Acquisition of Stand-up Behavior by a Real Robot using Hierarchical Reinforcement Learning

Jun Morimoto XMORIMO@ERATO.ATR.CO.JP
Nara Institute of Science and Technology; Kawato Dynamic Brain Project, JST
2-2 Hikaridai Seika-cho Soraku-gun Kyoto 619-0288 JAPAN

Kenji Doya DOYA@CTR.ATR.CO.JP
ATR International; CREST, JST
2-2 Hikaridai Seika-cho Soraku-gun Kyoto 619-0288 JAPAN

Abstract

In this paper, we propose a hierarchical reinforcement learning architecture for a robot with large degrees of freedom. In order to enable learning in a practical numbers of trials, we introduce a low-dimensional representation of the state of the robot for higher-level planning. The upper level learns a discrete sequence of sub-goals in a low-dimensional state space for achieving the main goal of the task. The lower-level modules learn local trajectories in the original high-dimensional state space to achieve the sub-goal specified by the upper level. We applied the hierarchical architecture to a three-link, two-joint robot for a task of learning to stand up by trial and error. The upper-level learning was implemented by Q learning, while the lower-level learning was implemented by a continuous actor-critic method. The robot successfully learned to stand up within 750 trials in simulation and then in an additional 170 trials using real hardware.

1. Introduction

Recently, there have been many attempts at applying reinforcement learning (RL) algorithms to acquisition of goal-directed behaviors in autonomous robots. However, a crucial issue in applying RL to real-world robot control is the curse of dimensionality. For example, control of a humanoid robot easily involves a forty or higher dimensional state space. Thus, the usual way of quantizing the state space with grids easily breaks down. We have recently developed RL algorithms for dealing with continuous-time, continuous-state control tasks without explicit quantization of state and time (Doya, 2000). However, the methods for high-dimensional function approximation and for global exploration remain open problems. The speed of learning is crucial in applying RL to real hardware control because, unlike in idealized simulations, such non-stationary effects as sensor drifts and mechanical aging are not neglectable and learning has to be quick enough to keep track of such changes in the environment.

In this article, we propose a hierarchical RL architecture that realizes practical learning speed in real hardware control tasks. Hierarchical RL methods have been developed for creating reusable behavioral modules (Singh, 1992; Tham, 1995; Digney, 1998), solving partially observable Markov decision problems (POMDP's) (Wiering & Schmidhuber, 1997), and for improving learning speed (Dayan & Hinton, 1993; Kimura & Kobayashi, 1999).

Most previous studies of hierarchical RL consider a 2-D maze-like state space and use coarse and fine grain quantization of the state space. However, in a high-dimensional state space, even the coarsest quantization into two bins in each dimension would create a prohibitive number of states. Thus, in designing a hierarchical RL architecture in high-dimensional space, dimension reduction of the state space is mandatory (Morimoto & Doya, 1998a).

In this study, we propose a hierarchical RL architecture in which the upper-level learner globally explores sequences of sub-goals in a low-dimensional state space, while the lower-level learners optimize local trajectories in the high-dimensional state space.

As a concrete example, we consider a "stand-up" task for a two-joint, three-link robot (see Figure 1). The

goal of the task is to find a path in a high-dimensional state space that links a lying state to an upright state under the constraints of the system dynamics. The robot is a non-holonomic system, as there is no actuator linking the robot to the ground, and thus trajectory planning is non-trivial. The geometry of the robot is such that there is no static solution; the robot has to stand up dynamically by utilizing the momentum of its body.

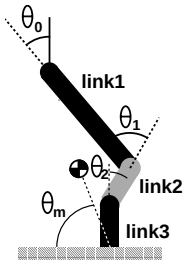


Figure 1. Robot configuration. θ_0:pitch angle, θ_1:hip joint angle, θ_2:knee joint angle, θ_m:the angle of the line from the center of mass to the center of the foot.

2. Hierarchical Reinforcement Learning

In this section, we propose a hierarchical RL architecture for non-linear control problems. The basic idea is to decompose a non-linear problem in a high-dimensional state space into two levels: a non-linear problem in a lower-dimensional space and nearly linear problems in the high-dimensional space (see Figure 2).

