Hindawi Publishing Corporation
Advances in Mechanical Engineering
Volume 2014, Article ID 276264, 12 pages
http://dx.doi.org/10.1155/2014/276264

Research Article

Autonomous Robust Skill Generation Using Reinforcement Learning with Plant Variation

Kei Senda and Yurika Tani

Department of Aeronautics and Astronautics, Kyoto University, Nishikyo-ku, Kyoto 615-8540, Japan

Correspondence should be addressed to Kei Senda; senda@kuaero.kyoto-u.ac.jp

Received 1 December 2013; Revised 24 March 2014; Accepted 24 March 2014; Published 24 April 2014

Academic Editor: Amir H. Gandomi

This paper discusses an autonomous space robot for a truss structure assembly using some reinforcement learning. It is difficult for a space robot to complete contact tasks within a real environment, for example, a peg-in-hole task, because of error between the real environment and the controller model. In order to solve problems, we propose an autonomous space robot able to obtain proficient and robust skills by overcoming error to complete a task. The proposed approach develops skills by reinforcement learning that considers plant variation, that is, modeling error. Numerical simulations and experiments show the proposed method is useful in real environments.

1. Introduction

This study discusses an unresolved robotics issue: how to make a robot autonomous. A robot with manipulative skills capable of flexibly achieving tasks, like a human being, is desired. Autonomy is defined as "automation to achieve a task robustly." Skills can be considered to be solutions to achieve autonomy. Another aspect of a skill is including a solution method. Most human skill proficiency is acquired by experience. Since how to realize autonomy is not clear, skill development must include solution methods for unknown situations. Our problem is how to acquire skills autonomously, that is, how to robustly and automatically complete a task when the solution is unknown.

Reinforcement learning [1] is a promising solution, whereas direct applications of existing methods with reinforcement learning do not robustly complete tasks. Reinforcement learning is a framework in which a robot learns a policy or a control that optimizes an evaluation through trial and error. It is teacherless learning. By means of reinforcement learning, a robot develops an appropriate policy as mapping from state to action when an evaluation is given. The task objective is prescribed, but no specific action is taught. Reinforcement learning often needs many samples. The large number of samples is due to the large number of states and actions. So, online learning in a real environment is usually impractical. Most learning algorithms consist of two processes [2]: (1) online identification by trial and error sampling and (2) finding the optimal policy for the identified model. These two processes are not separated in typical learning algorithms such as Q-learning [3]. Reinforcement learning is said to be adaptive because it uses online identification and on-site optimal control design. Robustness attained using this adaptability is often impractical. It takes a very long time for online identification by means of trial and error.

In our approach, by learning a robust policy rather than by online identification, reinforcement learning is used to achieve a solution to an unknown task.

Using our approach, this study addresses an autonomous space robot for a truss structure assembly. It is difficult for a space robot to achieve a task, for example, peg-in-hole task, by contact with a real environment, because of the error between the real environment and the controller model. In order to solve the problem, a space robot must autonomously obtain proficiency and robust skills to counter the error in the model. Using the proposed approach, reinforcement learning can achieve a policy that is robust in the face of plant variation, that is, the modeling error. Numerical simulations and experiments show that a robust policy is effective in a real environment and the proposed method is used.

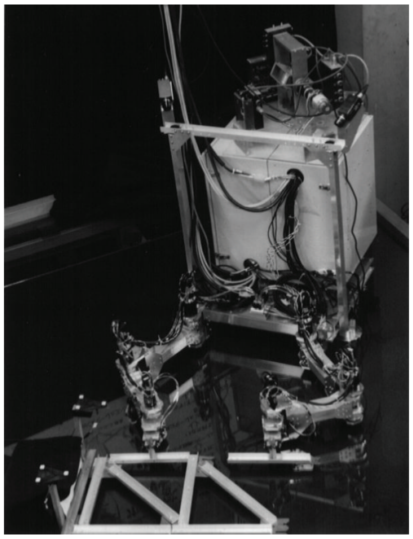


Figure 1: Photograph of space robot model and truss.

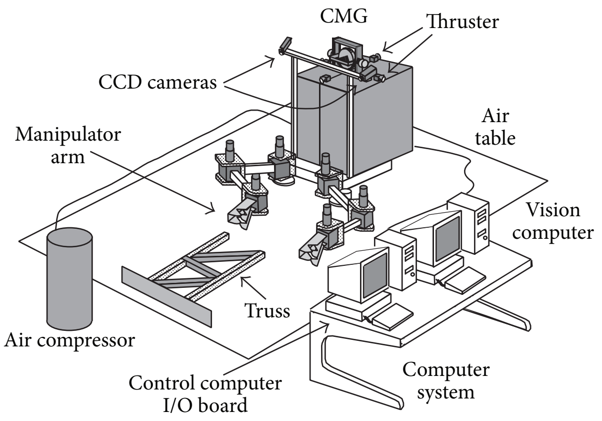


Figure 2: Schematic diagram of experimental system.

2. Need for Autonomy and Approach

2.1. Need for Autonomy. Autonomy is needed wherever robots work. Below, we discuss why and what kind of autonomy is required [4, 5] for space robots.

Space robots are required to complete tasks in the place of extra vehicular activity by an astronaut. Studies of autonomous systems are needed to realize space robots that can achieve missions under human-operator command. There are many applications for the autonomous space robots [6].

We developed ground experiments to simulate a free-flying space robot under orbital microgravity conditions (Figure 1). Using this apparatus, we have studied robot autonomy. Targeting control-based autonomy, we developed an automatic truss structure assembly, and so forth. However, it has been hard to achieve perfect robot automation because of various factors. To make a robot autonomous in the face of the following challenges, we must

(a) solve problems in the actual robot environment;

(b) operate robustly in the face of uncertainties and variations;

(c) overcome the difficulty of comprehensively predicting a wide variety of states;

(d) identify tasks and find unknown solutions to realize robust robot autonomy.

2.2. Approach to Autonomy Based on Reinforcement Learning. Human beings achieve tasks regardless of the above complicating factors, but robots cannot. Many discussions have rationalized the difficulties as originating in the nature of human skills. We have not established a means by which we can realize such skills. This study takes the following approach to this problem.

Section 4 approaches factors (a) and (b) in part by way of control-based automation taking account of the robustness of controls. The autonomous level of this approach is low because only small variations are allowable.

Section 5 considers how to surmount factors (a), (b), and (c) by learning. Using a predicted model and transferring the learned results to the actual system [7] is studied. This model is identified online and relearned. This procedure is applied where adaptability or policy modification is needed to thwart variations in the real environment. In some cases, the robot completes the targeted tasks autonomously. Learning is completed within an acceptable calculation time.

Section 6 considers factors (a), (b), (c), and (d). A peg-in-hole task has higher failure rates using the same approach as used in Section 5. Because of small differences between the model and the real environment, failure rates are excessive. The robot thus must gain skills autonomously. Skills similar to those of human beings must be generated autonomously. These skills are developed by reinforcement learning and additional procedures.

Section 7 evaluates approach robustness and control performance of the newly acquired skills. The skills obtained in Section 6 are better than those in Section 5.

3. Experimental System and Tasks

3.1. Outline of Experimental System. Our experimental system (Figures 1 and 2) simulates a free-flying space robot in orbit. Robot model movement is restricted to a two-dimensional plane [5].

The robot model consists of a satellite vehicle and dual three degrees of freedom (3-DOF) rigid SCARA manipulators. The satellite vehicle has CCD cameras for stereovision and a position/attitude control system. Each joint has a torque sensor and a servo controller for fine torque control of the output axis. Applied force and torque at the end-effector are calculated using measured joint torque. Air pads are used to support the space robot model on a frictionless planar table and to simulate the space environment. RTLinux is installed on a control computer to control the space robot model in real time. Stereo images from the two CCD cameras are captured by a video board and sent into an image-processing computer with a Windows OS. The image-processing computer measures the position and orientation of target objects in the worksite by triangulation. Visual information is sent to the control computer via Ethernet.

