Trial and Error

Using Previous Experiences as Simulation Models in Humanoid Motor Learning

By Norikazu Sugimoto, Voot Tangkaratt, Thijs Wensveen, Tingting Zhao, Masashi Sugiyama, and Jun Morimoto

Since biological systems have the ability to efficiently reuse previous experiences to change their behavioral strategies to avoid enemies or find food, the number of required samples from real environments to improve behavioral policy is greatly reduced. Even for real robotic systems, it is desirable to use only a limited number of samples from real environments due to the limited durability of real systems to reduce the required time to improve control performance. In this article, we used previous experiences as environmental local models so that the movement policy of a humanoid robot can be efficiently improved with a limited number of samples from its real environment. We applied our proposed learning method to a real humanoid robot and successfully achieve two challenging control tasks. We applied our proposed learning approach to acquire a policy for a cart-pole swing-up task in a real-virtual hybrid task environment, where the robot waves a PlayStation (PS) Move motion controller to move a cart-pole in a virtual simulator. Furthermore, we applied our proposed method to a challenging basketball-shooting task in a real environment.

Motor Learning in Real Environments

For biological systems, exploring outside worlds for long periods of time can be dangerous, since this increases the probability of encountering enemies. On the other hand, a biological system needs to explore its surrounding environment to sample the data and improve its behavioral policy to increase the probability of survival. In such cases, efficiently reusing previous experiences is crucial for improving its behavioral policies without actually interacting with real environments. A standard approach is using a parameterized simulation model.

Digital Object Identifier 10.1109/MRA.2015.2511681
Date of publication: 23 February 2016

1070-9932/16©2016IEEE

System identification methods can be used to build a parameterized model, and the acquired model can virtually sample the data for policy improvement [1], [4], [14], [17], [22], [35], [36]. However, for policy improvement, the learned simulation model's generalization performance needs to be carefully verified. Even though the simulation model can be useful for predicting state transitions, explicitly predicting them is not necessary for policy updates.

In this article, we propose using previously acquired data as a simulation model instead of building a parameterized simulation model. To utilize the previously acquired data to improve the current policy, we need to reevaluate the previous data in terms of the current policy. To do this in reinforcement-learning (RL) frameworks, importance-weighted policy gradients with parameter-based exploration (IW-PGPE) can be used [34]. The usefulness of this approach has been thoroughly evaluated by comparisons in numerical simulations with previously proposed RL methods [6], [9], [15], [25], [29].

However, no work has clearly stated and shown that this particular combination of PGPE and importance weighting to derive the gradient of objective functions is suitable for real robot learning. In our previous study, we presented preliminary results and showed that IW-PGPE is a useful approach for humanoid motor learning [23]. In this article, we extend our IW-PGPE algorithm to more efficiently reuse previous experiences by introducing a recursive operation to policy updates and show how our extended algorithm is useful for real robot learning in high-dimensional spaces. We successfully applied our proposed approach to two different tasks with two different conditions. First, we applied it to a cart-pole swing-up task in a real-virtual hybrid environment with a PS Move motion controller. Then we applied it to a challenging basketball-shooting task in a real environment.

Motor Learning Framework

Trajectories and Returns

We assume that the underlying control problem is a discrete-time Markov decision processes. At each discrete time step t, the agent observes state $\mathbf{x}(t) \in \mathcal{X}$, selects action $\mathbf{u}(t) \in \mathcal{U}$, and receives immediate reward $r(t)$ of the results from a state transition in the environment. The environment's dynamics are characterized by $p(\mathbf{x}(t+1) \,|\, \mathbf{x}(t), \mathbf{u}(t))$, which represents the transition probability density from current state $\mathbf{x}(t)$ to subsequent state $\mathbf{x}(t+1)$ when action $\mathbf{u}(t)$ is taken; $p(\mathbf{x}(1))$ is the probability density of the initial states. Immediate reward $r(t)$ is given based on reward function $r(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}(t+1))$.

The robot's decision-making procedure at each time step t is characterized by parameterized policy $p(\mathbf{u}(t)|\mathbf{x}(t), \mathbf{w})$ with parameter $\mathbf{w}$, which represents the conditional probability density of taking action $\mathbf{u}(t)$ in state $\mathbf{x}(t)$. We assume that the policy is continuously differentiable with respect to parameter $\mathbf{w}$.

A sequence of states and actions forms a trajectory denoted by $h := [\mathbf{x}(1), \mathbf{u}(1), \ldots, \mathbf{x}(T), \mathbf{u}(T)]$, where T denotes the number of steps, called the horizon length. We assume that T is a fixed deterministic number. Then the discounted cumulative reward along h, called the *return*, is given by

$$R(h) := \sum_{t=1}^{T-1} \gamma^{t-1} r(\mathbf{x}(t), \mathbf{u}(t)) + \Phi(\mathbf{x}(T)), \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor for future rewards. The immediate and terminal rewards are $r(\mathbf{x}(t), \mathbf{u}(t))$ and $\Phi(\mathbf{x}(T))$.

Policy Models

In this article, we consider feedback and feedforward policy models.

Feedback Policy Model

We use locally linear state-dependent basis functions in our feedback policy model [20], [21], [31]:

$$\mathbf{u}(t) = \mathbf{W}^{\text{fb}} \boldsymbol{\phi}^{\text{fb}}(\mathbf{z}(t)), \tag{2}$$

where $\mathbf{z}(t) \in \mathcal{Z}$ is a feedback state at time t. Note that state space $\mathcal{Z}$ can be a subset of the original state space, i.e., $\mathcal{Z} \subset \mathcal{X}$. $\mathbf{W}^{\text{fb}}$ is a state-dependent matrix, and $\boldsymbol{\phi}^{\text{fb}}(\mathbf{z}(t)) \in \Re^M$ is a vector that consists of state-dependent basis functions, where M is the number of basis functions.

Feedforward Policy Model

The feedforward policy model is formulated as:

$$\mathbf{u}(t) = \mathbf{W}^{\text{ff}} \boldsymbol{\phi}^{\text{ff}}(t), \tag{3}$$

where $\mathbf{w} \in \Re^M$ is the parameter vector, $\boldsymbol{\phi}^{\text{ff}}(t) \in \Re^{M \times 1}$ is a vector that consists of time-dependent basis functions, and $\mathbf{W}^{\text{ff}}$ is a parameter matrix.

Table 1. The supplemental equations for low-level controller and gradient of objective function with reference to policy parameters.

