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

Article *in* The International Journal of Robotics Research · February 2008

DOI: 10.1177/0278364907084980 · Source: DBLP

5 authors, including:

G. Endo
Tokyo Institute of Technology
358 PUBLICATIONS **3,423** CITATIONS
SEE PROFILE

Takamitsu Matsubara
Nara Institute of Science and Technology
190 PUBLICATIONS **2,131** CITATIONS
SEE PROFILE

Gordon Cheng
Technische Universität München
419 PUBLICATIONS **10,007** CITATIONS
SEE PROFILE

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

Gen Endo
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku
Tokyo, 152-8550, Japan
gendo@sms.titech.ac.jp

Jun Morimoto
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
xmorimo@atr.jp

Takamitsu Matsubara
ATR Computational Neuroscience Laboratories
2-2-2 Hikaridai, Seika-cho, Soraku-gun
Kyoto, 619-0288, Japan
and
Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi
Nara, 630-0192, Japan
takam-m@atr.jp

Jun Nakanishi
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
jun@atr.jp

Gordon Cheng
ATR Computational Neuroscience Laboratories
ICORP, Japan Science and Technology Agency
2-2-2 Hikaridai, Seika-cho, Soraku-gun
Kyoto, 619-0288, Japan
gordon@atr.jp

Abstract

In this paper we describe 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 controller obtained in numerical simulation achieves stable walking with a physical robot in the real world. Numerical simulations and hardware experiments evaluate the 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 an initial policy for a hardware robot to improve the controller within 200 iterations.

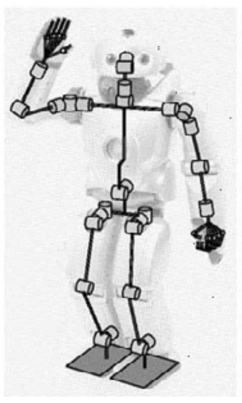
KEY WORDS—humanoid robots, reinforcement learning, bipedal locomotion, central pattern generator

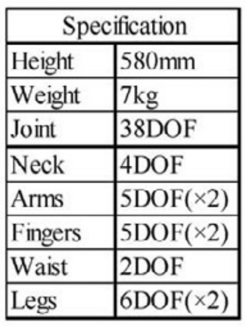
1. Introduction

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). Many of the 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 a broad range of applications working within unknown environments, equipping humanoid robots with a learning capability provides a promising avenue of research. 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, owing to the dimensionality explosion that is bound to occur. 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). See Figure 1.

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





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

Fig. 1. The hardware platform of the full-body humanoid robot: joint configuration and its specification.

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 properties in their joints and muscles, it is extremely difficult with the current hardware technology to design general humanoid robots that have both passive dynamic properties 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 walking. Moreover, developing a specific humanoid hardware with unifunctionality, for example walking, may lose other important features of the humanoid robot such as versatility and the capability to achieve 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 et al. 1999; Cohen 2003). It is known that during locomotion a feed-forward excitation to the muscles exists that can be independent of sensory feedback and brain input (Grillner et al. 1995). The feed-forward 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, owing to the entrainment property of the CPG (Taga 1995; Miyakoshi et al. 1998). However, designing appropriate feedback pathways in neural oscillators often requires much effort to manually tune the parameters of the oscillator. Thus, a genetic algorithm (Hase and 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 iterations to obtain a solution owing 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 owing to the 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 and 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), owing to the lack of information, which yields large variances in the estimated gradient of expected reward with respect to the policy parameters (Sutton et al. 2000; Konda and Tsitsiklis 2003). However, in the proposed framework, when the CPG and the mechanical system of the robot converge to a periodic trajectory owing to entrainment, the internal states of the CPG and the states of the robot will be synchronized. Thus, by using the state space that is only composed of the observables, thus reducing 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 five-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. As a 3D full-body humanoid robot has many degrees of freedom, the dynamics of the 3D humanoid is much more complicated than a 2D 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

In this section we describe 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. Figure 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 that specifies a turning radius defined on the ground to generate circular walking. R_{desired} modulates leg position, $\boldsymbol{p}^{\text{l,r}}$, based on geometrical constraints to walk along a specified circular walking trajectory. Then, the desired joint position for the joint PD servo is obtained through inverse kinematics and is used to control an actuator.

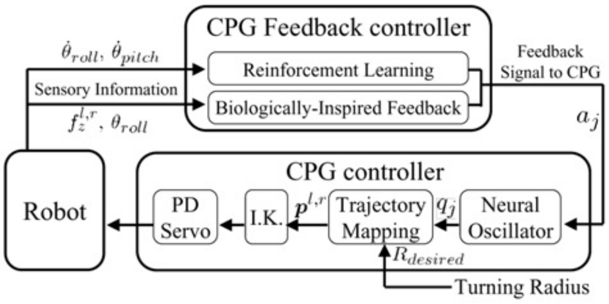


Fig. 2. Basic framework of CPG control architecture. It consists of three parts: CPG controller, robot and CPG feedback controller. The CPG controller generates the leg trajectory using a neural oscillator with a feedback signal. The leg trajectory is converted into joint torques via inverse kinematic calculation and PD servo. Then the robot interacts with the environment. The CPG feedback controller drives a feedback signal to CPG using incoming sensory information. The following notation is used: a_j is a feedback signal to CPG and q_j is the output of a neural oscillator; R_{desired} is a parameter that specifies the turning radius; $p^{\text{l,r}}$ indicates left/right leg position with respect to the body-fixed Cartesian coordinates; $\dot{\theta}_{\text{roll}}$, $\dot{\theta}_{\text{pitch}}$ 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; and $f_z^{\text{l,r}}$ is the vertical reaction force of left/right leg.