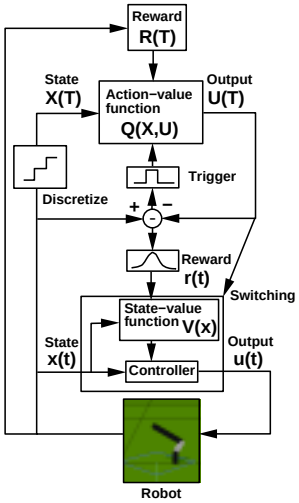


Figure 2. Hierarchical reinforcement learning architecture

2.1 Task Decomposition by Sub-goals

In the upper level, the learner deals with the entire task. The reward for the upper-level learner is given by the achievement of the entire task. In the lower level, each learner deals with a sub-task. The reward for the lower-level learner is given by the achievement of a given sub-goal. An action of the upper-level learner is the selection of the next sub-goal for the lower level. An action of the lower-level learner is the command for the actuators. The upper-level learner is activated when the lower-level learner achieves the current sub-goal. Then, the upper-level learner takes a new action, which is given as a new sub-goal for the lower-level learner. The state variables in the lower level are the physical variables, while those in the upper level are lower-dimensional state variables. The choice of low-dimensional state variables is a important issue in hierarchical RL. In general, the use of task-oriented kinematic variables, such as the positions of the end effector and the center of mass, in the upper level would be appropriate. In the stand-up task, we chose the angles of the joints and the center of mass as the state variables. In other words, we chose kinematic variables in the upper level and dynamic variables in the lower level as the input.

2.2 Upper-level Learning

In the upper level, the learner explores the whole relevant area of a low-dimensional sub-space of the original high-dimensional state space. In order to facilitate global search, the state space is coarsely discretized and the actions are defined as transitions to nearby states. We then use the $Q(\lambda)$-learning method (Peng & Williams, 1996) to learn a sub-goal sequence to achieve the goal of the entire task. Thus, a reward $R(T)$ to the upper level is given by the success or failure of the entire task, and the action-value function $Q(\boldsymbol{X}(T), \boldsymbol{U}(T))$ predicts accumulated future reward if the learner takes the action $\boldsymbol{U}(T)$ at the state $\boldsymbol{X}(T)$.

In the stand-up task, we chose the posture of the robot $\mathbf{X}=(\theta_m, \theta_1, \theta_2)$ as the state variables (see Figure 1). The action is given by $\mathbf{U}(T) = (U_m(T), U_1(T), U_2(T))$ whose components are integers. Then, the desired posture of the robot $\hat{X}_i(T) = X_i(T) + \Delta X_i U_i(T)$ is sent to the lower level as the next sub-goal, where ΔX_i is the action step size $(i = m, 1, 2)$.

The upper-level learner chose an action using Boltzmman distribution (Sutton & Barto, 1998). Thus we have

$$P(U(T) = a) = \frac{\exp[\beta Q(\mathbf{X}(T), a)]}{\sum_{b \in \mathcal{A}(\mathbf{X})} \exp[\beta Q(\mathbf{X}(T), b)]} \qquad (1)$$

where $\mathcal{A}(\mathbf{X})$ is the set of possible actions at state $\mathbf{X}$ and β is a parameter that controls the randomness in action selection for exploration. We define the reward for the upper-level learner as follows.

$$R(T) = R_{main} + R_{sub}$$

$$R_{main} = \begin{cases} 1 & \text{(on success of stand-up)} \\ 0 & \text{(on failure)} \end{cases}$$

$$R_{sub} = \begin{cases} 1 & \text{(final goal achieved)} \\ 0.25(\frac{Y}{L}+1) & \text{(subgoal achieved)} \\ 0 & \text{(on failure)} \end{cases}$$

where Y is the height of the head of the robot at a sub-goal posture and L is total length of the robot. The final goal is the upright stand-up posture $(90, 0, 0)$ [deg]. When the robot achieves a sub-goal, the upper-level learner gets a reward of less than 0.5. Note that archiving the final goal is a necessary but not sufficient condition of successful stand-up because there is the case of the robot falling down after achieving the final goal.

When the robot reaches to a neighborhood of the sub-goal, the next sub-goal is selected and the action-value function is updated in the upper level. When the robot stands up stably, we then think the stand-up task is accomplished, otherwise (e.g. when the robot falls down, or when a time limit has reached before the robot successfully stands up), we think the robot fails to stand-up.