The position and orientation measured by the stereo vision system involve errors caused by quantized images, lighting conditions at the worksite, and so forth. Time-averaged errors are almost constant in each measurement. Evaluated errors in the peg-in-hole experiments are modeled as described below. Hole position errors are modeled as a normal probability distribution, where the mean is $m = 0$ [mm] and standard deviation is $\sigma = 0.75$ [mm]. Hole orientation errors are modeled as a normal probability distribution, where the mean is $m = 0$ [rad] and standard deviation is $\sigma = 0.5\pi/180$ [rad].

We accomplish a hand-eye calibration to achieve tasks in the following sections. An end-effector, a manipulator hand, grasps a marker with a light-emitting diode (LED). The arm directs the marker to various locations. The robot calculates the marker location by using sensors mounted at the joints of the manipulator arm. The vision system also measures the marker location by using the stereo image by triangulation. Measurements using these joint-angle sensors have more precise resolution and accuracy. Hence, we calibrate the visual measurements based on measurements using the joint angle sensors. We consider the joint angle sensor measurement data to be the true value.

3.2. Tasks

3.2.1. Truss Assembly Task. Figure 3 illustrates the truss structure assembly sequence. The robot manipulates a truss component, connects it to a node, and proceeds each assembly step. Later this task is achieved by controls based upon mechanics understanding [5]. The truss design is robot friendly for easy assembling.

3.2.2. Peg-in-Hole Task. The peg-in-hole task is an example that is intrinsic to the nature of assembly. The peg-in-hole task involves interaction within the environment that is easily affected by uncertainties and variations, for example, errors in force applied by the robot, manufacturing accuracy, friction at contact points, and so forth. The peg easily transits to a state in which it can no longer move, for example, wedging or jamming [8]. Such variations cannot be modeled with required accuracy.

To complete a task in a given environment, a proposed method analyzes the human working process and applies the results to a robot [9]. Even if the human skill for a task can be analyzed, the results are not guaranteed to be applicable to a robot. Another method uses parameters in a force control designed by means of a simulation [10] but was not found to be effective in an environment with uncertainty. In yet another method [11], the task achievement ratios evaluated several predesigned paths in an environment with uncertainty. An optimal path is determined among the predesigned paths. There was the possibility a feasible solution did not exist among predesigned paths.

In the peg-in-hole experiment (Figure 4), the position and orientation of the hole are measured using a stereo camera. The robot manipulator inserts a square peg into a similar sized hole. This experiment is a two-dimensional plane problem (Figure 5). The space robot model coordinate system is defined as Σ_0, the end-effector coordinate system as Σ_E, and the hole coordinate system as Σ_{hl}. While the space robot completes its task, the robot grasps the structural site with another manipulator; the relative relation between Σ_0 and Σ_{hl} is fixed. State variables are defined as $[y_x, y_y, y_\theta, f_x, f_y, f_\theta]$, where (y_x, y_y) is the position of Σ_E in Σ_0, y_θ is the orientation about $\mathbf{k}_0$-axis, (f_x, f_y), and f_θ are the forces and torque in Σ_0 that end-effector applies to the environment.

The peg width is 74.0 [mm] and the hole width is 74.25 [mm]. The hole is only 0.25 [mm] wider than the peg. The positioning error is composed of the measurement error and the control error. The robot cannot insert the peg in the hole by position control if the positioning error is beyond ± 0.125 [mm]. Just a single measurement error by stereo vision often moves the peg outside of the acceptable region.

4. Control-Based Automation

4.1. Truss Assembly Task. Automatic truss assembly was studied via control-based automation with mechanics understanding. The robot achieved an automatic truss structure assembly [5] by developing basic techniques and integrating them within the experimental system.

The following sensory feedback control [12] is used for controlling manipulators:

$$\boldsymbol{\tau} = -\mathbf{J}^T \mathbf{K}_P (\mathbf{y} - \mathbf{y}_d) - \mathbf{K}_D \dot{\mathbf{q}}, \tag{1}$$

where $\boldsymbol{\tau}$ is the control input to the manipulator and $\mathbf{J}$ is the Jacobean matrix. The $\mathbf{y}$ is the manipulation variable whose elements are the hand position/orientation $[y_x, y_y, y_\theta]$, $\mathbf{y}_d$ is the reference value of $\mathbf{y}$, $\mathbf{q}$ is the joint angle vector, and $\mathbf{K}_P$ and $\mathbf{K}_D$ are feedback gains. When the end-effector contacts the environment and manipulation variable $\mathbf{y}$ is stationary under constraint, the end-effector applies force and torque to the environment:

$$\mathbf{f} = -\mathbf{K}_P (\mathbf{y} - \mathbf{y}_d). \tag{2}$$

The force and torque can be controlled by $\mathbf{y}_d$. This is a compliant control.

Figure 3 is a series of photographs of the experimental assembly sequence. As shown in panel (i), the robot holds on to the worksite with its right arm to compensate for any reaction force during the assembly. The robot installs the first component, member 1, during panels (ii) and (iii). The robot installs other members successively and assembles one truss unit, panels (iv)–(vi).

Control procedures for the assembly sequence are as follows. There are target markers in the experiment environment as shown in Figures 1 and 3. Target markers are located at the base of the truss structure and at the storage site for structural parts. Each target marker has three

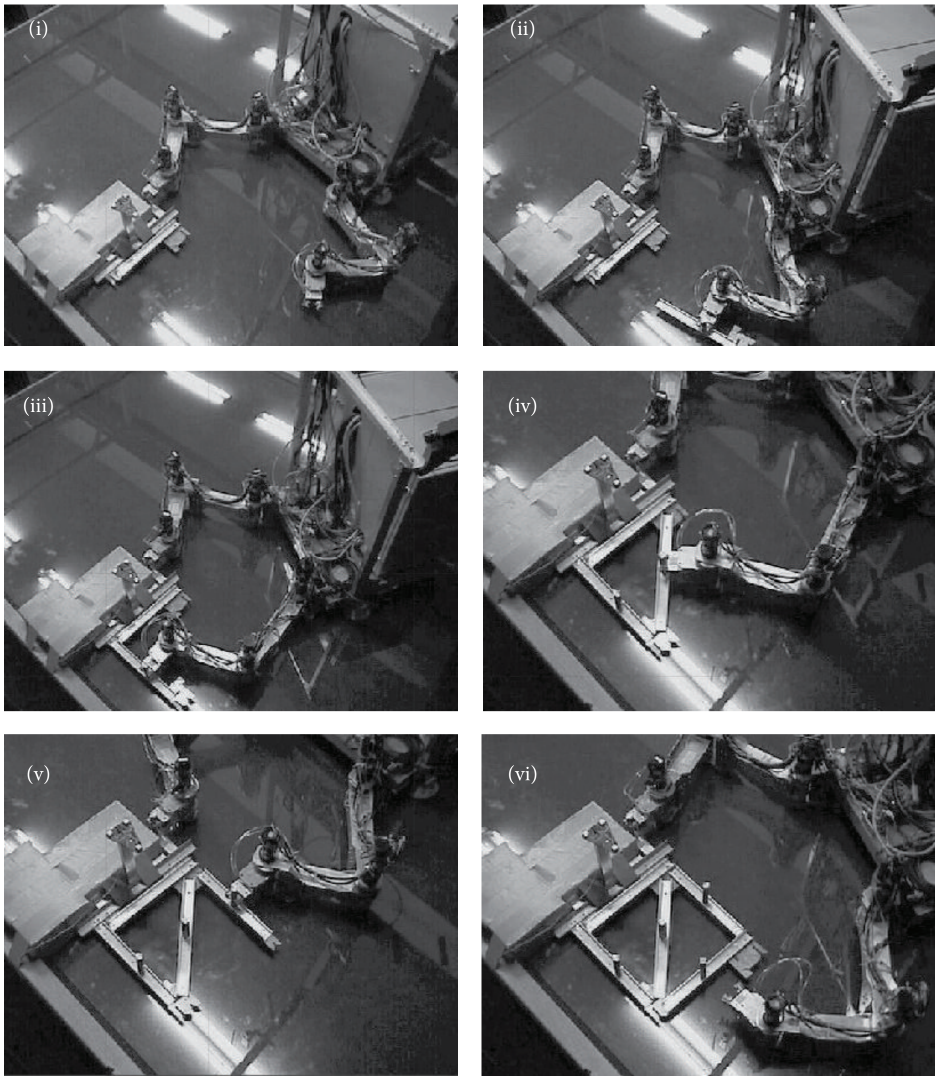


Figure 3: Sequence of truss structure assembly.