The CPG feedback controller generates the feedback signal to the 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 feedback arranged for the Z (vertical) direction. Finally, a_j is fed back to the neural oscillator which automatically adjusts its output due to the 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 sections.

2.1. Neural Oscillator Model

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

In the learning framework we present in this paper, we propose to change 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 (1985). The Matsuoka oscillator has a number of beneficial properties, most notably, it allows modulations of sensory feedback for adaptation to the environment and control of the amplitude with a single scale factor. These are some of the well-studied aspects of the Matsuoka oscillator, making it suitable for our investigation.

The oscillator dynamics of jth neural unit are

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

$$\tau'_{\text{CPG}}\dot{z}'_j = -z_j + q_j, \quad (2)$$

$$q_j = \max(0, z_j), \quad (3)$$

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

Figure 3 shows a schematic of a coupled oscillator where two neural units are connected by mutual inhibitions. The circles with numbers represent neural units whose dynamics are defined by (1)–(3). Lines ending with black or white circles indicate inhibitory or excitory neural connections, respectively.

The properties of the Matsuoka neural oscillator model have been explored numerically, 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 $\tau_{\text{CPG}}/\tau'_{\text{CPG}}$ is kept constant, the natural frequency of the oscillator is proportional to $1/\tau_{\text{CPG}}$. The tonic input c controls the amplitude of the output of the oscillator. It is demonstrated that the 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.

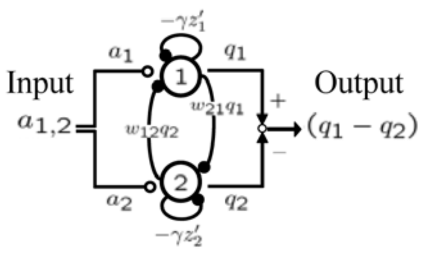


Fig. 3. Schematic of a coupled neural oscillator: two neural units have mutual inhibition; $a_{1,2}$ and $q_{1,2}$ are input/output signals, respectively. Lines ending with black or white circles indicate inhibitory or excitory neural connections, respectively.

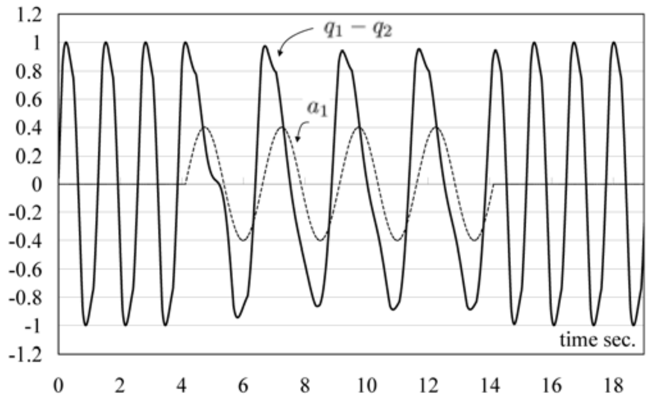


Fig. 4. Entrainment property of a neural oscillator: a sinusoid input is fed into the oscillator at 4.1 s for 10 s. The dashed, a_1, and solid, $(q_1 - q_2)$, lines are the input and output signal of the neural oscillator, respectively. The parameters are $\tau_{\text{CPG}} = 0.224$, $\tau'_{\text{CPG}} = 0.280$, $\omega_{12} = \omega_{21} = 2.0$, $\gamma = 2.5$, $c = 2.36$, 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 and the amplitude is also 0.4.

Figure 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 s for 10 s. 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. Figure 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 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 and Yamazaki 1997; Ishiguro et al. 2003). However, it is difficult to obtain appropriate feedback pathways for all of the oscillators to achieve the desired behavior with the increase in the number of degrees of freedom of the robot because neural oscillators are intrinsically non-linear. Moreover, the realization of precise torque control of each joint is also difficult 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 six 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 has been empirically demonstrated in our previous studies (Endo et al. 2004, 2005b).

Figure 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:

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

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

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

For propulsive motion in the sagittal plane, we introduce a quad-element neural oscillator to produce coordinated leg movements with a stepping motion based on the following observation: as illustrated in Figure 6, when the robot is walking forward, the leg trajectory with respect to the body coordinates in the sagittal plane can be roughly approximated with the shape of an ellipsoid. Suppose that the output trajectories of the oscillators can be approximated as $p_x^{\mathrm{l}} = A_x \cos(\omega t + \alpha_x)$ and $p_z^{\mathrm{l}} = A_z \cos(\omega t + \alpha_z)$, respectively. Then, to form the ellipsoidal trajectory on the X–Z plane, p_x^{l} and p_z^{l} need to satisfy the relationship $p_x^{\mathrm{l}} = A_x \cos\phi$ and $p_z^{\mathrm{l}} = A_z \sin\phi$, where ϕ is the angle defined in Figure 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 an inherent phase difference of $\pi/2$ between two coupled oscillators, $(q_3 - q_4)$ and $(q_5 - q_6)$ (see 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].