2.3 Lower-level Learning

In the lower level, the learner explores local areas of the high-dimensional state space without discretization. The lower-level learner learns to achieve the sub-goal specified by the upper level from any given initial state. Because each sub-goal is defined in the low-dimensional state space of the upper level, the sub-goal is not a point but a hyper-plane in the high-dimensional state space of the lower level. We use the continuous $TD(\lambda)$-learning with the actor-critic method (Doya, 2000) to learn the control command sequence. In addition, we use an Incremental Normalized Gaussian Network (INGnet) for implementing the actor and the critic (Morimoto & Doya, 1998b). A reward $r(t)$ is given to the lower level by the achievement of the sub-goal specified by the upper level (Dayan & Hinton, 1993).

In continuous $TD(\lambda)$-learning of the lower-level learner, the state-value function $V(\mathbf{x}(t))$ predicts the accumulated future reward at state $\mathbf{x}(t)$, while the control function $u_j(t) = u_{\max} h(f_j(\mathbf{x}(t)) + \sigma n_j(t))$ specifies a nonlinear feedback control law, where $h(x) = \frac{\pi}{2}\arctan(\frac{2}{\pi}x)$ is a sigmoid function to saturate output with maximum torque $u_{\max}$, and $\sigma n_j(t)$ is a noise term for exploration. We use INGnets for the critic and the actor.

In the stand-up task, we chose the pitch and joint angles $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2)$ and the corresponding angular velocities $\dot{\boldsymbol{\theta}} = (\dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2)$ as state variables $\mathbf{x}(t) = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ and chose torque $\mathbf{u}(t) = (\tau_1, \tau_2)$ for the two joints as the action variables. The output torque is the sum of two controllers, a linear servo controller and a non-linear feedback controller $f_i(\mathbf{x})$, which is acquired by the lower-level actor:

$$\tau_j = u_{\max} h\left(\frac{1}{u_{\max}}(k(\hat{\theta}_j - \theta_j) - b\dot{\theta}_j) + f_j(\mathbf{x}) + \sigma n_j\right) \tag{2}$$

where $k = 0.26$ [Nm/deg] and $b = 0.017$ [Nms/deg] are feedback gains, and $u_{\max} = 24$ [N.m] is maximum torque.

In this study, we used different lower-level learners for different sub-goals. When the robot reaches the neighborhood of a sub-goal ($||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}|| < 10$ [deg]), the upper-level learner switches the current lower-level learner module to the next one according to the choice of next sub-goals (see Fig. 2). Thus, one lower-level actor takes control until either the robot achieves the sub-goal, a time limit is reached, or the robot falls down. We used two types of reward for the lower level. One is given during the control according to the distance from the current posture $\boldsymbol{\theta}$ to the sub-goal posture $\hat{\boldsymbol{\theta}}(= \boldsymbol{U})$ given by the upper level

$$r(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \exp\left(-\frac{||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||^2}{s_\theta^2}\right) - 1, \tag{3}$$

where $s_\theta = 30$ [deg] gives the width of the reward function. Additional reward is given at the end of the control by the distance from the current pitch and joint angular velocity $\dot{\boldsymbol{\theta}}$ to the desired values $\hat{\dot{\boldsymbol{\theta}}}$ that are set by the memory of successful trials

$$r(t) = \begin{cases} \exp\left(-\frac{||\dot{\boldsymbol{\theta}}(t) - \hat{\dot{\boldsymbol{\theta}}}||^2}{s_{\dot{\theta}}^2}\right) & \text{(sub-goal achieved)} \\ -1.5 & \text{(The robot falls down)} \end{cases}, \tag{4}$$

where $s_{\dot{\theta}} = 60$ [deg/sec] gives the width of the reward function. If the time limit is reached, the lower-level learner is not updated at the end of control. The desired angular velocity $\hat{\dot{\boldsymbol{\theta}}}$ is initialized at the first successful stand-up as the angular velocity $\dot{\boldsymbol{\theta}}$ when the learner achieves the sub-goal area. It is then updated by $\hat{\dot{\boldsymbol{\theta}}} \leftarrow \eta\hat{\dot{\boldsymbol{\theta}}} + (1-\eta)\dot{\boldsymbol{\theta}}$ with $\eta = 0.9$ in subsequent successful trials. Note that we set reward $r(t) = 0$ in the

upper part of (4) before the robot achieves the first stand-up.