1.1) The expected return in the PGPE formulation is defined in terms of the expectations over both h and $\mathbf{w}$ as a function of hyperparameter ρ:

$$J(\rho) := \int\int p(h \,|\, \mathbf{w}) p(\mathbf{w} \,|\, \rho) R(h) \, dh d\mathbf{w}.$$

1.2) The derivative of the expected return using log arithmic derivative $\nabla_\rho \log p(\mathbf{w} \,|\, \rho) = \dfrac{\nabla_\rho p(\mathbf{w} \,|\, \rho)}{p(\mathbf{w} \,|\, \rho)}$:

$$\nabla_\rho J(\rho) = \int\int p(h \,|\, \mathbf{w}) p(\mathbf{w} \,|\, \rho) \nabla_\rho \log p(\mathbf{w} \,|\, \rho) R(h) \, dh d\mathbf{w}.$$

1.3) The expectations over h and $\mathbf{w}$ are approximated by empirical averages:

$$\nabla_\rho \hat{J}(\rho) = \frac{1}{N} \sum_{n=1}^{N} \nabla_\rho \log p(\mathbf{w}_n \,|\, \rho) R(h_n).$$

2) We can acquire the gradient information for policy updates that are weighted by importance weight v:

$$\nabla_\rho \hat{J}_{\text{IW}}(\rho) := \frac{1}{N'} \sum_{n=1}^{N'} v(\mathbf{w}'_n) \nabla_\rho \log p(\mathbf{w}'_n \,|\, \rho) R(h'_n).$$

PGPE Policy Updates

In our work, we use PGPE-based policy updates [6], [33]. Policy parameter $\mathbf{w}$ is stochastically sampled from prior distribution $p(\mathbf{w} \mid \boldsymbol{\rho})$ with hyperparameter $\boldsymbol{\rho}$. In other words, the policy is deterministic, but its parameter is stochastic.

In PGPE, hyperparameter ρ is optimized to maximize expected return $J(\boldsymbol{\rho})$ (Table 1, 1.1). Optimal hyperparameter $\boldsymbol{\rho}^*$ is given by $\boldsymbol{\rho}^* := \text{argmax}_{\rho} J(\boldsymbol{\rho})$. In practice, a gradient method is used to find $\boldsymbol{\rho}^*$: $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \varepsilon \Delta\boldsymbol{\rho}$, where $\Delta\boldsymbol{\rho} = \nabla_{\rho} J(\boldsymbol{\rho})$ is the derivative of J with respect to $\boldsymbol{\rho}$ (Table 1, 1.2) and ε is the learning rate. We approximated derivative $\nabla_{\rho} J(\boldsymbol{\rho})$ by the empirical average (Table 1, 1.3).

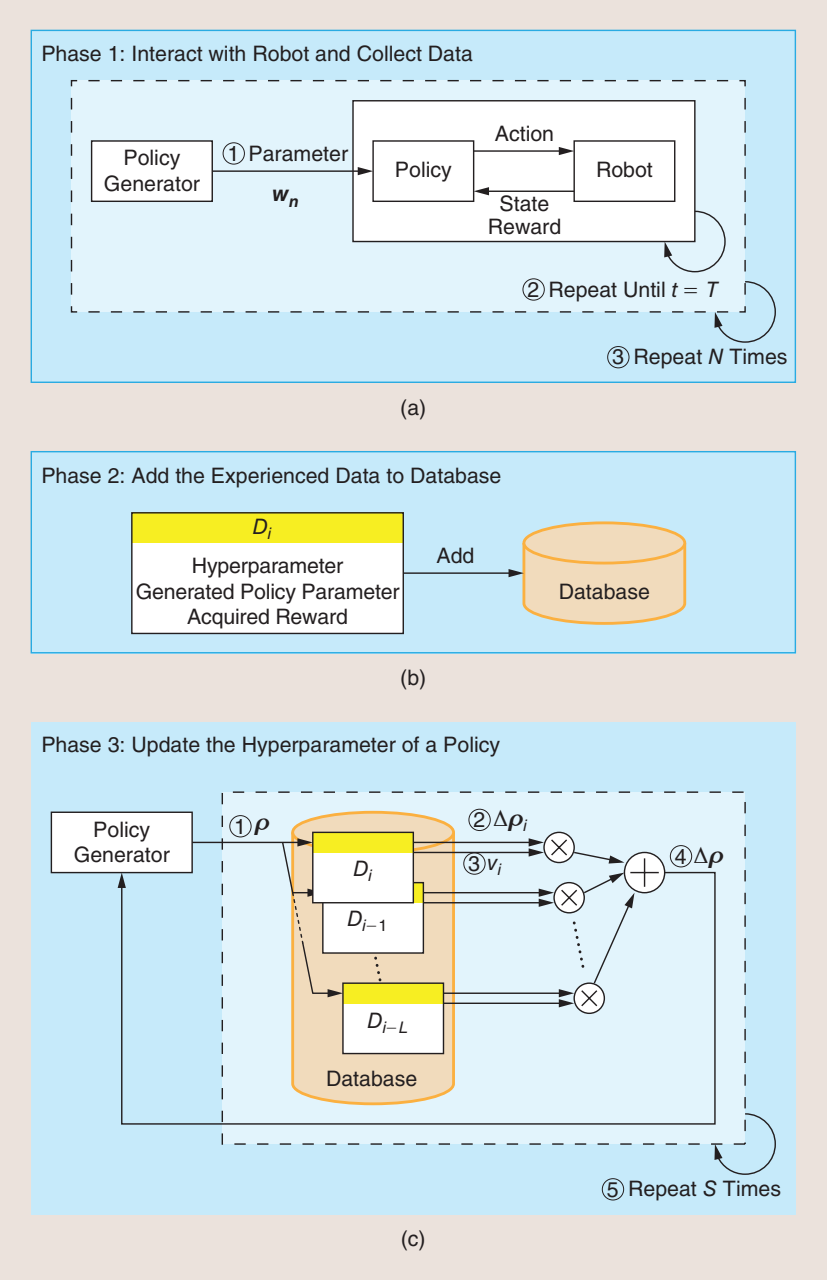

Figure 1. The flowchart of the proposed method: (a) Phase I, (b) Phase II, and (c) Phase III.

Efficient Reuse of Previous Experiences

Importance Weight

The original PGPE can be considered an on-policy algorithm [27], where the data collected from the current policy are used to estimate the policy gradients. However, to reuse the previous experiences, we need to evaluate the current policy with the data collected by the previous policies. To do this, we need an off-policy algorithm through which the data-collecting policy and the policy to be updated are different. Therefore, we use an off-policy version of the PGPE algorithm. In this off-policy method, importance weighting [8] is used to evaluate the previously collected data (experience) from the current policy's point of view. This method is called an IW-PGPE [16], [24], [26], [34].