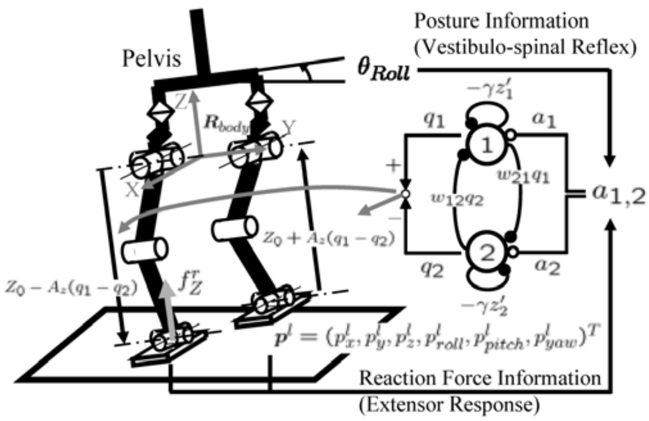


Fig. 5. Neural oscillator allocation and biologically inspired feedback pathways for a stepping motion in place. The neural oscillator output, $(q_1 - q_2)$, symmetrically controls the left and right leg position in the vertical direction Z with respect to the body-fixed coordinates $\boldsymbol{R}_{\mathrm{body}}$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

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:

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

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

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

1. At the beginning of the investigation, we directly used (q_1, q_2) for a quad-element oscillator. However, the oscillator dynamics of the stepping motion interfered with the dynamics of propulsive motion via uni-directional circular inhibitions. As a result, the 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 behavior similar to (q_1, q_2).

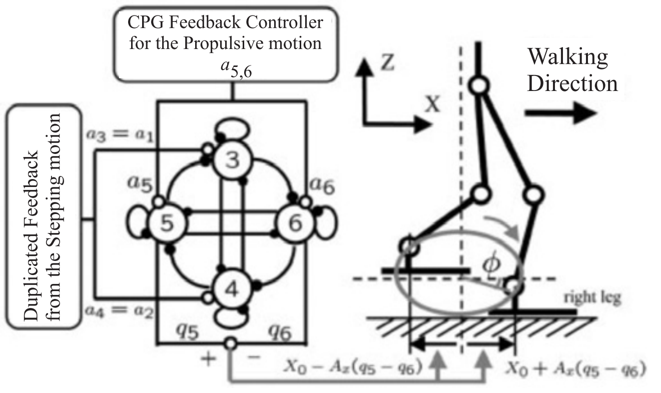


Fig. 6. A quad-element neural oscillator for propulsive motion in the sagittal plane. A schematic of a quad-element neural oscillator is shown on the left-hand side and an ellipsoidal leg trajectory in the X–Z plane is shown on the right-hand side. The propulsive leg trajectory can be estimated with the shape of an ellipsoid. The ellipsoid can be expressed as $p_x^{\mathrm{r}} = X_0 + A_x \cos\phi$, $p_z^{\mathrm{r}} = Z_0 + A_z \sin\phi$, 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 oscillators 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 $(q_3 - q_4)$. As duplicated feedback from the stepping motion is 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.

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

2.3. Turning Controller

We introduce an additional mechanism to control the walking direction by modulating the right and left step length as well as the 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. The desired circular radius R_{desired} modulates the mapping from a CPG output q_j to leg position $\boldsymbol{p}^{\mathrm{l,r}}$.

As illustrated in Figure 7, we assume that the origin of body-fixed coordinates moves with constant velocity along a particular circular arc defined by R_{desired} ($R_{\text{desired}} > 0$, when the robot makes a right turn), and left and right stance legs also move along the concentric circular arcs defined by $R_{\text{desired}} + Y_0^{\mathrm{l}}$ and $R_{\text{desired}} - Y_0^{\mathrm{r}}$, where $Y_0^{\mathrm{l,r}}$ is leg position offset in the lateral direction. In order to satisfy kinematic constraints without

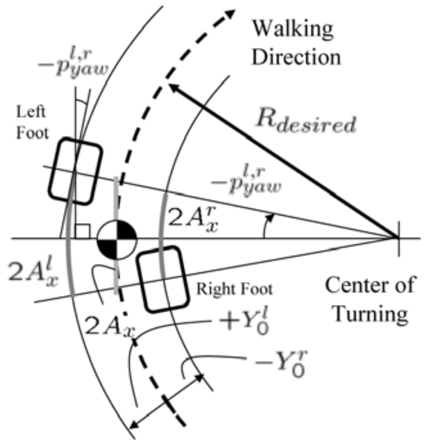


Fig. 7. Foot prints captured in double support phase during a turning walk. Step length and leg yaw rotation are modulated with respect to a desired circular radius, R_{desired}. We assume that the origin of body-fixed coordinates moves along the desired circular arc (dashed arc). We use $2A_x$ to denote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

slippage of the stance leg, the inner step length should be decreased while the outer step length should be increased due to leg position offset $Y_0^{\mathrm{l,r}}$ ($|Y_0^{\mathrm{l,r}}| < |R_{\text{desired}}|$):

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

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

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

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

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

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

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

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 possesses an entrainment property that to some extent could better cope with the effect of dynamics. In addition, 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 have been found that 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 and Boothe 1999). It is generally known that the vestibular system measures the body's inclination and activates contralateral muscles to keep upper body stabilized. This is one of the basic posture controls in humans and is called the *vestibulospinal reflex*. The effectiveness of these feedback pathways was experimentally demonstrated with a hardware quadruped robot (Kimura et al. 2007) to maintain the balance of the body when walking over unknown terrain.

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

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

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

We then introduce the extensor response and vestibulospinal reflex by adjusting the leg length according to the vertical reaction force or the inclination of the body as follows:

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

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

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 (Figure 6).

For the propulsive leg movements in the X direction, the feedback signal a_j is 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 is defined in the following section). The stochastic policy is estimated with a probability distribution $\pi_{\mathbf{w}^a}(\mathbf{x}, v_j) = P(v_j \mid \mathbf{x}; \mathbf{w}^a)$:

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

where $\mathbf{x}$ denotes partial states of the robot, and $\mathbf{w}^a = [(\mathbf{w}^\mu)^{\text{T}}, (\mathbf{w}^\sigma)^{\text{T}}]$ 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, $n_j(t) \sim N(0, 1)$. Here $N(0, 1)$ is a normal distribution which has a mean $\mu = 0$ and a variance $\sigma^2 = 1$. In the next section, we discuss the learning algorithm that we use to acquire a feedback controller $a_{j=5}$ for the propulsive motion.

