The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Planning-Space Shift Learning: Variable-space Motion Planning toward Flexible Extension of Body Schema

Yuichi Kobayashi and Shigeyuki Hosoe

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 tasks. 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.**

I. Introduction

Hierarchical learning is an important issue to improve learning ability of robots. In some researches of hierarchical learning, navigation in large building like complex mazes [1], [2] or controlling a robotic arm with multiple joints [3] are investigated as examples of large and complex problems. Apart from those high-dimensionalities, one aspect of the complexity in robot tasks is that it involves *objects* whose shapes can vary depending on situations and the objects can play various roles; a target to be carried, an obstacle to avoid collision, or a tool which can be used to achieve different objectives. The existence of the objects makes tasks diverse (various objectives and situations), complex (different dynamics depending on contact or non-contact) and high-dimensional (configuration space).

As an approach to the complex control problems, 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, even though the DOF of the total system is very large. For example, subsumption architecture [4] realized flexible and adaptive behaviors of locomotion robots with high DOFs, while individual module in the architecture played rather simple role such as collision avoidance or simple maneuver. Such architectures realize flexibility by focusing on just a part of the total system, where the part in focus flexibly varies depending on the situations and objectives.

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. In the proposed architecture, variables to be controlled can be variant, differing from the multiple-module learning model [5] where the state variables and control variables have to be invariant.

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. The ability of tool-use has gathered attention partly because it is deeply related to the issue of affordance [6] and body schema, that are frequently discussed in the field of developmental robotics [7]. Stoychev proposed a behavior-based approach to realize representation of tool use [8]. Nabeshima *et al.* also proposed a learning framework for tool-use [9]. This paper aims to propose a more general and more 'bottom-up' description of hierarchy generation. Thus, the proposed architecture does not utilize any explicit representation of 'tool' and it uses just the information of motion synchronousness.

In II, problem settings for the proposed learning architecture is described. The learning architecture is proposed in III, followed by evaluations by simulation in IV.

II. Problem Description

For simplicity, it is assumed that all motions of the robot and the objects are quasi-static.[1)] The objects are assumed to be polyhedral or spherical rigid bodies. Let n be the number of joints of the robot arm and m the number of the objects, which move only through contacting with the robot hand or other objects, *i.e.*, they do not move by themselves.[2)] The state variables of the system are the followings:

- Variables which express the configuration of the robot; $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^n$
- 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 should be moved and its desired configuration are given on-line. The agent can observe the following variables in addition to configuration of the robot $\boldsymbol{\theta}$:

- Position of the robot hand (end effector) $\boldsymbol{h}(\boldsymbol{\theta})$
- Configuration variables of the objects measured from image inputs to the robot

These variables are presented to the agent as the observation vector $\boldsymbol{y} \in \mathcal{Y} \subset \mathbb{R}^\ell$. Initially, these variables are not distin-

Y. Kobayashi is with Tokyo University of Agriculture and Technology, 2-24-16 Koganei Tokyo, Japan, yu-koba@cc.tuat.ac.jp

S. Hosoe is with RIKEN-TRI Collaboration Center for Human-Interactive Robot Research, Nagoya, Japan, hosoe@nagoya.riken.jp

1)Thus, velocity components can be omitted from state variables.

2)It is assumed that the robot contacts with objects only at its hand.

guished by the agent and the number of the object or DOFs of motions of them are also unknown. The configuration variables of objects are denoted by $[\boldsymbol{q}_1^T, \cdots, \boldsymbol{q}_m^T]^T$. The i-th component of observation variable $\boldsymbol{y}$ is denoted by $y_i(t), i = 1, \cdots, \ell$[3]. The agent does not know the correspondence between the elements of $\boldsymbol{y}$ and the observed position of the robot hand, nor the functional relations among the elements of $\boldsymbol{y}$ (including the kinematics of the arm).

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. The followings are assumptions on the tasks:

- Contact of a new object with moving ones and the hand happens only to the object which has begun moving last. It is also assumed that only one object will start moving at a time. Therefore the moving objects and the hand form a chain.
- The movement of an object is affected only by the object which is positioned closer to the robot hand and therefore the direction of the effect of object motions is unidirectional.
- The relationships between any two moving (and contacting) objects (hand) are described by mappings among the configuration variables of them without redundancy. The mappings do not depend on the contact positions on the objects.

III. Learning Structure

The learning structure proposed in this paper is constituted by the four basic functional components. An overview will first be presented in the following subsection and following to this the detail of the algorithm will be described.

A. Basic idea of planning-space shift

The agent repeats trial motions while resetting the configurations of the robot arm and the objects (the reset is done once a certain steps of the motion). The agent initially yields random action controls and observes $\boldsymbol{y}$ and collects them with control $\boldsymbol{u}$. An example with $\ell = 5$ is shown in Fig.1 and Fig.2. Initial random movements of the robot body causes changes of the corresponding observation variables. By detecting synchronous motions through the observation of changing variables in $\boldsymbol{y}$, one can find a group of observation variables which are directly affected by the motion of the robot itself (these correspond to the configuration variable of the robot hand). In this example, y_3, y_4 and y_5 are grouped as a set (Fig.1). Let $\mathcal{Y}_1 \subset \mathbb{R}^3$ (corresponding to $\{y_3, y_4, y_5\}$). By collecting the data of $\boldsymbol{\theta}$ and y_3, y_4, y_5 for some time period, one can obtain a functional relationship between $\boldsymbol{\theta}$ and y_3, y_4, y_5 represented as a mapping from Θ to $\mathcal{Y}_1$ and the inverse mapping from $\mathcal{Y}_1$ to Θ.

