Berthouze, L., Prince, C. G., Littman, M., Kozima, H., and Balkenius, C. (2007).
Proceedings of the Seventh International Conference on Epigenetic Robotics: Modeling
Cognitive Development in Robotic Systems. *Lund University Cognitive Studies, 135*.

Planning-Space Shift Learning toward Flexible Hierarchy Generation

Yuichi Kobayashi[*,**]
[*]Tokyo University of Agriculture and Technology
Nakacho 2-24-16, Koganei-city
Tokyo 184-8588 Japan
yu-koba@cc.tuat.ac.jp

Shigeyuki Hosoe[**]
[**]RIKEN BMC Research Center
Shimoshidami 2271-130,
Moriyama, Nagoya 463-0003 Japan
hosoe@bmc.riken.jp

Abstract

To improve the flexibility of robotic learning, it is important to realize an ability to generate a hierarchical structure. This paper proposes a learning framework which can dynamically change the planning space depending on the structure of the task. Synchronous motion information is utilized to generate modes and different modes correspond to different hierarchical structure of the controller. This enables efficient task planning and control using low-dimensional space. An object manipulation task is tested as an application, where an object is found and used as a tool (or as a part of the body) to extend the ability of the robot. The proposed framework is expected to be a basic learning model to account for body image acquisition including tool affordances.

1. Introduction

Ability of robots is still insufficient to realize flexible and adaptive robot systems that can act in household environments. One possible approach to improve the ability of robot systems is to learn from development of humans. This approach, called developmental robotics, is recently gathering more attention (Lungarella et al., 2003).

Hierarchical learning is an important issue to improve learning ability. In some researches of hierarchical learning, navigation in large building like complex mazes (Dayan and Hinton, 1993) or controlling a robotic arm with multiple joints (Miyamoto et al., 2004) are investigated as examples of large and complex problems. But the applications of robotic system in household environments have some specific characteristics that can be also found in problems discussed in developmental cognitive researches. That is, robotic tasks involve *objects* that can vary depending on situations and the objects can play various roles; a target to carry, an obstacle to avoid collision, or a tool which can be used to achieve different objectives. The existence of objects makes the task diverse (various objectives and situations), complex (different dynamics depending on contact or non-contact) and high-dimensional (configuration space).

On the other hand, it is known that some behaviors of the robotic system can be realized by combination of multiple modules, each of which has relatively small dimension, though the DOF of the total system is very large. For example, subsumption architecture proposed by Brooks (Brooks, 1986) realized flexible and adaptive behaviors of locomotion robots with many DOFs, while individual module in the architecture played rather simple role such as collision avoidance or simple maneuver. From the viewpoint of building learning architecture, flexibility of such architecture is realized where just a part of the total system should be brought into focus and the part in focus should flexibly vary depending on the situation and what to be done.

In this paper, a learning architecture that can account for such *variable focus* in robotic motion learning *with objects* is proposed. A 'part in focus' is interpreted as a space for motion planning in this research. By changing or 'shifting' the planning space, the architecture can be applied to diversity of tasks. This diversity is different from the one discussed in (Wolpert and Kawato, 1998) with multiple modules learning architecture in a sense that the input and output can be variant, while the multiple-module learning model deals with identical input and output.

One possible application of planning-space shift learning is a problem of adaptive tool-use. An object can be utilized to extend reachable region of a robot when the robot can move it. That is, the robot can use the object as a tool. Planning is originally done in task space of a robot hand and later done in a space of end of a tool after acquiring how to move it. The ability of tool-use has gathered attention partly because it is deeply related to the issue of affordance (Gibson, 1977) and body schema. Stoychev

proposed a behavior-based approach to realize representation of tool use (Stoychev, 2005). Nabeshima *et al.*also proposed a learning framework for tool-use (Nabeshima et al., 2006). This paper aims to propose a general description of hierarchy generation, where the ability of tool-use is regarded as one possible application of the learning framework. Thus, the proposed architecture does not utilize any explicit representation of 'tool' and it uses just the information of motion synchronousness.

In section 2., problem settings for the proposed learning architecture is described. The learning architecture is proposed in section 3., followed by evaluations by simulation in section 4..

2. Problem Settings

In this section, problem settings including basic assumptions are described. For simplicity, it is assumed that all of the motions of robots and objects are quasi-static. Thus, velocity components can be omitted from state variables.

Let n be the degrees of freedom (DOF) of the robot hand and m the number of objects, where objects move when they contact with the robot hand or other objects. The state variables of the system are followings:

- Variables which express the configuration of the robot; $\boldsymbol{\theta} \in \mathbb{R}^n$
- Variables which express the configuration of the object; $\boldsymbol{q}_1 \in \mathbb{R}^{n_1}, \cdots, \boldsymbol{q}_m \in \mathbb{R}^{n_m}$

The objective of the robot task is to move an object to a certain desired configuration. Which object to move and the desired configuration of the object are various.

