Reinforcement Learning for Balancer Embedded Humanoid Locomotion

Akihiko Yamaguchi, Sang-Ho Hyon, and Tsukasa Ogasawara

***Abstract*— Reinforcement learning (RL) applications in robotics are of great interest because of their wide applicability, however many RL applications suffer from large learning costs. We study a new learning-walking scheme where a humanoid robot is embedded with a primitive balancing controller for safety. In this paper, we investigate some RL methods for the walking task. The system has two modes: double stance and single stance, and the selectable action spaces (sub-action spaces) change according to the mode. Thus, a hierarchical RL and a function approximator (FA) approaches are compared in simulation. To handle the sub-action spaces, we introduce the *structured* FA. The results demonstrate that non-hierarchical RL algorithms with the structured FA is much faster than the hierarchical RL algorithm. The robot can obtain appropriate walking gaits in around 30 episodes (20∼30 min), which is considered to be applicable to a real humanoid robot.**

I. Introduction

Designing behavior by only its objective is essential to future robots, since this ability enables the end-users to easily teach their wish to the robots. Reinforcement learning (RL) is such a technology, so, RL applications in robotics are of great interest. Additionally, model-free RL algorithms are applicable without the dynamics model of the environment. Hence, a lot of works have been done in applying RL methods to robot control [1]∼[11]. In this paper, we focus on the RL application to humanoid locomotion.

Past research on RL applications to locomotion utilizes central pattern generators (CPGs) [4], [5], [7], or a property of passive dynamic walking [1], [6]. These methods restrict the behavior of the robot to certain patterns or dynamics. But such restriction is desirable for walking, which reduces the learning time greatly.

In contrast, we study a new learning-walking scheme where a humanoid robot embedded with a primitive balancing controller [12], [13] learns to walk (Fig. 1). This scheme has two advantages: (1) the robot can learn in safety, and (2) the size of a state-action space can be reduced. The balancing controller restricts the behavior of the robot to avoid falling down. Using the remaining DoF (degree of freedom), the robot learns to walk. Actually, the robot can still move its center of mass and a foot, which is capable to walk. However, the dynamics of the robot with the controller becomes so complex that we cannot easily identify the dynamics model. Thus, it is difficult to design optimal walking gaits. Even so, model-free RL methods are applicable to this case.

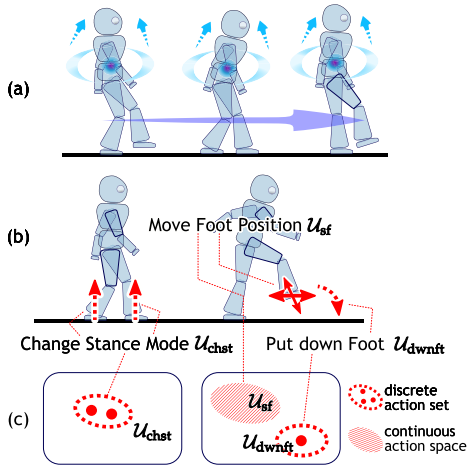

Fig. 1. Illustration of our learning-walking scheme. (a) A humanoid robot embedded with a primitive balancing controller learns walking. (b) Selectable actions in the double and the single stance modes. (c) Sub-action spaces.

Compared to the CPG approaches, our scheme is considered to be safer since falling down is automatically prevented by the balancing controller. In some passive dynamic walking approaches, the falling down of the robot is avoided due to its hardware property. For instance, falling down of the robot in [1] is a rare occasion. However, such kind of avoidance restricts the variety of robot behavior a lot. In contrast, our scheme is a software approach. That is, we can choose to use the balancing controller or not, which keeps the variety of the humanoid behavior.

In this paper, we investigate some applicable RL methods for our scheme. The features of our RL task are as follows:

(F1) The robot learns from scratch to obtain better performance than hand-coded policies.

(F2) The robot uses an on-line learning method (or batch mode of small sizes) to handle (F1) and reduce learning cost.

(F3) The robot has two modes: double stance and single stance, and the selectable action spaces (we call *sub*-action spaces) change according to the mode (Fig. 1(b), (c))

To handle (F3), we consider a hierarchical RL approach ([10], [14], [15]) and a *structured* function approximator (FA) approach. Kirchner applied a hierarchical version of

Part of this work was supported by a Grant-in-Aid for JSPS, Japan Society for the Promotion of Science, Fellows (22·9030)

A. Yamaguchi (JSPS Research Fellow) and T. Ogasawara are with the Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5, Takayama, Ikoma, Nara 630-0192, JAPAN {akihiko-y, ogasawar}@is.naist.jp

S. Hyon is with the Department of Robotics, Ritsumeikan University, 1-1-1 Noji Higashi, Kusatsu, Shiga 525-8577 JAPAN gen@fc.ritsumei.ac.jp

A. Yamaguchi and S. Hyon are also with the Computational Neuroscience Laboratories, Advanced Telecommunications Research Institute International (ATR), Kyoto 619-0288, JAPAN

Q-learning (HQL) to a similar task, the forward movement of a six-legged robot [10]. However, in our task, only two layers are needed to handle (F3). In this case, we can define a single FA into which the *sub*-FAs (corresponding to the sub-action spaces) are structured. Then we apply a normal (non-hierarchical) RL method to the structured FA. In general, if available prior knowledge of task is almost the same, a hierarchical RL method is inferior to a normal RL method since the former one has to limit its algorithm for converge. Thus, the structured FA is considered to be a better approach. Thus, we compare Peng's Q(λ)-learning [16] and fitted Q iteration [17], [18] with the structured FA, and Cohen's hierarchical RL (HRL) algorithm [15].