After some exploration, it may happen that variables y_1 and y_2 also change. This change of $\{y_1, y_2\}$ is regarded to be caused by the change of $\{y_3, y_4, y_5\}$. This observation corresponds to the fact that an object (corresponding to $\{y_1, y_2\}$) starts moving by contacting with the moving robot hand (corresponding to $\{y_3, y_4, y_5\}$). Similarly to the above, the second group of variables $\mathcal{Y}_2 \subset \mathbb{R}^2$ (corresponding to $\{y_1, y_2\}$)is introduced. The case where only variables y_3, y_4, y_5 change and the case where $y_1, \cdots, y_5$ change are called mode 1 and mode 2, respectively. Similarly to the case of mode 1, the inverse mapping from $\mathcal{Y}_2$ to $\mathcal{Y}_1$ is acquired. Using the mapping, it is possible to generate a trajectory in $\mathcal{Y}_1$ when a desired trajectory in $\mathcal{Y}_2$ is given.

[3]In this paper, i-th component of vector $\boldsymbol{x}$ is denoted by x_i.

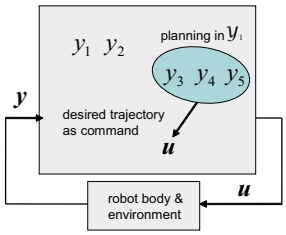


Fig. 1. Mode 1: generation of command $\boldsymbol{u}$ from $\mathcal{Y}_1$

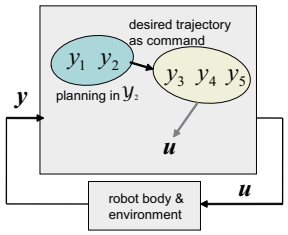


Fig. 2. Mode 2: generation of trajectory in $\mathcal{Y}_1$ from one in $\mathcal{Y}_2$

The constructed 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.3. First, a total trajectory with mode transitions (more strictly, a succession of points on boundaries) is planned as shown as a succession of arrows in the right part of the figure. Up to point p_{12} indicated in Fig.3, the trajectory has to be retained in mode 1, *i.e.*, only variables y_3, y_4, y_5 can change but y_1, y_2 must be constant. For this, we can use the constructed mapping from $\mathcal{Y}_1$ to $\mathcal{U}$. After transition to mode 2, the mapping from $\mathcal{Y}_2$ to $\mathcal{Y}_1$ is used while maintaining mode 2. Note that in the case of mode 1, the space for planning is $\mathcal{Y}_1$. Then later in mode 2, the planning space shifts to $\mathcal{Y}_2$.

From the perspective of hierarchical structure, control variables in $\mathcal{U}$ 'obeys' the trajectory defined by observation variables in $\mathcal{Y}_1$ in mode 1 (Fig.1). Next in mode 2, the trajectory in $\mathcal{Y}_1$ is dominated by the trajectory defined in $\mathcal{Y}_2$ (Fig.2). Thus, different relations of dependence among groups of variables can be seen in each mode, where some groups of variables in the upper layer dominates the behavior of other groups of variables in the lower layer. The relation between two groups of variables can be regarded as hierarchy. In this sense, the proposed architecture can be regarded to generate hierarchy autonomously.

This architecture brings two advantages:

- The motion-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 exploration space is divided into lower dimensional spaces ($\mathcal{Y}_1$ and $\mathcal{Y}_2$ in the present example) and this makes the exploration more efficient.

The proposed learning structure has the following functional components:

1) Mode generation by motion synchronousness:
   Modes are generated by grouping observation variables based on motion synchronousness. Observation vari-

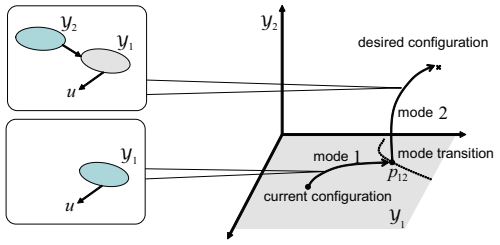


Fig. 3. Planning-space shift among different modes

ables are stored for estimating the boundary between modes.

2) Acquisition of motion mappings with consideration of mode keeping and mode transition:
Mapping from a group of observation variables to another group is estimated. This mapping is used to generate motion in a mode when a desired motion in another mode is given. 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.
3) Exploration and estimation of reachable region:
Using the motion mappings mentioned above, each mode is explored. During exploration, observation variables are stored to estimate reachable regions.
4) Planning and control via multiple modes:
Once a desired configuration of a certain object is given to the robot, the total trajectory is planned by finding via points on the boundaries among modes (p_{12} in Fig.3). Parameterization of boundaries is used for this.

These processes can be conducted simultaneously, but we execute and evaluate each process one by one, for simplicity. The following subsections give the details of the components.

B. Mode generation using motion synchronousness

Recall that vector $\boldsymbol{y} = [y_1, \cdots, y_\ell]^T \in \mathcal{Y} \subset \mathbb{R}^\ell$ is the observation vector. $\boldsymbol{y}(t)$ denotes the observed value of $\boldsymbol{y}$ at time t. In order to detect synchronousness among observation variables, time differences of observation variables are defined as

$$\Delta y_i(t) = y_i(t) - y_i(t-1), \quad i = 1, \cdots, \ell. \tag{III.1}$$

With $\Delta y_i(t)$, its indicator set I_C is defined as

$$I_C(t) = \{i_1, i_2, \cdots, i_k | \Delta y_{i_j}(t) \neq 0, j = 1, \cdots, k, \\ \Delta y_{i_j}(t) = 0, \text{otherwise}\}. \tag{III.2}$$

Hereafter, the terminology 'mode' will be used to denote different indicator sets. When a mode that has never been encountered is experienced at time t and let k be the current number of modes, mode $k+1$ is newly created and correspondingly the indicator set is represented as $I_C^{(k+1)} = I_C(t)$.