The robot can observe the following variables:

- Observation variables of robot hand
- Configuration variables of objects obtained by image inputs

These variables are not distinguished for the robot initially. Thus, they are both included in $\boldsymbol{y} \in \mathcal{Y} \subset \mathbb{R}^\ell$. The observation variables of the robot hand can be different from $\boldsymbol{\theta}$. E.g., position of the hand-tip in image coordinate can be a candidate for the observation variables. The configuration variables of objects equal to $[\boldsymbol{q}_1^T, \cdots, \boldsymbol{q}_m^T]^T$ for simplicity*. The i-th component of observation variable $\boldsymbol{y}$ is denoted by $y_i(t), i = 1, \cdots, \ell$. Note that though the values of the observation variable $\boldsymbol{y}$ are known, the relations among components of $\boldsymbol{y}$ are unknown for the robot. That is, the robot does not know even the physical meaning of $\boldsymbol{y}$ as well as the kinematic characteristics of robot hand and objects, such as link length, radius of circular objects, edge length of rectangular objects.

*In general, $\boldsymbol{y}$ may be more redundant visual information.

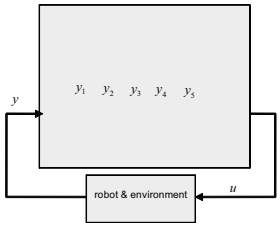

Figure 1: Proposed learning architecture

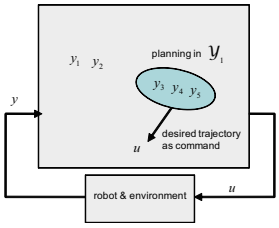

Figure 2: Mode one: getting mapping from $\mathcal{Y}_1$ to $\mathcal{U}$

At each time step, the robot can change its configuration variables. That is, $\boldsymbol{u} = \Delta\boldsymbol{\theta}$ is the control input to the system. When the robot moves its hand, the hand may contact with an object due to kinematic constraints between the hand and the objects or between objects. All constraints between the hand and objects and between objects are assumed to be holonomic.

3. Learning Structure

In this section, first an overview of the proposed learning structure is described. The following subsections describe each component of the learning structure.

3.1 Basic idea of planning-space shift

Fig.1 shows a schematic figure of a learning robot at the initial state without any knowledge on observation variables. In this example, $\ell = 5$ and the relations among $y_1, y_2, \cdots, y_5$ are unknown. Initial random movements of the robot body causes changes of these observation variables. By detecting synchronous motions (i.e., by finding that some of the observation variables change when the control input $\boldsymbol{u}$ changes), one can find a group of observation variables which are directly affected by the basic motion of the robot. In the example of Fig.2, y_3, y_4 and y_5 are in the same group of observation variables. Let $\mathcal{Y}_1 = \{[y_3, y_4, y_5]^T\}$ and $\mathcal{U} = \{\boldsymbol{u}\}$ in this example. By collecting the data of $\boldsymbol{u}$ and y_3, y_4, y_5 for some time period, one can obtain an inverse mapping from $\mathcal{Y}_1$ to $\mathcal{U}$†. Using this inverse mapping, it is possible to generate a control input $\boldsymbol{u}$ which realizes a desired trajectory specified in $\mathcal{Y}_1$ space.

After some exploration, it may happen that variables y_1 and y_2 also change with y_3, y_4 and y_5. This change of $[y_1, y_2]$ is regarded to be caused by the change of $[y_3, y_4, y_5]$, which is based on the idea that the motion of another object is caused by the motion of the robot hand through contact. By detecting the

†In general, it is not possible to acquire one-to-one correspondence when there exists redundancy in the relation. We omit such cases in this paper.

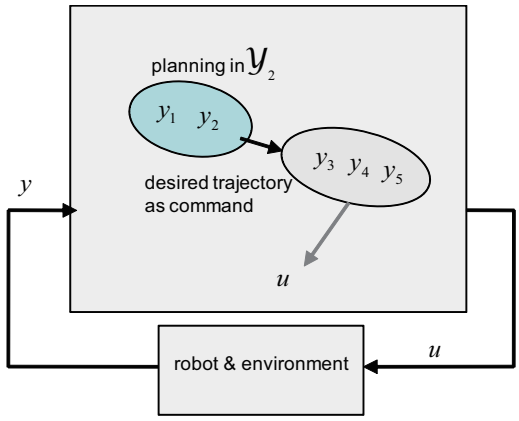

Figure 3: Mode two: getting mapping from $\mathcal{Y}_2$ to $\mathcal{Y}_1$

change of y_1 and y_2, the second group of variables in $\mathcal{Y}_2 = \{[y_1, y_2]^T\}$ is generated. This situation is distinguished as different 'mode', where the first basic motion is named mode one and the current motion of all components is named mode two. Similarly to the case of mode one, the inverse mapping from $\mathcal{Y}_2$ to $\mathcal{Y}_1$ is acquired by collecting data. Using the mapping, it is possible to generate a trajectory in $\mathcal{Y}_1$ when a desired trajectory in $\mathcal{Y}_2$ is given.