Our simulation results demonstrate that the FA approach is much superior to the hierarchical approach in both learning speed and the performance of the acquired walking gaits. The robot can obtain appropriate walking gaits in around 30 episodes (20$\sim$30 min), which is considered to be applicable to a real humanoid robot.

In the rest of this paper, we describe the RL algorithms used in this paper in section II, then define the FAs in section III. We present the simulation results in section IV, and conclude the paper in section V.

II. Reinforcement Learning Algorithms

The purpose of RL is for a learning system (agent) whose input is a state $x_n \in \mathcal{X}$ and a reward $R_n \in \mathbb{R}$, and whose output is an action $u_n \in \mathcal{U}$, to acquire the policy $\pi(x_n) : \mathcal{X} \to \mathcal{U}$ that maximizes the expected discounted return $\mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^{k-1} R_{n+k}\right]$ where $n \in \mathbb{N} = \{0, 1, \dots\}$ denotes the time step and $\gamma \in [0, 1)$ denotes a discount factor. In value-function-based RL algorithms, an action value function $Q(x, u) : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is learned to represent the expected discounted return by taking an action u from a state x. Then, the optimal action rule is obtained from the greedy policy $\pi(x) = \arg\max_u Q(x, u)$.

Another approach to find the optimal policy is searching directly in the policy space. The Natural Actor Critic (NAC) [3] is a typical example. However, this kind of approach strongly depends on the initial value of the policy parameter, especially in large domains. In our learning-from-scratch case, we assume that the value-function-based RL methods may obtain better policy since they can have richer policy parameterization. Thus, we test Peng's Q(λ)-learning algorithm [16] and fitted Q iteration algorithm [17], [18].

In general, fitted Q iteration requires appropriate samples, but it is difficult to obtain such samples with a random policy. In our preliminary experiments, though we applied fitted Q iteration in batch mode of small sizes, we were aware that Q(λ)-learning is better in the early stage of learning. A possible reason is that since we reduce the set of basis functions of a FA to improve the learning speed, the system loses the Markov property. Thus, we mainly use Q(λ)-learning, and apply fitted Q iteration to the same FA only with sample sequences of higher return.

In the rest of this section, we introduce these RL algorithms. Additionally, we briefly describe Cohen's hierarchical RL (HRL) algorithm [15].

A. Peng's Q($lambda$)-Learning Algorithm

The Peng's Q(λ)-learning algorithm [16] is an on-line RL method. The update procedure for a generic FA $Q(x, u)$ of the parameter $\theta \in \Theta$ is written as follows:

$$e_n = R_n + \gamma V_n(x_{n+1}) - V_n(x_n), \tag{1a}$$

$$e'_n = R_n + \gamma V_n(x_{n+1}) - Q_n(x_n, u_n), \tag{1b}$$

$$\theta_{n+1} = \theta_n + \alpha e_n Tr_n + \alpha e'_n \nabla_\theta Q_n(x_n, u_n), \tag{1c}$$

$$Tr_{n+1} = (\gamma\lambda)(Tr_n + \nabla_\theta Q_n(x_n, u_n)), \tag{1d}$$

where $\nabla_\theta Q(x, u)$ denotes the derivative of $Q(x, u)$ w.r.t. the parameter θ, α denotes a step-size parameter, Tr denotes the eligibility trace ($Tr_0 = \mathbf{0} \in \Theta$), and $V_n(x) \triangleq \max_u Q_n(x, u)$. This update procedure is applied after each action.

B. Fitted Q Iteration Algorithm

The fitted Q iteration algorithm [17], [18] is a batch mode RL method to learn from sample trajectories whose element is a four-tuple $F_n = (x_n, u_n, x_{n+1}, R_n)$. The idea of fitted Q iteration is as follows: first, we build a training set from the current FA and a set of four tuples. Then, we train the next FA with the training set by a supervised learning method. Iterating these two steps, the FA will converge to the action value function.

The action value FA Q_0 is initialized so that $Q_0(x, u) = 0$ for all $(x, u) \in \mathcal{X} \times \mathcal{U}$. In the N-th iteration, the training set $\{i_n, o_n\}$ is built from the current FA Q_{N-1} and the set of four tuples $\{F_n\}$ by

$$i_n = (x_n, u_n), \tag{2}$$

$$o_n = R_n + \gamma \max_u Q_{N-1}(x_{n+1}, u). \tag{3}$$

Then, Q_N is trained with $\{i_n, o_n\}$. We implement this supervised learning with a gradient descent for the least squares.