Suppose mode transition from i to j, which means the set of moving objects has changed. This include two cases; 1) an object that has been isolated begins to move by contacting with the moving objects, and 2) a part of a group of objects that have been moving is separated and stops moving. In general, any one of the group of moving objects can contact with an isolated object and any part of the group of moving objects can be separated. Based on the assumption described in II, however, connection of the objects is generated in a chain-form (*i.e.* not in a tree-form) and contact or separation happens only at the tip (object) of the chain. Thus, at mode transition from i to j, either of $I_C^{(i)} \subset I_C^{(j)}$ or $I_C^{(j)} \subset I_C^{(i)}$ holds. The former corresponds to the beginning of motion of an isolated object and the latter corresponds to the separation of the object at the tip of the group of moving objects.

Now let k be the current number of the group of observation variables, that is, $k-1$ objects have been found excluding the hand. We consider the former case of $I_C^{(i)} \subset I_C^{(j)}$, the case where a new object has been found at transition from mode i to j. The observation variables which correspond to an object that begins to move is defined as

$$\boldsymbol{y}^{(k+1)} = [y_{i_1}, y_{i_2}, \cdots, y_{i_{f(k+1)}}]^T, \ i_a \in I_C^{(j)} - I_C^{(i)}, \tag{III.3}$$

where $f(k+1)$ denotes the number of observation variables that correspond to $(k+1)$-th object. $\mathcal{Y}_{k+1}$ is defined as

$$\mathcal{Y}_{k+1} = \{\boldsymbol{y}^{(k+1)} | \boldsymbol{y}^{(k+1)} \in \mathbb{R}^{f(k+1)}\}. \tag{III.4}$$

Note that the distinction between 'what can be controlled' and 'what cannot be controlled' is obtained through the grouping of the observation variables. The indicator sets of $I_C^1, \cdots, I_C^{k+1}$ give the observation variables that can be controlled and the remaining indicators denote that those observation variables are impossible to control so far. The sense of 'agency' is expressed as controllable variables of $\mathcal{Y}_1, \cdots, \mathcal{Y}_{k+1}$ in this framework.

Suppose that the current mode is i and k-th group of observation variables is moving at the tip of the chain. Set of stored observation variables for k-th group is defined as

$$\mathcal{O}_{(k)} = \{\boldsymbol{y}^{(k)}(t) \subset \mathcal{Y}_k |^{\forall} t \ \text{ s.t. } \ I_C(t) = I_C^{(i)}\}. \tag{III.5}$$

The boundaries among different modes can be estimated by collected observation variables at the moment when a mode changes to another mode. When mode changes from i to j (or j to i) by connection of k-th object with ℓ-th object (or by separation of ℓ-th object from k-th object) [4], which means $I_C^{(i)} \subset I_C^{(j)}$, the database for boundary approximation is denoted by $B_{i,j}$ as

$$B_{i,j} = \{[\boldsymbol{y}_1^T, \boldsymbol{y}_2^T]^T | \boldsymbol{y}_1 \in \mathcal{O}_{(k)}, \boldsymbol{y}_2 \in \mathcal{O}_{(\ell)}\}. \tag{III.6}$$

$B_{i,j}$ is a set of observation variables between mode i and j. A transition mode set $\mathcal{M}_i$ is defined as the set of modes to which mode transition is possible from mode i. It is expressed by $B_{i,j}$ as $\mathcal{M}_i = \{j | B_{i,j} \neq \emptyset\}$.

C. Estimation of mode boundaries using SVM

To get boundary equations between modes, non-linear Support Vector Machine (SVM), which is known as a non-linear classifier with kernel functions [10], is used. For classification between mode i and j, vectors in $B_{i,j}$ are used.

[4] Here note that 'object' includes the robot hand.

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 (III.6)). Vectors in $B_{i,j}$ are rearranged into data vector $\boldsymbol{a}_k \in \mathbb{R}^{n_s}, k = 1, \cdots, m_s$. $\boldsymbol{d} \in \mathbb{R}^{m_s}$ is a vector with plus or minus ones, where plus and minus of d_k correspond respectively to modes i and j of $\boldsymbol{a}_k$, where $i < j$. 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

$$K(\boldsymbol{x}, \boldsymbol{a}_k) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{a}_k\|^2}{\sigma^2}\right), \quad \text{(III.7)}$$

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

$$\sum_{k=1}^{m_s} d_k w_k K(\boldsymbol{x}, \boldsymbol{a}_k) = 0, \quad \text{(III.8)}$$

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

$$\min_{\boldsymbol{w}} \left\{ \frac{1}{2}\boldsymbol{w}^T Q \boldsymbol{w} - \boldsymbol{e}^T \boldsymbol{w} \right\}, \ \boldsymbol{e} = [1, \cdots, 1]^T \in \mathbb{R}^{n_s}, \quad \text{(III.9)}$$

where Q is given by

$$Q = \frac{1}{\nu} + HH^T, \quad H = D[A - \boldsymbol{e}], \nu > 0. \quad \text{(III.10)}$$

$D = \mathrm{diag}[d_1, \cdots, d_{m_s}]$, $A = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_{m_s}]^T$ and ν is a parameter for the optimization problem . For implementation of optimization in (III.9), Lagrangian SVM is applied [11].

The boundary information obtained by SVM is used for 1) generation of mode keeping (and mode transition) motion mappings and 2) planning using via points on the boundaries with parameterization.

D. Mode-transition and mode-keeping motion mapping

This section describes how to generate motion in $\mathcal{Y}_i$ when a desired motion in $\mathcal{Y}_j$ is given, supposing that there exists transition of modes from i to j. This motion mapping component described in this section is used in control described in III-G.