The above-mentioned mappings can be used when a task is given to the robot system. For instance, let a desired configuration for $[y_1, y_2]$ is given as a cross depicted in Fig.4. First, a total trajectory with mode transition is planned as shown as a succession of arrows in the right part of the figure. To realize the planned trajectory in mode one, the mapping from $\mathcal{Y}_1$ to $\mathcal{U}$ is utilized to control y_3, y_4, y_5 along the trajectory while retaining mode one. After transition to mode two, the mapping from $\mathcal{Y}_2$ to $\mathcal{Y}_1$ is used while retaining mode two. Note that in the case of mode one, the space for planning is $\mathcal{Y}_1$. Then later in mode two, the planning space shifts to $\mathcal{Y}_2$. This architecture brings two advantages:

- The dimension of the planning space can be smaller than the case where the total observation variable space $\mathcal{Y}$ is used as the planning space. Thus the computation amount for the planning can be reduced.
- The division of planning space is not pre-defined. The planning framework can be flexibly applied to various tasks with different objects and different number of objects.

The proposed learning structure has the following functional components:

1. Mode generation by motion synchronousness:
   First, modes are generated using information of synchronous motion. Observation variables are also grouped. Observation variables are stored for estimating the boundary between modes.

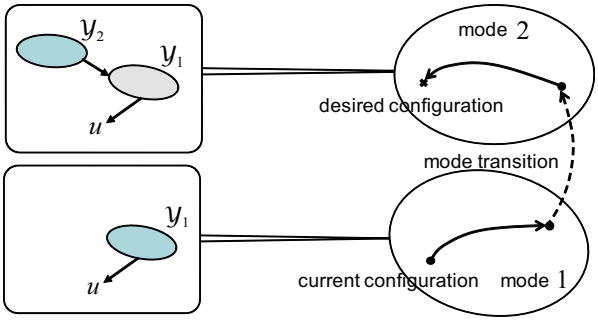

Figure 4: Planning-space shift among different modes

2. Acquisition of controller with mode retainment and mode transition:
   Within each mode, there can be two strategies for control; 1) to keep the same mode without any occurrence of mode transition, and 2) to make mode transition to a certain target mode. For both strategies, boundaries between modes (conditions for mode transitions) are needed to be estimated in addition to the mapping between groups of observation variables.
3. Planning via multiple modes:
   Once a task (a desired configuration of a certain object) is given to the robot, the total trajectory is planned as shown in of Fig.4. Parameterization of boundaries is necessary for this planning.

The following subsections give the details of the mode generation, the controller construction, parameterization of boundaries between modes and the planning via different modes.

3.2 Mode generation using motion synchronousness

Let $\mathcal{Y} \subset \mathbb{R}^n$ denote the set of observation variables. In order to detect synchronousness among observation variables, time differences of observation variables are defined as

$$\Delta y_i(t) = y_i(t) - y_i(t-1), \quad i = 1, \cdots, \ell \qquad (3..1)$$

Using $\Delta y_i(t)$, each component $c_i(t)$ of $\boldsymbol{c}(t)$ is defined as

$$c_i(t) = \begin{cases} 1 & \text{if} \quad \Delta y_i(t) \neq 0 \\ 0 & \text{if} \quad \Delta y_i(t) = 0 \end{cases}, \quad i = 1, \cdots, \ell \qquad (3..2)$$

This discrete-valued vector $\boldsymbol{c}(t) = [c_1(t), \cdots, c_\ell(t)]^T$ gives an identification of modes. For example, the initial mode described in 3.1 can be expressed by $\boldsymbol{c} = [0, 0, 1, 1, 1]^T$. Through exploration, new modes are found one by one. Mode information is stored as the list $\boldsymbol{c}^{(i)}, i = 1, 2, \cdots$. When a mode that has never been encountered is experienced at time t (and let i be the current number of modes), mode $i + 1$

is newly created and set as $\boldsymbol{c}^{(i+1)} = \boldsymbol{c}(t)$. In the case of the example in 3.1, the created modes are $\boldsymbol{c}^{(1)} = [0, 0, 1, 1, 1]^T$ and $\boldsymbol{c}^{(2)} = [1, 1, 1, 1, 1]^T$.

It is assumed that the number of variables with fluctuation (variables which satisfy $\Delta y_i(t) \neq 0$) increases when a new mode is created during on-line learning[‡]. The variables which begin to fluctuate by mode transition from mode i to mode j is defined as

$$\boldsymbol{y}^{(j,i)} = [y_{k_1}, y_{k_2}, \cdots, y_{k_{f(j,i)}}]^T,$$
$$c^{(i)}_{k_a} = 0, c^{(j)}_{k_a} = 1, \quad a = 1, 2, \cdots, f(j, i), \qquad (3..3)$$

where $f(j, i)$ denotes the number of observation variables that began to fluctuate by transition from mode i to mode j. The group of observation variables for mode j is defined as

$$\boldsymbol{y}^{(j)} = \boldsymbol{y}^{(j,i^*)}, \quad i^* = \arg\min_i f(j, i). \qquad (3..4)$$

Using this notation, newly-fluctuating observation parameter space $\mathcal{Y}_j$ is defined as

$$\mathcal{Y}_j = \{\boldsymbol{y}^{(j)} | \boldsymbol{y}^{(j)} \in \mathbb{R}^{f(j,i^*)}\}. \qquad (3..5)$$