LEDs at triangular vertices. The vision system measures the marker position and orientation simultaneously. The robot recognizes the position of a part relative to the target marker before assembly. The robot places the end-effector position at the pick-up point, which is calculated from the target marker position as measured by the stereo vision system. At the pick-up point, the end-effector grasps a handgrip attached to the targeted part. The position and orientation of the part to the end-effector are settled uniquely when the end-effector grasps the handgrip. The robot plans the path of the arm and part to avoid collision with any other object in the work environment. It controls the arm to track along the planned path. The robot plans a path from the pick-up point to the placement point, avoiding obstacles by means of an artificial potential method [13]. Objects in the environment, for example, the truss under assembly, are regarded as obstacles. The arm is then directed along a planned trajectory by the sensory feedback control (1). The end-effector only makes contact with the environment when it picks up or places the part. Hence, feedback gains in (1) are chosen to make it a compliant control.

Consequently, the truss assembly task is successfully completed by control-based automation. However, measurement error in the vision system sensors, and so forth, prevents assembly from being guaranteed. Irrespective of uncertainties and variations at the worksite, the space robot model requires autonomy to complete the task goal.

4.2. Peg-in-Hole Task. The positioning control of (1) tries to complete the peg-in-hole task. The peg first is positioned at $y_\theta = 0$ [rad], it transits to the central axis of the hole, and it moves in a positive direction, toward $\mathbf{i}_0$. The peg does not contact the environment during transition from the initial state to the goal. Insertion is successful if the position control of the peg relative to the hole is free from error. Unfortunately, the peg-in-hole task is often unsuccessful because of the existing error. The robot cannot insert the peg in the hole by position control if the positioning error

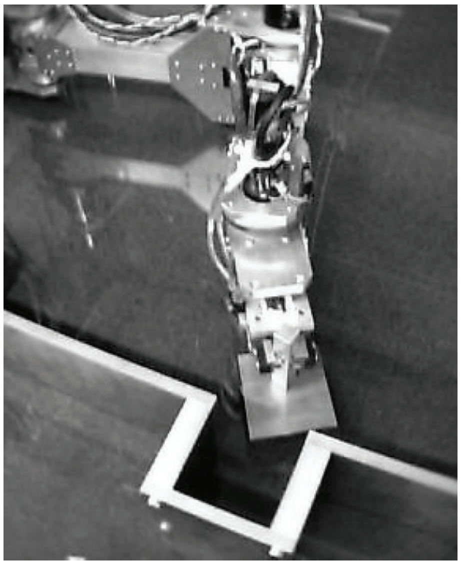


Figure 4: Photograph of peg-in-hole experiment setup.

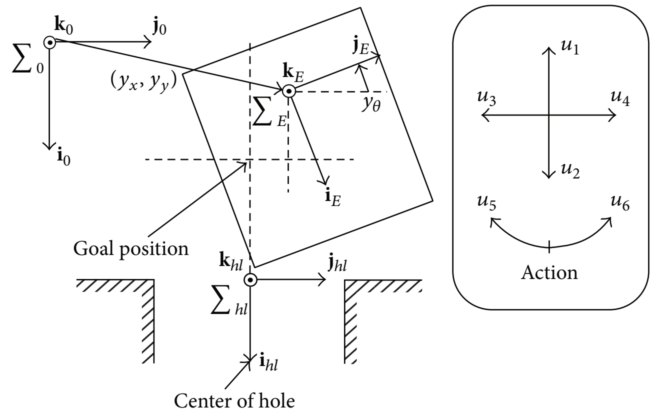


Figure 5: Definition of peg-in-hole task.

is greater than ±0.125 [mm]. So, single measurement error using stereo vision is often beyond the acceptable error. The manner in which the task fails is almost the same as shown in Figure 8 in the next section.

5. Existing Method for Autonomy with Reinforcement Learning

5.1. Outline of Existing Method with Reinforcement Learning. Reinforcement learning [1] is used to generate autonomous robot action.

In "standard" learning (Figure 6(a)), controller $\mathbf{K}_Q$ is designed in advance by learning the nominal plant model $\mathbf{P}_N$, and it is applied to real plant $\mathbf{P}$. We use a policy called controller $\mathbf{K}_Q$, which is designed with reinforcement learning methods. When variations exist, for example, measurement error in the vision system, unexpected obstacles appear in the environment, and so forth, and $\mathbf{K}_Q$ cannot complete tasks due to poor robustness and adaptability.

As shown in Figure 6(b), new plant model $\mathbf{P}'_N$ is reconstructed using visual measurement. Controller $\mathbf{K}'_Q$ is designed for the reconstructed model $\mathbf{P}'_N$. Controller $\mathbf{K}'_Q$ is then applied to real plant $\mathbf{P}$. Learning converges within a practical calculation time and the new policy is applicable to the truss structure assembly [5]. This method works well because it treats the kinematic problem without force interaction between the robot and the environment. The plant model for learning is reconstructed by visual measurement within a short time. This method has adaptability only if the model can be reconstructed accurately within a short time.

If the robot cannot complete the task with the controller due to error between the model and the real plant, the robot switches to online learning. Adaptability is realized by online identification and learning. However, this cannot be used for peg-in-hole task, because online identification requires too much time. In the next section, a reinforcement learning problem for a peg-in-hole task requires several tens of thousands of state-action pairs. It requires tens of days for online identification if a hundred samples are selected for each state-action pair and each sampling takes one second.

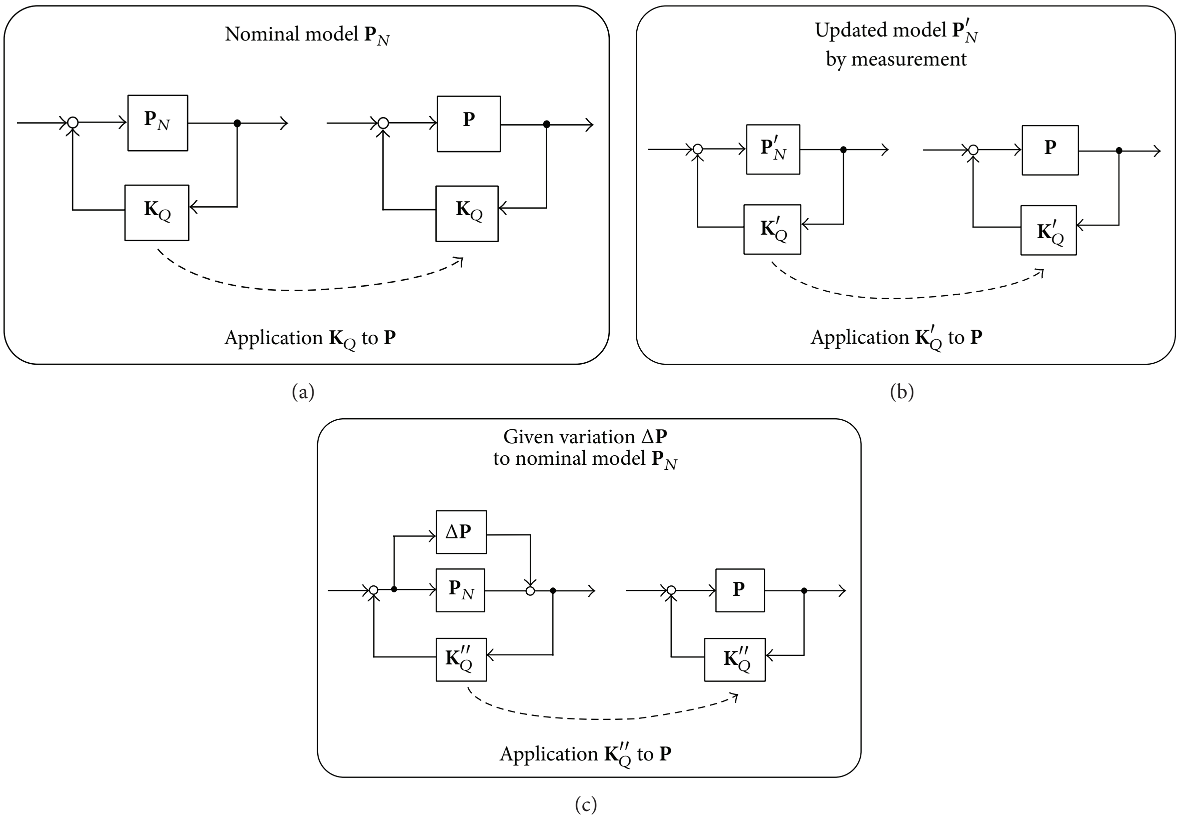
5.2. Existing Method with Reinforcement Learning

