We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,300

Open access books available

172,000

International authors and editors

190M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Reinforcement learning approach to object-contact motion with estimation of low-dimensional submanifold and mode-boundary

Yuichi Kobayashi and Shigeyuki Hosoe

*Tokyo University of Agriculture and Technology, RIKEN RTC
Japan*

1. Introduction

One of the challenging issues for robotic motion learning is the acquisition of object-contact motions. It is known that reinforcement learning is an effective approach to robot motion learning. For example, multi-legged locomotion (Kimura et al., 2001), biped locomotion (Nakanishi et al., 2004), and stand-up motion of a link robot (Morimoto & Doya, 2001) are known as successful applications of reinforcement learning to robot motion learning. However, it is difficult to realize object-contact motion learning by simply applying reinforcement learning. The object-contact motions include complex dynamics caused by mode-switching of contact/non-contact or stick/slip. The object-contact motions include both continuous dynamics and discrete events, which has been investigated as hybrid systems control problems (van der Schaft & Schumacher, 2000). In robotic applications, such mode switching is an important issue for the development of learning methods but hitherto not so much attention has been paid from reinforcement learning viewpoint. Apart from reinforcement learning researches, mode switching has been actively discussed in robotic literature. For example, (Schlegl et al., 2002) analysed multi-fingered grasping problem as a hybrid system and proposed a hybrid control architecture with grasping and regrasping. (Yashima et al., 2003) proposed a planning strategy for multi-fingered robotic hand with contact mode switching. There are also general approaches for systematic modelling, planning and control such as Mixed Logic Dynamical model (e.g. (Yin et al., 2007)). Most of these approaches, however, assume that the dynamics of the system is completely known. From the viewpoint of increasing abilities of learning robots, it is very important to consider how to deal with unknown factors in hybrid control problems. In this chapter two different learning schemes are proposed for the acquisition of contact-motion control, incorporating some specific characteristics of the contact motion dynamics. The first approach utilizes the characteristics that the actual motion of the system is restricted to a lower-dimensional submanifold by holonomic constraint when a contact mode is maintained (Fig.1). This helps for narrowing the learning space. The other one depends on the characteristics of object contact motions that the dynamics of the system changes with mode switching of

contact/non-contact or stick/slip. By explicitly approximating the boundary of the mode switching, efficiency of learning can be improved.

In the first learning scheme, an acquisition method of mappings with a parameterization of the low-dimensional submanifold and the approximation method of the boundary of a discontinuous reward function have been proposed. The method is evaluated in an object rotating task. The effectiveness of the proposed learning method was verified by making comparisons with ordinal Q-learning and Dyna.

In the second learning scheme, object manipulation with stick/slip mode switching is realized by a hierarchical structure. The method is applicable to tasks when the desired configuration of the object are specified so that they can be attained only when the switching between stick and slip modes is utilized. The upper layer determines a global trajectory and a sequence of mode switching by reinforcement learning while giving the lower layer a command for small motion not accompanying any (sliding/sticking) mode changes. The lower layer realizes a desired small regional motion using the estimated boundary information about mode switches by Support vector machine (SVM). The proposed learning architecture is evaluated in simulation of an object control task with stick/slip mode switching.

The remainder of the chapter is composed as follows. In section 2, learning of manipulation with constraint is discussed, followed by evaluation by simulation in section 3. The second manipulation task with stick/slip mode switching is described in section 4 including the proposal of hierarchical learning architecture. The learning method is evaluated by simulation in section 5. Section 6 gives discussion and conclusions.

2. Learning of Manipulation with Constraint

2.1 Problem assumptions and overview of learning method

The reinforcement learning problem discussed in this section is classified as a control problem with continuous state space $X \subset \mathbb{R}^{N_1}$ and discrete state set A . The learning agent decides actions by $\pi: X \rightarrow A$ and receives reward $R_t = R(x_t, a_t)$, $x_t \in X, a_t \in A$ with state transition from x_t to x_{t+1} , where $a_t = \pi(x_t)$ and $R(x, a)$ is unknown by the agent. The objective of the agent is to obtain the optimal policy π^* that maximizes cumulated expectation of discounted reward $E_{\pi^*}[\sum_t \gamma^t R_t]$ with discount factor $\gamma (0 < \gamma \leq 1)$.

Fig. 1. Constrained motion of manipulation and submanifold in configuration space

In addition to these ordinal formulations, we assume the followings.

- (a) The state variable $\mathbf{x} \in X$ can be expressed as

$$\mathbf{x} = \begin{bmatrix} \mathbf{z} \\ \dot{\mathbf{z}} \end{bmatrix}, \mathbf{z} \in \mathbb{R}^N, 2N = N_1. \quad (1)$$

- (b) The motion of $\mathbf{z}$ is constrained to a lower-dimensional submanifold P of the configuration space Z as long as the agent is successfully keeping the motion without any failure.

- (c) Reward $R^G(p, \dot{p}, a)$ is given for reaching the goal region where $p \in P$.

- (d) When the agent fails the task, a trial is finished and a new trial is started from an initial state.

The first assumption is a standard framework for dynamic systems. The second assumption is valid for the problems with holonomic constraints. For simplicity, it will be assumed that $N=2$ i.e., Z is a plane. This implies that P must be a curve in the plane since it is a real submanifold in Z . Therefore, the motion of $\mathbf{z}$ can be expressed by one-dimensional parameter $p \in \mathbb{R}$ and state of the system is expressed by $[p, \dot{p}] \in X \subset \mathbb{R}$ as far as the constraint is maintained.

Failing in the fourth assumption means that the agent cannot continue the task because of some fatal error. When the failure happens, it is assumed that the agent can recognize the situation by getting some negative reward. That is, failure is expressed as

$$R_a^F \begin{cases} R_{fail} < 0 & \text{if fails with action } a \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The boundary between the region where R_a^F takes 0 and the region where R_a^F takes R_{fail} is assumed to be continuous in S as indicated in the right hand of Fig. 2. In the application to reinforcement learning framework, the value function is calculated using the sum of reward as $R = R^G(p, \dot{p}, a) + R^F(p, \dot{p}, a)$.

Fig. 2. Constrained motion of manipulation and submanifold in configuration space.

Based on the above assumptions, a learning scheme composed of the following three basic elements will be proposed.

- (a) A mapping from Z to one-dimensional parameter space $P \subset \mathbb{R}$.
- (b) Model-based reinforcement learning with system dynamics and reward function approximation

(c) Approximated representation for reward function R_a^F with discontinuous jump as indicated by Eqn. (2)

The reward function approximation in (c) mainly focuses on the reward R^F for failing the task. The function approximation of system dynamics and reward function are presented based on the mapping in (a).