3. Simulations

First, we show simulation results of the stand-up task with a two-joint, three-link robot using the hierarchical RL architecture. We then investigate the basic properties of the hierarchical architecture in a simplified stand-up task with one joint. We show how the performance changes with the action step size in the upper level. We also compare the performance between the hierarchical RL architectures and non-hierarchical RL architectures. Finally, we show the role of the upper-level reward R_{sub} for reaching a sub-goal.

3.1 Stand-up Task using a Two-joint, Three-link Robot

We tested the performance of the hierarchical RL architectures in the stand-up task using the two-joint, three-link robot (see Figure 1). We chose $\mathbf{X} = (\theta_m, \theta_1, \theta_2)$ in the upper level and $\mathbf{x} = (\theta_0, \theta_1, \theta_2, \dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2)$ in the lower level. In this study, we used a prior knowledge that the angle of the center of mass θ_m should not decrease in successful stand-up trajectories. Then, the upper level chose an action from $U_m \in \{0, 1, 2\}$, $U_1 \in \{0, \pm 1, \pm 2\}$, $U_2 \in \{0, \pm 1\}$ by using equation 1. We chose the action step size in the upper level as $\mathbf{\Delta X} = (\Delta\theta_m, \Delta\theta_1, \Delta\theta_2) = (30, 50, 25)$ [deg]. Each trial was started with the robot lying on the ground, $\mathbf{x} = (90, 0, 0, 0, 0, 0)$ [deg], and was continued for $t < 2(T + 1)$ seconds in simulated time, where T is the discrete time in the upper level. When the robot fell down and hit its hip or head on the ground, the trial was terminated and restarted again. Each simulation was continued up to 1000 trials.

The physical structure and the parameters of the robot are shown in Figure 8 and Table 2. The physical system was simulated by a dynamic simulator made by Boston Dynamics Inc., with a time step of 0.001 [sec]. We used the number of trials made before achieving 10 successful trials as the measure of the learning speed.

The robot successfully learned to stand up in 7 out of 10 simulation runs. The average number of learning trials was 749, and it took 30 minutes in simulated time (averaged over 7 successful runs). The upper-level learner used 4.3 sub-goals (averaged over 7 successful runs) for successful stand-up.

Figure 3(a) shows the time course of learning. The vertical axis shows the performance index given by the integral of the head height $\int_0^{t_e} y(t)dt$, where t_e is terminal time of the trial. Figure 3(b) shows the number of sub-goals used in each trial. In the first stage of learning, the upper-level learner used only a few sub-goals, but after the middle stage of learning, the number of sub-goals increased because the lower-level learner learned to achieve sub-goals.

Figure 4 shows an example of a sub-goal sequence acquired in the upper level. Figure 5 shows an example of a stand-up trajectory acquired in the lower level. Each learner successfully learned the appropriate action sequence for the stand-up task.

Next, we tested the affect of variability in the initial position. Trials were started from randomly generated postures on a constraint that the foot and the head are touched on the ground ($90 \leq \theta_0 \leq 140$, $0 \leq \theta_1 \leq 150$, $0 \leq \theta_2 \leq 150$ [deg]). In this condition, the robot successfully learned to stand up in 7 out of 10 simulation runs. The average number of learning trials was 706, and it took 18 minutes in simulated time (averaged over 7 successful suns). Thus, the variability in the initial position improved learning speed. The result indicates that it is not just a particular trajectory but a robust control policy that was learned by the hierarchical network.

3.2 Comparison between Different Step Sizes

To investigate the effects of the action step size, we compared the upper-level learners with different $\mathbf{\Delta X}$. For simplicity, we fixed θ_2 to 0 [deg] by servo control and chose action steps as $\Delta\theta_1 = 25, 30, 50$ [deg] and $\Delta\theta_m = 30$ [deg]. Thus, we chose $\mathbf{X} = (\theta_m, \theta_1)$ and $\mathbf{x} = (\theta_0, \theta_1, \dot{\theta}_0, \dot{\theta}_1)$ as state variables in the upper and the lower levels, respectively. Each simulation was continued up to 1000 trials.