For mode j, motion information is collected through random exploration and stored, where a motion in $\mathcal{Y}_j$ is caused by a motion in $\mathcal{Y}_i$. The stored data is used to construct an inverse mapping (regarded as motion transformation) between groups of observation variables in $\mathcal{Y}_i$ and $\mathcal{Y}_j$. Fig.4 shows an example of motion transformation between groups of variables $\mathcal{Y}_i$ and $\mathcal{Y}_j$, denoted by $G_{i,j}$. Motion transformation $G_{i,j}$ receives a vector in $\mathcal{Y}_j$ as an input and gives a vector in $\mathcal{Y}_i$ as an output. As a result it can generate a trajectory 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 the feed-forward neural network (hereafter denoted by NN)[12]. 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

$$\Delta\boldsymbol{y}^{(i)} = \tilde{f}_{i,j}(\boldsymbol{y}^{(j)}(t) + \Delta\boldsymbol{y}^{(j)}) - \tilde{f}_{i,j}(\boldsymbol{y}^{(j)}(t)) \quad \text{(III.11)}$$

using the outputs of NN.

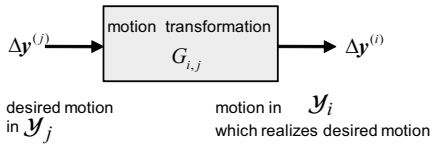


Fig. 4. Motion transformation between $\mathcal{Y}_i$ and $\mathcal{Y}_j$

To generate a motion which keeps the same mode, it is necessary to explicitly avoid the mode transition in addition to using the above-mentioned inverse-mapping. Here we consider that the current mode is j and suppose that it is required to avoid mode transition from j to k, where $k \in \mathcal{M}_j$. Mode keeping motion generator can be designed using boundary information of SVM. Recall that the discrimination between two modes is given by $F(\boldsymbol{x}) \lessgtr 0$where

$$F(\boldsymbol{x}) = \sum_{k=1}^{m_s} d_k w_k K(\boldsymbol{x}, \boldsymbol{a}_k). \quad \text{(III.12)}$$

By introducing a potential function as $\Phi(\boldsymbol{x}) = \{F(\boldsymbol{x})\}^2$, an algorithm to derive a motion in $\mathcal{Y}_i$ to explore in mode j randomly is shown in Algorithm I.

Algorithm I Motion generation with mode keeping

1) Set $found = false$
2) While $found = false$ repeat:
   - Set $\Delta\boldsymbol{y}_{\text{tmp}}^{(j)} = \text{rand}$
   - If $F([\boldsymbol{y}^{(j)T} + \Delta\boldsymbol{y}_{\text{tmp}}^{(j)T}, \boldsymbol{y}^{(k)}]^T) > \varepsilon, \quad k \in \mathcal{M}_j$, then $found = true$
   - Else repeat for n_o times ($n_o > 0$):
     a) Set $\Delta\boldsymbol{y}_{\text{tmp}}^{(j)} = \Delta\boldsymbol{y}_{\text{tmp}}^{(j)} - \eta\dfrac{\partial\Phi}{\partial\boldsymbol{y}^{(j)}}$
     b) If $F([\boldsymbol{y}^{(j)T} + \Delta\boldsymbol{y}_{\text{tmp}}^{(j)T}, \boldsymbol{y}^{(k)}]^T) > \varepsilon, \quad k \in \mathcal{M}_j$, then $found = true$ and break
3) Output $\Delta\boldsymbol{y}^{(i)}$ with $\Delta\boldsymbol{y}^{(j)} = \Delta\boldsymbol{y}_{\text{tmp}}^{(j)}$ using (III.11)

Here, η is a coefficient to determine the displacement in $\mathcal{Y}_j$ 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. Before deciding $\Delta\boldsymbol{y}^{(j)}$, it is checked whether current mode j is maintained using discrimination function F. If it is judged that the mode might change, the small displacement $\Delta\boldsymbol{y}^{(j)}$ is modified to be further from the boundary of the modes using gradient of potential function Φ.

E. Parameterization of mode boundary

As described in III-A, 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. Let us consider parameterization of boundary between mode i and mode. The parameterization is done for each discretized $\boldsymbol{y}^{(j)}$ as shown in Fig.5. In this paper, a method for one dimensional parameterization using nodes is introduced (here assuming for simplicity that boundaries

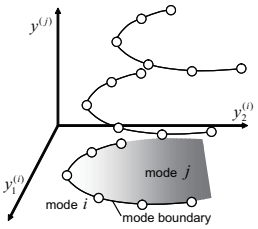


Fig. 5. Parameterization of boundary by nodes

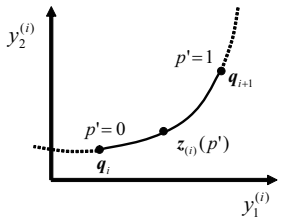


Fig. 6. A curve model constructed by nodes

are one-dimensional manifolds in two-dimensional space [5]). Thus, a curve for boundary estimation is generated in $\mathcal{Y}_i$ space (here we assume that $\mathcal{Y}_i \subset \mathbb{R}^2$) by nodes.

Now let $\boldsymbol{q}_k \in \mathbb{R}^2$ denote a point on the boundary between mode i and mode j which we call hereafter k-th node and L denote the number of nodes. A curve from k-th node to $(k+1)$-th node, which is defined as k-th segment of the curve, is defined using following equation as shown in Fig.6.

$$\boldsymbol{z}^{(k)}(p') = \boldsymbol{q}_k + \boldsymbol{v}_{1k}p' + \boldsymbol{v}_{2k}p'^2, \tag{III.13}$$

where $\boldsymbol{v}_{1k}, \boldsymbol{v}_{2k} \in \mathbb{R}^2$ are coefficient vectors that give the shape of k-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 $\boldsymbol{q}_k$ to $\boldsymbol{q}_{k+1}$. By differentiating (III.13) by p', $\frac{\partial \boldsymbol{z}^{(k)}}{\partial p'}(p') = \boldsymbol{v}_{1k} + 2\boldsymbol{v}_{2k}p'$ 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 $\boldsymbol{z}^{(k)}(1) = \boldsymbol{z}^{(k+1)}(0)$,

$$\boldsymbol{q}_k + \boldsymbol{v}_{1k} + \boldsymbol{v}_{2k} = \boldsymbol{q}_{k+1}, \quad k = 0, \cdots, L-1 \tag{III.14}$$