2.2 Mapping to one-dimensional space

It is assumed that z belongs to a plane, i.e., $N=2$ in (1) and $z(t)$ moves along a one-dimensional submanifold (a curve) as long as no failure happens. In on-line learning, the agent constructs an estimated representation of the curve using observed sampled data of $z(t)$. For this, we can use various kind of curve fitting theories such as Spline, Bezier or Ferguson curve (Farin, 2001). Here, we will apply the second order polynomial interpolation method for simplicity. In on-line learning, $z(t)$ is observed at every time step with interval Δt and new samples are added if necessary. Let Q denote the set of sample points. We add new sample points when

$$Q_L < \min_{q_i \in Q} \|q_i - z(t)\|, \quad (3)$$

where Q_L denotes the threshold length used for judgment. The threshold value Q_L should be sufficiently small for the precise representation of the nonlinear curve.

Fig. 3. Mapping to 1D submanifold and finding nearest point on curve

When given n sampled points $q_i \in Z, i=0, \dots, n-1$ of $z(t)$, i -th interval on the one-dimensional curve can be approximated by

$$z^{(i)}(p') = q_i + v_{1i}p' + v_{2i}p'^2, p' \in [0,1], \quad (4)$$

where $z^{(i)}$ denotes i -th interval of the curve, as shown in the right hand of Fig. 3. $v_{1i}, v_{2i} \in \mathbb{R}^{N'}$ are coefficient vectors which specify the shape of the curve in the i -th interval. When we differentiate this equation by p' , we have

$$\frac{\partial \mathbf{z}^{(i)}}{\partial \mathbf{p}'}(\mathbf{p}') = \mathbf{v}_{1i} + 2\mathbf{v}_{2i}\mathbf{p}', \quad (5)$$

which gives the tangential vector of the curve at $\mathbf{z}^{(i)}(\mathbf{p}')$. Considering the continuity of i -th and $(i+1)$ -th intervals ($\mathbf{z}^{(i)}(1) = \mathbf{z}^{(i+1)}(0)$), Eqn. (4) has to satisfy

$$\mathbf{q}_i + \mathbf{v}_{1i} + \mathbf{v}_{2i} = \mathbf{q}_{i+1}, \quad i = 0, \dots, n-1. \quad (6)$$

Similarly considering the continuity of the tangents ($\frac{\partial \mathbf{z}^{(i)}}{\partial \mathbf{p}}(1) = \frac{\partial \mathbf{z}^{(i+1)}}{\partial \mathbf{p}}(0)$) the followings have to hold:

$$\mathbf{v}_{1i} + 2\mathbf{v}_{2i} = \mathbf{v}_{1i+1}, \quad i = 0, \dots, n-2. \quad (7)$$

Let $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ denote the identity matrix. By letting

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{I}_N \mathbf{I}_N & & \mathbf{O} \\ & \mathbf{I}_N \mathbf{I}_N & \\ & & \ddots \\ \mathbf{O} & & & \mathbf{I}_N \mathbf{I}_N \end{bmatrix} \in \mathbb{R}^{nN \times 2nN}, \quad (8)$$

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{I}_N & 2\mathbf{I}_N & -\mathbf{I}_N & & \mathbf{O} \\ & \mathbf{I}_N & 2\mathbf{I}_N & -\mathbf{I}_N & \\ & & \ddots & & \\ \mathbf{O} & & & \mathbf{I}_N & 2\mathbf{I}_N & -\mathbf{I}_N \end{bmatrix} \in \mathbb{R}^{(n-1)N \times 2nN}, \quad (9)$$

and

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_{10} \\ \mathbf{v}_{20} \\ \vdots \\ \mathbf{v}_{1n-1} \\ \mathbf{v}_{2n-1} \end{bmatrix} \in \mathbb{R}^{2nN}, \quad \mathbf{Q}_1 = \begin{bmatrix} \mathbf{q}_1 - \mathbf{q}_0 \\ \vdots \\ \mathbf{q}_n - \mathbf{q}_{n-1} \end{bmatrix} \in \mathbb{R}^{nN}, \quad (10)$$

we can rewrite (6) and (7) in the matrix form as

$$\mathbf{A}\mathbf{V} = \mathbf{Q}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 \\ \mathbf{O}_{(n-1)N \times 1} \end{bmatrix}. \quad (11)$$

$\mathbf{V}$ can be calculated using pseudo-inverse matrix as

$$\mathbf{V} = \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{Q}. \quad (12)$$

Thus, we can decide the parameters $\mathbf{v}_{1i}$ and $\mathbf{v}_{2i}$ from sample points $\mathbf{q}_i$. Using $\mathbf{V}$, we can specify $\mathbf{p}$ for observed $\mathbf{z}$ by finding the nearest point on curve. The squared error for i -th interval on the curve is defined as

$$E_i(p', z) = \|\mathbf{q}_i + \mathbf{V}_{1i}p' + \mathbf{V}_{2i}p'^2 - \mathbf{z}\|^2, \quad p' \in [0, 1], i = 0, \dots, n-1. \quad (13)$$

Parameter p and i giving the minimum will be represented as

$$[i_{\min}, p_{\min}] = \arg \min_{i, p'} E_i(p', \mathbf{z}). \quad (14)$$

The minimum of $E_i(p, z)$ can be obtained by solving the cubic equation derived from the

local minimum condition $\frac{\partial E_i(p, z)}{\partial p} = 0$. Eqn. (14) is calculated at every time step t to obtain $i_{\min}(t)$ and $p_{\min}(t)$. By connecting parameterization (4) for subintervals, the whole curve can be parameterized by:

$$p(t) = i_{\min}(t) + p_{\min}(t) \quad (15)$$

$$p(t) = (p(t) - p(t - \Delta t)) / \Delta t. \quad (16)$$

Since $p_{\min}$ and $i_{\min}$ are determined from z by Eqn(14), $p_{\min}$, $i_{\min}$ and therefore p is described as a function of z . We interpret this as Eqn. (14) and Eqn. (15) defines a mapping from $z(t)$ to $p(t)$. When the minimum in Eqn. (14) is not attained in any intervals, that is, z lies in the outside of the curve, $p(t)$ is calculated by

$$p(t) = \begin{cases} -\|q_0 - z\| / Q_L & \text{if } \|q_0 - z(t)\| < \|q_{n-1} - z(t)\| \\ n + \|q_{n-1} - z\| / Q_L & \text{otherwise} \end{cases} \quad (17)$$

instead of Eqn. (15). We define $s(t) \equiv [p(t), \dot{p}(t)]^T$ for later discussion. In reinforcement learning application, the sample points are obtained successively while exploration.