The basic idea of importance weighting is to weight samples drawn from a sampling distribution to match the target distribution. For PGPE, importance weight v was defined for current hyperparameter $\boldsymbol{\rho}$ that was used in previous experiences:

$$v(\mathbf{w}') = \frac{p(\mathbf{w}' \mid \boldsymbol{\rho})}{p(\mathbf{w}' \mid \boldsymbol{\rho}')}. \tag{4}$$

This weight indicates how much the previous experience contributed to the current policy update. The approximated derivative of the expected return is then weighted by this importance weight for reusing the previous experiences. Table 1 shows the weighted derivatives using the previous experiences, where $\mathbf{w}'_n$ represents a policy parameter generated from previous hyperparameter $\boldsymbol{\rho}'$ and h'_n represents the trajectory of the previous experiences.

Learning Procedure

The learning procedure of our proposed method is shown in Figure 1, which repeats from Phases 1 to 3 until the learning performance is converged:

- *Phase 1:* Collect data in a real environment.
- *Phase 2:* Add the collected data to a database.
- *Phase 3:* Update the hyperparameters of the current policy using the stored data in the database.

In Phase 1, ① policy parameter w_n is sampled from prior distribution $p(\mathbf{w} \mid \boldsymbol{\rho})$. ② Then, a trajectory is acquired from the real environment using the policy with the sampled policy parameter. In ③, ① and

② are repeated N times. Note that for each trial, different policies with different policy parameters are used.

In Phase 2, the acquired data composed of N trials are added to a database. The hyperparameter, the sampled policy parameters, and the acquired reward are stored as ith data set D_i. The database stores the latest $L+1$ data sets. We limit the data size to avoid unnecessary computation for the parameter updates in Phase 3.

In Phase 3, the hyperparameter is updated based on all the data stored in the database. In ④, the weighted sum of each derived gradient for each dataset is used to update the policy (Table 1),

Finally, in ⑤ we recursively applied the procedure composed of ①, ②, ③, and ④ S times.

Recursive Reuse of Previous Experiences

Here we recursively use the IW-PGPE method so that we can more efficiently reuse the previously collected data. This recursive update, which has not been systematically explored for real robot learning, is a novel contribution of this study. By recalculating the importance weights introduced in (4), the previous experiences can be repeatedly reused for hyperparameter updates. Here, we introduce a new operator, H, which outputs the amount of updates of hyperparameter $\boldsymbol{\rho}$:

$$\Delta\boldsymbol{\rho} = H(\boldsymbol{\rho}, \{D\}_{i-L}^{i}), \tag{5}$$

$$= \frac{1}{L+1}\sum_{i'=i-L}^{i} \nabla_{\rho}\widehat{J}_{\mathrm{IW}}(\boldsymbol{\rho}, D_{i'}), \tag{6}$$

where D_i represents the ith experienced data needed for the derivative calculation. Note that for the current iteration, weighted derivative $\nabla_{\rho}\widehat{J}_{\mathrm{IW}}(\boldsymbol{\rho})$ equals nonweighted derivative $\nabla_{\rho}\widehat{J}(\boldsymbol{\rho})$, because importance weight $\nu(\mathbf{w}_n)$ is one. With this operator, a recursive formula for the hyperparameter updates is derived:

$$\boldsymbol{\rho}_{s+1} = \boldsymbol{\rho}_s + H(\boldsymbol{\rho}_s, \{D\}_{i-L}^{i}), \tag{7}$$

where $s = \{1, \ldots, S\}$ is the number of recursions.

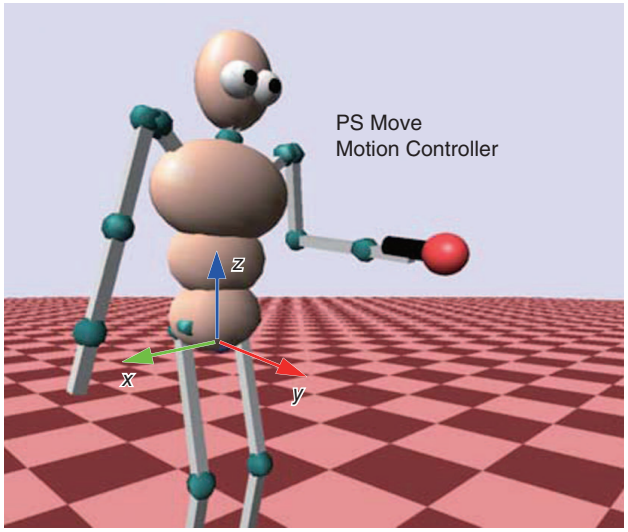

Figure 2. The humanoid robot simulator [31]. (Image courtesy of Prof. S. Schaal.)

With each recursion, the importance weight is estimated automatically and adaptively. The importance weight gradually becomes small depending on prior distribution $p(\mathbf{w}\,|\,\boldsymbol{\rho})$. In other words, the number of recursions is adapted based on the relationship between the trajectory generated from the previous and current policies.

Cart-Pole Swing-Up Experiment

We first applied our proposed approach to a simulated environment and tested it using a humanoid robot simulator (Figure 2). In addition, we developed a virtual dynamics simulator and our robot interacted with its virtual environment using a PS Move motion controller [Figure 3(a)]. Since the cart-pole dynamics are underactuated, the robot can only apply force to

Small variations of the movement trajectories were observed due to the uncertainty of the real system.