5.2.1. Problem Definition. Following general dynamic programming (DP) formulations, this paper treats a discrete-time dynamic system in a reinforcement learning problem. A state s_i and an action u_k are the discrete variables and the elements of finite sets $\mathcal{S}$ and $\mathcal{U}$, respectively. The state set $\mathcal{S}$ is composed of N_s states denoted by $s_1, s_2, \ldots, s_{N_s}$ and an additional termination state s_0. The action set $\mathcal{U}$ is composed of K actions denoted by $u_1, u_2, \ldots, u_K$. If an agent is in state s_i and chooses action u_k, it will move to state s_j and incur a one-step cost $g(s_i, u_k, s_j)$ within state transition probability $p_{ij}(u_k)$. This transition is denoted by (s_i, u_k, s_j). There is a cost-free termination state s_0, where $p_{00}(u_k) = 1$, $g(s_0, u_k, s_0) = 0$, and $Q(s_0, u_k) = 0$, $\forall u_k$. We assume that the state transition probability $p_{ij}(u_k)$ is dependent on only current state s_i and action u_k. This is called a discrete-time finite Markov decision process (MDP). The system does not explicitly depend on time. Stationary policy μ is a function mapping states into actions with $\mu(s_i) = u_k \in \mathcal{U}$, and μ is given by the corresponding time-independent action selection probability $\pi(s_i, u_k)$.

In this study, we deal with an infinite horizon problem where the cost accumulates indefinitely. The expected total cost starting from an initial state $s^0 = s_i$ at time $t = 0$ and using a stationary policy μ is

$$J^{\mu}(s_i) = \underset{s^1, s^2, \ldots}{E} \left[\sum_{t=0}^{\infty} g\left(s^t, \mu\left(s^t\right), s^{t+1}\right) \mid s^0 = s_i \right], \quad (3)$$

where $E_x[\cdot]$ denotes an expected value, and this cost is called J-factor. Because of the Markov property, a J-factor of



Nominal model $\mathbf{P}_N$
$\mathbf{P}_N$
$\mathbf{P}$
$\mathbf{K}_Q$
$\mathbf{K}_Q$
Application $\mathbf{K}_Q$ to $\mathbf{P}$
(a)
Updated model $\mathbf{P}'_N$ by measurement
$\mathbf{P}'_N$
$\mathbf{P}$
$\mathbf{K}'_Q$
$\mathbf{K}'_Q$
Application $\mathbf{K}'_Q$ to $\mathbf{P}$
(b)
Given variation $\Delta\mathbf{P}$ to nominal model $\mathbf{P}_N$
$\Delta\mathbf{P}$
$\mathbf{P}_N$
$\mathbf{P}$
$\mathbf{K}''_Q$
$\mathbf{K}''_Q$
Application $\mathbf{K}''_Q$ to $\mathbf{P}$
(c)

FIGURE 6: Learning using (a) nominal plant model, (b) updated plant model, and (c) plant model with variation.

a policy μ satisfies

$$J^{\mu}(s_i) = \sum_{k=1}^{K} \pi(s_i, u_k) \sum_{j=0}^{N_s} p_{ij}(u_k) \left\{ g(s_i, u_k, s_j) + J^{\mu}(s_j) \right\}, \quad \forall s_i. \tag{4}$$

A policy μ is said to be proper if μ satisfies $J^{\mu}(s_i) < \infty, \forall s_i$.

We regard the J-factor of every state as an evaluation value, and the optimal policy μ^* is defined as the policy that minimizes the J-factor:

$$\mu^*(s_i) \equiv \arg\min_{\mu} \sum_{i=1}^{N_s} J^{\mu}(s_i), \quad \forall s_i. \tag{5}$$

The J-factor of the optimal policy is defined as the optimal J-factor. It is denoted by $J^*(s_i)$.

The optimal policy defined by (5) satisfies Bellman's principle of optimality. Then, the optimal policy is stationary and deterministic. The optimal policy can be solved by minimizing the J-factor of each state independently. Hence, the optimal J-factors satisfy the following Bellman equation, and the optimal policy is derived from the optimal J-factors:

$$J^*(s_i) = \min_{u_k} \sum_{j=0}^{N_s} p_{ij}(u_k) \left\{ g(s_i, u_k, s_j) + J^*(s_j) \right\}, \tag{6}$$

$$\mu^*(s_i) = \arg\min_{u_k} \sum_{j=0}^{N_s} p_{ij}(u_k) \left\{ g(s_i, u_k, s_j) + J^*(s_j) \right\}. \tag{7}$$

5.2.2. Solutions. The existing type of reinforcement learning problem is solved as "standard" learning in Figure 6(a). It obtains the optimal policy μ^*_{nom}, which minimizes the J-factors of (7) for the nominal plant. It corresponds to controller $\mathbf{K}_Q$ in Figure 6(a). The optimal J-factor J^*_{nom} of μ^*_{nom} can be obtained by the DP-based solutions. The solutions are mentioned in [1, 2], but they are omitted here.

5.3. Learning Skill for Peg-in-Hole by Existing Method

5.3.1. Problem Definition of Peg-in-Hole. Here, the peg-in-hole task defined in Section 3.2.2 is redefined as a reinforcement learning problem.

State variables $[y_x, y_y, y_\theta, f_x, f_y, f_\theta]$ in Section 3.2.2 are continuous but discretized into 1.0 [mm], 1.0 [mm], $0.5\pi/180$ [rad], 2.0 [N], 1.0 [N], and 0.6 [Nm] in the model for reinforcement learning. The discrete state space has 4,500 discrete states, where the number of each state variable is

$[5, 5, 5, 4, 3, 3]$. Robot action at the end-effector is u_1, u_2, u_3, and u_4, at each of the end-effector states transiting by ± 1 in the direction of the $\mathbf{i}_0$-axis or $\mathbf{j}_0$-axis, and u_5 and u_6, at each of the end-effector states transiting by ± 1 about the $\mathbf{k}_0$-axis of rotation. State-action space is described in the space robot model coordinate system Σ_0. The hole is 0.25 [mm] wider than the peg, and (y_x, y_y) are quantized larger than this difference.

Control in (1) is used to transit from present state s_i to the next state s_j by action u_k. The reference manipulation variable to make the transition to s_j is $\mathbf{y}_d^{(s_j)} = \mathbf{y}_d^{(s_i)} + \delta\mathbf{y}_d^{(s_i)}$ given by $\delta\mathbf{y}_d^{(s_i)}(u_k)$ $(k = 1, 2, \ldots, 6)$, where $\delta\mathbf{y}_d^{(s_i)}$ is kept constant during transition. When the end-effector contacts the environment and manipulation variable $\mathbf{y}$ is stationary under constraint, the end-effector applies force and torque to the environment $\mathbf{f} = -\mathbf{K}_P(\mathbf{y} - \mathbf{y}_d)$ as (2), where $\mathbf{y}_d$ is the reference manipulation variable. Force and torque are controlled by $\mathbf{y}_d$, which is changed by an action. This is a compliant control, a force control. Tasks in which the end-effector contacts the environment, for example, peg-in-hole, demand a control with compliance. Therefore, a compliant control is essential as a basic control.