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

$$\boldsymbol{v}_{1k} + 2\boldsymbol{v}_{2k} = \boldsymbol{v}_{1k+1}, \quad k = 0, \cdots, L-2. \tag{III.15}$$

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

$$AV = Q, \quad A \equiv \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, Q \equiv \begin{bmatrix} Q_1 \\ O_{2(L-1)\times 1} \end{bmatrix}, \tag{III.16}$$

where matrices are defined as

$$A_1 = \begin{bmatrix} I_2 & I_2 & & & O \\ & I_2 & I_2 & & \\ & & \ddots & & \\ O & & & I_2 & I_2 \end{bmatrix} \in \mathbb{R}^{2L\times 4L}, \tag{III.17}$$

$$A_2 = \begin{bmatrix} I_2 & 2I_2 & -I_2 & & & O \\ & I_2 & 2I_2 & -I_2 & & \\ & & \ddots & & & \\ O & & & I_2 & 2I_2 & -I_2 \; O_2 \end{bmatrix} \in \mathbb{R}^{2(L-1)\times 4L} \tag{III.18}$$

[5] One dimensional boundary corresponds to the case with objects and hand expressed in 2-D space.

and

$$V = \begin{bmatrix} \boldsymbol{v}_{10} \\ \boldsymbol{v}_{20} \\ \vdots \\ \boldsymbol{v}_{1L-1} \\ \boldsymbol{v}_{2L-1} \end{bmatrix} \in \mathbb{R}^{4L}, \; Q_1 = \begin{bmatrix} \boldsymbol{q}_1 - \boldsymbol{q}_0 \\ \vdots \\ \boldsymbol{q}_L - \boldsymbol{q}_{L-1} \end{bmatrix} \in \mathbb{R}^{2L}. \tag{III.19}$$

When positions of nodes are given, V can be calculated using pseudo-inverse as the minimum-norm solution by $V = A^T(AA^T)^{-1}Q \equiv A^\dagger Q$. By connecting all segments, a one-dimensional submanifold $\boldsymbol{z}(s)$ can be defined as follows:

$$\boldsymbol{z}(s) \equiv \boldsymbol{z}^{(k)}(p'), \quad i \le s < k+1, p' = s-k, k \in \mathbb{Z} \tag{III.20}$$

The positions of nodes are modified so that the curve $\boldsymbol{z}(s)$ lie close to the 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

$$E = \alpha_{\text{ext}}E_{\text{ext}} + \alpha_{\text{int}}E_{\text{int}}. \tag{III.21}$$

An internal energy for the curve is defined as

$$E_{\text{int}} = \int_0^L \left\| \frac{\partial^2 \boldsymbol{z}(s)}{\partial s^2} \right\|^2 ds. \tag{III.22}$$

Using (III.13), E_{int} can be expressed as

$$E_{\text{int}} = \sum_{k=1}^{L} \int_0^1 \left\| \frac{\partial^2 \boldsymbol{z}^{(k)}}{\partial p'^2} \right\|^2 dp' = 4\sum_{k=1}^{L} \|\boldsymbol{v}_{k2}\|^2. \tag{III.23}$$

E_{ext} is defined so that the curve coincides with the boundary when E_{ext} is minimized. By using the potential function Φ defined between mode i and j, it can be defined as

$$E_{\text{ext}} = \sum_{k=1}^{L} \Phi([\boldsymbol{q}_k^T, \boldsymbol{y}_d^{(j)T}]^T), \tag{III.24}$$

where $\boldsymbol{y}_d^{(j)}$ denotes discretized value of $\boldsymbol{y}^{(j)}$. [6] By iterating gradient descent update

$$\boldsymbol{q}_k \leftarrow \boldsymbol{q}_k - \eta_n \frac{\partial E}{\partial \boldsymbol{q}_k}, \quad k = 1, \cdots, L \tag{III.25}$$

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 (III.25), any point on the boundary can be expressed by parameter s. In the following, parameterization of boundary between mode i and j is referred as $\boldsymbol{z}_{i,j}(s)$.

F. Exploration and estimation of reachable region

Observation variables are stored during exploration to estimate reachable region for motion planning. The reachable region is identified by the parameters on mode transition boundaries and the observation variables. For memorization of reachable region (and utilization of it for total trajectory

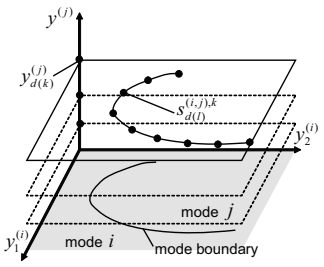


Fig. 7. Discretization of mode boundary and observationvariable

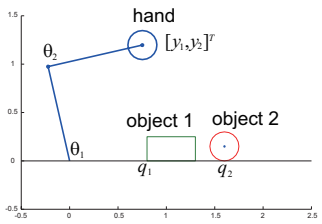


Fig. 8. Simulation settings: robot hand and objects

planning), parameter for boundary and configuration of object are discretized as indicated as small circles in Fig.7.

Discretized values of observation variables for mode j is defined as $\mathcal{Y}_D^{(j)} = \{\boldsymbol{y}_{d(1)}^{(j)}, \cdots, \boldsymbol{y}_{d(n(j))}^{(j)}\}$, where $n(j)$ denotes number of discretization of the observation variable. For identification of boundary parameter, discretized values of boundary parameter (between mode i and j) is defined as $\mathcal{S}_D^{(i,j),k} = \{s_{d(1)}^{(i,j),k}, \cdots, s_{d(n(i,j))}^{(i,j),k}\}$,where $s_{d(\ell)}^{(i,j),k}$ denotes ℓ-th discretized parameter on boundary of mode transition from i to j with k-th discretized observation variable $\boldsymbol{y}_{d(k)}^{(j)}$ and $n(i,j)$ denotes the number of discretization.