Table 1 shows the results of the learning to stand up with different $\Delta\theta_1$. The robot acquired good performance with $\Delta\theta_1 = 25,\ 30$ [deg], and poor performance with $\Delta\theta_1 = 50$ [deg]. The upper level with smaller $\Delta\theta_1$ used more sub-goals for standing up. Figure 6 shows examples of the stand-up trajectories and the sub-goal points in joint angle space with different $\Delta\theta_1$. The set of sub-goal points with $\Delta\theta_1 = 20$ [deg] and $\Delta\theta_1 = 30$ [deg] were different, but both were good via points for generating stand-up trajectories. On the other hand, the set of sub-goal points with $\Delta\theta_1 = 50$ [deg] lacked the important via point that represents a maximum curvature of the stand-up trajectory (see Figure 6). Without this via point, the lower-level learner had to learn a difficult sub-task and often failed to acquire a part of the stand-up trajectories.

Thus we showed that the proposed hierarchical reinforcement learning method was not so sensitive to the

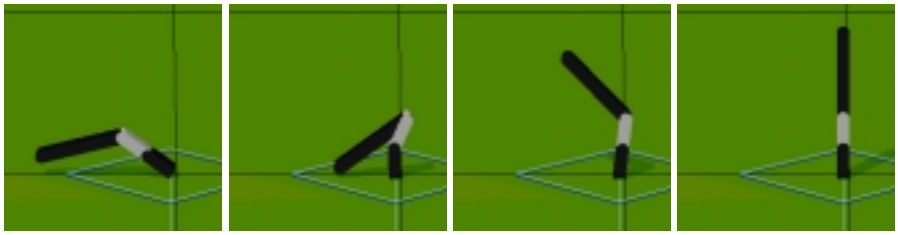


Figure 4. Example of a successful sub-goal sequence

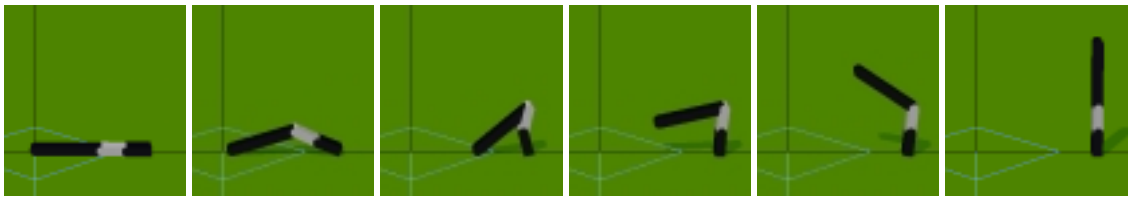
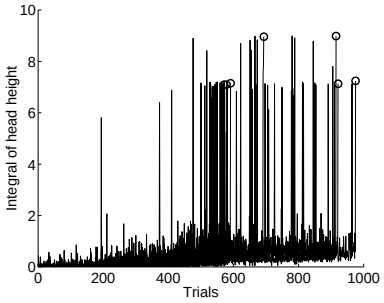
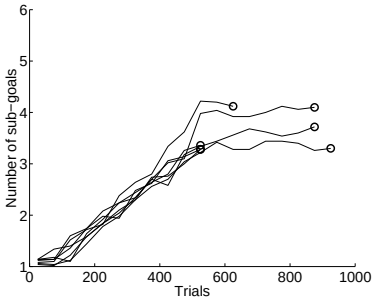


Figure 5. Example of a successful stand-up trajectory



(a) Performance index



(b) Number of sub-goals

Figure 3. Time course of learning. Circles show 10th successful stand-up in which a simulation run was terminated. (a)Performance index. (b)Average number of sub-goals in each 50 trials.

choice of $\boldsymbol{\Delta X}$ but has a certain range of $\boldsymbol{\Delta X}$ in which the upper-level learner successfully acquired the appropriate sub-goals for stand-up.