The robot takes the next action after (1) control settles and the peg becomes stationary. Regardless of whether the peg is in contact with the environment, the robot waits for settling and proceeds to the next action. For this reason, state variables do not include velocity.

The goal is to achieve states with the largest y_x, the peg position in $\mathbf{i}_0$-direction, in the state space. The one-step cost is $g(s_i, u_k, s_j) = 1$ for all states other than the goal state. Hence, the J-factor is the expected step number from s_i to the goal.

5.3.2. Learning Method. State transition probabilities for the state-action space in the previous section are calculated with sample data. Sample data are calculated with a dynamic simulator in a spatially continuous state space. The dynamic simulator is constructed with an open-source library, the open dynamics engine (ODE) developed by Russell Smith. The numerical model of this simulator has continuous space, force, and time in contrast to the discretized models for reinforcement learning in Section 5.2. This discretized state transition model is regarded as plant $\mathbf{P}_N$, and the method in Figure 6(a) is applied. The optimal policy μ^*_{nom}, that is, controller $\mathbf{K}_Q$, is derived from the solution in Section 5.2. The optimal policy μ^*_{nom} is applied to the dynamic simulator with a continuous state-action space or the hardware experimental setup, the real plant $\mathbf{P}$ in Figure 6(a). This study does not deal with the online learning.

5.3.3. Learning Result. The result of a numerical simulation in which controller $\mathbf{K}_Q$ is applied to the environment with no position/orientation error is shown. The peg moves and arrives at the goal as shown in Figure 7. Peg positioning is first changed to $y_\theta = 0$ [rad]. After the peg transits to the hole central axis, it is moved in a positive direction toward $\mathbf{i}_0$. Then the peg is inserted into the hole. During the transition from the initial state to the goal, the peg does not make contact

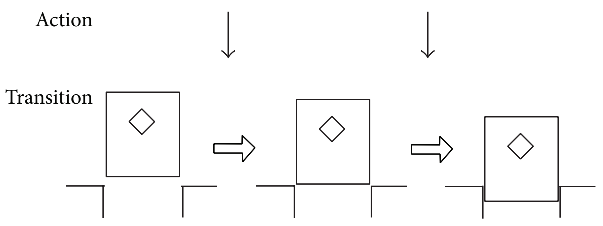


FIGURE 7: Trajectory of controller $\mathbf{K}_Q$ in a simulation without any hole position error.

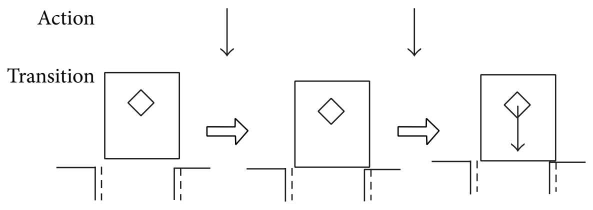


FIGURE 8: Trajectory of controller $\mathbf{K}_Q$ in a simulation with hole position error −0.5 [mm] in $\mathbf{j}_0$.

with the environment, and the end-effector applies force and torque, $[f_x, f_y, f_\theta] = [0, 0, 0]$.

In an environment with the hole position error of −0.5 [mm] in $\mathbf{j}_0$ direction, the peg does not arrive at the goal with controller $\mathbf{K}_Q$; see Figure 8. The task is not completed using $\mathbf{K}_Q$ due to small errors caused by visual measurement, and so forth.

6. Autonomous Acquisition of Skill by Learning

6.1. Autonomous Acquisition of Skill for Peg-in-Hole. A robot can achieve peg positioning or movement with contact force, and it must have basic control functions same as a human being. Human vision measurement and positioning control are not accurate enough. However, the rate of a human failure in the same task is not as high as that of a robot. One reason for this may be the skills a human being brings to the task.

A human being conducting peg-in-hole task uses a typical sequence of actions (Figure 9). First, the human being puts a corner of the peg inside the hole. The peg orientation is inclined. The peg is in contact with the environment. Two points of the peg, the bottom and a side, are in contact with the environment, as shown in the close-up in Figure 9. The human then rotates the peg and pushes it against the hole and maintains the two contact points. The two corners are then inserted into the hole. Finally, the human inserts the peg into the hole and completes the task.

Human vision measurement accuracy and positioning control accuracy are not high. A human presumably develops skill while manipulating this situation. We conducted an experiment to check whether robot learning in the same situation could achieve the task as well as a human.

This situation conceptually corresponds to Figure 6(c). Plant $\mathbf{P}_N + \Delta\mathbf{P}$ denotes a variation plant with error caused by visual measurement, and so forth. Variation plant set $\{\mathbf{P}_N +$

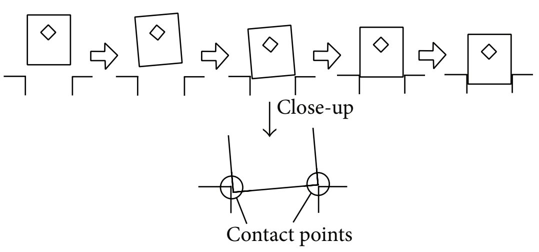


Figure 9: Human skill.

$\Delta\mathbf{P}\}$ is composed of all the variation plants that can exist. Real plant $\mathbf{P}$ is supposed to be a member of variation plant set $\{\mathbf{P}_N + \Delta\mathbf{P}\}$. The learning robot obtains controller $\mathbf{K}_Q''$. The controller is able to complete the task for all of the variation plants in $\{\mathbf{P}_N + \Delta\mathbf{P}\}$.

6.2. Problem Definition for Reinforcement Learning with Variation. We assume there are N variation plants around the estimated plant (the nominal plant). We use a set composed of N variation plants for learning.

We consider difference w_l between a variation plant and the nominal plant in each state, which is a discrete variable and the element of finite set $\mathscr{W}$. Finite set $\mathscr{W}$ is composed of L differences denoted by $w_0, w_1, \ldots, w_{L-1}$. Difference w_0 indicates no difference. If an agent is in state s_i with difference w_l and chooses action u_k, it will move to s_j within a state transition probability $p_{ij}(u_k; w_l)$ and incur a one-step cost $g(s_i, u_k, s_j; w_l)$. This transition is denoted by $(s_i, u_k, s_j; w_l)$. Difference w_l can be considered as the disturbance that causes state transition probability $p_{ij}(u_k; w_0)$ to vary to $p_{ij}(u_k; w_l)$. We assume that the $p_{ij}(u_k; w_l)$ and $g(s_i, u_k, s_j; w_l)$ are given.

Difference w_l at each state is determined by a variation plant. Variation η is a function mapping states into difference with $\eta(s_i) = w_l \in \mathscr{W}$. The nominal plant is defined by $\eta_0(s_i) = w_0$ for all states s_i. The plant does not explicitly depend on time, so variation η is time-invariant. We assume that $\eta(s_i) = w_l$ is given.

A plant set composed of N plants used for learning is represented by $\mathscr{H} = \{\eta_0, \eta_1, \ldots, \eta_{N-1}\}$. Set $\mathscr{H}$ corresponds to $\{\mathbf{P}_N + \Delta\mathbf{P}\}$. Let $\rho(\eta_n)$ denote the probability that the plant variation is η_n. We call this the existing probability of variation plant η_n. We assume that $\rho(\eta)$ is given at time $t = 0$.