Let i_k denote current mode and assume that mode i_k has been reached by mode transitions $i_1 \rightarrow i_2 \rightarrow \cdots \rightarrow i_k$. Mode transition sequence is expressed by $\boldsymbol{\tau} = [i_1, \cdots, i_k]^T \in \mathbb{N}^k$. Let $\mathcal{T}$ denote set of mode transition sequences and $\mathcal{T}$ is updated while exploration as

$$\mathcal{T} \leftarrow \mathcal{T} \cup \{\boldsymbol{\tau}(t)\} \quad \text{if} \quad \boldsymbol{\tau}(t) \notin \mathcal{T} \wedge \left|I_C^{(i)}\right| < \left|I_C^{(j)}\right|, \quad i < j, \tag{III.26}$$

where $\boldsymbol{\tau}(t)$ denotes mode transition sequence at current time t. The second condition stands for that new mode sequence is memorized only when the mode transition keeps increasing number of changing observation variables. By this, mode transitions with separation between objects (or an object and the hand) are omitted. Using discretizations defined above, identification of mode transition for mode transition parameter $s^{(i,j)}$ and observation variable $\boldsymbol{y}^j$ can be given as $\boldsymbol{g}_{\boldsymbol{\tau}} = [\boldsymbol{g}_{\boldsymbol{\tau}}^{s\,T}, \boldsymbol{g}_{\boldsymbol{\tau}}^{y\,T}]^T$, where

$$\boldsymbol{g}_{\boldsymbol{\tau}}^{y} = [\ell_y^{(i_2)}, \cdots, \ell_y^{(i_k)}]^T, \ell_y^{(j)} = \arg\min_k \|\boldsymbol{y}_{d(k)}^{(j)} - \boldsymbol{y}^{(j)}\| \tag{III.27}$$

$$\boldsymbol{g}_{\boldsymbol{\tau}}^{s} = [\ell_s^{(i_1,i_2)}, \cdots, \ell_s^{(i_{k-1},i_k)}]^T,$$
$$\ell_s^{(i,j)} = \arg\min_\ell \|s_{d(\ell)}^{(i,j),k} - s^{(i,j),k}\|, k = \ell_y^{(j)}. \tag{III.28}$$

Let $\mathcal{D}_{\boldsymbol{\tau}}^{(i)}(\boldsymbol{g}_{\boldsymbol{\tau}})$ denote set of observation variables in mode i with mode transition specified by $\boldsymbol{\tau}$ and $\boldsymbol{g}_{\boldsymbol{\tau}}$. When $\boldsymbol{y}^{(i)}(t)$ is observed during exploration, $\mathcal{D}_{\boldsymbol{\tau}}^{(i)}(\boldsymbol{g}_{\boldsymbol{\tau}})$ is updated as

$$\mathcal{D}_{\boldsymbol{\tau}}^{(i)}(\boldsymbol{g}_{\boldsymbol{\tau}}) \leftarrow \mathcal{D}_{\boldsymbol{\tau}}^{(i)}(\boldsymbol{g}_{\boldsymbol{\tau}}) \cup \{\boldsymbol{y}^{(i)}(t)\}$$
$$\text{if} \quad {}^\forall\boldsymbol{y}_k^{(i)} \in \mathcal{D}_{\boldsymbol{\tau}}^{(i)}(\boldsymbol{g}_{\boldsymbol{\tau}}) \text{ s.t. } \|\boldsymbol{y}^{(i)} - \boldsymbol{y}_k^{(i)}\| > R_r, \tag{III.29}$$

[6]The discretization intervals are set equal in the simulation.

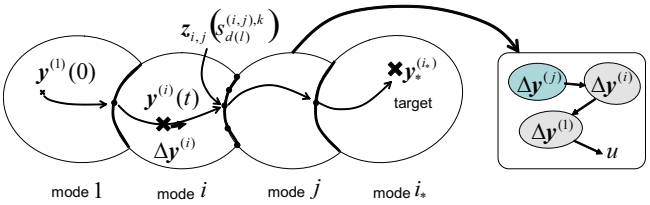
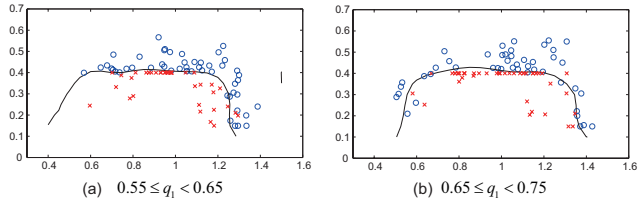


Fig. 9. Planning with mode transition



(a) $0.55 \le q_1 < 0.65$ (b) $0.65 \le q_1 < 0.75$

Fig. 10. Estimated boundaries between mode 1 and 2

where $R_r > 0$ denotes a threshold value and $\boldsymbol{y}_k^{(i)}$ denotes k-th element in $\mathcal{D}_{\boldsymbol{\tau}}^{(i)}(\boldsymbol{g}_{\boldsymbol{\tau}})$. Reachable region in mode i via mode transition $\boldsymbol{\tau}$ and $\boldsymbol{g}_{\boldsymbol{\tau}}$ can be expressed as

$$\mathcal{R}_{\boldsymbol{\tau}}^{(i)}(\boldsymbol{g}_{\boldsymbol{\tau}}) =$$
$$\{\boldsymbol{y}^{(i)} \in \mathcal{Y}_i | {}^\exists\boldsymbol{y}_k^{(i)} \in \mathcal{D}_{\boldsymbol{\tau}}^{(i)}(\boldsymbol{g}_{\boldsymbol{\tau}}), \|\boldsymbol{y}^{(i)} - \boldsymbol{y}_k^{(i)}\| \le R_r\}, \tag{III.30}$$

which will be used in motion planning described in III-G.