3. Learning the Sensory Feedback Controller

Promising results have been demonstrated in applications of the policy gradient technique to POMDP biped walking locomotion (Tedrake et al. 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 2D biped walking task. The policy gradient method could acquire steady walking with 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 for the sensory feedback controller of the neural oscillator model in the X direction.

Humanoid robots have many degrees of freedom with a large 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 a reasonable number of state variables for learning. 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 $\mathbf{x} = (\dot{\theta}_{\text{roll}}, \dot{\theta}_{\text{pitch}})^{\text{T}}$ as the input state to the learning algorithm, and $\dot{\theta}_{\text{roll}}$, $\dot{\theta}_{\text{pitch}}$ are measured with gyro sensors located on the pelvis of the hardware. Therefore, the other states of the robot, such as inclinations of the pelvis, the linear velocity with respect to world coordinates, and the 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 Figure 8. First, the CPG controller generates leg trajectory in the X direction, allowing the robot to interact 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 the temporal difference (TD) error, represented in continuous time and space. Then, an actor generates a CPG feedback signal for the 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 Section 3.2 for further details).

In the following sections, we introduce the definition of the value function in continuous time and space to derive the TD error (Doya 2000). Then, we discuss the learning method of a policy for the sensory feedback controller. We use a normalized Gaussian network (NGnet) to approximate both the value function and the policy (see the 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,

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

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

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

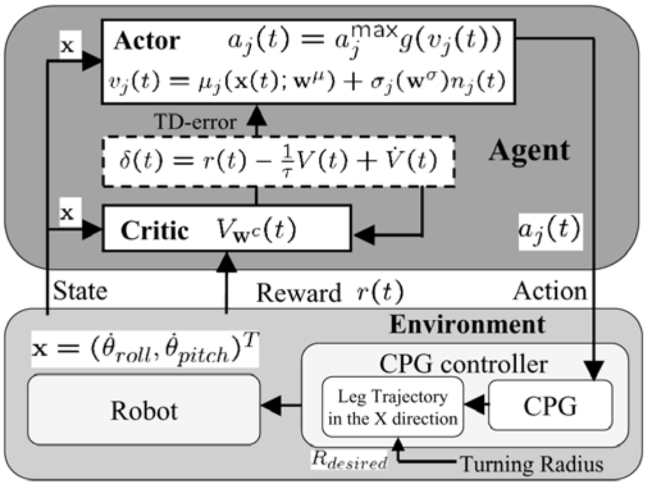

Actor $a_j(t) = a_j^{\max} g(v_j(t))$
$v_j(t) = \mu_j(\mathrm{x}(t); \mathrm{w}^\mu) + \sigma_j(\mathrm{w}^\sigma) n_j(t)$
x
TD-error
$\delta(t) = r(t) - \frac{1}{\tau}V(t) + \dot{V}(t)$
Agent
x
Critic $V_{\mathbf{w}^c}(t)$
$a_j(t)$
State
Reward $r(t)$
Action
Environment
$\mathrm{x} = (\dot{\theta}_{roll}, \dot{\theta}_{pitch})^T$
CPG controller
Robot
Leg Trajectory in the X direction
CPG
$R_{desired}$
Turning Radius

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

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

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

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

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

We denote the current estimate of the value function as $V(\mathbf{x}^{\text{all}}(t)) = V(\mathbf{x}^{\text{all}}(t); \mathbf{w}^c)$, 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).

As we consider a learning framework in POMDPs, i.e. we only observe partial states $\mathbf{x}$ from all states $\mathbf{x}^{\text{all}}$, the TD error usually does not converge to zero. However, Kimura and

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. The eligibility trace is used to assign credit for the TD error backwards 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

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

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

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 the following intuitions. For the learning rate, α, we try to maximize the learning rate that is small enough to estimate the expectation in (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 a smaller value than the actual value. We set a sufficiently large value for this parameter to take into account the negative reward that is caused when the robot falls over. The time constant κ^c controls credit assignment of the reward to the previously visited state. As we consider a partially observable environment, it is important to assign the credit from an actual acquired reward, which does not depend on the estimated value function. However, too large a 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

Kimura and 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

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

where

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

$\mathbf{w}^a$ 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. The eligibility trace is used to assign credit for the TD error backwards 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

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

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

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

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

The basic intuition for updating the policy with TD error and the eligibility trace is as follows: larger TD error indicates that the generated action gave a better result, i.e. acquired a larger reward and/or achieved a state that has a larger estimated value than the expected value. To increase the chances of acquiring a larger reward, we can increase the policy parameters that contribute to increasing the TD error. The eligibility traces represent the contribution of each policy parameter.

We manually tune β, κ^a based on the following intuitions. For a learning rate β, we try to maximize the learning rate that is small enough to estimate the expectation in (25) by averaging the sample data (using a small learning rate has the same effect as averaging the sample data). The time constant κ^a controls the credit assignment of the reward to the previously generated action. As 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, too large a κ^a leads to a large variance in the gradient estimation. In this study, we set the learning parameters to $\beta^\mu = \beta^\sigma = 195, \kappa^\mu = 1.0, \kappa^\sigma = 0.1$.

3.3. Rewards

We design a reward function:

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

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 as the first term, while at the same time to achieve forward progress with the second term. In this study, the parameters were chosen as k_S in the range 0.0–10.0, $k_H = 10.0$, $h' = 0.272$, where the unit for h_1 and h' is meters and for v_x is meters per second. 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 has 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 ms and the learning was performed at 16 ms interval. The total computation including learning algorithm required 3.8 times longer than real time.