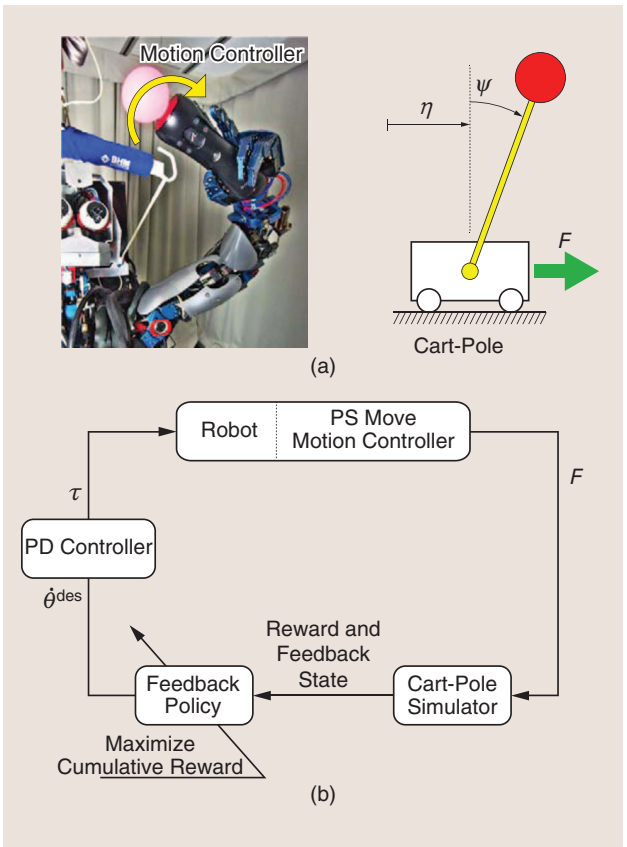

Figure 3. (a) The robot controls a motion controller to swing-up a cart-pole simulated in virtual environment from a hanging position. The angular velocity (yellow arrow) is converted into the cart's driving force. (b) The diagram of the experimental system. A desired trajectory of controlled joint (θdes) is given to the robot, and the motion controller's angular velocity is given to the cart-pole simulator. The reward is based on cart-pole state (η, ψ), and policy is optimized to maximize cumulative reward. (Photo courtesy of ATR.)

the cart. The motion controller's angular velocity is converted into the cart's driving force. In this task, the robot controls a cart-pole to swing up from a hanging position by waving the motion controller and also controls the five joints of its upper body (the torso's yaw joint, three joints of the left-shoulder, and the left-elbow joint). The length of one trial was 2 s.

The performance reached a maximum value around the 120th iteration.

For this cart-pole swing-up task, we implemented a feedback policy (2) that outputs a desired joint angular velocity by a mixture of local feedback policies with basis functions defined in the cart-pole state space [13]. The derived desired joint velocities $\theta^{\dot{\text{des}}}$ were converted into joint torques using a proportional-derivative (PD) controller (see also Table 2). Here, the state-dependent matrix in (2) was defined as

$$\mathbf{W}^{\text{fb}} = \tilde{\mathbf{W}}_{\text{fb}}\tilde{\mathbf{Z}}, \tag{8}$$

where $\tilde{\mathbf{Z}} = [\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_M](\tilde{z}_m = [(\mathbf{z} - \mathbf{c}_m)^\top, 1]^\top$ and $\mathbf{z}(t) = [\psi(t)\ \dot{\psi}(t)\ \dot{\eta}(t)]^\top)$ $\tilde{\mathbf{W}}_{\text{fb}}$ is a parameter matrix. As shown in Figure 3(a), angle position ψ, the angular velocity of pole $\dot{\psi}$, and the horizontal velocity of cart $\dot{\eta}$ were considered in the feedback policy. We used 12 basis functions $(M = 12)$ for each controlled joint. Each basis function is formulated, as shown in Table 2 with center $\mathbf{c}_m^{\text{fb}}$, which is allocated with grid pattern $(4 \times 3 \times 1)$ in the range of ψ: $0, \pi/4, \pi/2, 3\pi/4$ rad, $\dot{\psi}$: -2π, 0, 2π rad/s, and $\dot{\eta}$: 0.

The force applied to cart F was derived according to the PS Move motion controller's angular velocity.

We also defined the objective function as the sum of state-dependent reward $q(\mathbf{x}(t))$ and cost of action $c(\mathbf{x}(t), \mathbf{u}(t))$. Here $\mathbf{x}$ and $\mathbf{u}$ are the state vector and the desired position of each joint. A state-dependent reward is given based on the pole's angle:

$$q(\mathbf{x}(t)) = \exp[-\alpha\psi(t)^2], \tag{9}$$

where parameter $\alpha = 0.5$ is a constant. When the pole's angle is upright $(\psi = 0)$, the state-dependent reward takes a maximum value. The control cost is given based on the difference between the actual and desired angles:

$$c(\mathbf{x}(t), \mathbf{u}(t)) = \beta\sum_{j=1}^{5}(\theta_j(t) - \theta_j^{\text{des}}(t))^2, \tag{10}$$

where $\beta(= 5 \times 10^{-4})$ is a constant and j is an index of the controlled joint.

Table 2. The supplemental equations about basis functions.

1) The basis functions for feedback policy are defined in state space of $\mathbf{z}$ with center $\mathbf{c}_m^{\text{fb}}$ and size $\sum_m$:

$$\phi_m^{\text{fb}}(\mathbf{z}(t)) = \frac{g_m^{\text{fb}}(\mathbf{z}(t))}{\sum_{m'=1}^{M} g_{m'}^{\text{fb}}(\mathbf{z}(t))},$$

$$g_m^{\text{fb}}(\mathbf{z}(t)) = \exp\left[-\frac{1}{2}(\mathbf{z}(t) - \mathbf{c}_m^{\text{fb}})^\top \sum_m^{-1}(\mathbf{z}(t) - \mathbf{c}_m^{\text{fb}})\right].$$

2) The basis functions are defined along the time trajectory with center c_m^{ff} and size σ:

$$\phi_m^{\text{ff}}(t) = \frac{g_m^{\text{ff}}(t)}{\sum_{m'=1}^{M} g_{m'}^{\text{ff}}(t)},$$

$$g_m^{\text{ff}}(t) = \exp\left[\frac{1}{2\sigma^2}(t - c_m^{\text{ff}})^2\right].$$

3) The PD controller outputs torque command τ for each joint to track the desired trajectories with positive constants (K_{P} and K_{D}):

$$\tau(t) = -K_{\text{P}}(\theta(t) - \theta^{\text{des}}(t)) - K_{\text{D}}(\dot{\theta}(t) - \dot{\theta}^{\text{des}}(t)).$$

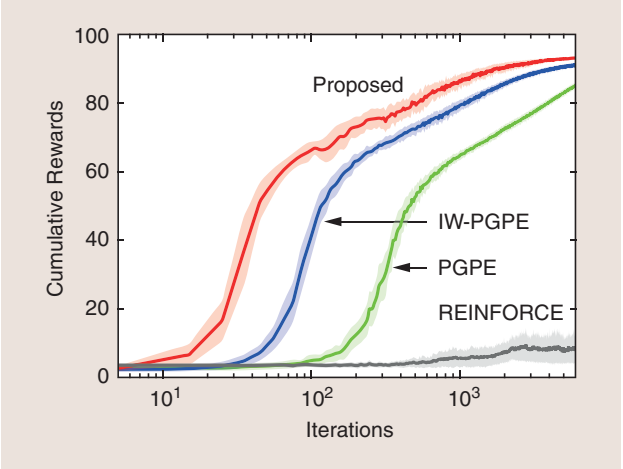

Figure 4. The simulation results of cart-pole swing-up. The mean and standard deviation of cumulative rewards are plotted. Blue and red lines are learning curves with and without recursive updates formulated in the "Recursive Reuse of Previous Experiences" section.