G. Trajectory generation and control with mode transitions

Planning with mode transitions described in this section corresponds to the dynamical shift of planning space, because different observation variables are used in planning within each mode. The total algorithm of trajectory generation and control is given in Algorithm II. A task is given to the robot system as a target configuration of a certain object[7]. The robot interprets the given information as target mode and target observation variables in the target mode (1). Using mode transition database and reachable region database, the robot finds an appropriate transition sequence of modes and pair of mode transition parameters and observation variable at mode transition (2,3). At step 4, the robot starts control of its body to realize mode transition sequence of $\boldsymbol{\tau}^\dagger$ with transition parameter $\boldsymbol{g}_{\boldsymbol{\tau}^\dagger}^\dagger$. At each mode, the robot aims at a target (subgoal) point on the mode transition boundary which is specified by discretized parameter of boundary curve (b). Once a subgoal is obtained, the robot repeats small movements toward the subgoal while trying to keep current mode. For mode keeping, Algorithm I is utilized by replacing random value to decide $\Delta\boldsymbol{y}_{\text{tmp}}^{(i)}$ by $\Delta\boldsymbol{y}^{(i)}$ given in (c)-i. The desired motion $\Delta\boldsymbol{y}^{(i)}$ is realized by transformations of motions as shown in Fig.9.

In the case when impossible tasks are given to the robot, the procedures of 1) or 3) can not be achieved.

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

Algorithm II Trajectory generation and control

1) Given: target mode i_* and configuration $\boldsymbol{y}_*^{(i_*)}$
2) Find $\boldsymbol{\tau}^\dagger \in \mathcal{T}$ s.t. $(^\exists k \in \mathbb{N}$ s.t. $(i_* = \tau_k^\dagger \wedge k \leq k_1, {}^\forall k_1$ s.t. $i_* = \tau_{k_1}^\dagger))$
3) Find $\boldsymbol{g}_{\boldsymbol{\tau}^\dagger}^\dagger$ s.t. $\boldsymbol{y}_*^{(i_*)} \in \mathcal{R}_{\boldsymbol{\tau}^\dagger}^{(i_*)}(\boldsymbol{g}_{\boldsymbol{\tau}^\dagger}) \wedge \boldsymbol{g}_{\boldsymbol{\tau}^\dagger}^{\dagger y} = [\ell_{y0}^{(i_2)}, \cdots, \ell_{y0}^{(i_k)}]^T$,
where $\ell_{y0}^{(i)} = \arg\min_\ell \|\boldsymbol{y}_{d(\ell)}^{(i)} - \boldsymbol{y}^{(i)}(0)\|$
4) Set current mode as $i = 1$ and repeat following trajectory generation and control procedures until target configuration is achieved, i.e. $\boldsymbol{y}^{(i_*)}(t) = \boldsymbol{y}_*^{(i_*)}$ holds:
   a) If $i \neq i_*$, find next mode as $j = \tau_\iota^\dagger$ where $\iota = \kappa + 1, \tau_\kappa^\dagger = i$
   b) If $i = i_*$, set target observation variable as $\boldsymbol{y}_\dagger^{(i)} = \boldsymbol{y}_*^{(i_*)}$, else set target point on mode boundary by $\boldsymbol{y}_\dagger^{(i)} = \boldsymbol{z}_{i,j}\left(s_{d(\ell)}^{(i,j),k^\dagger}\right)$, where $\ell = g_{\boldsymbol{\tau}^\dagger i}^s, k^\dagger = \arg\min_k \|\boldsymbol{y}_{d(k)}^{(j)} - \boldsymbol{y}^{(j)}(t)\|$
   c) Repeat small movement of $\Delta\boldsymbol{y}^{(1)}$ until target point is reached, i.e., $\boldsymbol{y}^{(i)} = \boldsymbol{y}_\dagger^{(i)}$
      i) Calculate displacement using $\Delta\boldsymbol{y}^{(i)} = \beta\frac{(\boldsymbol{y}_\dagger^{(i)} - \boldsymbol{y}^{(i)}(t))}{\|\boldsymbol{y}_\dagger^{(i)} - \boldsymbol{y}^{(i)}(t))\|}$ and mode keeping
      ii) Calculate $\Delta\boldsymbol{y}^{(\iota_1)}$ using $\Delta\boldsymbol{y}^{(i)}$, $\Delta\boldsymbol{y}^{(\iota_2)}$ using $\Delta\boldsymbol{y}^{(\iota_1)}$, $\cdots$, $\Delta\boldsymbol{y}^{(1)}$ using $\Delta\boldsymbol{y}^{(\iota_m)}$ with mode keeping, where $\boldsymbol{\tau}^\dagger = [1, \iota_m, \cdots, \iota_2, \iota_1, i, j, \cdots]^T$
   d) Set $i \leftarrow j$

IV. Simulation

The proposed mode generation and trajectory generation algorithms are implemented and evaluated in simulation. The robot manipulator has two joints and there are two objects, rectangular and circular objects as shown in Fig.8. 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) $[y_1, y_2]^T$, the position of the rectangular object (object 1) $y_3 (= q_1)$ and the position of the circular object (object 2) $y_4 (= q_2)$. Initial position of object 1 is randomly chosen in $0.6 \leq q_1 \leq 0.9$, while position of object 2 is fixed at $q_2 = 1.6$. The manipulator and the objects are initialized every 100 steps.