As mentioned above, we combine fitted Q iteration and Q(λ)-learning. At the end of each action, we apply the update rule of Q(λ)-learning and store the sample. At the end of every N_{FQI} episodes, we execute an iteration of fitted Q iteration with the samples in the top N_{smpl} episodes ranked by the return of the episode.

C. Cohen's Hierarchical Reinforcement Learning

As a hierarchical RL method, we use Cohen's hierarchical RL (HRL) [15]. Many hierarchical RL methods require that a task can be decomposed into sub-tasks. But, in our case, we only design a single reward function. Thus, the hierarchical RL methods requiring sub-tasks are not applicable to our case. However, Cohen's HRL uses a single reward function, which is suitable for our case. Thus, we choose it.

Though Cohen's HRL is developed for discrete state-action spaces, we extend this algorithm in a straightforward way so that a FA is available for each module.

For our task, we construct a two layer modular structure. Specifically, it has one higher module and several lower

modules. Each lower module has its unique sub-action space. The higher module selects a lower module as an action.

III. Function Approximators

Next, we describe some FAs for the action value functions $Q(x,u)$. For a continuous state space $\mathcal{X}$ and a discrete action space $\mathcal{U}$, we use a linear FA because of its stability. If both the state and the action spaces are continuous, we employ wire-fitting [19]. The remarkable feature of wire-fitting is that we can maximize $Q(x,u)$ w.r.t. u by evaluating only on a fixed number of points. We define the structured FA for our walking task, where the action space consists of discrete sets and continuous vector spaces whose selectability depends on the state.

A. Linear Function Approximator

For a continuous state $x \in \mathcal{X}$ and a discrete action $u \in \mathcal{U}$, we let $Q(x,u) = \theta_u^\top \phi(x)$, where $\phi(x) = (\phi_1(x), \dots, \phi_{|\mathcal{K}|}(x))^\top$ denotes the feature vector of a state x, $\mathcal{K} = \{\phi_k \mid k = 1, 2, ..\}$ denotes a set of basis functions, and $\theta_u \in \mathbb{R}^{|\mathcal{K}|\times 1}$ denotes a parameter related to an action u. The parameter vector is defined as $\theta = (\theta_1^\top, \dots, \theta_{|\mathcal{U}|}^\top)^\top \in \mathbb{R}^{|\mathcal{K}||\mathcal{U}|\times 1}$. The derivative of the $Q(x,u)$ w.r.t. θ is given by $\nabla_\theta Q_n(x,u) = (\delta_{1u}\phi_1^\top, \dots, \delta_{|\mathcal{U}|u}\phi_{|\mathcal{U}|}^\top)^\top$ where δ denotes the Kronecker's delta.

As basis functions, we use Normalized Gaussian Network (NGnet) [20] which is sometimes used as the basis functions of FAs in RL applications [11]. In NGnet, $\phi_k(x)$ is given by

$$\phi_k(x) = \frac{\mathrm{G}(x; \mu_k, \Sigma_k)}{\sum_{k' \in \mathcal{K}} \mathrm{G}(x; \mu_{k'}, \Sigma_{k'})}, \tag{4}$$

where $\mathrm{G}(x; \mu, \Sigma)$ denotes a Gaussian with mean μ and covariance matrix Σ. In the case of a linear FA, $\mathcal{K}$ is predefined and $\{\mu_k, \Sigma_k \mid k \in \mathcal{K}\}$ are treated as fixed parameters. In the following, a linear FA with NGnet is referred to as LFA-NGnet.

B. Wire-Fitting

For a continuous state $x \in \mathcal{X}$ and a continuous action $u \in \mathcal{U}$, wire-fitting is defined as:

$$Q(x,u) = \lim_{\epsilon \to 0^+} \frac{\sum_{i \in \mathcal{W}} (d_i + \epsilon)^{-1} q_i(x)}{\sum_{i \in \mathcal{W}} (d_i + \epsilon)^{-1}}, \tag{5}$$

$$d_i = \|u - u_i(x)\|^2 + C\big[\max_{i' \in \mathcal{W}}(q_{i'}(x)) - q_i(x)\big]. \tag{6}$$

Here, a pair of the functions $q_i(x) : \mathcal{X} \to \mathbb{R}$ and $u_i(x) : \mathcal{X} \to \mathcal{U}$ ($i \in \mathcal{W}$) is called a *control wire*; wire-fitting is regarded as an interpolator of the set of control wires $\mathcal{W}$. C is a smoothing factor of the interpolation; we choose $C = 0.001$ in the experiments. Any FA is available for $q_i(x)$ and $u_i(x)$. Regardless of the kind of the FAs, one of $q_i(x)$, $i \in \mathcal{W}$ is equal to $\max_u Q(x,u)$ and the corresponding $u_i(x)$ is the greedy action at x.