Set of stored observation variables in i-th mode is denoted by $\mathcal{M}_i$ and $\mathcal{M}_{(i)}$. They are defined as

$$\mathcal{M}_i = \{\boldsymbol{y}(t) \subset \mathcal{Y} | \boldsymbol{c}(t) = \boldsymbol{c}^{(i)}\} \qquad (3..6)$$
$$\mathcal{M}_{(i)} = \{\boldsymbol{y}^{(i)}(t) \subset \mathcal{Y}_i | \boldsymbol{c}(t) = \boldsymbol{c}^{(i)}\}. \qquad (3..7)$$

The boundaries among different modes can be approximated by collecting observation variables when a mode changes to another mode. The database for boundary approximation is denoted by $B_{i,j}$ as

$$B_{i,j} = \{[\boldsymbol{y}_1^T, \boldsymbol{y}_2^T]^T | \boldsymbol{y}_1 = \boldsymbol{y}^{(i)}(t) \in \mathcal{M}_{(i)},$$
$$\boldsymbol{y}_2 = \boldsymbol{y}^{(j)}(t) \in \mathcal{M}_{(j)}, \boldsymbol{y}(t + \Delta t) \in \mathcal{M}_j\}, \qquad (3..8)$$

where Δt denotes sampling time for control and observation. $B_{i,j}$ is a set of observation variables where mode has changed from i to j.

To approximate the boundaries among modes, non-linear Support Vector Machine (SVM), which is known as a non-linear classifier with kernel functions, is used (Vapnik, 1995). For classification between mode i and j, vectors in $B_{i,j}$ and $B_{j,i}$ are used. Let m_s denote the total number of data and n_s denote the dimension of vectors ($[\boldsymbol{y}_1^T, \boldsymbol{y}_2^T]^T$ in (3..8)). Vectors in $B_{i,j}$ and $B_{j,i}$ are rearranged into data matrix $\boldsymbol{a}_i \in \mathbb{R}^{n_s}, i = 1, \cdots, m_s$. $\boldsymbol{d} \in \mathbb{R}^{m_s}$ is a vector with plus or minus ones, where plus and minus correspond respectively to modes i and j. Hereafter, let $\boldsymbol{x} = [\boldsymbol{y}_1^T, \boldsymbol{y}_2^T]^T$. In non-linear SVM with Gaussian kernel, by introducing kernel function K as

$$K(\boldsymbol{x}, \boldsymbol{a}_i) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{a}_i\|^2}{\sigma^2}\right), \qquad (3..9)$$

[‡]This assumption corresponds to the fact that all of objects (including the robot hand) currently moving keep the motion when another object begins to move.

where σ denotes a width parameter for the Gaussian kernel, separation surface between two classes is expressed as

$$\sum_{i=1}^{m_s} d_i w_i K(\boldsymbol{x}, \boldsymbol{a}_i) = 0, \qquad (3..10)$$

where $\boldsymbol{w}$ is a solution of the following optimization problem:

$$\min_{\boldsymbol{w}} \left\{\frac{1}{2}\boldsymbol{w}^T Q \boldsymbol{w} - \boldsymbol{e}^T \boldsymbol{w}\right\}, \qquad (3..11)$$

where Q is given by

$$Q = \frac{1}{\nu} + HH^T, \quad H = D[A - \boldsymbol{e}], \nu > 0 \qquad (3..12)$$

and $\boldsymbol{e} \in \mathbb{R}^{n_s}$ denote the vector of ones. $D = \mathrm{diag}[d_1, \cdots, d_{m_s}]$, $A = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_{m_s}]^T$ and ν is a parameter for the optimization problem . For implementation of optimization in (3..11), Langrangian SVM is applied (Mangasarian and Musicant, 2001).

The boundary information obtained by SVM is used for 1) construction of mode retainment (and mode transition) controller and 2) planning on submanifolds through parameterization of the boundary surface.

3.3 Mode-transition and mode-retainment controller

In each mode, motion information is stored through random exploration. The stored data is used to construct an inverse mapping (motion transformation) between two groups of observation variables. Fig.5 shows an example of motion transformation between variable set $\mathcal{Y}_i$ and $\mathcal{Y}_j$, denoted by $G_{i,j}$. Motion transformation $G_{i,j}$ receives a vector in $\mathcal{Y}_j$ as input and gives a vector in $\mathcal{Y}_i$ as output. As a result it can generate a trajectory (how to move) in $\mathcal{Y}_i$ so as to realize a given input trajectory in $\mathcal{Y}_j$. One way to realize such a transformation is to apply a non-linear function approximator such as a feed-forward neural network (hereafter denoted by NN)(Rumelhart et al., 1986). Let $\tilde{f}_{i,j}(\boldsymbol{y}^{(j)})$ denote the output of NN (after training) for input $\boldsymbol{y}^{(j)}$. When desired displacement $\Delta\boldsymbol{y}^{(j)}$ is given at time t, displacement in $\mathcal{Y}_i$ can be calculated by

$$\Delta\boldsymbol{y}^{(i)} = \tilde{f}_{i,j}(\boldsymbol{y}^{(j)}(t) + \Delta\boldsymbol{y}^{(j)}) - \tilde{f}_{i,j}(\boldsymbol{y}^{(j)}(t)) \quad (3..13)$$

using the outputs of NN.