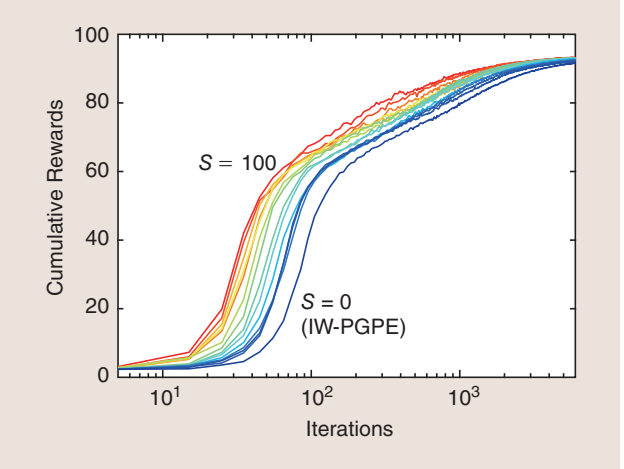

Figure 5. The simulation results of different numbers of recursions. The difference of color represents the different parameters, $S = 0$ (blue), 1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 60, 80, 100 (red).

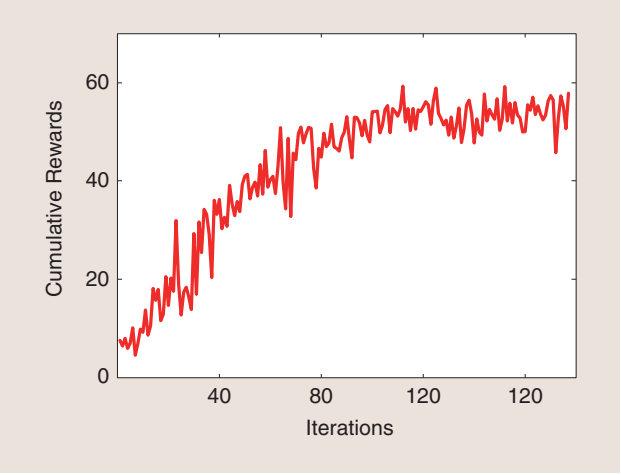

Figure 6. The learning performance of the cart-pole swing-up task using real humanoid robot, where reward function is based on previous work of (9) and (10).

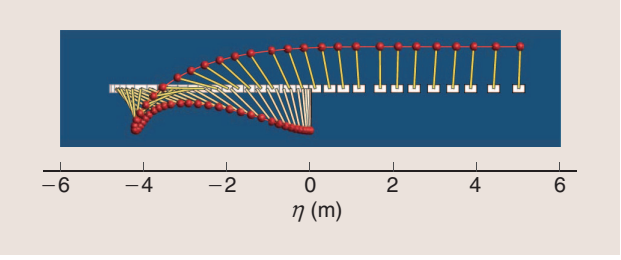

Figure 7. The acquired behavior of the cart-pole in the swing-up task.

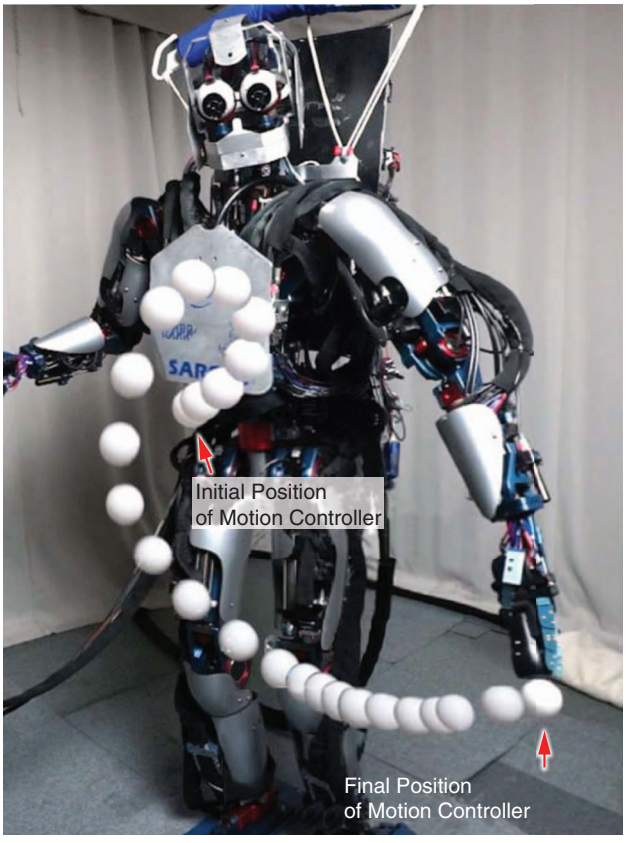

Figure 8. The acquired behavior of a humanoid robot in the cart-pole swing-up task. (Photo courtesy of ATR.)

Hyperparameter $\boldsymbol{\rho}$ was updated with a discount factor of $\gamma = 0.999$, a learning rate of $\varepsilon = 0.05$, ten trials in one iteration of $N = 10$, and 100 iterations for reuse ($L = 100$). We also employed the recursive updates introduced in the "Recursive Reuse of Previous Experiences" section and terminated each recursion at the 100th time ($S = 100$). To efficiently reuse the old data, we set sufficiently large numbers to L and S, so that the importance weight becomes close to zero when we use the oldest data.

Small variations of the movement trajectories were observed due to the uncertainty of the real system.

To evaluate the average learning performances of our proposed approach, we compared the following methods:

- REINFORCE: The REINFORCE algorithm [25]
- PGPE: Standard PGPE [6]
- IW-PGPE: Standard IW-PGPE [34]
- Proposed: Proposed recursive IW-PGPE.

For each method, we updated the parameters every ten trials and used the same learning rate.