We use NGnet for $q_i(x)$ and a constant vector for $u_i(x)$, that is, we let $q_i(x) = \theta_i^\top \phi(x)$ and $u_i(x) = U_i$, where $\phi(x)$ is the feature vector of the NGnet. The parameter vector θ is defined as $\theta^\top = (\theta_1^\top, U_1^\top, \theta_2^\top, U_2^\top, \dots, \theta_{|\mathcal{W}|}^\top, U_{|\mathcal{W}|}^\top)$, and the gradient $\nabla_\theta Q(x,u)$ can be calculated analytically.

As an exploration policy in using wire-fitting, we use the Boltzmann-like selection method proposed in [21]. A control wire i is considered to be a discrete action whose action value is $q_i(x)$, and one of the control wires is chosen by Boltzmann selection. Then the corresponding $u_i(x)$ is the selected action. We refer to this method as WF-Boltzmann.

C. Structured Function Approximator

Next, we define a FA for our walking task where the action space consists of discrete sets and continuous vector spaces whose selectability depends on the state. Such an action space can be defined as a direct sum of discrete sets and vector spaces. Thus, we denote the action space as $\mathcal{U} = \bigsqcup_{p \in \mathcal{P}} \mathcal{U}_p$, where $\mathcal{U}_p$ is a sub-action space and $\mathcal{P}$ is a set of sub-space indexes. Again, see Fig. 1(b), (c) as an example of a set of sub-action spaces. In the following, we denote $u = (p, u_p) \in \mathcal{U}$ for convenience. We use $\mathcal{P}(x) \subseteq \mathcal{P}$ to express the selectable sub-action spaces at a state x.

We simply define a FA over the action space $\mathcal{U}$ by structuring (combining) sub-action value functions. First of all, we define each sub-action value function $Q_p(x, u_p)$ over $\mathcal{X} \times \mathcal{U}_p$. We choose the LFA-NGnet for a discrete set $\mathcal{U}_p$, and wire-fitting for a continuous space $\mathcal{U}_p$. Then we define the overall Q as

$$Q(x,u) \triangleq \sum_{p' \in \mathcal{P}} \delta_{pp'} Q_{p'}(x, u_{p'}) = Q_p(x, u_p), \tag{7}$$

where $u = (p, u_p)$. We let θ_p the parameter of a sub-action value function Q_p. The parameter vector of the Q can be defined as $\theta = (\theta_1^\top, \dots, \theta_{|\mathcal{P}|}^\top)^\top$. The derivative of Q w.r.t. θ is given by

$$\nabla_\theta Q(x,u)^\top = \big(\delta_{p1} \nabla_{\theta_1} Q_1(x, u_1)^\top, \dots, \delta_{p|\mathcal{P}|} \nabla_{\theta_{|\mathcal{P}|}} Q_{|\mathcal{P}|}(x, u_{|\mathcal{P}|})^\top\big), \tag{8}$$

where $u = (p, u_p)$.

The greedy action at x can be given by

$$u^\star = \operatorname*{arg\,max}_{u \in \bigsqcup_{p \in \mathcal{P}(x)} \mathcal{U}_p} Q(x,u). \tag{9}$$

This can be evaluated as follows: (1) calculating $\hat{u}_p = \arg\max_{u_p \in \mathcal{U}_p} Q_p(x, u_p)$ for all $p \in \mathcal{P}(x)$, (2) calculating $u^\star = \arg\max_{u \in \{\hat{u}_p\}} Q(x,u)$.

As an exploration policy, we define a two-stage action selection method so that the RL agent can explore as broadly as possible and it has a scalability for any kind of sub-FAs. In the first stage, for each $p \in \mathcal{P}(x)$, select a sub-action $\hat{u}_p$ from $\mathcal{U}_p$ based on $Q_p(x, u_p)$. Here, we use Boltzmann selection if Q_p is the LFA-NGnet and WF-Boltzmann selection if Q_p is wire-fitting.

In the second stage, select an action u from $\{\hat{u}_p | p \in \mathcal{P}(x)\}$ based on their action values $\{Q_p(x, \hat{u}_p)\}$. We use a weighted version of Boltzmann selection to consider the size of $\mathcal{U}_p$ so that the RL agent can broadly explore. That is,

$$\pi(\hat{u}_p | x) = \frac{w_p \exp(\frac{1}{\tau} Q_p(x, \hat{u}_p))}{\sum_{p' \in \mathcal{P}(x)} w_{p'} \exp(\frac{1}{\tau} Q_{p'}(x, \hat{u}_{p'}))}, \tag{10}$$

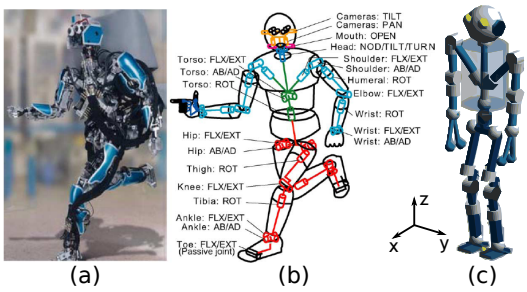

Fig. 2. SARCOS biped humanoid robot developed by NICT/ATR. (a) Hardware. (b) DoF configuration. (c) Simulation model.

where w_p denotes the weight to compensate the size of $\mathcal{U}_p$. We decide the weights $\{w_p\}$ so that they are proportional to the size of the action set $\mathcal{U}_p$ if $\mathcal{U}_p$ is discrete, or to the number of the control wires of Q_p if Q_p is wire-fitting. Note that in the early stage of learning, the probability of $\hat{u}_p$ is nearly proportional to w_p, which makes the exploration appropriate for the size of $\mathcal{U}_p$.