For set $\mathscr{H}$, the expected cost of a policy μ starting from an initial state $s^0 = s_i$ at $t = 0$ is

$$\bar{J}^{\mu}(s_i) = \underset{\eta, s^1, s^2, \ldots}{E}\left[\sum_{t=t_0}^{\infty} g\left(s^t, \mu\left(s^t\right), s^{t+1}; \eta\left(s^t\right)\right) \mid s^0 = s_i,\ \eta \in \mathscr{H}\right], \tag{8}$$

which is the J-factor of this problem. This J-factor formula using the plant existing probability is

$$\bar{J}^{\mu}(s_i) = \sum_{n=0}^{N-1} \rho(\eta_n) J^{\mu,\eta_n}(s_i), \tag{9}$$

where $J^{\mu,\eta_n}(s_i)$ denotes the expected cost using the policy μ on a plant η_n starting from an initial state s_i. It satisfies

$$J^{\mu,\eta_n}(s_i) = \sum_{k=1}^{K} \pi(s_i, u_k) \sum_{j=0}^{N_s} p_{ij}(u_k; \eta_n(s_i)) \times \left\{g\left(s_i, u_k, s_j; \eta_n(s_i)\right) + J^{\mu,\eta_n}\left(s_j\right)\right\}. \tag{10}$$

We define the optimal policy as

$$\mu^*(s_i) \equiv \arg\min_{\mu} \sum_{i=1}^{N_s} \bar{J}^{\mu}(s_i), \quad \forall s_i, \tag{11}$$

which minimizes the J-factor of every state. The J-factor of the optimal policy μ^* is defined as the optimal J-factor, represented by $\bar{J}^*(s_i)$. The objective is to obtain the optimal policy. We assume that there is at least one policy μ satisfying $\bar{J}^{\mu}(s_i) < \infty, \forall s_i$, in this problem. Henceforth, we will call this problem the original problem.

The variation plant in the original problem correlates with differences between any two states. Due to this correlation, the optimal policy does not satisfy Bellman's principle of optimality [14]. Therefore, the optimal policy and the optimal J-factor in this problem do not satisfy (6) and (7). In general, the optimal policy is not stationary. If policies are limited to stationary, the optimal policy is stochastic.

Therefore, another problem definition or another solution method is needed.

6.3. Solutions for a Relaxed Problem of Reinforcement Learning with Variation. We relax the original problem to recover the principle of optimality. Then, we can find the optimal J-factor efficiently by applying DP algorithms to the relaxed problem. We treat a reinforcement learning problem based on a two-player zero-sum game.

We assume that differences, $w_0, w_1, \ldots, w_L$, exist independently in each state s_i. Then the original problem is relaxed to a reinforcement learning problem [15–17] based on a two-player zero-sum game [18] whose objective is to obtain the optimal policy for the worst variation maximizing the expected cost. Since the correlations of differences in a variation plant are ignored, the principle of optimality is recovered.

The $\mathscr{H}_{2pzs}$ is defined as the set of variation plants consisting of all possible combinations of any differences. Since each state has L types of differences, the number of variation plants in $\mathscr{H}_{2pzs}$ is L^{N_s}. We define the optimal policy μ^*_{2pzs} as the policy minimizing the expected cost against the worst variation η^*_{2pzs} maximizing the expected cost

$$\left(\mu^*_{2pzs}, \eta^*_{2pzs}\right) \equiv \arg\min_{\mu} \max_{\eta \in \mathscr{H}_{2pzs}} \sum_{i=1}^{N_s} J^{\mu,\eta}(s_i), \tag{12}$$

and the optimal J-factor $J^*_{2pzs}(s_i)$ is defined as the J-factor of the optimal policy and the worst variation.

Since the principle of optimality is recovered, the optimal J-factor satisfies the following Bellman equation:

$$J^*_{2pzs}(s_i) = \min_{u_k}\max_{w_l}\sum_{j=0}^{N_s} p_{ij}(u_k; w_l)\left\{g\left(s_i, u_k, s_j; w_l\right) + J^*_{2pzs}\left(s_j\right)\right\}. \tag{13}$$

Therefore, the optimal J-factor can be obtained by a DP algorithm. Using the optimal J-factor, the optimal policy and the worst variation are obtained by

$$\left(\mu^*_{2pzs}(s_i), \eta^*_{2pzs}(s_i)\right) = \arg\min_{u_k}\max_{w_l}\sum_{j=0}^{N_s} p_{ij}(u_k; w_l) \times \left\{g\left(s_i, u_k, s_j; w_l\right) + J^*_{2pzs}\left(s_j\right)\right\}. \tag{14}$$

The optimal policy μ^*_{2pzs} is applicable to all L^{N_s} variation plants in $\mathscr{H}_{2pzs}$. The optimal policy μ^*_{2pzs} is proper for all plants in $\mathscr{H}$ of Section 6.2 because $\mathscr{H} \subseteq \mathscr{H}_{2pzs}$ holds. However, the actual number of plants to which the policy should be applied is only N and $N \ll L^{N_s}$. Hence, the optimal policy of the reinforcement learning problem based on the two-player zero-sum game is often conservative and yields poor performance because the problem does not consider the existence of variation plants. We cannot solve this problem if there is no policy satisfying $J^{\mu,\eta}(s_i) < \infty, \forall s_i, \forall \eta \in \mathscr{H}_{2pzs}$, even though the policy μ exists and satisfies $\sum_i J^{\mu,\eta_n}(s_i) < \infty, \forall \eta_n \in \mathscr{H}$. Hence, a solution method to solve the original problem is desired.

6.4. Learning of Peg-in-Hole Task with Variation

6.4.1. Problem Definition of Peg-in-Hole Task with Variations. This section uses the same problem definition for peg-in-hole as Section 5.3.1. The following is added to take variations into account.

The hole position and orientation are measured by the stereo vision system. These measurements involve errors caused by quantized images, lighting conditions at a worksite, and so forth. Time-averaged errors are almost constant while the space robot performs the task, unlike white noise whose time-averaged error is zero. Error evaluations are modeled as described below. Hole position errors are modeled as normal probability distributions, where the mean is $m = 0$ [mm] and standard deviation is $\sigma = 0.75$ [mm]. Hole orientation errors are modeled as normal probability distributions, where the mean is $m = 0$ [rad] and standard deviation is $\sigma = 0.5\pi/180$ [rad]. If the error's statistical values gradually vary, we have to estimate them online. The relative position and orientation between Σ_0 and Σ_{hl} are fixed during the task. The plant variations are modeled as hole position and orientation measurement errors.

Consider these errors as variations $\Delta\mathbf{P}$ added to nominal model $\mathbf{P}_N$ (Figure 6(c)). We constructed 9 plants $\eta_0 \sim \eta_8$ for learning, as listed in Table 1, where each plant has a combination of errors among $[-1.0, 0.0, 1.0]$ [mm] in $\mathbf{j}_0$-axis direction and $[-(0.5/180)\pi, 0.0, (0.5/180)\pi]$ [rad] in $\mathbf{k}_0$-axis rotation. Plant η_0 with no error both in $\mathbf{j}_0$-axis direction and in $\mathbf{k}_0$-axis rotation is the nominal plant. The plant existing probabilities followed the above-mentioned normal probability distributions.

Table 1: Hole position of variation plant η_n from the nominal plant.

Plants	Variations	
	Position in $\mathbf{j}_0$ (mm)	Rotation about $\mathbf{k}_0$ (rad)
η_0	0.0	0.0
η_1	0.0	$-(0.5/180)\pi$
η_2	-1.0	0.0
η_3	-1.0	$-(0.5/180)\pi$
η_4	-1.0	$(0.5/180)\pi$
η_5	0.0	$(0.5/180)\pi$
η_6	1.0	0.0
η_7	1.0	$(0.5/180)\pi$
η_8	1.0	$-(0.5/180)\pi$

In the original problem, plant η_n determines the state transition probability as $p_{ij}(u_k; \eta_n)$ for all state transitions (s_i, u_k, s_j) simultaneously. The state transition probability is represented by $p_{ij}(u_k; w_n(s_i))$ where $\mathscr{W}(s_i) = \{w_0(s_i), \ldots, w_{N-1}(s_i)\}$ and $w_n(s_i) = \eta_n(s_i)$. On the other hand, the two-player zero-sum game allows difference w_l at state s_i to be chosen arbitrarily from $\mathscr{W}(s_i)$. The one-step cost is $g(s_i, u_k, s_j; w_l) = 1$.

In the later simulations and experiments to evaluate the learned results, the hole position and orientation are derived from the above normal probability distribution. In the simulations, a variation in the hole position and attitude is chosen for each episode, but the variation is invariant during the episode.

6.4.2. Learning Method. Under the conditions of the above problem definition, a policy is obtained by the solution in Section 6.3. It is the optimal policy μ^*_{2pzs} of the two-player zero-sum game, that is, $\mathbf{K}''_Q$ in Figure 6(c). The optimal policy μ^*_{2pzs} is applied to the dynamic simulator with a continuous state-action space or the experimental hardware setup, which is a real plant $\mathbf{P}$ in Figure 6(c). No online learning is needed.

There is no proper policy for all plants in $\mathscr{H}_{2pzs}$ if the variations in Table 1 are too large. In this case, there is no policy satisfying the reinforcement learning problem based on the two-player zero-sum game. A typical approach for this situation is to make the variations smaller, to reconstruct $\mathscr{H}_{2pzs}$, and to solve the two-player zero-sum game again. This approach is repeated if we cannot obtain any solutions. This approach reduces the robustness of solutions.

6.4.3. Control Results. In results for numerical simulation (Figure 10), where the peg arrives at the goal using controller $\mathbf{K}''_Q$ in the environment without hole position/orientation error. Peg positioning is firstly inclined, and the peg moves

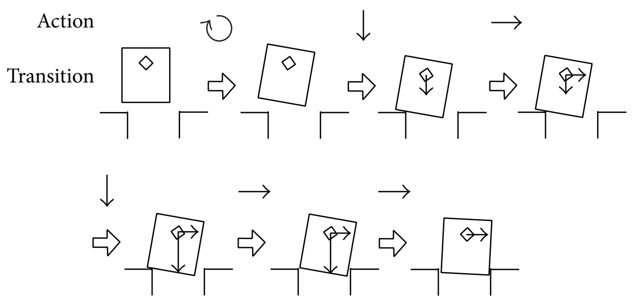


Figure 10: Trajectory of controller $\mathbf{K}_Q''$ in a simulation using nominal plant η_0 without any hole position error.

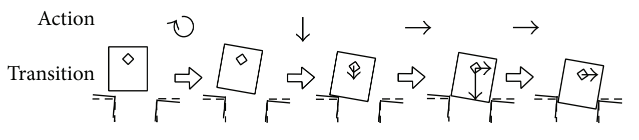


Figure 11: Trajectory of controller $\mathbf{K}_Q''$ in a simulation using plant η_1.

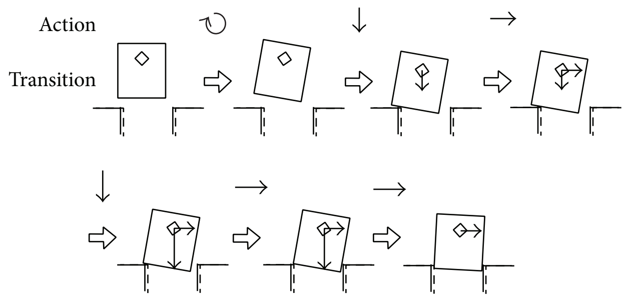


Figure 12: Trajectory of controller $\mathbf{K}_Q''$ in a simulation using plant η_2.

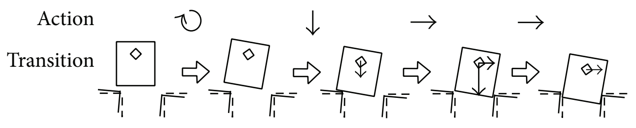


Figure 13: Trajectory of controller $\mathbf{K}_Q''$ in a simulation using plant η_3.

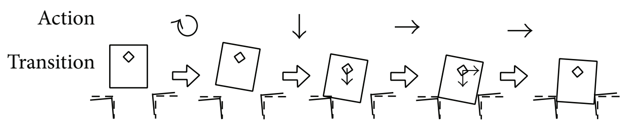


Figure 14: Trajectory of controller $\mathbf{K}_Q''$ in a simulation using plant η_4.

in the positive direction, toward $\mathbf{i}_0$. Then, the peg's corner is inserted in the hole. The peg makes contact with a corner of the hole. The peg transits in a positive direction, toward $\mathbf{j}_0$, while maintaining contact. Another corner of the peg is put inside the hole when the action in the direction of $\mathbf{i}_0$ and $\mathbf{j}_0$ is repeated. Peg positioning is changed to $y_\theta = 0$ [rad], and the peg slips into the hole. The task is completed. The learned result is similar to that of human skill for the peg-in-hole task in Figure 9.

The peg has arrived at the goal using controller $\mathbf{K}_Q''$ for variation plants η_1–η_8. The numerical results for η_1–η_4 are shown in Figures 11, 12, 13, and 14. Each transition is similar to the case of η_0, and the peg is inserted into the hole.

The task is achieved with controller $\mathbf{K}_Q''$ in the same environment with error, where $\mathbf{K}_Q$ previously did not work at all. This means that the action generated by controller $\mathbf{K}_Q''$ is robust against variations as well as human skill. We judge the robot, that is, controller $\mathbf{K}_Q''$, to have obtained a skill, the ability to complete a task when the vision measurement accuracy is low.

7. Evaluation of Obtained Skill

7.1. Results of Hardware Experiments. Example results in the hardware experiment using controllers $\mathbf{K}_Q$ and $\mathbf{K}_Q''$ are shown in Figure 15. The following variations are used: +0.3 [mm] in $\mathbf{i}_0$, +1.2 [mm] in $\mathbf{j}_0$, and $+0.5\pi/180$ [rad] rotation about the $\mathbf{k}_0$-axis. Controller $\mathbf{K}_Q$ cannot complete the task due to environmental variations, but controller $\mathbf{K}_Q''$ can.

7.2. Evaluation of Robustness and Control Performance. Robustness and control performance of controllers $\mathbf{K}_Q$ and $\mathbf{K}_Q''$ are evaluated by simulations and hardware experiments, the peg-in-hole task.

Variation plants, that is, error in hole position and orientation, are derived from the normal probability distribution in Section 6.4. The robustness and the control performance are evaluated, respectively, by the task achievement ratio and the average step number to the goal. The achievement ratio equals the number of successes divided by the number of trials. Table 2 shows the achievement ratios and the average step number of $\mathbf{K}_Q$ and $\mathbf{K}_Q''$ as evaluated by simulations and hardware experiments. The simulations and the experiments are executed 10,000 times and 50 times, respectively. The achievement ratios of $\mathbf{K}_Q$ are 59% and 64% in simulation and hardware experiments. Those of $\mathbf{K}_Q''$ dramatically increase to 99% in numerical simulation and 96% in hardware experiments. These results show that the robot autonomously

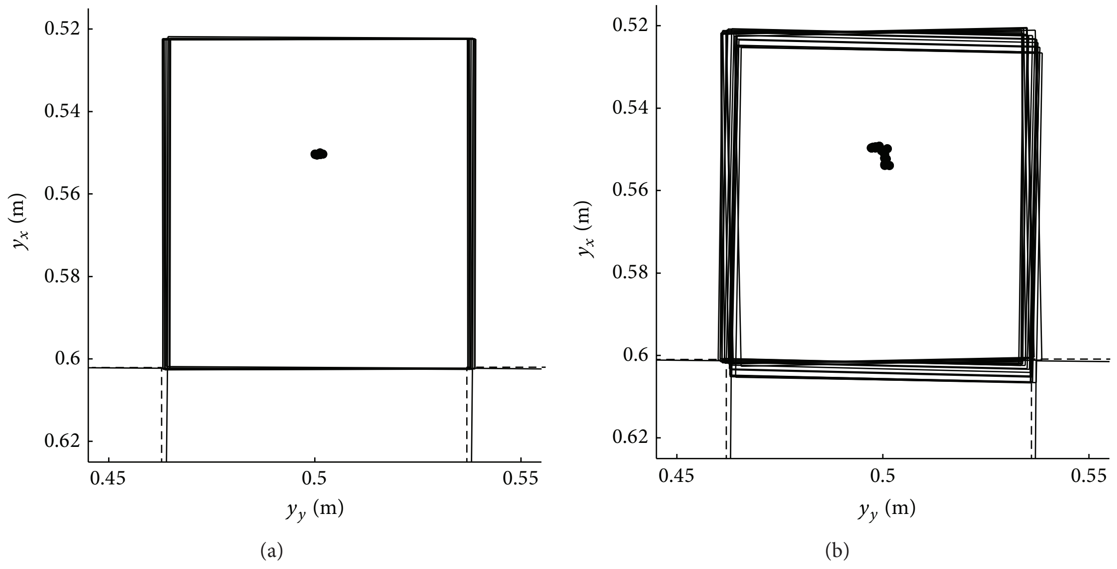


FIGURE 15: Experimental trajectories using two controllers in an environment with error (a) controller $\mathbf{K}_Q$ and (b) controller $\mathbf{K}''_Q$.