Table 1. Comparison with different $\Delta\theta_1$

$\Delta\theta_1$	Success rate	Trials (Average over successful trials)	Time (Average over successful trials)	Sub-goals (Average over successful trials)
25 [deg]	90%	408	19 [min]	6.3
30 [deg]	100%	375	16 [min]	4.5
50 [deg]	20%	463	16 [min]	4

3.3 Comparison between Hierarchical and Plain Architectures

We then compared the hierarchical RL architectures with non-hierarchical, plain RL architectures. We again used a one-joint, two-link robot and compared the results in section 3.2 with the results in this section. We used only one actor and critic pair in the plain architecture, as in conventional reinforcement learning. In such a case, the actor and the critic have to learn highly non-linear control function and value function, respectively. In preliminary experiments, we used a simple reward like the height of the head for the plain architecture without success. Thus, we used a hand-crafted reward

$$r(y) = \begin{cases} 0.3(\frac{y}{L}) + 0.3\sin(\theta_m) \\ \quad +0.4\exp(-(\frac{\theta_0^2+\theta_1^2}{s_\theta^2} + \frac{\dot{\theta}_0^2+\dot{\theta}_1^2}{s_{\dot{\theta}}^2})) - 1 \text{ (during trial)} \\ -1 \quad \text{(The robot falls down)} \end{cases} \tag{5}$$

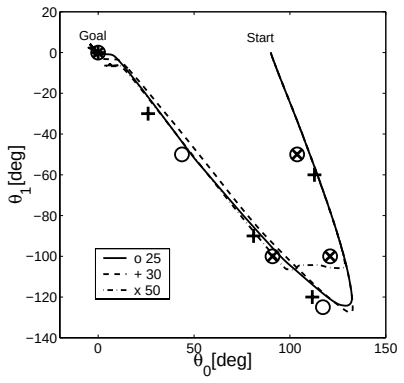


Figure 6. Stand-up trajectories and sub-goals using different $\Delta\theta_1$

for the plain architecture, where y is the hight of the head of the robot, L is the total length of the robot, and $s_\theta = 60$ [deg] and $s_{\dot{\theta}} = 240$ [deg/sec] give the width of the reward function. Each simulation was continued up to 2000 trials.

The robot successfully learned to stand up within 1685 trials, which took 56 minutes in simulated time (averaged over 5 successful runs out of 10 simulation runs). Figure 7 shows the time course of learning with plain architecture. As a result, the robot with hierarchical architecture learned to stand up four times faster than the one with plain architecture. In addition, we can say that the robot with hierarchical architecture (with $\Delta\theta_1 = 30, 25[deg]$) learned to stand up in a more robust way than the one with plain architecture because the robot with hierarchical architecture achieved about twice as many successful runs as the robot with plain architecture.

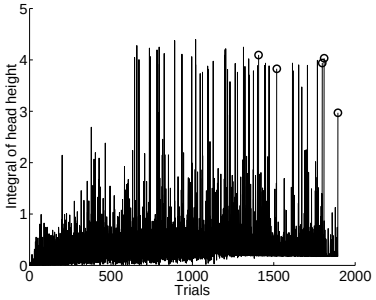


Figure 7. Time course of learning with plain architecture. Circles show 10th successful stand-up in which a simulation run was terminated.

4. Real Robot Experiments

Next, we applied the hierarchical RL to a real robot. As the initial condition for the real robot learning, we used the sub-goal sequence and non-linear controllers acquired by the simulation in section 3.1. We then applied the hierarchical RL to a real robot.

We used a PC/AT with Pentium 233MHz and RT-Linux as the operating system for controlling the robot. The time step of the lower-level learning was $\Delta t = 0.01$ [sec], and that of the servo control was $\Delta t = 0.001$ [sec].

The robot has a inclination sensor to detect the pitch angle and angular velocity of the link3 (see Figure 1) and two rotary encoders to detect joint angles (θ_1, θ_2). We derived joint angular velocity $(\dot{\theta}_1, \dot{\theta}_2)$ by numerically differentiating the joint angles. We calculated the pitch angle and angular velocity (θ_0, $\dot{\theta}_0$) using the above sensor data (see Figure 1).

The physical parameters of the real robot are shown in Table 2.