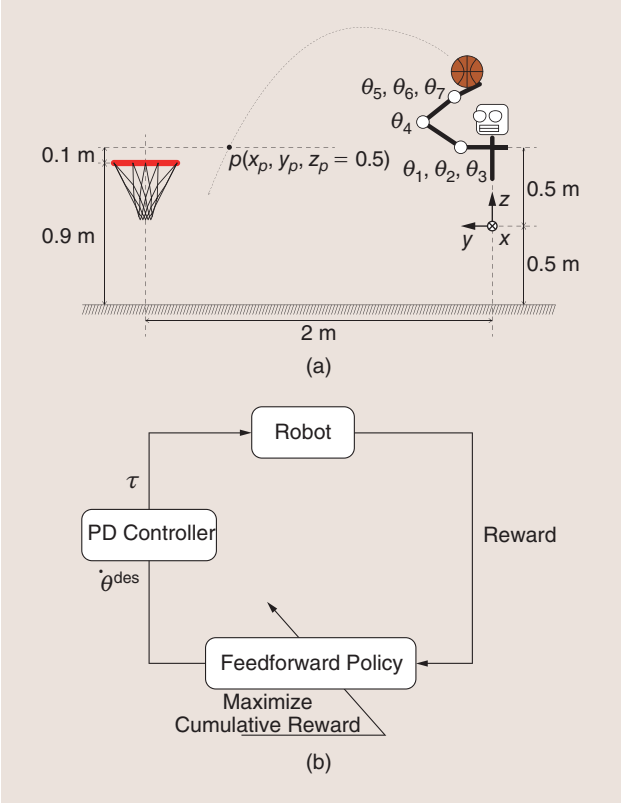

Figure 9. (a) The setup of the basketball-shooting task: horizontal distance between CB-i and the goal was 2 m, and goal's height was 0.9 m. A reward was given based on the distance between the ball and the goal when the ball crosses horizontal plane at $z = 0.5$ m. The ball's position was observed by a stereo camera. (b) A schematic diagram of the experimental system.

The learning results are shown in Figure 4. The horizontal and vertical axes are the learning iterations and the cumulative rewards. The gray, green, blue, and red lines represent the performances of REINFORCE, PGPE, IW-PGPE, and the proposed method, respectively. The mean and standard deviation over five simulation runs are plotted. Since system noise is not considered in our simulation, the standard deviation of the learning curves was relatively small. The cart-pole swing-up policy was improved by our proposed method much faster than by the other approaches.

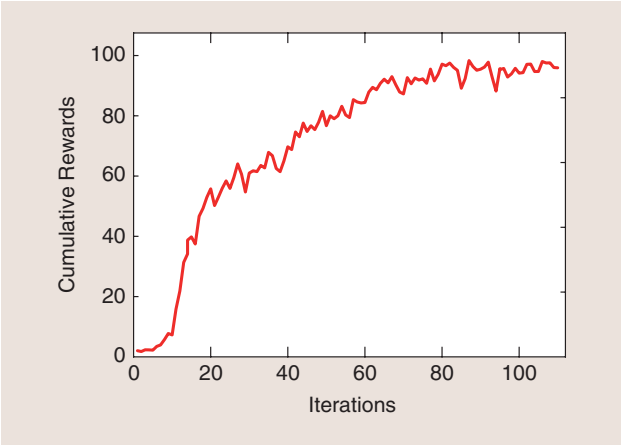

Figure 10. The learning performance of basketball-shooting task using a real humanoid robot.

The effects of the number of recursions (S) are shown in Figure 5. We conducted five simulation runs for each parameter $S = (0, 1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 60, 80, 100)$ with different random seeds. The improvements of the task performance became faster with larger S. In this simulation, we did not observe any slowdown or bias in the performance improvement even with a relatively large S value. This is possibly due to the effect of the importance weight.

We also evaluated our proposed method using our real humanoid robot. The learning performance is shown in Figure 6. The horizontal and vertical axes are the iterations and the mean of the cumulative rewards in each iteration. The performance reached a maximum value around the 120th iteration. Figures 7 and 8 show the acquired swing-up movements, although the pitch angular velocity of the PS Move corresponds to the force input to the cart. In this cart-pole swing-up task using the real-virtual hybrid environment, since the robot did not know which PS Move sensor corresponded to the input to the cart, it explored the movements by using the five joints of its upper body. As a result, the hand position of the robot moved around the three-dimensional Cartesian space (see also Figure 8).

