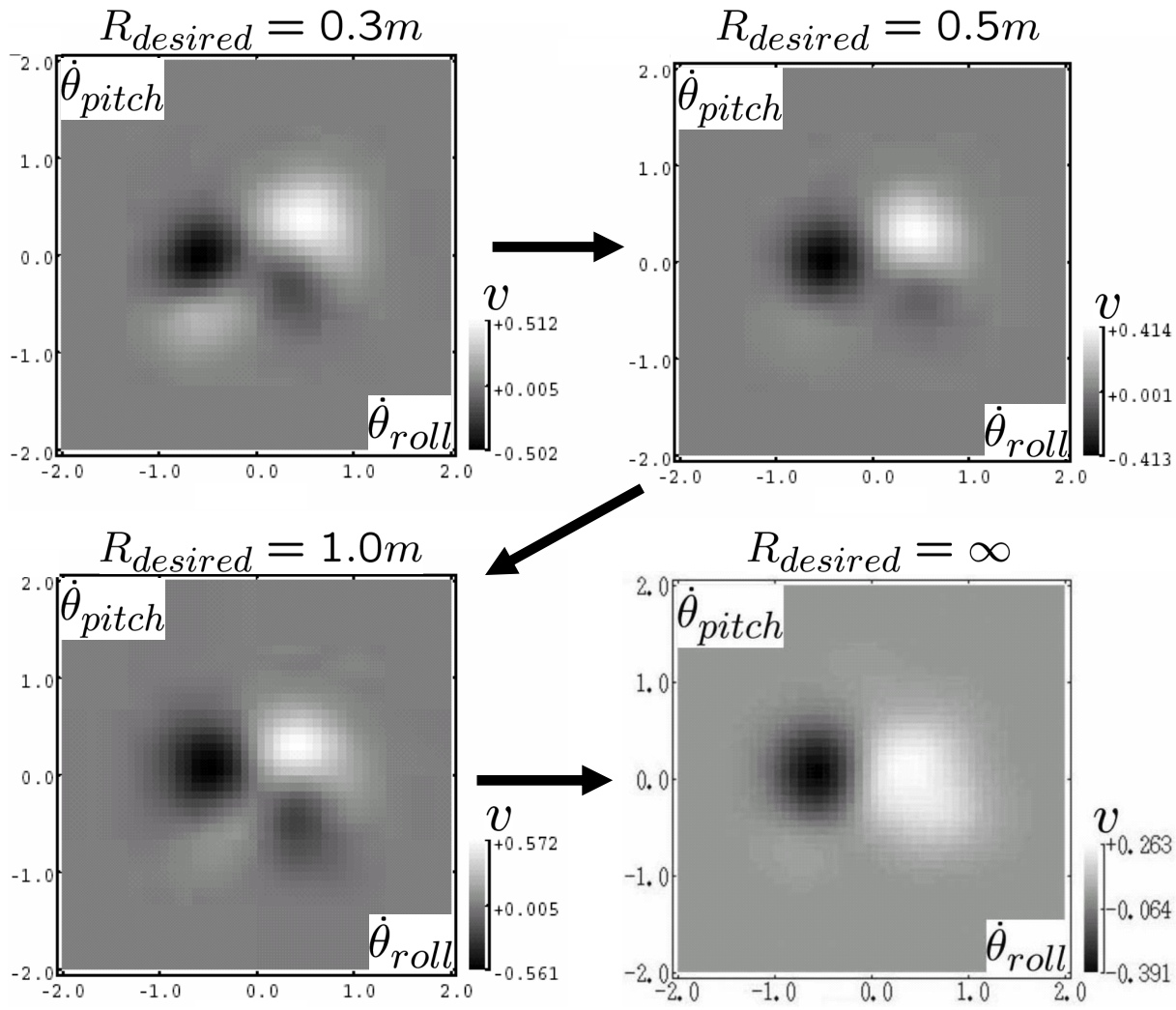


Figure 15: Typical acquired policy with different $R_{desired}$. The smaller $R_{desired}$ utilizes $\dot{\theta}_{pitch}$ information more compared with the straight walking case.