Table 2. Physical parameters of the real robot

	length	weight	inertia
link1	0.40 m	0.85 kg	0.064 $kg\ m^2$
link2	0.15 m	3.5 kg	0.11 $kg\ m^2$
link3	0.15 m	0.46 kg	0.011 $kg\ m^2$

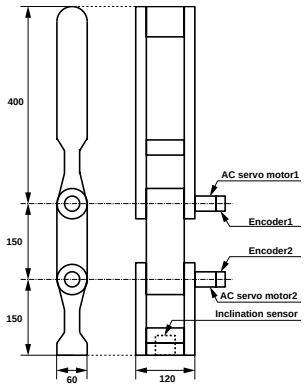


Figure 8. Real robot configuration

We used the sub-goal sequence and non-linear controllers acquired by the learning with 7 successful simulation runs as the initial setting for the real robot experiments. Each experiment was continued up to 200 trials. The robot successfully learned to stand up in 6 out of 7 experiments within 164 trials, and it took 21 minutes (averaged over 6 successful runs). Figure 9 shows the time course of learning with the real robot, and Figure 10 shows the time course of a successful stand-up trajectory. These results showed that

Figure 11. Example of a stand-up trajectory using the real robot

the proposed hierarchical RL method enabled the real robot to accomplish the stand-up task and that the sub-goal sequence and non-linear controllers acquired by the simulation is useful for the learning of the real robot.

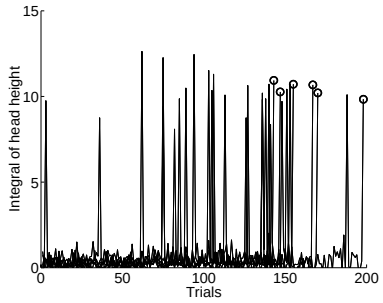


Figure 9. Time course of learning with the real robot. Circles show 10th successful stand-up in which a simulation run was terminated.

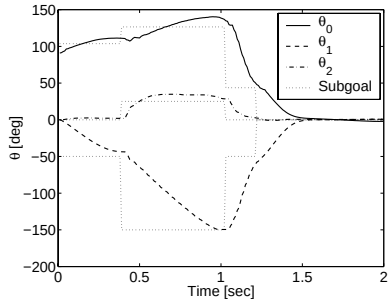


Figure 10. Example of a time course of a stand-up trajectory and a sub-goal sequence (θ_0:pitch angle, θ_1, θ_2:joint angle)

5. Discussion

In this section, we briefly summarize the achievement of this study in relation to the previous studies of hierarchical RL.

5.1 Improving Learning Speed

Here we summarize the reasons for the successful learning of the stand-up task by the hierarchical architecture, which can be helpful in other tasks as well. First, the upper level decomposed the original task into simplified sub-tasks. Furthermore, the upper level reward of the success of a sub-task (R_sub) encouraged the upper level to set realizable subgoals. Second, the dimension reduction in the upper-level dramatically reduced the number of state in high-dimensional state space. Third, the coarse exploration in the upper level enabled the robot to explore efficiently in the entire state space and prevented it from getting in stuck local optimum. Fourth, in the hierarchical architecture, prior knowledge can be easily included. We set the appropriate size and direction of action steps in the upper level and provided linear feedback component in the lower level.

5.2 Selection of Reduced Variables

In this study, we chose the angles of the joints and the center of mass as the low-dimensional state variables for the upper level. However, this strategy has limitation that the dimension can be reduced at most to the half of the original dimension. For tasks with much higher-dimensional state space, for example, arms or legs with excess degrees of freedom, we should consider the use of task-oriented kinematic variables in the upper level, such as the positions of the end effector and the center of mass. How to select such essential variables by learning remains as a subject of future work.

In addition, we chose an appropriate step size $\boldsymbol{\Delta X}$ in the upper level, but a method of automatically choosing and adapting step size is also a subject of future work.

5.3 Multiple Tasks

We applied the lower-level modules to a single task. However, for robots cope with multiple tasks, reusing the lower-level modules is desired.

In compositional Q-Learning (CQ-L) (Singh, 1992), a gating module stochastically switches reusable lower level modules, and a bias module estimates the state-value for compositional tasks.

In nested Q-learning (Digney, 1998), a state is detected as a sub-goal if non-typical reinforcement is given in

the state or the learner visits the state many times. Each sub-task then becomes one of the actions that the learner can choose as a primitive action.

We will incorporate these ideas into our hierarchical RL method as future work.