2.3 Approximation of Reward Function

Now, based on the obtained mapping as above, the reward function can be represented by using lower dimensional parameter as $R_a(s)$ instead of $R(x, a)$. Based on the assumption for the reward function, two components of $R_a(s)$ are to be approximately constructed through learning.

$$\tilde{R}_a(s) = \tilde{R}_a^G(s) + \tilde{R}_a^F(s), \quad \forall a \in A \quad (18)$$

The approximate learning of $\tilde{R}_a^F(s)$ is done by following procedures. Suppose the reward for failure R_{fail} is given at time t_{fail} . Sample is added at this moment as shown below.

$$M_a \leftarrow M_a \cup \{u^a\}, \quad u^a = \left[s(t_{\text{fail}})^T, R_{\text{fail}} \right]^T, \quad (19)$$

where $a = a(t_{\text{fail}})$. Similarly, observed data at the previous time step of t_{fail} is also added by

$$M_a \leftarrow M_a \cup \{u^a\}, \quad u^a = \left[s(t_{\text{fail}} - \Delta t)^T, 0 \right]^T. \quad (20)$$

Using the sample set M_a , the approximate function for $\tilde{R}_a^F(s)$ is constructed by the nearest neighbor method. First, we find $u_{\min}(s)$ as

$$u_{\min}(s) = \arg \min_{u^a \in M_a} \|s - [u_1^a, u_2^a]^T\| \quad (21)$$

Using above, the approximation function is defined by

$$\tilde{R}_a^F(s) = u_{\min 3}^a(s), \quad (22)$$

where u_i^a denotes i -th component of vector u^a . Left hand of Fig. 4. shows an example of approximating $\tilde{R}_a^F(s)$ using nearest neighbour method, which corresponds to Voronoi division of the state space. A circle in the figure corresponds to $u_3^a = 0$ and a cross

corresponds to $\mathbf{u}_3^a = \mathbf{R}_{\text{fail}}$.

Fig. 4. Voronoi division of state space and state transition approximation

2.4 Reinforcement learning of Controlling Constrained Motion

The basic idea of the proposed learning method is based on the model-based learning method, which is known as Dyna (Sutton, 1991). The main procedures for model-based reinforcement learning are as follows:

- state transition and reward function are approximated by online sampling,
- value function $V(s)$ is updated using these approximations, and
- policy π is improved successively using $V(s)$.

State transition can be denoted as a function of state variables and action.

$$\dot{s}(t) = f(s(t), a(t)). \quad (23)$$

The state transition function is approximated by the same framework of the nearest-neighbor method described in section 2.3. Value function $V(s)$ is updated with respect to each grid as

$$V(s) = \max_{a \in A} \left[\gamma \sum_{i=1}^d \rho(s_i | s, a) V(s_i) + \tilde{R}_a(s) \right] \quad (24)$$

as discussed in optimal control (e.g. (Munos & Moore, 2001)). The state transition probability ρ is calculated for the neighbors in the two-dimensional grid as

$$\rho(s_i | s, a) = \frac{\exp\left(-\frac{\|s_i - \bar{s}\|^2}{\sigma^2}\right)}{\sum_{k=1}^d \exp\left(-\frac{\|s_k - \bar{s}\|^2}{\sigma^2}\right)}, \quad \bar{s} = s + \tilde{f}(s, a)\Delta t, \quad (25)$$

where $\tilde{f}$ denotes the approximated state transition function. The right hand of Fig.4. shows an example of $\bar{s}$ and its neighbor grids in the two-dimensional case. In the ϵ -greedy policy, action is chosen randomly with a small probability ϵ and decided by

$$\pi(\mathbf{p}) = \arg \max_{a \in A} \left[\gamma \sum_{i=1}^d \rho(\mathbf{p}_i | \mathbf{p}, a) V(\mathbf{p}_i) + \tilde{R}_a(\mathbf{p}) \right], \quad (26)$$

with probability $1 - \varepsilon$.

The total learning algorithm proposed for the constrained motion acquisition including successive addition of sample $\mathbf{q}_i$ is described as Table 1.

1. $Q \leftarrow \{\mathbf{q}_0\}, \mathbf{q}_0 = [q_1(t_0), q_2(t_0)]^T$
2. Initialize state $t = 0, \mathbf{x}(0) = \mathbf{x}_0$
3. Observe $\mathbf{z}(t)$
4. If Eqn. (3) holds, add sample data
 $\mathbf{q}_j = \mathbf{q}(t), j = |Q| + 1, Q \leftarrow Q \cup \{\mathbf{q}_j\}$
5. Obtain $p(t)$ by the mapping using Eqn. (15)
6. Decide control input based on ε -greedy:
 (a) Random action selection at probability ε
 (b) Otherwise, $a = \pi(\mathbf{s}(t))$.
7. State transition with reward R_t and judgement for failing the task (whether $R_t = R_{\text{fail}}$ or not)
8. Data sampling for identification of state transition and reward functions.
9. If trial completed or failed, update $V(\mathbf{s})$ and $\pi(\mathbf{s})$ and go to 2.
10. $t \leftarrow t + \Delta t$ and go to 3.

Table 1. Learning algorithm for constrained motion

3. Simulation and evaluation of Constrained Motion Learning

3.1 Dynamics of the model

A two-link manipulator and an object with one-DOF rotational movement are shown in Fig. 5. The contact between the link and the object is point contact without rolling. The objective of the task is to stabilize the object around some posture by the link manipulator.

Fig. 5. Rotation of one-DOF object

Let q_1 and q_2 be the angles of the link manipulator and θ be angle of the object. τ_1 and τ_2 are the torques for joints 1 and 2, respectively. Because of the constraint, the actual motion of this problem has one DOF as long as the manipulator remains in contact with the object. Letting $q = [q_1, q_2]^T$, the position constraint while the end of link 2 contacts with the object can be described as

$$l_1 \cos q_1 + l_2 \cos(q_1 + q_2) = x_r L \cos \theta \quad (27)$$

$$l_1 \sin q_1 + l_2 \sin(q_1 + q_2) = y_r L \sin \theta, \quad (28)$$

where L , l_1 and l_2 are the lengths of the object, link 1 and link 2, respectively, and $[x_r, y_r]^T$ is the center of object rotation. The motion equation of the manipulator can be written as

$$M_q(q)\ddot{q} + h_q(q, \dot{q}) + J_q^T F = [\tau_1, \tau_2]^T, \quad (29)$$

where $F \in \mathbb{R}^2$ denotes the contact force between the link and the object. Let m and I_0 denote the mass and inertia of the object, respectively. μ_R denotes the viscosity resistance coefficient. The dynamics of the object can be written as

$$I_0 \ddot{\theta} + \mu_R \dot{\theta} + \frac{1}{2} L m g \cos \theta - J_\theta^T F = 0. \quad (30)$$

We define h_θ as

$$h_\theta(\theta, \dot{\theta}) = \mu_R \dot{\theta} + \frac{1}{2} L m g \cos \theta. \quad (31)$$