IV. Experiments: Learning Walking of a Human-Size Humanoid Robot

We apply the RL methods mentioned above to the walking task of a human-size biped humanoid robot shown in Fig. 2. The humanoid robot has 50 DoF and torque controllability with hydraulic actuation [22]. Its height is 1.58 m, and its hip height is 0.82 m when at an upright posture. It weighs 93.7 kg. Its DoF configuration is shown in Fig. 2(b). The arms and legs each have 7 DoFs, and the neck and torso each have 3 DoFs. In this section, we demonstrate a simulation comparison of the RL methods. In the experiments, we use a dynamics simulator with a presice model (Fig. 2(c)) including a well-tuned contact model.

A. Robotic System Setup for RL

The robot is embedded with the balancing controller [13]. It regulates the center of mass (CoM) to the center of the supporting region through the optimal force control. For biped walking case, the desired CoM, as well as the position of the swinging foot should be varied. Although the controller has compliant stabilization and terrain-adaptation abilities, its performance is not satisfactory for dynamic situations because of sensory delays and limitations in the low-level joint controllers. Thus, we investigate which RL method is most appropriate for this situation.

The state given to the RL agent consists of the stance mode, $\{\mathsf{left}, \mathsf{double}, \mathsf{right}\}$, the position of the CoM, and the previous stance mode. We also tested the velocity of the CoM instead of the previous stance mode. But, the velocity is so sensitive to sensor noise and is therefore not suitable for the real robot case. Thus, the state x is defined as

$$x = (mode_{\rm st}, x_{\rm cm}, y_{\rm cm}, mode_{\rm pst}). \tag{11}$$

In the following experiments, we allocate 405 Gaussians on a $3 \times 9 \times 5 \times 3$ grid as the basis functions of the NGnet.

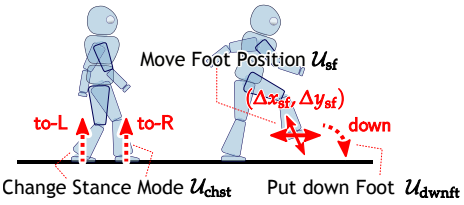

Fig. 3. Illustration of the sub-action spaces of the walking task.

Though the robot with the balancing controller can move the position of the CoM and the swinging foot, we pre-implement some primitive CoM movements so that the RL agent can learn the task easily. Thus, the available sub-action spaces are defined as illustrated in Fig. 3. Specifically, the available sub-action space in the double stance mode is defined as follows.

$\mathcal{U}_{\rm chst} = \{\mathsf{to\text{-}R}, \mathsf{to\text{-}L}\}$: Changing the stance mode to right or left. Specifically, the position of CoM is moved to above the left/right foot, then the other foot is raised up. These actions are executed in 1.0 second respectively.

And the available sub-action spaces in the single stance mode are defined as follows.

$\mathcal{U}_{\rm dwnft} = \{\mathsf{down}\}$: Putting down the swinging foot. This action is executed in 1.0 second.

$\mathcal{U}_{\rm sf} = \{(\Delta x_{\rm sf}, \Delta y_{\rm sf}) \mid \Delta x_{\rm sf}, \Delta y_{\rm sf} \in \mathbb{R}\}$: Moving $x_{\rm sf}$ and $y_{\rm sf}$ which denote the x and y positions of the swinging foot. $\Delta x_{\rm sf}$ and $\Delta y_{\rm sf}$ denote their differences. This action is executed in 0.1 second.

Namely, the action space is $\mathcal{U} = \mathcal{U}_{\rm chst} \sqcup \mathcal{U}_{\rm dwnft} \sqcup \mathcal{U}_{\rm sf}$, and the selectable action spaces at a state x can be written by

$$\mathcal{P}(x) = \begin{cases} \{\rm chst\} & \text{if } mode_{\rm st} = \mathsf{double}, \\ \{\rm dwnft, sf\} & \text{if } mode_{\rm st} = \mathsf{left \text{ or } right}. \end{cases} \tag{12}$$

B. RL Methods Configurations

For the discrete action spaces $\mathcal{U}_{\rm chst}$ and $\mathcal{U}_{\rm dwnft}$, we define $Q_{\rm chst}$ and $Q_{\rm dwnft}$ as the LFA-NGnet, respectively. For the continuous action space $\mathcal{U}_{\rm sf}$, we define $Q^{\rm wf}_{\rm sf}$ as wire-fitting. For comparison, we also define $Q^{\rm disc}_{\rm sf}$ as the LFA-NGnet over $\mathcal{U}^{\rm disc}_{\rm sf}$ defined by discretizing $\mathcal{U}_{\rm sf}$ with a 3×3 grid. The parameters of every LFA-NGnet are initialized to zero. About $Q^{\rm wf}_{\rm sf}$, $\{\theta_i | i \in \mathcal{W}\}$ are initialized to zero, while $\{U_i | i \in \mathcal{W}\}$ are initialized with points of a 3×3 grid on $\mathcal{U}_{\rm sf}$.