6. Conclusions

We proposed a hierarchical RL architecture based on a reduced dimensional state representation in the upper level, and showed that the stand-up task of a two-joint, three-link real robot was successfully accomplished by the hierarchical architecture. We showed that the hierarchical architecture learned faster and more robustly than a plain architecture.

Acknowledgments

We would like to thank Mitsuo Kawato, Stefan Schaal, Tsukasa Ogasawara, and Kazuyuki Samejima for their helpful discussions.

References

Dayan, P., & Hinton, G. E. (1993). Feudal Reinforcement Learning. *Advances in Neural Information Processing Systems 5* (pp. 271–278). San Francisco, CA: Morgan Kaufmann.

Digney, B. L. (1998). Learning Hierarchical Control Structures for Multiple Tasks and Changing Environments. *Proceedings of the Fifth Conference on the Simulation of Adaptive Behavior* (pp. 321–330). Cambridge, MA: The MIT Press.

Doya, K. (2000). Reinforcement Learning in Continuous Time and Space. *Neural Computation*, *12*, 219–245.

Kimura, H., & Kobayashi, S. (1999). Efficient Nonlinear Control by Combining Q-learning with Local Linear Controllers. *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 210–219). San Francisco, CA: Morgan Kaufmann.

Morimoto, J., & Doya, K. (1998a). Hierarchical Reinforcement Learning of Low-dimensional Subgoals and High-dimensional Trajectories. *Proceedings of the Fifth International Conference on Neural Information Processing* (pp. 850–853). Burke, VA: IOS Press.

Morimoto, J., & Doya, K. (1998b). Reinforcement Learning of Dynamic Motor Sequence: Learning to Stand Up. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1721–1726). OMNIPRESS.

Peng, J., & Williams, R. (1996). Incremental Multi-step Q-learning. *Machine Learning*, *22*, 283–290.

Singh, S. (1992). Transfer of Learning by Composing Solutions of Elemental Sequential Tasks. *Machine Learning*, *8*, 323–339.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* Cambridge, MA: The MIT Press.

Tham, C. K. (1995). Reinforcement Learning of Multiple Tasks using a Hierarchical CMAC Architecture. *Robotics and Autonomous Systems*, *15*, 247–274.

Wiering, M., & Schmidhuber, J. (1997). HQ-learning. *Adaptive Behavior*, *6*, 219–246.

Appendix

In this appendix, we show parameters which used in this study.

Limits for the joint angles

In the stand-up task, we limited the posture angles in the upper level as $0 \leq \theta_m \leq 90$, $-150 \leq \theta_1 \leq 0$, $0 \leq \theta_2 \leq 25$ [deg]. In the lower level, we limited the joint angles as $-150 \leq \theta_1 \leq 150$, $-150 \leq \theta_2 \leq 150$ [deg] which correspond to the limits of the real robot.

Exploration parameters

We set the exploration parameter in the upper level as $\beta = 0.2M(T)$, where $M(T)$ is the number of trials lasting no fewer than T steps.

The noise term in the lower level $n_j(t)$ is low-pass filtered noise $\tau_n \dot{n}_j(t) = -n_j(t) + N_j(t)$, where $N_j(t)$ denotes normal Gaussian noise and $\tau_n = 0.1$ [sec] is a time constant for the low-pass filter. The size of the noise term σ in the lower level was modulated as $\sigma = \sigma_s \min[1, \max[0, V_1 - V(t)]]$. $V(t)$ is the lower level state value function and $V_1 = 0.5$ is a exploration parameter for the lower-level learner. The maximal noise level σ_s was also changed according to the sub-goal and the number of trials m as

$$\sigma_s = \begin{cases} \sigma_0 & \text{for final sub-goal} \\ \sigma_1 & \text{otherwise if } m \leq m_1 \\ \frac{(m_2-m)\sigma_1+(m-m_1)\sigma_2}{m_2-m_1} & \text{if } m_1 < m < m_2 \\ \sigma_2 & \text{if } m \geq m_2 \end{cases}. \tag{6}$$

The parameters were $m_1 = 300$ [trial], $m_2 = 600$ [trial], $\sigma_1 = 0.5$, $\sigma_2 = 0.1$, and $\sigma_0 = 0.01$.