To construct a controller which keeps the same mode, it is necessary to explicitly avoid mode transition in addition to using the above-mentioned inverse-mapping. Mode retainment controller can be designed using boundary information obtained by SVM. Recall that the discrimination function between two modes are given by $F(\boldsymbol{x}) > 0$ and

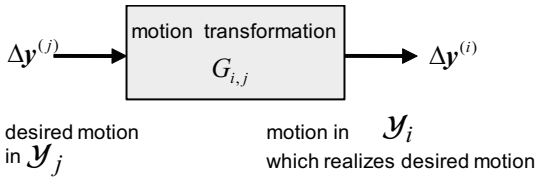

Figure 5: Motion transformation between $\mathcal{Y}_i$ and $\mathcal{Y}_j$

$F(\boldsymbol{x}) < 0$ where

$$F(\boldsymbol{x}) = \sum_{i=1}^{m_s} d_i w_i K(\boldsymbol{x}, \boldsymbol{a}_i). \quad (3..14)$$

By introducing a potential function $\Phi(\boldsymbol{x})$ as

$$\Phi(\boldsymbol{x}) = \|F(\boldsymbol{x})\|^2, \quad (3..15)$$

the boundary can be expressed as $\{\boldsymbol{x}|\Phi(\boldsymbol{x}) = 0\}$.

Thus, a policy to explore inside a mode can be expressed as

$$\Delta \boldsymbol{y}^{(i)} = \begin{cases} -\eta \dfrac{\partial \Phi}{\partial \boldsymbol{y}^{(i)}}, & \text{if } \Phi(\boldsymbol{x}) < \varepsilon \\ \text{rand}, & \text{otherwise} \end{cases}, \quad (3..16)$$

where η is a coefficient to determine the displacement in $\mathcal{Y}_i$ and ε denotes a threshold value to judge that it is close enough to the boundary of mode transition. 'rand' denotes a random vector generator.

The controller for mode transition is described in the next sub-section after introducing parameterization along mode boundaries.

3.4 Parameterization of mode boundary

As described in the example in 3.1, it is necessary to parameterize boundaries between modes to identify where to make transition on the boundary for the total planning accompanied with mode transitions. In this paper, a method for one dimensional parameterization using nodes is introduced (here assuming for simplicity that boundaries are one-dimensional manifolds). Now let $\boldsymbol{q}_i \in \mathbb{R}^N$ denote a point on the boundary which we call hereafter i-th node and L denote the number of nodes. A curve from node i to node $i+1$, which is defined as i-th segment of the manifold, is defined using following equation (Fig.6).

$$\boldsymbol{x}^{(i)}(p') = \boldsymbol{q}_i + \boldsymbol{v}_{1i} p' + \boldsymbol{v}_{2i} p'^2, \quad (3..17)$$

where $\boldsymbol{v}_{1i}, \boldsymbol{v}_{2i} \in \mathbb{R}^N$ are coefficient vectors that give the shape of i-th segment of the curve. This curve is designed so that changing parameter p' from zero to one corresponds to the change on the curve from $\boldsymbol{q}_i$ to $\boldsymbol{q}_{i+1}$. By differentiating (3..17) by p',

$$\frac{\partial \boldsymbol{x}^{(i)}}{\partial p'}(p') = \boldsymbol{v}_{1i} + 2\boldsymbol{v}_{2i} p' \quad (3..18)$$

is obtained. This gives a tangential vector of the curve. By considering the continuity of positions of the curve at the both edge of the curve segment, that is, by letting $\boldsymbol{x}^{(i)}(1) = \boldsymbol{x}^{(i+1)}(0)$,

$$\boldsymbol{q}_i + \boldsymbol{v}_{1i} + \boldsymbol{v}_{2i} = \boldsymbol{q}_{i+1}, \quad i = 0, \cdots, L-1 \quad (3..19)$$

is obtained as a condition for continuity. Similarly, the condition for continuity of tangential vector is given by $\frac{\partial \boldsymbol{x}^{(i)}}{\partial p'}(1) = \frac{\partial \boldsymbol{x}^{(i+1)}}{\partial p'}(0)$, that is,

$$\boldsymbol{v}_{1i} + 2\boldsymbol{v}_{2i} = \boldsymbol{v}_{1i+1}, \quad i = 0, \cdots, L-2. \quad (3..20)$$

The above-mentioned conditions can be expressed in matrix form as

$$AV = Q, \quad A \equiv \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, Q \equiv \begin{bmatrix} Q_1 \\ O_{(L-1)N \times 1} \end{bmatrix}, \quad (3..21)$$

where matrices are defined as

$$A_1 = \begin{bmatrix} I_N & I_N & & & O \\ & & I_N & I_N & \\ & & & \ddots & \\ O & & & & I_N \ I_N \end{bmatrix} \in \mathbb{R}^{LN \times 2LN}, \quad (3..22)$$

$$A_2 = \begin{bmatrix} I_N & 2I_N & -I_N & & & O \\ & & I_N & 2I_N & -I_N & \\ & & & \ddots & & \\ O & & & & I_N \ 2I_N & -I_N \ O_N \end{bmatrix} \in \mathbb{R}^{(L-1)N \times 2LN} \quad (3..23)$$

and

$$V = \begin{bmatrix} \boldsymbol{v}_{10} \\ \boldsymbol{v}_{20} \\ \vdots \\ \boldsymbol{v}_{1L-1} \\ \boldsymbol{v}_{2L-1} \end{bmatrix} \in \mathbb{R}^{2LN}, \ Q_1 = \begin{bmatrix} \boldsymbol{q}_1 - \boldsymbol{q}_0 \\ \vdots \\ \boldsymbol{q}_L - \boldsymbol{q}_{L-1} \end{bmatrix} \in \mathbb{R}^{LN}. \quad (3..24)$$