The Jacobian matrices J_q and J_θ in Eqn. (29) and (30) are obtained by differentiating the constraints (27) and (28) by time. Using above-mentioned notations, the integrated motion equation may be written as

$$\begin{bmatrix} M_q(q) & 0 \\ 0 & I_0 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} h_q(q, \dot{q}) \\ h_\theta(\theta, \dot{\theta}) \end{bmatrix} + \begin{bmatrix} J_q^T \\ J_\theta^T \end{bmatrix} F = \begin{bmatrix} \tau_1 \\ \tau_2 \\ 0 \end{bmatrix}, \quad (32)$$

or equivalently as

$$M(z)\ddot{z} + h(z, \dot{z}) + J^T F = \tau, \quad (33)$$

where $z = [q_1, q_2, \theta]^T$ and $\tau = [\tau_1, \tau_2, 0]^T$. The contact force F can be written as

$$F = (JM^{-1}J^T)^{-1} \{ JM^{-1}(\tau - h) + \dot{J}\dot{z} \}. \quad (34)$$

When we define F_c in the object coordinate

$$F_c = \begin{bmatrix} F_{cx} \\ F_{cy} \end{bmatrix} = R_\theta F \quad (35)$$

by rotating F by θ , the conditions for maintaining contact with the object are written as

$$F_{cy} > 0 \quad (36)$$

$$\mu F_{cy} > F_{cx}, \quad (37)$$

where μ denotes the static friction coefficient. Eqn. (37) gives the condition for avoiding slipping. If either of the conditions is broken, a trial is aborted and the agent restarts the trial from the initial state z_0 . These conditions for maintaining contact give the boundary of reward function for maintaining constraints.

3.2 Assumption for learning agent

It is assumed that the agent

- observes q_1 and q_2 and their velocities $\dot{q}_1$ and $\dot{q}_2$
- he force F_c and the object angle θ , but receives the reward for reaching goal region and the reward for failing to maintain contact with the object.

In addition to these assumptions for agent observation, the agent utilizes the knowledge described in section 3.1 through the proposed mapping method and reward function approximation.

3.3 Simulation Conditions

We evaluate the proposed learning method in the problem described in section 3.1. Although we show the effectiveness of the proposed learning method through a problem where analytical solutions can be easily found, it does not mean this method is restricted to such problems. The method can be applied to other problems where we can not easily derive analytical solutions, e.g., manipulation problems with non-spherical fingertips or with moving joints structures, which can be seen in human arms.

Physical parameters are set as $l_1 = 2, l_2 = 2, L = 1/2$ [m], $m_0 = 0.8$ [kg], $\mu = 0.8$. $[x_r, y_r] = [2.5, 0]$ and the initial state is set as $\mathbf{z}_0 = [3/\pi, -2/3\pi, \pi]^T$. Sampling time for the control is 0.25[sec] and is equivalent to one step in a trial. We have 4×4 actions by discretizing τ_1 and τ_2 into $[60, 30, 0, -60]$ [Nm]. One trial is finished after 1,000 steps or when either of conditions (27) or (28) is broken. If either $\theta(t)$ or $\dot{\theta}(t)$ goes out of the interval $[\theta_{\min}, \theta_{\max}] = [0, \pi]$ or $[\dot{\theta}_{\min}, \dot{\theta}_{\max}] = [-5, 5]$, a trial is also aborted. The reward function is given as

$$R(x, a) = R^1(x, a) + R^2(x), \quad (38)$$

where each component is given by

$$R^1(x, a) = \begin{cases} 10 & \text{if } |\theta - \theta_d| < \pi/10 \\ -|\theta - \theta_d| & \text{otherwise} \end{cases} \quad (39)$$

and

$$R^2(x) = \begin{cases} 0 & \text{if (27) and (28) hold} \\ -100 & \text{otherwise} \end{cases} \quad (40)$$

The desired posture of the object is $\theta_d = \pi/2$. The threshold length for adding new samples in the mapping construction is set as $Q_L = 0.05$. The state space constructed by $s \in \mathcal{R}^2$ is divided into 40×40 grids with the the regions $[p_{\min}, p_{\max}] = [0, 5]$ and $[\dot{p}_{\min}, \dot{p}_{\max}] = [-5, 5]$. The parameters for reinforcement learning are set set as $\mathcal{E} = 0.1$ and $\gamma = 0.95$

The proposed reinforcement learning method is compared with two candidates.

- Model-based reinforcement learning without mapping F_Q using $[q_1, q_2, \dot{q}_1, \dot{q}_2]$ as state variables.
- Ordinal Q-learning with state space constructed by the state variables $s = [p, \dot{p}]$

The first method is applied to evaluate the effect of introducing the mapping to one-dimensional space. The second method is applied to see that the explicit approximation of discontinuous reward function can accelerate learning.

3.4 Simulation Results

The obtained mapping is depicted in the left hand of Fig. 6. The bottom circle corresponds to the initial state with $\mathbf{z}_0$ and each circle in the figure denotes a sample. The right hand of Fig. 6. shows the reward profiles obtained through trials. We can see that performance is not always sufficiently good even after many trials. This is caused by the $\mathcal{E}$ -greedy policy and the nature of the problem. When the agent executes random action based on the $\mathcal{E}$ -greedy policy, it can easily fail to maintain contact with the object even after it acquired a sufficiently good policy not to fail.

Fig. 6. Obtained 1-D mapping and learning curve obtained by the proposed method

The left hand of Fig.7 shows the state value function $V(\mathbf{s})$. It can be seen that the result of exploration in the parameterized state space is reflected in the figure where the state value is non-zero. The positive state value means that it was possible to reach the desired configuration through trials. The right hand of Fig.7 shows the learning result with Q-learning as a comparison. In the Q-learning case, the object did not reach the desired goal region within 3,000 trials. With four-dimensional model-based learning, it was possible to reach the goal region. Table 2 shows comparisons between the proposed method and the model-based learning method without lower-dimensional mapping. The performances of the obtained controllers after 3,000 trials learning are evaluated without random exploration (that is, $\mathcal{E}=0$) with ten test sets. The average performance of the proposed method was higher. This is caused by the fact that the controller obtained by the learning method without the mapping failed to keep contact between the arm and the object at earlier stages of the rotating task in many cases, which resulted in smaller cumulated rewards. Additionally in the case of the method without the mapping, calculation time for the control was three times as long as the proposed method case.