Exploration and learning are done by the followings:

1) Move randomly for 3000 steps.
2) Build mapping between $[\theta_1, \theta_2]^T$ and $[y_1, y_2]^T$, estimate boundary between mode 1 and mode 2, [8] and generate parameterization on the boundary.
3) Explore mode 2 using the motion generator which keeps mode 2 for 200 steps.
4) Build mapping between q_1 and q_2, estimate boundary between mode 2 and mode 3 (object 2 moving).
5) Explore mode 2 and 3 using mode keeping motion generator for 200 steps to estimate reachable region.

[8] They correspond respectively to the movement of the robot hand and the movement of the rectangular object by contacting with the robot hand.

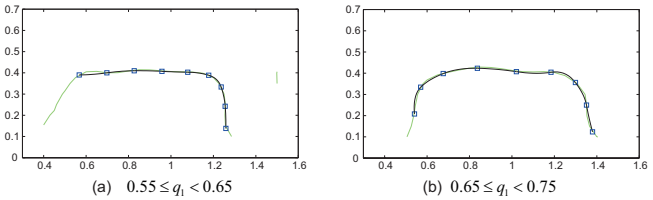


Fig. 11. Parameterization on boundary with nodes

It is expected that while procedure 1, contact between the hand and object 1 happens frequently enough to estimate boundary. Similarly it is expected that sufficient motion information of object 2 is obtainable to finally estimate the reachable region. The mapping and boundary building in 4. is formally done but actually not so important because motions of object 1 and 2 are restricted on a line and the boundary of mode switching is just a point.

After the total exploration, three tasks are given to the robot system:

- Task 1: move the end effector to $[y_1, y_2] = [1.3, 0.6]$.
- Task 2: move object 1 to $q_1 = 0.5$.
- Task 3: move object 2 to $q_2 = 1.9$.

First, the robot acquires the relation between its joint angles and the hand position by randomly changing the joint angles (procedure 1). The first group of observation variables is built as $[y_1, y_2]^T \in \mathcal{Y}_1$. When the hand contacts object 1, the second group of observation variable is constructed as $y_3 \in \mathcal{Y}_2$. While the hand keeps contact with the object 1, the object moves together with the hand. When moving object 1 contacts with object 2, object 2 begins to move and 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 1 and mode 2. 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(a) and (b) show boundaries of two modes where $0.55 \leq q_1 < 0.65$ for (a) and $0.65 \leq q_1 < 0.75$ for (b). Circles in the figure denote variables in mode 1, crosses denote mode 2 and curves denote boundaries of $F(\boldsymbol{x}) = 0$.

The discrimination function $F(\boldsymbol{x})$ is utilized for parameterization on the boundary explained in III-E. Fig.11(a) and (b) are resultant node distributions with $L = 9$ after 100 iterations of node movements, where nodes are initially located around the center of the boundary. It can be seen that nodes are distributed along the contour of $F(\boldsymbol{x}) = 0$. The both end nodes are constrained to the nearest observed data, which can be understood by comparing Fig.10 with Fig.11.

Fig.12 shows the trajectory of task 1, where a cross in the figure denotes the target configuration. This motion was realized by using the mapping between $\mathcal{Y}_1$ and $\mathcal{Y}_2$, which can be regarded as inverse kinematics learning. In the case of task 2, first reachable region of y_3 was checked by Algorithm II. A cross in Fig.13 denotes the destination on the boundary

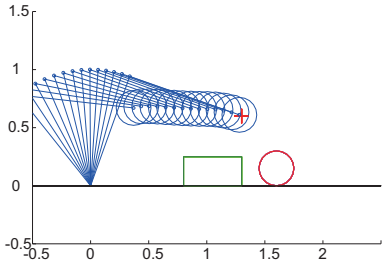


Fig. 12. Trajectory of task 1

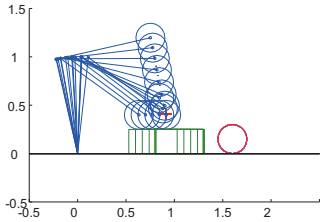


Fig. 13. Trajectory of task 2

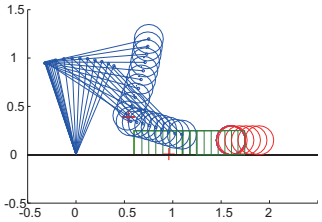


Fig. 14. Trajectory of task 3

between mode 1 and 2. First the robot hand aims at the destination on the boundary, and then keeps contact to realize the desired configuration of y_3. In the case of task 3, first reachable region of y_4 was checked. The lower cross in Fig.14 indicates the destination on boundary between mode 2 and 3, while the upper cross indicates the destination on boundary between mode 1 and 2. After realizing contact between the hand and object 1, the robot keeps contact to move object 1 to the right direction and to realize contact between object 1 and object 2. It can be seen that finally the robot realized the desired configuration of $y_4 = 1.9$ by extending its body by using object 1.

V. Discussion

In the simulation, it was shown that the proposed learning architecture enables various task execution by finding relations among variables and using the relations for planning with various observation variables.

Motor babbling [13] has been discussed as human learning of motor coordination. While the first stage of mapping learning in the proposed method can be equivalent to inverse-kinematics learning [14], the total framework deals with wider problem of finding relations among various observation variables. The proposal of this paper can be regarded as an construction method of body image [15], where 'self' is identified through the criterion of 'what can be controlled' and 'what cannot be controlled'.

On the other hand, the applications presented in the simulation is rather simple and easy. The proposed 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 the 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 framework of hybrid dynamical system control [16] 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' [17].

VI. 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

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

[2] Hauskrecht, M., Meuleau, N., Boutilier, C., Kaelbling, L. P., and Dean, T. (1998). Hierarchical solution of markov decision processes using macro-actions. In *Proc. of 14th Annual Conference on Uncertainty in Artificial Intelligence*, pages 220–222.

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

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

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

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

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

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

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

[10] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

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

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

[13] 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.

[14] 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.

[15] 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.

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

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