4.2. Simulation Results and Hardware Verifications

First, we carried out numerical experiments using the dynamics simulator to acquire a feedback controller (policy) for the CPG for biped walking. Then, 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 utilized 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 switched 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 s. 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 cases where an additional 3,000 trials were done after successful acquisition or 5,000 trials even without successfully acquiring a policy.

As expected, at the beginning of the 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 rewards, k_S, walking motion was successfully acquired for 20 experiments (Table 1); we could not observe policy convergence for seven failed experiments. The required trials are the averaged value of each successful experiment with the same k_S. Typically, it takes 20 h to run one simulation for 1,000 trials and the policy was acquired on average after 696 trials. Figures 9 and 10 show a typical example of accumulated reward at each trial and an acquired policy, respectively. Figure 10 shows that while $\dot{\theta}_{\text{roll}}$ dominates the policy output, $\dot{\theta}_{\text{pitch}}$ does not assert much influence. The reason is that $\dot{\theta}_{\text{roll}}$ is always being generated by the stepping motion regardless of the propulsive motion. Therefore, the policy tries to utilize $\dot{\theta}_{\text{roll}}$ to generate synchronized leg movements. We also observed that $\dot{\theta}_{\text{pitch}}$ is suppressed by the reward function because $\dot{\theta}_{\text{pitch}}$ lowers the pelvis height by a pitching oscillation, which can cause falling.

We transferred the acquired 20 successful learning processes onto the robot by using a series of policies with different trials but the same learning process. We then used these

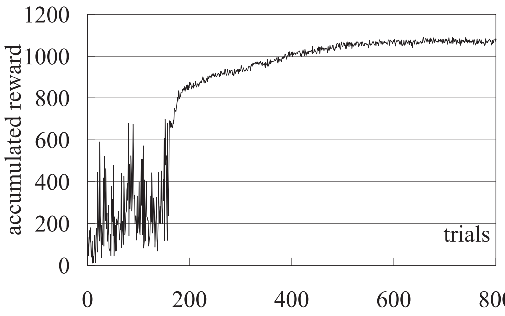


Fig. 9. Typical learning curve. Steady walking without falling over is acquired after 180 trials in this example.

Table 1. Achievements in the acquisition of straight walking.

		Number of achievements	
k_S	Number of experiments	Simulation (trials)	Hardware (trials)
0.0	4	1 (2,385)	0 (—)
1.0	3	3 (528)	3 (1,600)
2.0	3	3 (195)	1 (800)
3.0	4	4 (464)	2 (1,500)
4.0	3	2 (192)	2 (350)
5.0	5	2 (1,011)	1 (600)
10.0	5	5 (95)	5 (460)
Sum (average)	27	20 (696)	14 (885)

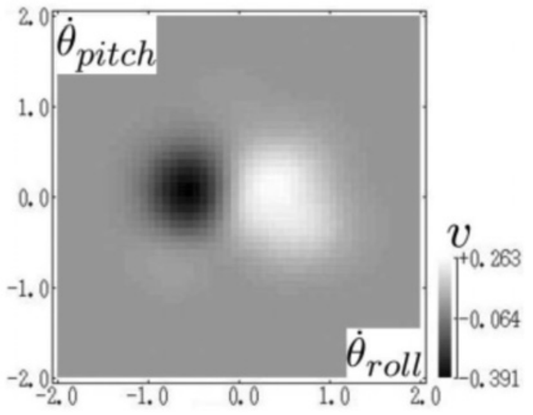


Fig. 10. Typical learned policy. The gray scale shows the acquired stochastic policy where the horizontal and vertical axes are input states used for learning $\dot{\theta}_{\text{roll}}$ and $\dot{\theta}_{\text{pitch}}$, respectively.

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 m without falling over. We verified improvements of the policy in accordance with the 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 the successful learning processes in the numerical experiments performed successful walking on the hardware (Table 1). Figure 11 shows snapshots of an acquired walking pattern. The six policies, which could not succeed on the hardware experiment, had similar profiles to a typical policy shown in Figure 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 $\dot{\theta}_{\text{roll}}$ was across zero. Consequently, the leg movement exceeded the current limit of the actuator of the robot, which was not modeled in the dynamics simulator, and the robot fell over. The policy with slightly smaller output can be improved via online learning, which is discussed in the following section.

In our experiments, an 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 $+3^\circ$ to -4° inclination, suggesting sufficient walking stability.

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 (1) and (2) to $\tau_{\text{CPG}} = 0.105$, $\tau'_{\text{CPG}} = 0.132$, $c = 2.08$, $\gamma = 2.5$ to generate an inherent oscillation, where its amplitude is 1.0 and period is 0.8. As 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^{-1}.

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

This result shows that 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 fixed point using a return map. We perturbed the target trajectory on $(q_5 - q_6)$ to change the step length at random timings during steady walking, and captured the states of the robot when the left leg touched down. We measured two steps, just after the perturbation ($\mathbf{d}_n$) and the next step ($\mathbf{d}_{n+1}$). If acquired walking motion is locally stable, the absolute eigenvalue of the return map should be less than 1.