When positions of nodes are given, V can be calculated using pseudo-inverse as the minimum-norm solution by

$$V = A^T (AA^T)^{-1} Q \equiv A^\dagger Q. \quad (3..25)$$

By connecting all segments, a one-dimensional submanifold $\boldsymbol{x}(s)$ can be defined as follows (see Fig.7):

$$\boldsymbol{x}(s) \equiv \boldsymbol{x}^{(i)}(p'), \quad i \le s < i+1, p' = s - i, i \in \mathbb{Z} \quad (3..26)$$

The positions of nodes are modified so that the curve $\boldsymbol{x}(s)$ lie along a boundary. It is assumed that the boundary is smooth. This is realized by an incremental procedure to reduce energy defined on the curve, where energy function of the curve is defined with coefficient α_{ext} and α_{int} as

$$E = \alpha_{\text{ext}} E_{\text{ext}} + \alpha_{\text{int}} E_{\text{int}}. \quad (3..27)$$

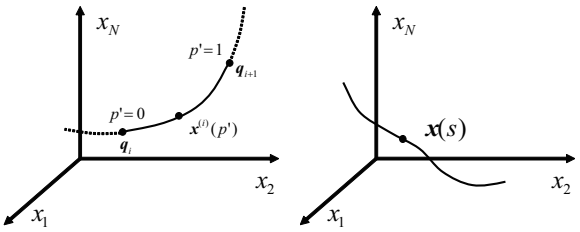

Figure 6: A curve model constructed by nodes

Figure 7: A curve parameterized by s

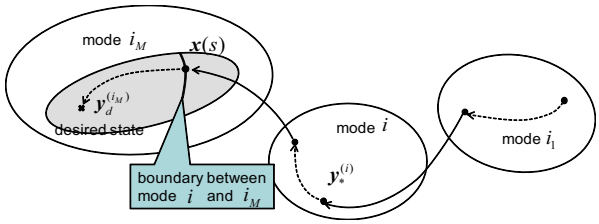

Figure 8: Planning with mode transition

An internal energy for the curve is defined as

$$E_{\text{int}} = \int_0^n \left\| \frac{\partial^2 \boldsymbol{x}(s)}{\partial s^2} \right\|^2 ds. \qquad (3..28)$$

Using (3..17), E_{int} can be expressed as

$$E_{\text{int}} = \sum_{i=0}^{n-1} \int_0^1 \left\| \frac{\partial^2 \boldsymbol{x}^{(i)}}{\partial p'^2} \right\|^2 dp' = 4 \sum_{i=0}^{n-1} \|\boldsymbol{v}_{i2}\|^2. \quad (3..29)$$

E_{ext} is defined so that the curve coincides with the boundary when E_{ext} is minimized. By simply using Φ, it can be defined as $E_{\text{ext}} = \Phi(\boldsymbol{x})$. By iterating the update

$$\boldsymbol{q}_i \leftarrow \eta_n \frac{\partial E}{\partial \boldsymbol{q}_i}, \quad i = 1, \cdots, L \qquad (3..30)$$

the positions of nodes $1, \cdots, L$ are modified so that the curve coincides the boundary while keeping smoothness. After convergence of the iteration of (3..30), any point on the boundary can be expressed by parameter s.

Using the parameterization, a policy to make mode transition at a boundary $\boldsymbol{x}(s^*)$ is given by

$$\Delta \boldsymbol{y}^{(i)} = \begin{cases} -\eta_1 \dfrac{\partial \Phi}{\partial \boldsymbol{y}^{(i)}}, & \text{if } \Phi(\boldsymbol{x}) < \varepsilon \\ \eta_2 (\boldsymbol{x}(s^*) - \Delta \boldsymbol{y}^{(i)}(t)), & \text{otherwise} \end{cases}, \qquad (3..31)$$

where s^* is given by the planner for the total trajectory. The planning method for the total trajectory is described in the next subsection.

3.5 Planning with mode transition

Planning with mode transition described in this section corresponds to the dynamical shift of planning space, because different observation variables are used in each planning within each mode. Let a desired configuration of an object is given to the robot system[§]. First, a group of observation variables that corresponds to the motion of the object is specified as $\boldsymbol{y}^{(i)}$. Then the desired configuration is interpreted as $\boldsymbol{y}_d^{(i)}$. Next, mode transition from the initial mode to the mode where $\boldsymbol{y}^{(i)}$ changes is determined using a database which is constructed by memorizing sequences of mode transitions. Let $[i_1, \cdots, i_M]$ denote a sequence of mode transitions.

The algorithm for planning the total trajectory is described in Alg. I.

Alg. I Motion Planning with Mode Transitions

1. Set $C(i) = 0$, $^\forall i$
2. Set $i = i_M$ and $\boldsymbol{y}_*^{(i)} = \boldsymbol{y}_d^{(i)}$
3. Select a point on the boundary between mode i and mode $i-1$ randomly as $\boldsymbol{x}(s)$
   (a) If $\boldsymbol{y}_*^{(i)}$ is reachable from $\boldsymbol{x}(s)$: If $i = i_1$ planning is finished. Otherwise, set $i \leftarrow i-1, \boldsymbol{y}_*^{(i)} \leftarrow \boldsymbol{x}(s)$ and go to 3.
   (b) Else $C(i) \leftarrow C(i) + 1$
      i. If $C(i) > C_{\text{th}}$, set $i \leftarrow i+1$ [a] and go to 3.
      ii. Else go to 3.

[a] if $i = i_M$, i is not changed.

4. Simulation

The simulation was implemented by MATLAB. The robot manipulator has two joints and there are two objects, rectangular and circular objects as shown in Fig.9. The end effector of the robot hand is also a circle. The state variables are the joint angles of the manipulator $\boldsymbol{\theta} \in \mathbb{R}^2$ and positions of two objects $q_1, q_2 \in \mathbb{R}$. The observation variables are the position of the robot hand (center of circle) y_1, y_2, the position of the rectangular object $y_3 (= q_1)$ and the position of the circular object $y_4 (= q_2)$.

First, the robot acquires the relation between its joint angles and the hand position by randomly changing the joint angles. The neural network introduced in 3.3 is implemented to approximate the inverse mapping. The first group of observation variables is built as $[y_1, y_2]^T \in \mathcal{Y}_1$. When the hand contacts the rectangular object, the second group of observation variable is constructed as $y_3 \in \mathcal{Y}_2$. While the hand keeps contact with the rectangular object,

[§] Here it is assumed that configurations of other objects are not specified as a task.

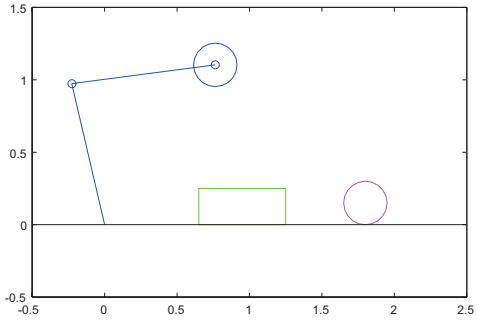

Figure 9: Simulation settings: robot hand and objects

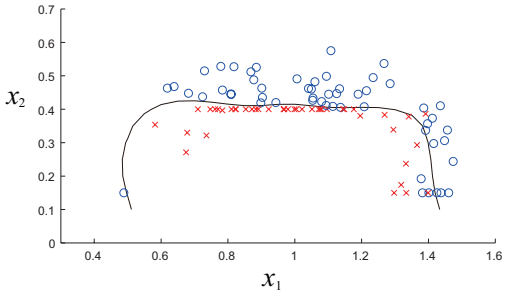

Figure 10: Estimated boundary by SVM

the object moves together with the hand. Then the third group is constructed as $y_4 \in \mathcal{Y}_3$.

While collecting mapping data by the random motion, the robot also collects boundary information between mode one and mode two. In this case discrimination function $F(\boldsymbol{x})$ of SVM is defined in three-dimensional space, that is, $\boldsymbol{x} = [y_1, y_2, y_3]^T$. Fig.10 shows a boundary of two modes where q_1 is fixed as $y_3 = 0.65$. The boundary in the figure is drawn by using contour plot function of MATLAB after 3000 steps of random motion. The collected data for boundary estimation is plotted for $0.6 < y_3 < 0.7$. Circles in the figure denote variables in mode one and crosses denote mode two.

The discrimination function $F(\boldsymbol{x})$ is utilized for parameterization on the boundary which is explained in 3.4. Fig.11 is a resultant node distribution with $L = 9$ after 100 steps iteration of node movements, where nodes are initially located randomly around the center of the figure. It can be seen that nodes are distributed along the contour of $F(\boldsymbol{x}) = 0$.

A task is given to the robot system to move the circular object from an initial position to $q_2 = 2.2$. Alg. I described in 3.5 is utilized for the planning with mode transitions from mode one to mode three via mode two. Fig.12 shows the trajectory realized by the learning architecture. First the hand moves toward a point on the parameterized boundary between mode one and two using the inverse mapping of NN, where the point on the boundary is decided by the planning. After contacting the object, the hand

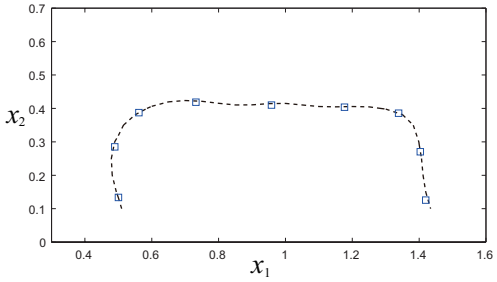

Figure 11: Parameterization on boundary with nodes

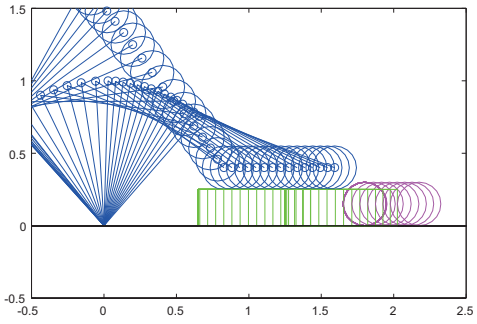

Figure 12: Trajectory of robot hand and objects

keeps contact with the object while moving rightward. By letting the rectangular object touch the circular object, the task is accomplished.

5. Discussion

From the viewpoint of developmental robotics, the proposed learning architecture has following suggestions to related researches:

- Motor babbling(Demiris and Dearden, 2005) has been discussed as a way of learning motor coordination. While the first stage of learning of mapping in the proposed framework is composed of random actions and can be equivalent to inverse-kinematics learning(Oyama et al., 2001), the total framework deals with wider problem of finding relations among various kinds of representations.
- The proposed architecture can be regarded as an construction method of body image (e.g., (Yoshikawa et al., 2003)), where 'self' is identified through the criterion of 'what can be controlled' and 'what cannot be controlled'.
- The proposed learning architecture includes a criterion for autonomous exploration, which is based on the structure of the task. Exploration in the proposed learning method is done while retaining a mode or aiming at different modes, which helps building an exploration strategy.

On the other hand, the applications presented in the simulation is rather simple and easy. The pro-

posed architecture is applicable to more complex and high-dimensional problems, but to apply the framework to higher-dimensional problems, it is required to consider parameterization of high-dimensional sub-manifolds (boundaries between modes). Other possible extensions are followings:

- The influence of noise should be taken into account in order to implement the proposed framework in the real world. One possible way to deal with noise in motion synchronousness detection is to approximate the effect of noise by preceding measurement.
- Observation variables directly correspond to configuration variables in this paper. Extraction of observation variables from various image feature vectors should be taken into consideration.
- The proposed framework does not include integration of multi-modal sensor information. E.g., tactile information is of great importance to generate the representation of the object (Metta and Fitzpatrick, 2003). Generation of modes can be done more efficiently with integration of other sensor information.
- The framework of hybrid dynamical system control (van der Schaft and Schumacher, 2000) is closely related to the control with mode switchings. Besides, a control strategy that utilizes a partial information of state variable is known as 'backstepping' (Khalil, 2001).

6. Conclusion

In this paper, an approach to hierarchy generation is proposed as planning-space shift learning. The learning framework consists of generation of modes based on motion synchronousness, estimation of boundaries between modes, and planning via multiple modes. In simulation, an illustrative example was shown where an object is utilized as a tool to move another object. The consideration of the proposed architecture even with simple examples will be an important base for constructing more complex representations of hierarchical learning.

References

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:253–262.

Dayan, P. and Hinton, G. E. (1993). Feudal reinforcement learning. *Advances in Neural Information Processing Systems*, (5):271–378.

Demiris, Y. and Dearden, A. (2005). From motor babbling to hierarchical learning by imitation: a robot developmental pathway. In *Proc. of the Fifth Int. Workshop on Epigenetic Robotics*, pages 31–37.

Gibson, J. J. (1977). The theory of affordances. In *Perceiving, Acting, and Knowing: Toward Ecological Psychology*, pages 62–82.

Khalil, H. K., (Ed.) (2001). *Nonlinear Systems*. Prentice Hall.

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connection Science*, 15(4):151–190.

Mangasarian, O. L. and Musicant, D. R. (2001). Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177.

Metta, G. and Fitzpatrick, P. (2003). Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128.

Miyamoto, H., Morimoto, J., Doya, K., and Kawato, M. (2004). Reinforcement learning with via-point representation. *Neural Networks*, 17:299–305.

Nabeshima, C., Kuniyoshi, Y., and Lungarella, M. (2006). Adaptive body schema for robotic tool use. *Advanced Robotics*, 20(10):1105–1126.

Oyama, E., Agah, A., MacDorman, K. F., Maeda, T., and Tachi, S. (2001). A modular neural network architecture for inverse kinematics model learning. *Neurocomputing*, 38-40:797–805.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel Data Processing*, 1:318–362.

Stoychev, A. (2005). Behavior-grounded representation of tool affordances. In *Proc. of IEEE Int. Conf. on Robotics and Automation*.

van der Schaft, A. and Schumacher, H. (2000). *An Introduction to Hybrid Dynamical Systems*. Springer.

Vapnik, V. N. (1995). *The Nature of Statictical Learning Theory*. Springer.

Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329.

Yoshikawa, Y., Hosoda, K., and Asada, M. (2003). Does the invariance in multi-modalities represent the body scheme? - a case study with vision and proprioception -. In *Proc. of the 2nd Int. Symp. on Adaptive Motion of Animals and Machines*, volume SaP-II-1.