TABLE 2: Achievement ratios and averaged step numbers of peg-in-hole task with controllers $\mathbf{K}_Q$ and $\mathbf{K}''_Q$.

Controller	Simulation		Experiment	
	Ratio	Step number	Ratio	Step number
$\mathbf{K}_Q$	58.7%	19.3	64%	21
$\mathbf{K}''_Q$	98.7%	9.83	96%	17

generates robust skill using the proposed learning method. The difference in step numbers between hardware experiments and simulations, an increase in hardware steps, may be due to variations, for example, irregular friction in the environment, joint flexibility, and so forth. Such variables are not considered in the numerical simulation.

Robust skills are thus autonomously generated by learning in this situation, where variations make task achievement difficult.

8. Conclusions

We have applied reinforcement learning to obtain successful completion of a given task when a robot normally cannot complete the task using controller designed in advance. Peg-in-hole achievement ratios are usually low when we use conventional learning without consideration of plant variations. In the proposed method, using variation consideration, the robot autonomously obtains robust skills which enabled the robot to achieve the task. Simulation and hardware experiments have confirmed the effectiveness of our proposal. Our proposal also ensures robust control by conducting learning stages for a set of plant variations.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, Mass, USA, 1998.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[3] S. Fujii, K. Senda, and S. Mano, "Acceleration of reinforcement learning by estimating state transition probability model," *Transactions of the Society of Instrument and Control Engineers*, vol. 42, no. 1, pp. 47–53, 2006 (Japanese).

[4] K. Senda, "An approach to autonomous space robots," *Systems, Control and Information*, vol. 45, no. 10, pp. 593–599, 2001 (Japanese).

[5] K. Senda, Y. Murotsu, A. Mitsuya et al., "Hardware experiments of a truss assembly by an autonomous space learning robot," *Journal of Spacecraft and Rockets*, vol. 39, no. 2, pp. 267–273, 2002.

[6] S. B. Skaar and C. F. Ruoff, Eds., *Teleoperation and Robotics in Space*, AIAA, Washington, DC, USA, 1995.

[7] M. Asada, "Issues in applying robot learning and evolutionary methods to real environments," *Journal of Society of Instrument & Control Engineers*, vol. 38, no. 10, pp. 650–653, 1999 (Japanese).

[8] D. E. Whitney, "Quasi-static assembly of compliantly supported rigid parts," *Journal of Dynamic Systems, Measurement and Control*, vol. 104, no. 1, pp. 65–77, 1982.

[9] D. Sato and M. Uchiyama, "Peg-in-hole task by a robot," *Journal of the Japan Society of Mechanical Engineers*, vol. 110, no. 1066, pp. 678–679, 2007 (Japanese).

[10] N. Yamanobe, Y. Maeda, T. Arai et al., "Design of force control parameters considering cycle time," *Journal of the Robotics Society of Japan*, vol. 24, no. 4, pp. 554–562, 2006 (Japanese).

[11] T. Fukuda, W. Srituravanich, T. Ueyama, and Y. Hasegawa, "A study on skill acquisition based on environment information (task path planning for assembly task considering uncertainty)," *Transactions of the Japan Society of Mechanical Engineers C*, vol. 66, no. 645, pp. 1597–1604, 2000 (Japanese).

[12] F. Miyazaki and S. Arimoto, "Sensory feedback for robot manipulators," *Journal of Robotic Systems*, vol. 2, no. 1, pp. 53–71, 1985.

[13] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using Laplace's equation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2102–2106, May 1990.

[14] K. Senda and Y. Tani, "Optimality principle broken by considering structured plant variation and relevant robust reinforcement learning," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '11)*, pp. 477–483, October 2011.

[15] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of International Conference on Machine Learning*, pp. 157–163, 1994.

[16] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural Computation*, vol. 17, no. 2, pp. 335–359, 2005.

[17] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control," *Automatica*, vol. 43, no. 3, pp. 473–481, 2007.

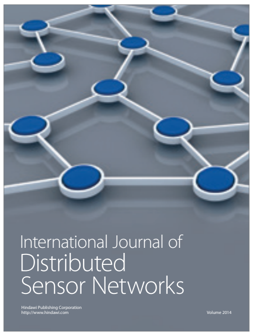
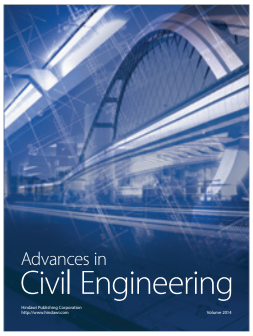
[18] T. Başar and P. Bernhard, *H_∞-Optimal Control and Related Minimax Design Problems*, Birkhäuser, Boston, Mass, USA, 1995.

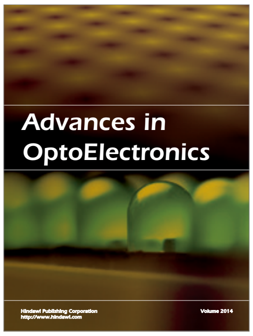
The Scientific World Journal

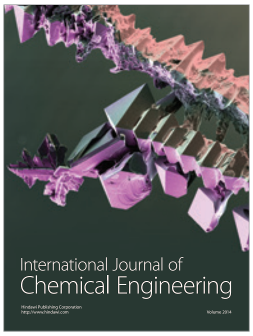
International Journal of Rotating Machinery

Journal of Engineering

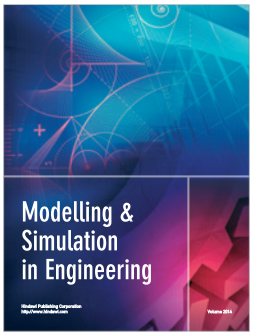
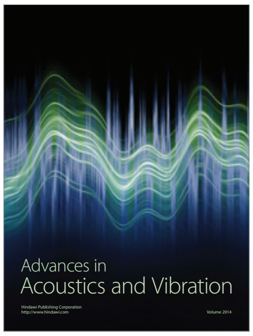
Journal of Sensors

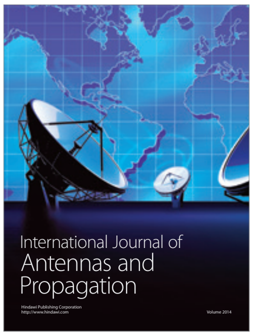
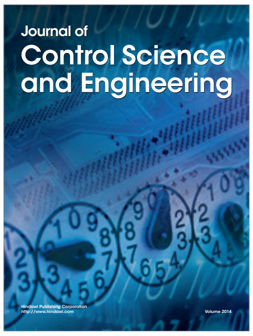
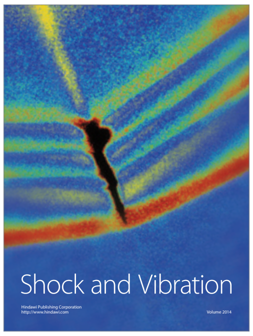
Advances in Mechanical Engineering

International Journal of Distributed Sensor Networks

Advances in Civil Engineering


Advances in OptoElectronics

Journal of Robotics

International Journal of Chemical Engineering

VLSI Design

International Journal of Navigation and Observation

Modelling & Simulation in Engineering

Advances in Acoustics and Vibration

Active and Passive Electronic Components

International Journal of Antennas and Propagation

Journal of Control Science and Engineering

Shock and Vibration

Journal of Electrical and Computer Engineering