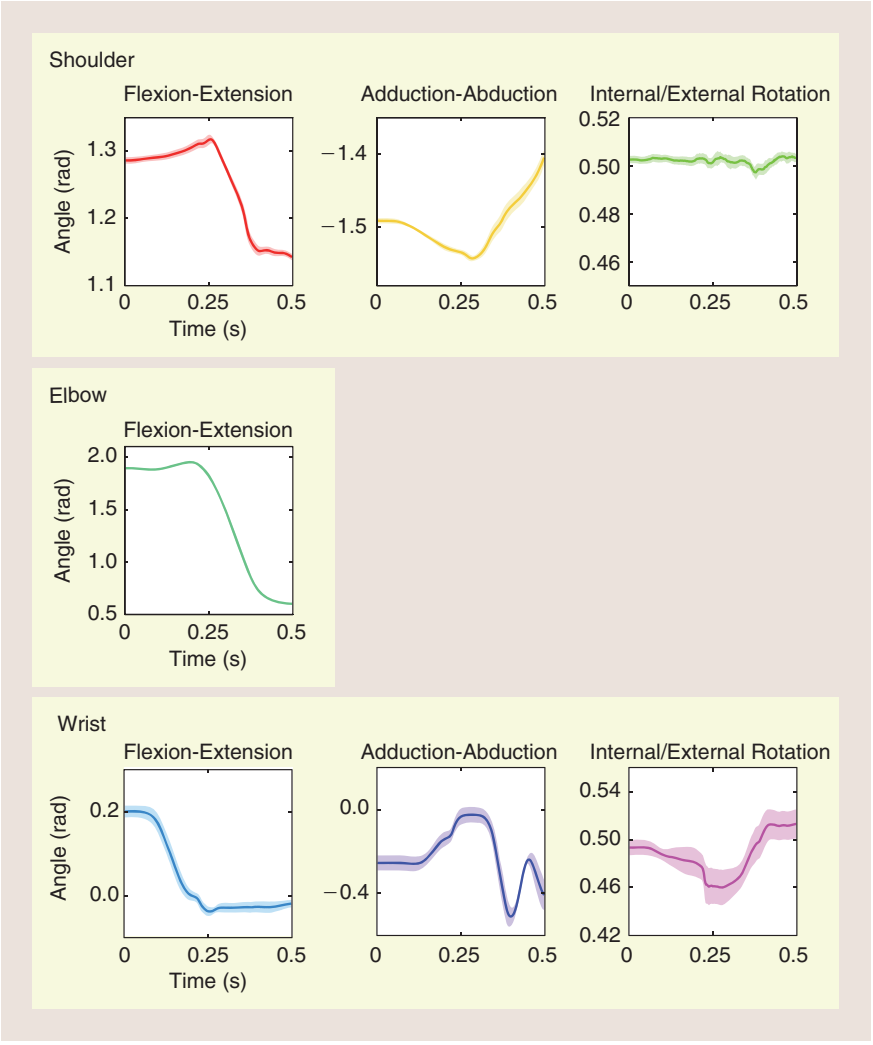

Figure 11. The generated trajectories of the right arm in the basketball-shooting task. Horizontal and vertical axes are time and joint angle. Mean and standard deviation over 50 trials are plotted.

Basketball-Shooting Experiment

Next, we tested the basketball-shooting task using all seven degrees of freedom of the robot's right arm. The policy outputs the desired angular velocity for seven joints of the right arm: three shoulder joints, the elbow joint, and three wrist joints. Since the robot does not have any sensors to detect the state of the ball when it is on its hand, we simply used the feedforward policy model in (3). Desired angular velocities $\dot{\theta}_j^{\mathrm{des}}(j = \{1, 2, 3, 4, 5, 6, 7\})$ are learned through the trials. A simple nominal trajectory was provided for the elbow joint that slowly extended it. The center of basis functions c_m^{ff} was defined along the

Figure 12. The acquired behavior of the humanoid robot in the basketball-shooting task. (Photos courtesy of ATR.)

time at regular intervals. The length of one trial was 0.5 s with 0.002-s time steps [13].

The task setup is shown in Figure 9. The position of the goal was 2 m from the robot. When the ball (weight: 0.5 kg, radius: 0.11 m) crossed a horizontal plane at height of 1 m, a terminal reward is given based on the horizontal distance between the ball and the basket:

$$\Phi = \alpha_0 \exp[-\alpha_1 d^2], \tag{11}$$

where the distance is defined as $d^2 = p_x^2 + (p_y - 2)^2$. The ball's positions on the horizontal plane are p_x and p_y. $\alpha_0 (= 100)$and $\alpha_1 (= 5)$ are constant parameters. The control cost was given as follows:

$$c(\mathbf{x}(t), \mathbf{u}(t)) = \beta \sum_{j=1}^{7} (\theta_j(t) - \theta_j^{\mathbf{des}}(t))^2, \tag{12}$$

where constant parameter β was 5×10^{-4}. The state-dependent cost was given only in the terminal condition.

For one data set, the numbers of trials, databases, and recursive updates were $N = 10$, $L = 10$, and $S = 10$, respectively. The policy was updated with discount factor $\gamma = 0.999$ and learning rate $\varepsilon = 0.05$.

The learning performance is shown in Figure 10. The learning converged around the 80th iteration. After the learning stage, the successful shooting rate was 100%; 50 of 50 shots went in.

The means and standard deviations of all the joint trajectories of the humanoid right arm in the basketball-shooting task are shown in Figure 11. This result shows that properly coordinated joint movements had to be learned for a successful basketball-shooting task. On the other hand, small variations of the movement trajectories were observed due to the uncertainty of the real system. The acquired behavior after the learning is shown in Figure 12.

Discussion

In our PGPE-based approach, the policy's output is deterministic and does not need noisy control output for exploration. Thus, the control output can be smooth, and smooth control output is highly suitable for robot hardware. In comparison, the previously proposed model-based policy update algorithms must first identify the real environment [1], [4], [14], [17], [22], [35], [36]. Therefore, exploratory noisy input using a real robot is mandatory, although successful applications of model-based RL methods to real systems have been presented [14], [18]. In addition, the previous approaches used an approximated dynamics model with a function approximator, such as Gaussian process

We applied our proposed method to a challenging basketball-shooting task in a real environment.

regression [2]. With a function approximator, the sampled data from the approximated model can be generated by inappropriate interpolation or extrapolation that improperly updates the policy parameters. In addition, if we aggressively derive the analytical gradient of the approximated model to update the policy, the approximated gradient might be far from the true gradient of the objective function due to the model approximation error. If we consider using these function approximation methods for high-dimensional systems like humanoid robots, this problem becomes more serious due to the difficulty of approximating high-dimensional dynamics models with a limited amount of data sampled from real systems. On the other hand, if the environment is extremely stochastic, a limited amount of previously acquired data might not be able to capture the real environment's property and could lead to inappropriate policy updates. However, rigid dynamics models, such as a humanoid robot model, do not usually include large stochasticity. Therefore, our approach is suitable for a real robot learning for high-dimensional systems like humanoid robots.

Efficiently reusing previous experiences is crucial to improve its behavioral policies without actually interacting with real environments.

Figure 13. The humanoid robot CB-i [7]. (Photo courtesy of ATR.)

Moreover, applying RL to actual robot control is difficult, since it usually requires many learning trials that cannot be executed in real environments, and the real system's durability is limited. Previous studies used prior knowledge or properly designed initial trajectories to apply RL to a real robot and improved the robot controller's parameters [1], [4], [10], [19], [32].

We applied our proposed learning method to our humanoid robot [7] (Figure 13) and show that it can accomplish two different movement-learning tasks without any prior knowledge for the cart-pole swing-up task or with a very simple nominal trajectory for the basketball-shooting task.

The proposed recursive use of previously sampled data to improve policies for real robots would also be useful for other policy search algorithms, such as reward weighted regression [11] or information theoretic approaches [12], and it might be interesting to investigate how these combinations work as a future study.

Conclusions

In this article, we proposed reusing the previous experiences of a humanoid robot to efficiently improve its task performance. We proposed recursively using the off-policy PGPE method to improve the policies and applied our approach to cart-pole swing-up and basketball-shooting tasks. In the former, we introduced a real-virtual hybrid task environment composed of a motion controller and virtually simulated cart-pole dynamics. By using the hybrid environment, we can potentially design a wide variety of different task environments. Note that complicated arm movements of the humanoid robot need to be learned for the cart-pole swing-up. Furthermore, by using our proposed method, the challenging basketball-shooting task was successfully accomplished.

Future work will develop a method based on a transfer learning [28] approach to efficiently reuse the previous experiences acquired in different target tasks.