Fig. 7. State value function and learning curve obtained by Q-learning

	Proposed method	Without mapping
Average reward	-3.87	-90.7
Standard deviation	8.53	34.8

Table 2. Comparison with model-based reinforcement learning without mapping

The examples of the sampled data for reward approximation are shown in Fig. 8. Circles in the left hand figure denote $u_3^a = 0$ and the crosses denote $v_3^a = R_{\text{fail}}$. The reward functions $\tilde{R}_{13}^F(\mathbf{s})$ approximated using corresponding sample data are also shown in the figure. Fig. 9 shows an example of the trajectories realized by the obtained policy $\pi(s)$ without random action decisions in the parameterized state space and in the physical space, respectively.

Fig. 8. Sampled data for reward estimation ($a=13$) and approximated reward $\tilde{R}_{13}^F(s)$

Fig. 9. Trajectory in the parameterized state space and trajectory of links and object

3.5 Discussion

The result of simulation showed that the reinforcement learning approach effectively worked for the manipulation task. Through comparison between Q-learning and model-based reinforcement learning without the proposed mapping, we saw that the proposed mapping and reward function approximation improved the learning performance including calculation time. Some parameter settings should be adjusted to make the problem more realistic, e.g., friction coefficient, which may require more trials to obtain a sufficient policy by learning. For the purpose of focusing on the state space construction, we assumed discrete actions in the learning method. In the example of this manipulation task, however, the continuous control of input torques plays an important role in realizing more dexterous manipulation. It is also useful for the approximation of reward to consider the continuity of actions. The proposed function approximation with low-dimensional mapping is expected to be a base for such extensions.

4. Learning of Manipulation with Stick/Slip contact mode switching

4.1 Object Manipulation Task with Mode Switching

This section presents a description of an object manipulation task and a method for simulating motions with mode switching. Note that mathematical information described in this section is not used by the learning agent. Thus, the agent can not predict mode switching using equations described in this section. Instead, it estimates the mode boundary by directly observing actual transitions (off-line).

Fig. 10. Manipulation of an object with mode switching

An object manipulation task is shown in Fig.10. The objective of the task is to move the object from initial configuration to a desired configuration. Here, it is postulated that this has to be realized by putting robot hand onto the object and moving it forward and backward by utilizing friction between the hand and the object as shown in the figure. Note that, due to the limited working ranges of joint angles, mode change (switching contact conditions between the hand and the object from slipping mode to stick mode and vice versa) is generally indispensable to achieve the task. For example, to move the object close to the manipulator, it is necessary once to slide the hand further (from the initial position) on the object so that the contact point becomes closer to point B in Fig.11.

Physical parameters are as described in Fig.11. The followings are assumed about physical conditions for the manipulation:

- The friction is Coulomb type frictions and the coefficient of static friction is equal to the coefficient of kinetic friction
- The torque of the manipulator is restricted to $\tau_{1\min} \leq \tau_1 \leq \tau_{1\max}$ and $\tau_{2\min} \leq \tau_2 \leq \tau_{2\max}$.
- The joint angles have limitations of $q_{1\min} \leq q_1 \leq q_{1\max}$ and $q_{2\min} \leq q_2 \leq q_{2\max}$.
- The object and the floor contact at a point and the object does not do rotational motion.
- A mode where both contact points (hand and object / object and floor) are slipping is omitted (Controller avoids such mode).

In what follows the contact point between the hand and the object will be referred as point 1 and the contact point between the object and the floor as point 2. It is assumed that the agent can observe at each control sampling time the joint angles of the manipulator and their velocities and also

- position and velocity of the object and the ones of contact point 1.
- contact mode at contact point 1 and 2 (stick/slip to positive direction of x axis/slip to negative direction of x axis/apart).

Concerning the learning problem, the agent is assumed to know or not know the following factors: It knows the basic dynamics of the manipulator, i.e., gravity compensation and Jacobian matrix are known (they correspond to g_q and J_q in Eqn. (41)). On the other hand, the agent does not know conditions for the mode switching. That is, friction conditions are unknown including friction coefficients. The agent also does not know the limitation of joint angles and sizes (vertical and horizontal lengths) of the object.

From the viewpoint of application to the real robot, it might be not easy to measure the contact mode precisely, because 1) it is difficult to detect small displacement of the object (e.g. assuming visual sensor) and 2) the slipping phenomenon could be stochastic. In the real application, estimation of mode boundary might require further techniques such as noise reduction.

Fig. 11. Manipulator and a rectangular object

4.2 System Dynamics and Physical Simulation

Motion equation of the manipulator is expressed by

$$M_q(q)\ddot{q} + h_q(q, \dot{q}) = [\tau_1, \tau_2]^T + J_t^T F_t + J_n^T F_n, \quad (41)$$

where $q = [q_1, q_2]^T$, $F_t = [F_{1t}, F_{2t}]^T$, $F_n = [F_{1n}, F_{2n}]^T$, $J_t = [J_{1t}^T, 0_{2 \times 1}]^T$, $J_n = J_n = [J_{1n}^T, 0_{2 \times 1}]^T$ and $J_q = [J_{1t}^T, J_{1n}^T]^T$ is Jacobian matrix of the manipulator. F_{it} and F_{in} denote tangential and normal force at point i , respectively. Zero vectors in J_t and J_n denote that the contact forces at point 2 do not affect the dynamics of the manipulator. Letting $\phi = [x, y]^T$, motion equation of the object is expressed by

$$M_o \ddot{\phi} + g_o = W_t F_t + W_n F_n, \quad (42)$$

where $g_o = [0, mg]^T$ and

$$W_t = \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix}, W_n = \begin{bmatrix} 0 & 0 \\ -1 & 1 \end{bmatrix}. \quad (43)$$

δ_i denotes contact mode at contact point i and defined as

$$\delta_i(t + \Delta t) = \begin{cases} 0(\text{stick}) & \text{if } v_{it}(t + \Delta t) = 0 \\ 1(\text{slip to +direction}) & \text{if } v_{it}(t + \Delta t) > 0, \\ -1(\text{slip to -direction}) & \text{if } v_{it}(t + \Delta t) < 0 \end{cases} \quad (44)$$

where v_{it} denotes relative (tangential) velocity at contact point i . At each contact point, normal and tangential forces satisfy the following relation based on Coulomb friction law.

$$\mu F_n - |F_t| \geq 0. \quad (45)$$

Relative velocities of the hand and the object at contact point 1 are written as

$$v_n = J_n \dot{q} + W_n^T \dot{\phi}, v_t = J_t \dot{q} + W_t^T \dot{\phi}. \quad (46)$$

By differentiating and substituting Eqns. (41) and (42), the relation between relative acceleration and contact force can be obtained as

$$a = AF + a^0, a = [a_n \ a_t]^T, F = [F_n \ F_t]^T, \quad (47)$$

where

$$A = JMJ^T, M = \begin{bmatrix} M_q^{-1} & \\ & M_o^{-1} \end{bmatrix}, J = \begin{bmatrix} J_n & W_n^T \\ J_t & W_t^T \end{bmatrix}, a^0 = J \begin{bmatrix} \dot{q} \\ \dot{\phi} \end{bmatrix} + JM \begin{bmatrix} \tau - h_q \\ g_o \end{bmatrix}. \quad (48)$$