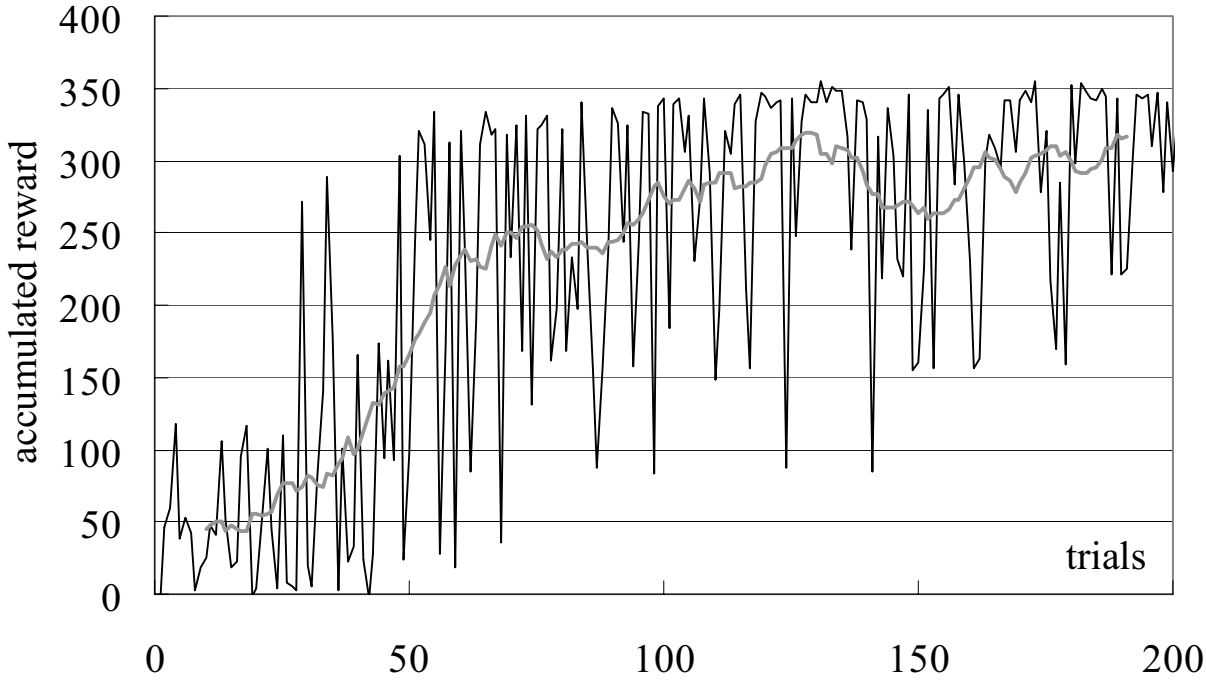


Figure 16: Additional online learning for straight walking. The black and gray line are accumulated reward for one trial and running average of accumulated reward for 20 trials, respectively.

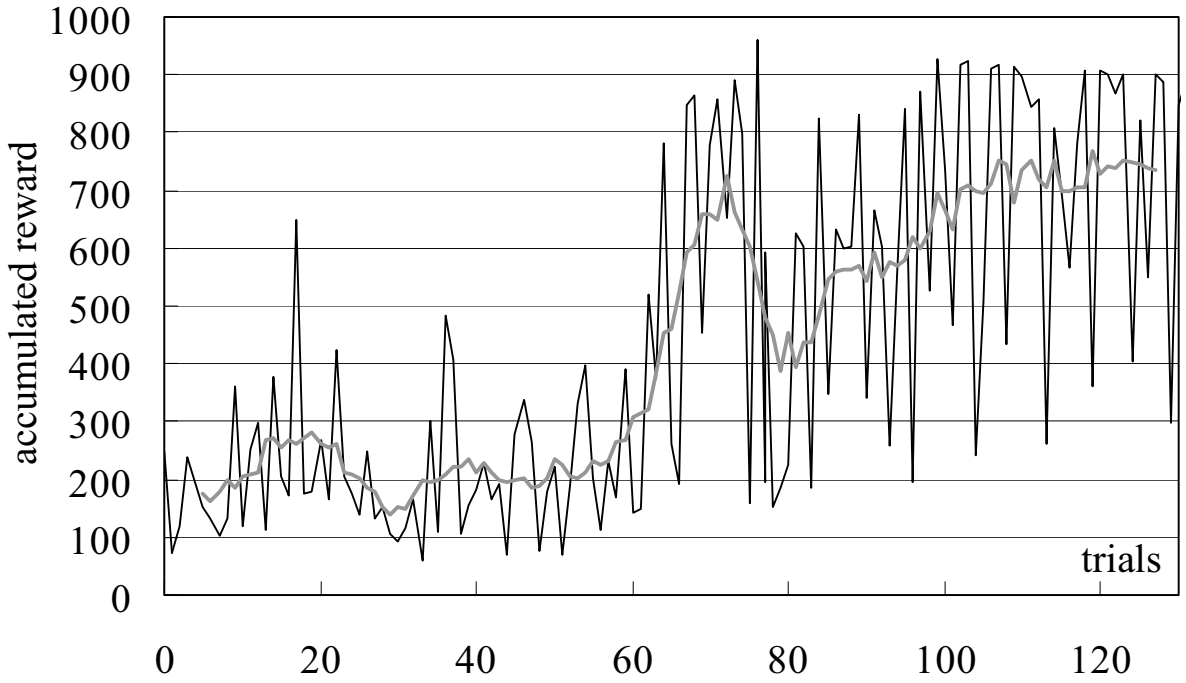


Figure 17: Additional online learning for circular walking. The black and gray line are accumulated reward for one trial and running average of accumulated reward for 20 trials, respectively.