Acknowledgment

This work was supported by MEXT KAKENHI Grant 23120004, MIC-SCOPE, ``Development of BMI Technologies for Clinical Application" carried out under SRPBS by AMED, and NEDO. Part of this study was supported by JSPS KAKENHI Grant 26730141. This work was also supported by NSFC 61502339.

References

[1] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann, "Data-efficient contextual policy search for robot movement skills," in *Proc. National Conf. Artificial Intelligence*, 2013.

[2] C. E. Rasmussen and C. K. I. *Williams Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[3] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *Proc. 14th Int. Conf. Machine Learning*, 1997, pp. 12–20.

[4] C. G. Atkeson and J. Morimoto, "Nonparametric representation of policies and value functions: A trajectory-based approach," in *Proc. Neural Information Processing Systems*, 2002, pp. 1643–1650.

[5] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *J. Machine Learning Res.*, vol. 5, pp. 1471–1530, Nov. 2004.
[6] F. Sehnke, C. Osendorfer, T. Rückstiess, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Netw.*, vol. 23, no. 4, pp. 551–559, 2010.
[7] G. Cheng, S. Hyon, J. Morimoto, A. Ude, G. H. Joshua, G. Colvin, W. Scroggin, and C. J. Stephen, "Cb: A humanoid research platform for exploring neuroscience," *Adv. Robotics*, vol. 21, no. 10, pp. 1097–1114, 2007.
[8] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*. Berlin, Germany: Springer-Verlag, 1996.
[9] H. Hachiya, J. Peters, and M. Sugiyama, "Reward weight regression with sample reuse for direct policy search in reinforcement learning," *Neural Comput.*, vol. 23, no. 11, pp. 2798–2832, 2011.
[10] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, 2006, pp. 2219–2225.
[11] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proc. Int. Conf. Machine Learning*, 2007, pp. 745–750.
[12] J. Peters, K. Mülling, and Y. Altün, "Relative Entropy Policy Search," in *Proc. 24th AAAI Conf. Artificial Intelligence*, 2010, pp. 1607–1612.
[13] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989.
[14] J. Morimoto, C. G. Atkeson, "Nonparametric representation of an approximated Poincare map for learning biped locomotion," *Auton. Robots*, vol. 27, no. 2, pp. 131–144, 2009.
[15] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 79, pp. 1180–1190, 2008.
[16] L. Weaver and N. Tao, "The optimal reward baseline for gradient-based reinforcement learning," in *Proc. 7th Conf. Uncertainty Artificial Intelligence*, 2001, pp. 538–545.
[17] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data efficient approach to policy search," in *Proc. Int. Conf. Machine Learning*, 2011, , pp. 465–472.
[18] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, 2015.
[19] N. Sugimoto and J. Morimoto, "Phase-dependent trajectory optimization for CPG-based biped walking using path integral reinforcement learning," in *Proc. IEEE/RAS Int. Conf. Humanoid Robots*, 2011, pp. 255–260.
[20] N. Sugimoto, M. Haruno, K. Doya, and M. Kawato, "MOSAIC for multiple-reward environments," *Neural Comput.*, vol. 24, no. 3, pp. 577–606, 2012.
[21] N. Sugimoto, J. Morimoto, S. Hyon, and M. Kawato, "eMOSAIC Model for Humanoid Robot Control," *Neural Netw.*, vol. 29–30, pp. 8–19, May 2012.
[22] N. Sugimoto and J. Morimoto, "Trajectory-model-based reinforcement Learning: Application to bimanual humanoid motor learning with a closed-chain constraint," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2013, pp. 429–434.
[23] N. Sugimoto, V. Tangkaratt, T. Wensveen, T. Zhao, M. Sugiyama, and J. Morimoto, "Efficient reuse of previous experiences in humanoid motor learning," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 554–559.
[24] R. J. Williams, "Toward a theory of reinforcement-learning connectionist systems," Tech. Rep. NU-CCS-88-3, College of Computer Science, Northeastern Univ., Boston, MA, 1988.
[25] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, 1992.
[26] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph.D. dissertation, Univ. Massachusetts, 1984.
[27] R. S. Sutton and G. A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[28] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowledge Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.
[29] S. Kakade, "A natural policy gradient," in *Proc. Advances Neural Information Processing Systems 14*, 2002, pp. 1531–1538.
[30] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Comput.*, vol. 10, no. 8, pp. 2047–2084, 1998.
[31] S. Schaal, "The SL simulation and real-time control software package," Univ. of Southern California, Los Angeles, CA, Tech. Rep., 2009.
[32] T. Matsubara, J. Morimoto, J. Nakanishi, M. Sato, and K. Doya, "Learning CPG-based biped locomotion with a policy gradient method," *Robot. Auton. Syst.*, vol. 54, no. 11, pp. 911–920, 2006.
[33] T. Zhao, H. Hachiya, G. Niu, and M. Sugiyama, "Analysis and improvement of policy gradient estimation," *Neural Netw.*, vol. 26, pp. 118–129, Feb. 2012.
[34] T. Zhao, H. Hachiya, V. Tangkaratt, J. Morimoto, and M. Sugiyama, "Efficient sample reuse in policy gradients with parameter-based exploration," *Neural Comput.*, vol. 25, no. 6, pp. 1512–1547, 2013.
[35] V. Tangkaratt, S. Mori, T. Zhao, J. Morimoto, and M. Sugiyama, "Model-based policy gradients with parameter-based exploration by least-squares conditional density estimation," *Neural Netw.*, vol. 57, pp. 128–140, Sept. 2014.
[36] S. Schaal, C. G. Atkeson, "Learning control in robotics," *IEEE Robot. Automat. Mag.*, vol. 17, no. 2, pp. 20–29, 2010.

Norikazu Sugimoto, National Institute of Information and Communications Technology, Osaka, Japan. E-mail: xsugi@nict.go.jp.

Voot Tangkaratt, The University of Tokyo, Japan. E-mail: voot@ms.k.u-tokyo.ac.jp.

Thijs Wensveen, Delft University of Technology, The Netherlands. E-mail: thijswensveen@gmail.com.

Tingting Zhao, the Tianjin University of Science and Technology, China. E-mail: tingting@tust.edu.cn.

Masashi Sugiyama, The University of Tokyo, Japan. E-mail: sugi@k.u-tokyo.ac.jp.

Jun Morimoto, ATR Computational Neuroscience Labs, Kyoto, Japan. E-mail: xmorimo@atr.jp.