By applying Euler integration to (47) with time interval Δt , relation between relative velocity and the contact force can be obtained as

$$v(t + \Delta t) = KF + b, \quad K \equiv A\Delta t, \quad b \equiv a^0 \Delta t + v(t). \quad (49)$$

On the other hand, normal components of contact force and relative velocity have the following relation.

$$F_{in} \geq 0, \quad (50)$$

$$v_{in} \geq 0, \quad (51)$$

$$F_{in} = 0 \quad \text{or} \quad v_{in} = 0. \quad (52)$$

This relation is known as linear complementarity. By solving (49) under conditions of (45) and (50)-(52), contact forces and relative velocities at next time step can be calculated. In this chapter, projected Gauss-Seidel method (Nakaoka, 2007) is applied to solve this problem.

4.3 Hierarchical Architecture for Manipulation Learning

The upper layer deals with global motion planning in $x-l$ plane using reinforcement learning. Unknown factors on this planning level are 1) limitation of state space of $x-l$ plane caused by the limitation of joint angles and 2) reachability of each small displacement by lower layer. The lower layer deals with local control which realizes small displacement

given by the upper layer as command. The estimated boundary between modes by SVM is used for control input (torque) generation.

Fig.12 shows an overview of the proposed learning architecture. Configuration of the system is given to the upper layer after discretization and interpretation as discrete states. Actions in the upper layer are defined as transition to adjacent discrete states. Policy defined by reinforcement learning framework gives action a as an output. The lower layer gives control input τ using state variables and action command a . Physical relation between two layers is explained in Fig.4. Discrete state transition in the upper layer corresponds to small displacement in x - l plane. When an action is given as command, the lower layer generates control inputs that realizes the displacement by repeating small motions for small time period Δt until finally s' is reached. In this example in the figure, l is constant during state transition.

Fig. 12. Hierarchical learning structure

4.4 Upper layer learning for Trajectory Generation

For simplicity and easiness of implementation, Q-learning (Sutton, 1998) is applied in the upper layer. The action value function is updated by the following TD-learning rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_a Q(s', a) - Q(s, a)] \quad (53)$$

The action is decided by the ϵ -greedy method. That is, a random action is selected by small probability ϵ and otherwise the action is selected as $a = \arg \max Q(s, a)$. The actual state transition is achieved by the lower layer. The reward is given to the upper layer depending on the state transition.

4.5 Lower Controller Layer with SVM Mode-Boundary Learning

When current state $X(t) = [x(t), l(t), \dot{x}(t), \dot{l}(t)]^T$ control input $\tau(t)$ are given, contact mode at

next time $\delta(t+\Delta t)$ can be calculated by projected Gauss-Seidel method. This relation between X , u and δ can be learned as a classification problem in X - u space. A nonlinear Support Vector Machine is used in our approach to learn the classification problem. Thus, mode transition data are collected off-line by changing $x, l, \dot{x}, \dot{l}, \tau_1, \tau_2$. Let m_s denote training set size and $\mathbf{d} \in \mathbb{R}^{m_s}$ denote a vector with plus or minus ones, where plus and minus correspond respectively to different two modes. In non-linear SVM with Gaussian kernel, by introducing kernel function K (with query point v) as

$$K(v, \mu_i) = \exp\left(-\frac{\|v - \mu_i\|^2}{\sigma^2}\right), \quad (54)$$

where $\mu_i = [x, l, \dot{x}, \dot{l}, \tau_1, \tau_2]^T$ denotes i -th data for mode boundary estimation and σ denotes a width parameter for the Gaussian kernel, separation surface between two classes is expressed as

$$\sum_{i=1}^{m_s} d_i w_i K(v, \mu_i) = 0, \quad (55)$$

where w is a solution of the following optimization problem:

$$\min_w \left\{ \frac{1}{2} w^T Q w - e^T w \right\}, \quad (56)$$

where Q is given by

$$Q = \frac{1}{v} + H H^T, \quad H = D[\mu_0 - e], \quad v > 0 \quad (57)$$

and $\mathbf{e} \in \mathbb{R}^{n_s}$ denote the vector of ones. $D = \text{diag}[d_1, \dots, d_{m_s}]$, $\mu_0 = [\mu_1, \dots, \mu_{m_s}]^T$ and v is a parameter for the optimization problem. Note that matrix D gives labels of modes. For implementation of optimization in (56), Lagrangian SVM (Mangasarian & Musicant, 2001) is used. After collecting data set of D and μ_0 and calculating SVM parameter w , (55) can be used to judge the mode at next time step when $X = [x(t), l(t), \dot{x}(t), \dot{l}(t)]^T$ is given.

When the action command a is given by the upper layer, the lower layer generates control input by combining PD control and mode boundary estimation by SVM. Let $\Delta\chi(a) = [\Delta x, \Delta l]^T$ denote displacement in x - l space which corresponds to action a (notice that here χ is different from X because velocities are not necessary in the upper layer). When $\Delta l = 0$, the command a means that the modes should be maintained as $\delta_1 \neq 0$ and $\delta_2 = 0$. When $\Delta l = 0$ on the other hand, it is required that the modes should be $\delta_1 = 0$ and $\delta_2 \neq 0$. Thus, the desired mode can be decided depending on the command $\Delta\chi(a)$. First, PD control input u_{PD} is calculated as

$$u_{PD} = -K_p J_q^T [\Delta x + \Delta l, 0]^T - K_D \dot{q} + g_q - J_q^T F_d, \quad (58)$$

where F_d is desired contact force and K_p , K_D are PD gain matrices. In order to realize the desired mode retainment, u_{PD} is verified by (55). If it is confirmed that u_{PD} maintains the

desired mode, $\mathbf{u}_{PD}$ is used as control input. If it is found that $\mathbf{u}_{PD}$ is not desirable, a searching algorithm for finding $\mathbf{u}$ is applied until a desirable control input is found. $\tau_1 - \tau_2$ space is discretized into small grids. The grid points are tested one by one using (55) until the desirable condition is satisfied. The total learning algorithm is described in Table 3.