Figure 13 shows the return map with 50 data points with a white dot indicating a fixed point derived by averaging 100 steps without perturbation. The estimated eigenvalue is -0.0101 calculated with 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 $R_{\text{desired}} = 0.3, 0.5, 1.0$ m and $A_x = 0.015$ m, $Y_0^{\text{l}} = 0.04$ m, $Y_0^{\text{r}} = -0.04$ m. As $p_{\text{yaw}}^{\text{l,r}}$ is small, we

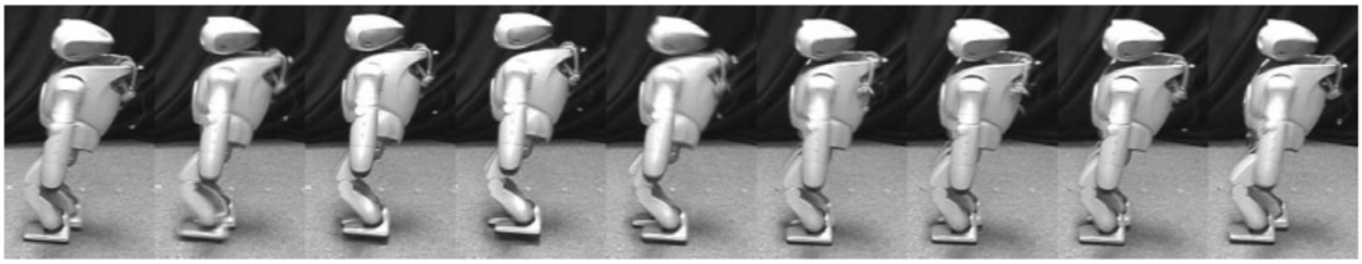


Fig. 11. Snapshots of straight steady walking with the acquired feedback controller ($A_x = 0.015$ m, $A_z = 0.005$ m, $v_x = 0.077$ m s^{-1}. Photos were captured every 0.1 s.)

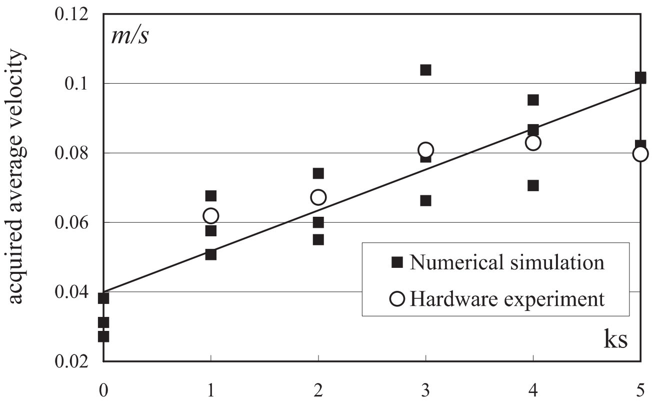


Fig. 12. The relationship between acquired average velocity and velocity reward k_S. Each data point used different policy and average velocity was derived by averaging steady walking velocity for 10 s. The solid line shows linear approximation for all data points.

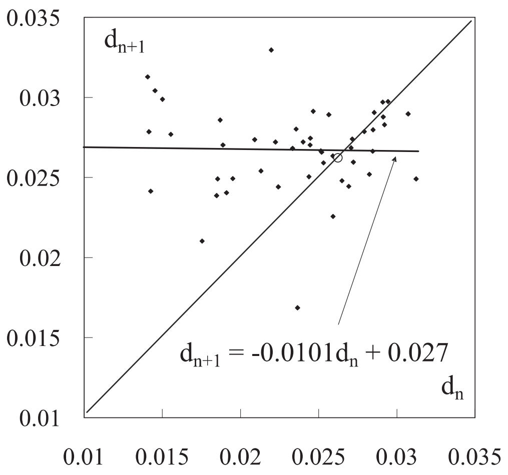


Fig. 13. Return map of the step length. We captured the step length when the left leg touched down. Here $\mathbf{d}_n$ and $\mathbf{d}_{n+1}$ are the step length just after the perturbation and the next step, respectively. The solid line indicated by an arrow is linear approximation for all data points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

can abbreviate the second term in the right-hand side of (12). To calculate the forward walking velocity v_x for the reward in (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 were performed on a PC cluster using 45 nodes with AMD Opteron 248 CPU over three 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.

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

Table 2. Achievements of acquisition.

		Number of achievements	
R_{desired}	Number of experiments	Simulation (trials)	Hardware (trials)
0.3	15	11 (68)	5 (820)
0.5	15	12 (73)	7 (1,529)
1.0	15	12 (293)	10 (2,355)

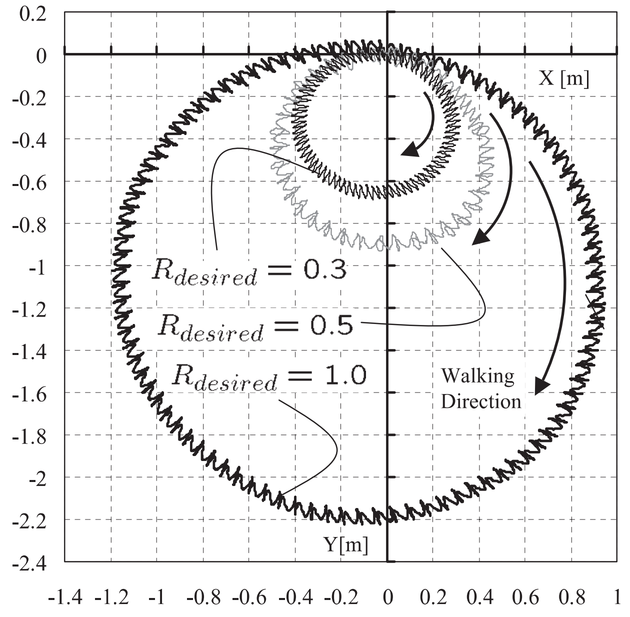


Fig. 14. Trajectories of COM with different R_{desired}. The robot started walking at $(X, Y) = (0, 0)$ in the direction of the positive X axis.

We implemented all of the acquired policies with different numbers of trials on the real robot (over two days of experimentation). We investigated the numbers of achievements and the required numbers of trials in numerical simulations, as in the case of straight walking experiments. In Table 2, the number of required trials to acquire a turning walk was much less than in the case of straight walking. A possible reason is that the walking velocity of a turning walk is relatively smaller than the straight walking owing to slip between the stance leg and the ground, which is caused by modeling error of frictional forces in the horizontal direction. As robustness for stepping motion in place is guaranteed by (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. Therefore, the acquisition of a turning walk becomes easier in comparison with straight walking in numerical simulations.

Figure 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 profiles 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 $\dot{\theta}_{\text{pitch}}$ more information compared with the straight walking case. The reason would be that we cannot clearly decompose a turning walk into sagittal motion and a lateral motion due to a yaw rotational movement around the center of turning. In addition, a smaller R_{desired} would lead to larger interference between the two motions. Thus, the learning algorithm tried to utilize both $\dot{\theta}_{\text{roll}}$ and $\dot{\theta}_{\text{pitch}}$ 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 2,062 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. However, even under these conditions, we could make use of an additional online learning on the real hardware. This is because the 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 that 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 this information only by using equipped onboard sensors. We can derive the walking velocity from the relative velocity of the stance leg with respect to the pelvis. This can be an 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 a cutoff frequency of 1 Hz to smooth out the velocity estimation. The body height was measured by 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 h to perform). Figure 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 Figure 9 owing to a different time duration 16 s for one trial.) Figure 17 shows the circular

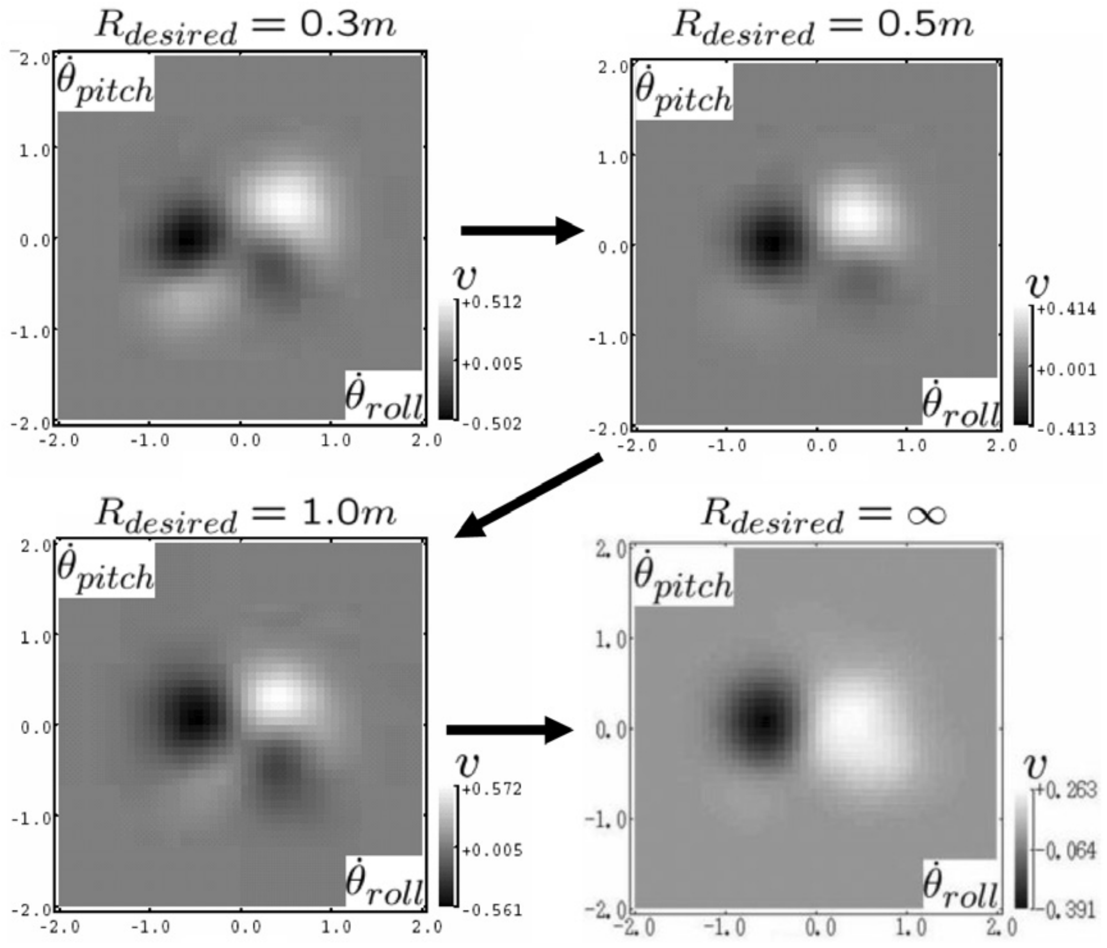


Fig. 15. Typical acquired policy with different R_{desired}. The smaller R_{desired} utilizes $\dot{\theta}_{\text{pitch}}$ more information compared with the straight walking case.

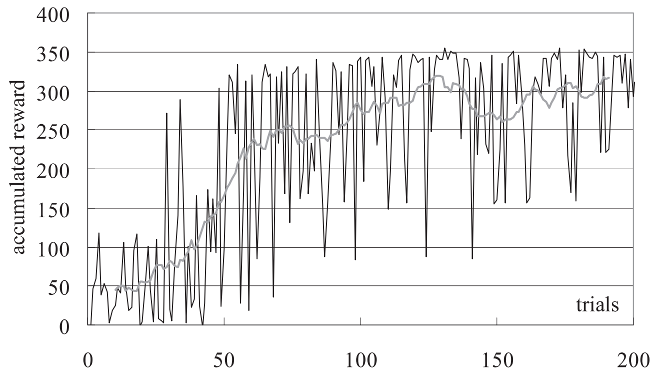


Fig. 16. Additional online learning for straight walking. The black and gray lines are the accumulated reward for one trial and the running average of the accumulated reward for 20 trials, respectively.

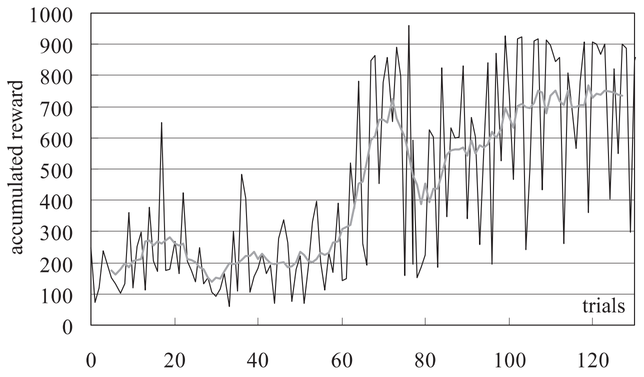


Fig. 17. Additional online learning for circular walking. The black and gray lines are the accumulated reward for one trial and the running average of the accumulated reward for 20 trials, respectively.

walking case where $k_S = 10.0$, $k_h = 10.0$ and $h' = 0.275$ with a time duration of 20 s for one trial. The gray line indicates the running average of the 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 have proposed an efficient learning framework for CPG-based biped locomotion control using the policy gradient method. We have 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 the considerable number of hidden variables, the proposed framework has successfully obtained a steady walking pattern for straight walking within 1,000 trials, on average, in the simulation. The acquired feedback controllers have been implemented on a 3D hardware robot and demonstrated robust walking in the physical environment. We have discussed velocity control and stability for straight steady walking, as well as an extension to circular walking. Finally, we have 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 that can be applied to a full-body hardware humanoid robot.

In this paper, we have only considered a policy gradient method based on Kimura's update rule. A comparison with other alternative policy search approaches such as value-function-based reinforcement learning (Doya 2000) and GPOMDP developed by Baxter and Bartlett (2001) will 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 the Initial Research Project, Okinawa Institute of Science and Technology, for productive and invaluable discussions. We would like to thank all of the persons concerned in the hardware development of the humanoid platform for supporting this research.

Appendix: Function Approximator for the Value Function and the Policy

We use a 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 and Kobayashi 1998; Peters et al. 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

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

and 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 kth 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 ($-2.0 \leq \dot{\theta}_{\mathrm{roll}}, \dot{\theta}_{\mathrm{pitch}} \leq 2.0$). We used 225 (=15 × 15) basis functions to approximate the value function and the policy.

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

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

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

Endo, G. et al. (2004). An empirical exploration of a neural oscillator for biped locomotion control. *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA'04)*, New Orleans, LA, pp. 3036–3042.

Endo, G. et al. (2005a). Learning CPG sensory feedback with policy gradient for biped locomotion for a full-body humanoid. *Proceedings of the 20th National Conference on*

Artificial Intelligence (AAAI-05), Pittsburgh, PA, pp. 1267–1273.

Endo, G. et al. (2005b). Experimental studies of a neural oscillator for biped locomotion with QRIO. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA'05)*, Barcelona, Spain, pp. 598–604.

Grillner, S. et al. (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. *Proceedings of the 6th International Symposium on Computer Simulation in Biomechanics*, Tokyo, Japan, pp. 9–12.

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

Hirai, K. et al. (1998). The development of Honda Humanoid Robot. *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, Leuven, Belgium, pp. 1321–1326.

Hirukawa, H. et al. (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. *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, Madison, WI, pp. 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. R. and Tsitsiklis, J. N. (2003). On actor–critic algorithms. *SIAM Journal on Control and Optimization*, **42**(4): 1143–1166.

Kuroki, Y. et al. (2001). A small biped entertainment robot. *Proceedings of the 2001 IEEE-RAS International Conference on Humanoid Robots (Humanoids2001)*, Tokyo, Japan, pp. 181–186.

Matsubara, T. et al. (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, NJ, Princeton University Press.

Miyakoshi, S. et al. (1998). Three dimensional bipedal stepping motion using neural oscillators—towards humanoid motion in the real world. *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, Victoria, BC, pp. 84–89.

Mori, T. et al. (2004). Reinforcement learning for a CPG-driven biped robot. *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI'04)*, San Jose, CA, pp. 623–630.

Morimoto, J. et al. (2005). Poincare-map-based reinforcement learning for biped walking. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA'05)*, Barcelona, Spain, pp. 2381–2386.

Nishiwaki, K. et al. (2000). Design and development of research platform for perception–action integration in humanoid robot: H6. *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00)*, Takamatsu, Japan, pp. 1559–1564.

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

Park, I. et al. (2005). Mechanical design of humanoid robot platform KHR-3 (KAIST Humanoid Robot—3: HUBO). *Proceedings of the 2005 IEEE-RAS International Conference on Humanoid Robots (Humanoids2005)*, Tsukuba, Japan, pp. 321–326.

Peters, J., Vijayakumar, S. and Schaal, S. (2003). Reinforcement learning for humanoid robots—policy gradients and beyond. *Proceedings of the 3rd IEEE International Conference on Humanoid Robots (Humanoids2003)*, Karlsruhe and Munich, Germany (CD-ROM).

Sutton, R. S. et al. (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. *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04)*, Sendai, Japan, pp. 2849–2854.

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