In this experiments, we compare the following combinations of the RL algorithms and the sub-FAs[1].

S-WF-QL : Q(λ)-learning for $Q^{\rm wf}$ where $Q_{\rm chst}$, $Q_{\rm dwnft}$, and $Q^{\rm wf}_{\rm sf}$ are structured.

S-WF-QLFQI : The combination of Q(λ)-learning and fitted Q iteration for $Q^{\rm wf}$.

S-DISC-QL : Q(λ)-learning for $Q^{\rm disc}$ where $Q_{\rm chst}$, $Q_{\rm dwnft}$, and $Q^{\rm disc}_{\rm sf}$ are structured.

S-DISC-QLFQI : The combination of Q(λ)-learning and fitted Q iteration for $Q^{\rm disc}$.

S-DISC-Q0LFQI : The combination of Q(0)-learning and fitted Q iteration for $Q^{\rm disc}$.

[1]We used the RL library, SkyAI: `skyai.sourceforge.net`

HRL : Cohen's HRL for a two layer modular structure where the lower modules learn Q_{chst}, Q_{dwnft}, and $Q_{\mathrm{sf}}^{\mathrm{disc}}$. The higher module learns the policy to select a lower module. The S-DISC-Q0LFQI is compared to verify the effect of the eligibility trace (λ). The reason why $Q_{\mathrm{sf}}^{\mathrm{disc}}$ is used in the HRL rather than $Q_{\mathrm{sf}}^{\mathrm{wf}}$ is due to the stability of the linear FA.

For every RL method, we set $\gamma = 0.95$. We use a decreasing step size parameter $\alpha = \max(0.05, 0.3\exp(-0.002N_{\mathrm{eps}}))$ for Q(λ)-learning and the HRL. N_{eps} denotes a number of episodes. For fitted Q iteration, we use a constant step size parameter $\alpha = 0.05$. For Q(λ)-learning, we set $\lambda = 0.9$ and apply the *replacing trace* [23] to make the eligibility trace stable (see also [24]). For the combination of Q(λ)-learning and fitted Q iteration, we set $N_{\mathrm{FQI}} = 3$ and $N_{\mathrm{smpl}} = 10$. As the exploration policy, we use Boltzmann (or Boltzmann-like) selection, with a decreasing temperature $\tau = 1.0\exp(-0.002N_{\mathrm{eps}})$.

C. Task Setting

The objective of the walking task is to move forward along the x-axis as far as possible. Though the balancing controller is embedded, the robot still has a probability of falling down. The balancing controller employed in this paper does not consider the swinging leg motions explicitly. When the swinging leg moves too fast and CoP locates near the supporting edge, then the robot can lose the stability. Thus, we design the reward function as follows:

$$r(t) = r_{\mathrm{mv}}(t) - r_{\mathrm{sc}}(t) - r_{\mathrm{fd}}(t), \tag{13a}$$

$$r_{\mathrm{mv}}(t) = 200 v_{\mathrm{cm}x}(t)\delta t, \tag{13b}$$

$$r_{\mathrm{sc}}(t) = 1\delta t, \tag{13c}$$

$$r_{\mathrm{fd}}(t) = \begin{cases} 50 & \text{if falling-down,} \\ 0 & \text{otherwise,} \end{cases} \tag{13d}$$

where r_{mv} means a reward for moving, r_{sc} means a step cost, and r_{fd} means a penalty for falling down[2]. The reward function $r(t)$ is integrated from the start time to the finish time of an action, and given to the RL agent. Each episode starts with the initial state where the robot is standing up (first snapshot in Fig. 6) and stationary, and ends if $t > 75\,\mathrm{s}$ or the robot is falling down.

D. Result

Fig. 4 shows the resulting learning curves of the walking task (the mean of the return over 10 runs per episode). The horizontal axis is logarithmic scale. The horizontal line (MANUAL) shows the performance of a manually manipulated walking with a keyboard interface[3]. Fig. 5 shows the trajectory in an episode of the CoM position of a walking gait acquired by the S-WF-QLFQI, and Fig. 6 shows the corresponding snapshots.

The HRL and the S-DISC-Q0LFQI are very slow, that is, they take a lot of episodes to acquire performance. The possible reason is that the update rule of the HRL is similar to Q(0)-learning which may be poor for the walking task. About the S-DISC-Q0LFQI, Q(0)-learning is dominant in the early stage of learning rather than fitted Q iteration since suitable samples are not obtained yet. Thus, the S-DISC-Q0LFQI is considered to be as slow as the HRL.

[2]Specifically, the falling-down is defined as $(|\phi| > 25.0^\circ) \vee (|\theta| > 25.0^\circ)$ where ϕ and θ denote the x and y-Euler angles respectively.

[3]The maximum return value in 30 trials is plotted.

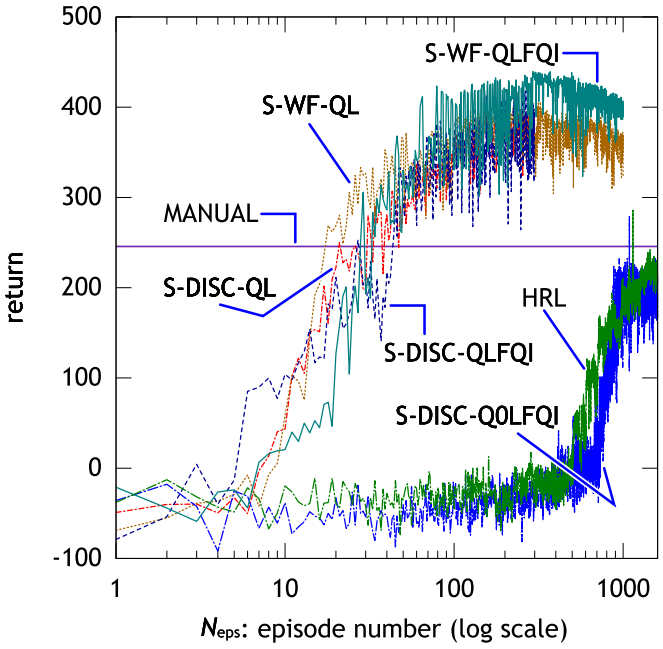

Fig. 4. Resulting learning curves of the walking task. Each curve shows the mean of the return over 10 runs per episode. The horizontal line (MANUAL) shows the performance of a manually manipulated walking with a keyboard interface.

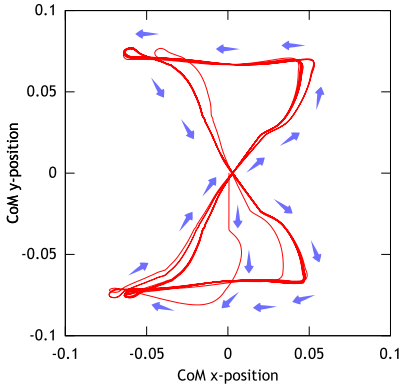

Fig. 5. Trajectory of the CoM position. Small arrows indicate the direction of the movement. Note that the origin of the CoM is the center of the feet.

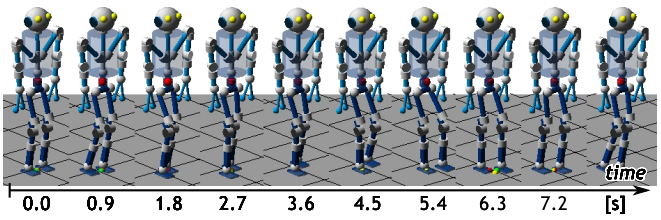

Fig. 6. Animation snapshots.

All of the methods using Q(λ)-learning are much faster than these two methods. Thus, the eligibility trace (λ) is considered to be very effective for the walking task.

The acquired performance of the S-WF-QLFQI is the best, that is, it converges to the highest value of the return. It substantially exceeds the performance of the manually manipulated walking. Compared with Q^{disc}, the FA Q^{wf} has an ability to acquire better performance since Q^{wf} directly approximate over the continuous action space $\mathcal{U}_{\mathrm{sf}}$ by wire-fitting. However, there is no performance difference between the S-DISC-QL and the S-WF-QL because of the instability of wire-fitting caused by its nonlinearity. Thus, we consider the reason for the best performance of the S-WF-QLFQI is due to both using wire-fitting and updating by fitted Q iteration which is more stable than Q(λ)-learning. We should also note that the S-WF-QLFQI is slightly slower than the S-WF-QL. A possible reason is that Q(λ)-learning and fitted Q iteration conflict since the definition of the action value function is slightly different in the two algorithms.

V. Conclusion and Future Work

We investigated some applicable RL methods for the new learning-walking scheme where a humanoid robot is embedded with a primitive balancing controller. This scheme has two advantages: (1) the robot can learn in safety, and (2) the size of the state-action space can be reduced. We considered a hierarchical RL approach and FA approaches, and compared them in simulation. The results demonstrated that Cohen's hierarchical RL algorithm did not work well; it took long learning time. On the other hand, the structured FA was defined for our situation. The RL methods based on Peng's Q(λ)-learning could obtain a suitable policy much faster than the HRL. Especially, applying the combination of fitted Q iteration and Q(λ)-learning to the structured FA acquired the best performance. The reason is considered as both the accuracy of wire-fitting and the stability of fitted Q iteration.

Though we investigated for the walking task, our findings are considered to be applicable to similar situations. That is, for a task where the action space consists of discrete sets and continuous vector spaces whose selectability depends on the state, the structured FA approach should be considered rather than hierarchical RL methods. The hierarchical RL methods have wider applicability, but in a specific situation, specialized methods sometimes work more efficiently like the structured FA.

We did not compare the policy gradient methods in this paper, since we assumed the value-function-based RL methods can obtain better policy. However, testing some policy gradient methods, such as NAC [3], is part of our future work.

References

[1] R. Tedrake, T. Zhang, and H. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," in *the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04)*, vol. 3, 2004, pp. 2849–2854.

[2] H. Kimura, T. Yamashita, and S. Kobayashi, "Reinforcement learning of walking behavior for a four-legged robot," in *Proceedings of the 40th IEEE Conference on Decision and Control*, 2001.

[3] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *Humanoids2003, IEEE-RAS International Conference on Humanoid Robots*, 2003.

[4] T. Matsubara, J. Morimoto, J. Nakanishi, M. Sato, and K. Doya, "Learning CPG-based biped locomotion with a policy gradient method," *Robotics and Autonomous Systems*, vol. 54, no. 11, pp. 911–920, 2006.

[5] L. Righetti and A. Ijspeert, "Programmable Central Pattern Generators: an application to biped locomotion control," in *the IEEE Internactional Conference in Robotics and Automation (ICRA'06)*, 2006.

[6] K. Hitomi, T. Shibata, Y. Nakamura, and S. Ishii, "Reinforcement learning for quasi-passive dynamic walking of an unstable biped robot," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 982–988, 2006.

[7] Y. Nakamura, T. Mori, M. Sato, and S. Ishii, "Reinforcement learning for a biped robot based on a CPG-actor-critic method," *Neural Networks*, vol. 20, no. 6, pp. 723–735, 2007.

[8] T. Matsubara, J. Morimoto, J. Nakanishi, S. Hyon, J. G. Hale, and G. Cheng, "Learning to acquire whole-body humanoid CoM movements to achieve dynamic tasks," in *the IEEE Internactional Conference in Robotics and Automation (ICRA'07)*, 2007, pp. 2688–2693.

[9] A. Yamaguchi, J. Takamatsu, and T. Ogasawara, "Constructing action set from basis functions for reinforcement learning of robot control," in *the IEEE Internactional Conference in Robotics and Automation (ICRA'09)*, Kobe, Japan, 2009, pp. 2525–2532.

[10] F. Kirchner, "Q-learning of complex behaviours on a six-legged walking machine," *Robotics and Autonomous Systems*, vol. 25, no. 3-4, pp. 253–262, 1998.

[11] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," *Robotics and Autonomous Systems*, vol. 36, no. 1, pp. 37–51, 31 July 2001.

[12] S. Hyon, J. Hale, and G. Cheng, "Full-body compliant human-humanoid interaction: Balancing in the presence of unknown external forces," *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 884–898, 2007.

[13] S. Hyon, "Compliant terrain adaptation for biped humanoids without measuring ground surface and contact forces," *Robotics, IEEE Transactions on*, vol. 25, no. 1, pp. 171–178, 2009.

[14] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, no. 4, pp. 341–379, 2003.

[15] S. Cohen, O. Maimon, and E. Khmlenitsky, "Reinforcement learning with hierarchical decision-making," in *ISDA '06: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications*. USA: IEEE Computer Society, 2006, pp. 177–182.

[16] J. Peng and R. J. Williams, "Incremental multi-step Q-learning," in *International Conference on Machine Learning*, 1994, pp. 226–232.

[17] D. Ernst, P. Geurts, and L. Wehenkel, "Iteratively extending time horizon reinforcement learning," in *Proceedings of the 14th European Conference on Machine Learning*, N. Lavra, L. Gamberger, and L. Todorovski, Eds. Dubrovnik, Croatia: Springer-Verlag Heidelberg, September 2003, pp. 96–107.

[18] ——, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.

[19] L. C. Baird and A. H. Klopf, "Reinforcement learning with high-dimensional, continuous actions," Wright Laboratory, Wright-Patterson Air Force Base, Tech. Rep. WL-TR-93-1147, 1993.

[20] M. Sato and S. Ishii, "On-line EM algorithm for the normalized Gaussian network," *Neural Computation*, vol. 12, no. 2, pp. 407–432, 2000.

[21] A. Yamaguchi, J. Takamatsu, and T. Ogasawara, "Constructing continuous action space from basis functions for fast and stable reinforcement learning," in *the 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN'09)*, Toyama, Japan, 2009, pp. 401–407.

[22] M. Kawato, "From 'understanding the brain by creating the brain' towards manipulative neuroscience," *Phil. Trans. R. Soc. B*, vol. 363, no. 1500, pp. 2201–2214, 2008.

[23] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine Learning*, vol. 22, no. 1-3, pp. 123–158, 1996.

[24] J. N. Tsitsiklis and B. V. Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.