1. Lower layer: Estimate mode switching boundary by sampling mode transition data off-line.
2. Environment: Initialize state.
3. Observe $X(t)$ and obtain discrete state s .
4. Upper layer: Action selection. Action is given to the lower layer as a subgoal ($\Delta\chi$).
5. Lower layer: Mode retainment/transition operation corresponding action a using mode boundary estimation:
  - a) Calculate PD control input $\mathbf{u}_{PD}$ using (58).
  - b) Judge whether desired mode retainment is possible using (55).
  - c) If it is possible to realize desired motion, use $\mathbf{u}_{PD}$ as control input. Otherwise, explore and find an appropriate input $\mathbf{u}$ using SVM.
  - d) Environment: State transition to $X(t + \Delta t)$
  - e) Repeat a)-d) until subtotal is reached. If mode retainment is failed or the hand went out of limitation, negative reward is given and state is initialized.
6. Environment: State transition and reward $r(s,a)$ is given to the upper layer.
7. Upper layer: Update $Q(s, a)$ using (53)
8. Repeat 3.-7. until goal is reached or a fixed number of steps are repeated.
9. Go to 2.

Table 3. Algorithm for hierarchical learning of stick/ slip switching motion control

5. Simulation results of Stick/Slip Switching Motion Learning

Physical parameters for simulation are set as followings:

- Lengths of links and sizes of the object: $l_1 = 1.0, l_2 = 1.0, a = 0.336[m]$ (Object is a square.)
- Masses of the links and the object: $m_1 = 1.0, m_2 = 1.0$ [kg]
- Time interval for one cycle of simulation and control: $\Delta t = 0.02[\text{sec}]$
- Coefficients of static (and kinetic) friction: $\mu_1 = 0.6, \mu_2 = 0.2$
- Joint angle limitation is set as $q_{1\min} = 0, q_{1\max} = 1.6[\text{rad}]$ (No limitation for q_2).
- Torque limitations are set as $\tau_{1\min} = -5, \tau_{1\max} = 20$ and $\tau_{2\min} = -20, \tau_{2\max} = 5$.

Initial states of the manipulator and the object are set as $[x_0, 1_0, \dot{x}_0, \dot{1}_0]^T = [1.440, 0.1090, 0, 0]^T$. Corresponding initial conditions for the manipulator are $[q_1, q_2, \dot{q}_1, \dot{q}_2]^T = [\pi/3, -\pi/2, 0, 0]^T$. Goal state is given as $[x_d, l_d, \dot{x}_d, \dot{1}_d]^T = [0.620, 0.3362, 0, 0]^T$ (as indicated in Fig.10)

Parameters for Q-learning algorithm are set as $\gamma = 0.95$, $\alpha = 0.5$ and $\epsilon = 0.1$. The state space is defined as $0.620 < x < 1.440$, $0 < l < 0.336(= a)$ and x and l axes are discretized into 6. Thus total number of discrete states is 36. There are four actions in the upper layer Q-learning, each corresponds to the transition to adjacent state in x, l space. Reward is defined as $r(s, a) = r^1(s, a) + r^2(s, a)$ and r^1 and r^2 are specified as followings. Let s_d denote the goal state in discrete state space and r^1 is given as

$$r^1(s, a) = \begin{cases} 10 & \text{if } s = s_d \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

r^2 is given as $r^2(s, a) = -1$ when constraints are broken or the hand moves out of the state space.

5.1 Mode boundary estimation by SVM

Before applying reinforcement learning, mode transition data are collected and used for mode boundary estimation by SVM. Data are sampled for grid points in X , by discretizing $[x, l, \dot{x}, \dot{l}, \tau_1, \tau_2]$ by $[5, 10, 10, 10, 10, 10]$. Two graphs in Fig. 13. show examples of mode boundary estimation. In the left hand, $x - \dot{x}$ plane is shown by fixing other variables as $l = 0.183$ and $\tau = [-1, 5]^T$ by setting $\dot{l} = 0$. The curve in the figure shows the region where mode 'stick' for contact point 1 and mode 'slip to negative direction of x-axis' for contact point 2 are maintained. In the left hand, $l - \dot{l}$ plane is shown by fixing other variables as $l = 0.966$ and $\tau = [5.5, 2.5]^T$ by setting $\dot{x} = 0$. The curve shows the region where mode 'slip to positive direction of x-axis' for contact point 1 and mode 'stick' for contact point 2 are maintained.

Fig. 13. Examples of estimated boundary by SVM

5.2 Learning of manipulation

The profile of reward per step (average) is shown in the left hand of Fig.14. Trajectories from initial configuration to the desired one were obtained after 200 trials. It takes value of around 6 or 7 because it is an average of one trial, in which reward of -1 is obtained at the beginning and later reward of 10 is obtained, as far as it stays at the desired configuration. The right hand of Fig.14 shows state value function $V(s)$, which is calculated from action

value function by $\max_a Q(s,a)$ (s_1 and s_2 correspond to discretization of l and x , respectively). It can be seen that the value of the desired state is the highest in the state space. 500 steps trials are tested for 20 times. For all cases, it was possible to achieve the control to the desired state, though numbers of trials required to achieve learning are different (around several hundred trials).

The left hand of Fig.15 shows a trajectory obtained by the hierarchical controller with the greedy policy. Totally five mode switching are operated to achieve desired configuration. The right hand of Fig.15 shows the profiles of joint torques. Continuous torques are calculated by the lower layer.

Fig. 14. Learning profile and obtained state value function

Fig. 15. Trajectory on l - x plane and joint torque profiles

Fig.16. shows contact modes δ for contact point 1 and 2. By comparing two figures, it can be seen that when $\delta_1 = 1$ (contact point 1 is slipping and the hand is moving to right), $\delta_2 = 0$ (contact point 2 is stick mode and the object is stopping) holds. On the contrary, when $\delta_1 = 0$ (contact point 1 is stick mode), $\delta_2 = -1$ (contact point 2 is slipping and the object is moving to left) holds. That is, the hand is moving together with the object. Thus, the manipulator is switching 'slipping the hand on the object to right' mode and 'moving the object to left' mode. Note that there are instances when both contact modes becomes stick, that is, $\delta_1 = \delta_2 = 0$. This is caused by the learning architecture, which requires stopping at the end of each action of Q-learning. If there is no stop for each action, the total motion

would be much more smooth and faster.

Fig. 16. Examples of estimated boundary by SVM

5.3 Discussion

The lower layer controller achieved local control of the manipulator using SVM boundary obtained off-line sampling. On-line data sampling and on-line boundary estimation of the mode boundaries will be one of our future works. On the other hand, there were some cases where the lower layer controller could not find appropriate torques to realize desired mode. Improvement of the lower layer controller will realize faster learning in the upper layer. One might think that it would be much easier to learn mode boundary in $F_{it} - F_{in}$ space using measurement of contact force F_i for contact point i , because the boundary can be expressed by simple linear relation in contact force space. There are two reasons for applying boundary estimation in the torque space: 1) In more general cases, it is not appropriate to assume that contact forces can be always measured. E.g., in whole body manipulation (Yoshida et al., 2006), it is difficult to measure contact force because contact can happen at any point on the arm. 2) From the viewpoint of developing learning ability, it is also an important learning problem to find an appropriate transformation of coordinate systems so that boundaries between modes can be simply expressed. This will be also one of our future works.

In order to extend the proposed framework to more useful applications such as multi-finger object manipulation, a higher-dimensional state space should be considered. If dimension of the state space is higher, the boundary estimation problem by SVM will require more computational load. The problem 2) mentioned above will be a key-technique to realize a compact and effective boundary estimation to the high-dimensional problems. The dimension of state space for the reinforcement learning should remain low enough so that the learning approach is applicable. Otherwise, other planning techniques might be better to be applied.

6. Conclusion

In this chapter, we proposed two reinforcement learning approaches for object contact robotic motion. The first approach realized a holonomic constrained motion control by making use of a function giving a map from the general motion space to the constrained lower dimensional one and the reward function approximation. This mapping can be regarded as giving function approximation for the extraction of nonlinear lower dimensional parameters. By comparing the proposed method with the ordinal reinforcement learning method, the superiority of the proposed learning method was confirmed. From a more general perspective, we are investigating multidimensional mapping for broader applications. In addition, it is important to consider the continuity of action (force control input) in the manipulation task.

In the second approach, a hierarchical approach of mode switching control learning was proposed. In the upper layer, reinforcement learning was applied for global motion planning. In the lower layer, SVM was applied to learn the boundaries between contact modes and utilized to generate control input which realized mode retainment control. In simulation, it was shown that an appropriate trajectory was obtained by reinforcement learning with mode switching of stick/slip. For further development, fast learning of mode boundaries will be required.

7. References

- Andrew G. Barto, Steven J. Bradke & Satinder P. Singh: Learning to Act using Real-Time Dynamic Programming, Artificial Intelligence, Special Volume: Computational Research on Interaction and Agency, 72, 1995, pp. 81-138. Gerald Farin: Curves and Surfaces for CAD, Morgan Kaufmann Publishers, 2001.
- Z. Gabor, Z. Kalmar, & C. Szesvari: Multi-criteria reinforcement learning, Proc. of the 15th Int. Conf. on Machine Learning, pp. 197-205, 1998.
- Peter Geibel: Reinforcement Learning with Bounded Risk, Proc. of 18th Int. Conf. on Machine Learning, pp. 162-169, 2001.
- H. Kimura, T. Yamashita and S. Kobayashi, Reinforcement Learning of Walking Behavior for a Four-Legged Robot, Proc. of IEEE Conf. on Decision and Control, 411-416, 2001.
- Cheng-Peng Kuan & Kuu-Young Young: Reinforcement Learning and Robust Control for Robot Compliance Tasks, Journal of Intelligent and Robotic Systems, 23, pp. 165-182, 1998.
- O. L. Mangasarian and David R. Musicant, Lagrangian Support Vector Machines, Journal of Machine Learning Research, 1, 161-177, 2001.
- H. Miyamoto, J. Morimoto, K. Doya and M. Kawato: Reinforcement learning with via-point representation, Neural Networks, 17, 3, 299-305, 2004.
- Saleem Mohideen & Vladimir Cherkassky, On recursive calculation of the generalized inverse of a matrix, ACM Transactions on Mathematical Software 17, Issue 1, pp. 130 - 147, 1991
- J. Morimoto and K. Doya, Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. Robotics and Autonomous Systems 36 (1): 37-51, 2001.

- R. Munos, A. Moore, Variable Resolution Discretization in Optimal Control, Machine Learning, No.1, pp.1-31,2001.
- J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, M. Kawato, Learning from demonstration and adaptation of biped locomotion. Robotics and Autonomous Systems 47(2-3): 79-91, 2004
- S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita and H. Hirukawa, Constraint-based Dynamics Simulator for Humanoid Robots with Shock Absorbing Mechanisms, The 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007
- A. van der Schaft & H. Schumacher: An Introduction to Hybrid Dynamical Systems, Springer, 2000.
- Richard S. Sutton: Dyna, an Integrated Architecture for Learning, Planning, and Reacting, Proc. of the 7th Int. Conf. on Machine Learning, pp. 216-224, 1991.
- Richard S. Sutton: Learning to Predict by the Methods of Temporal Differences, Machine Learning, 1988, 3, 9-44.
- T. Schlegl, M. Buss, and G. Schmidt, Hybrid Control of Multi-fingered Dextrous Robotic Hands, S. Engell. G. Frehse, E. Schnieder (Eds.): Modelling, Analysis and Design of Hybrid Systems, LNCIS 279, 437-465, 2002.
- V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, 1995. M. Yashima, Y. Shiina and H. Yamaguchi, Randomized Manipulation Planning for A Multi-Fingered Hand by Switching Contact Modes, Proc. 2003 IEEE Int. Conf. on Robotics and Automation, 2003.
- Y. Yin, S. Hosoe, and Z. Luo, A Mixed Logic Dynamical Modelling Formulation and Optimal Control of Intelligent Robots, Optimization Engineering, Vol.8, 321-340,2007.
- E. Yoshida, P. Blazevic, V. Hugel, K. Yokoi, and K. Harada, Pivoting a Large Object: Whole-body Manipulation by a Humanoid Robot, Applied Bionics and Biomechanics, vol. 3, no. 3, 227-235, 2006

Robotics 2010 Current and Future Challenges

Edited by Houssem Abdellatif

ISBN 978-953-7619-78-7

Hard cover, 494 pages

Publisher InTech

Published online 01, February, 2010

Published in print edition February, 2010

Without a doubt, robotics has made an incredible progress over the last decades. The vision of developing, designing and creating technical systems that help humans to achieve hard and complex tasks, has intelligently led to an incredible variety of solutions. There are barely technical fields that could exhibit more interdisciplinary interconnections like robotics. This fact is generated by highly complex challenges imposed by robotic systems, especially the requirement on intelligent and autonomous operation. This book tries to give an insight into the evolutionary process that takes place in robotics. It provides articles covering a wide range of this exciting area. The progress of technical challenges and concepts may illuminate the relationship between developments that seem to be completely different at first sight. The robotics remains an exciting scientific and engineering field. The community looks optimistically ahead and also looks forward for the future challenges and new development.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Yuichi Kobayashi and Shigeyuki Hosoe (2010). Reinforcement learning Approach to Objectcontact Motion with Estimation of Lowdimensional Submanifold and Mode-Boundary, Robotics 2010 Current and Future Challenges, Houssem Abdellatif (Ed.), ISBN: 978-953-7619-78-7, InTech, Available from: <http://www.intechopen.com/books/robotics-2010-current-and-future-challenges/reinforcement-learning-approach-to-objectcontact-motion-with-estimation-of-lowdimensional-submanifol>

INTECH
open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen