Learning to Generate Articulated Behavior Through the Bottom-Up and the Top-Down Interaction Processes

Jun Tani

Brain Science Institute, RIKEN

2-1 Hirosawa, Wako-shi, Saitama, 351-0198 Japan

Tel +81-48-467-6467, FAX +81-48-467-7248

E-mail tani@brain.riken.go.jp

(Neural Networks, 16-1, pp11-23, 2003)

Abstract

A novel hierarchical neural network architecture for sensory-motor learning and behavior generation is proposed. Two levels of forward model neural networks are operated on different time scales while parametric interactions are allowed between the two network levels in the bottom-up and top-down directions. The models are examined through experiments of behavior learning and generation using a real robot arm equipped with a vision system. The results of the learning experiments showed that the behavioral patterns are learned by self-organizing the behavioral primitives in the lower level and combining the primitives sequentially in the higher level. The results contrast with prior work by Pawelzik et al. (1996), Tani & Nolfi (1998), and Wolpert & Kawato (1998) in that the primitives are represented in a distributed manner in the network in the present scheme whereas, in the prior work, the primitives were localized in specific modules in the network. Further experiments of on-line planning showed that the behavior could be generated robustly against a background of real world noise while the behavior plans could be modified flexibly in response to changes in the environment. It is concluded that the interaction between the bottom-up

process of recalling the past and the top-down process of predicting the future enables both robust and flexible situated behavior.

1 Introduction

It is generally understood that higher-order cognition involves structural information processing which deals with the level of abstraction. When an agent perceives sensory flow, the agent might need to recognize only sequences of essential event-like features out of raw sensory signals. On the other hand, when the agent attempts to generate motor behavior, it is reasonable to assume that an abstract event sequence is generated in a higher level and that its detailed motor program is generated in a lower level. The question is how this sort of abstraction of information in the processes of sensation and behavior generation can be achieved based solely on the sensory-motor experiences of the agent. Kuniyoshi etal. (1994) argued that the key issue lay in the question of how the sensory-motor flow could be recognized and generated as *articulated*. Here, *articulated* means that continuous sensory-motor flow is temporally segmented into a sequence of chunks that are reusable as parts for reconstructing other experiences of sensory-motor flows.

The problems of segmentation and chunking have been studied in the context of sequence learning by Wang and Arbib (1990), Hochreiter and Schmidhuber (1997) utilizing a local short term memory architecture, and Sun and Sessions (2000) utilizing adaptive reinforcement learning modules. In the field of motor learning, Gomi and Kawato (1993), Wolpert and Kawato (1998), Haruno, Wolpert and Kawato (2001), and Bapi and Doya (2000) have shown that behavior primitives can be self-organized at each local network using a mixture of experts (Jacobs et al, 1991) type architecture by following Pawelzik's (1996) idea of segmenting sequences. Their models utilize a network consisting of multiple adaptive modules in which each adaptive module learns competitively to be an expert in predicting a specific primitive sensory-motor profile. When the winner of the experts switches from one module to another module corresponding to the structural changes in the characteristics of the sensory-motor flow, the sensory-motor flow is considered to have been segmented. The idea of utilizing a mixture of experts was further extended by introducing level structures to the system (Tani & Nolfi, 1998). Multiple levels of modular networks were considered. When the system repeatedly experiences a similar sequence of module activations in the lower level, this sequence itself can be learned by a module in the higher level network. Each module in the higher level network learns to encode a different sequence of the prim-

itives that is reusable in various situations. By cascading the networks into different levels, the sensory-motor flow can be articulated hierarchically. Those prior works utilizing a local short term memory architecture or a mixture of experts scheme share the same characteristic that primitives are represented locally in corresponding modules. A major drawback of such localist representation schemes is that the number of primitive patterns stored is limited to that of local modules.

The current paper proposes an alternative scheme in which multiple primitive patterns are stored distributedly. We call the scheme the "forwarding forward model" (FF-model) since a cascade of forward dynamics models (Kawato, Furukawa, & Suzuki, 1987; Jordan & Rumelhart, 1992) is utilized. It will be explained how behavior primitives are self-organized as well as evoked through the processes of bottom-up and top-down interactions utilizing the level structures proposed in the distributed representation scheme. Particularly, we propose a scheme of combining the recall of past experiences and the planning of future behavior for achieving both robust and flexible behavior generation. In order to generate adequate future behavior, the behavior programs have to be prepared prior to their execution. On the other hand, the behavior program currently generated should be modified flexibly even after its generation in response to situational changes in the environment. Therefore, the process of motor planning for the future should be performed in an on-line manner, i.e. integrated with the process of on-line recall of the past so that future behavior can be programmed while being situated in the past behavioral context. In other words, since the current behavior is generated by going through the interactive processes between top-down prediction of the future and bottom-up recall of the past, the behavior can be adaptively modified in response to a changing environment while also being performed robustly with respect to noise.

The proposed scheme will be examined through experiments of learning and generation of guided behaviors with a real robot arm. We will investigate (i) how behavior primitives are articulated and how they are integrated between the levels using limited sets of learning examples, and (ii) how the behavioral program is generated adaptively in response to situational changes using simple task examples. At the end of the paper, we discuss the advantages and disadvantages of the distributed representation scheme compared to the local one.

2 Model

2.1 Overview

We posit two levels of dynamical systems in which each dynamical system is operated on a different time scale. These two dynamical systems interact with each other in a top-down, bottom-up, or bidirectional manner, depending on the operational processes. We explain the basic ideas behind our model first by showing abstract dynamical systems models. The actual neural network modeling will be explained in the next subsection.

The diagram of the top-down process is shown in Figure 1 (a). In this figure the lower and higher level dynamics are represented on two different time scales, t and T, respectively. The lower level dynamics are represented in the form of a forward model (Kawato, Furukawa, & Suzuki, 1987; Jordan & Rumelhart, 1992) with the current sensory-motor state x_t as input and the next sensory-motor state x_{t+1} as output. A vector p_t is the "parametric bias" that functions as a set of parameters to modulate the characteristics of the lower level forward dynamics. The forward dynamics in the higher level determines the time development of the parametric bias P_T that is fed into the lower level forward dynamics as p_t. The idea is that the dynamics in the lower level is structurally changed when the values of the parametric bias are modulated since a mapping exists between the parametric bias and the dynamical structure in the lower level. This mechanism is analogous to bifurcations of nonlinear dynamical systems corresponding to parameter changes. Consequently, a specific sensory-motor pattern is generated by means of the modulation of the lower level forward dynamics as mapped from the parametric bias

The mapping from the parametric bias to the sensory-motor pattern is self-organized utilizing the error signal generated in the lower level network during the repeated learning of a given set of behavior patterns, as will be described later. From the dynamic constraints implemented in the model, the parametric bias values tend to change in a stepwise fashion only when the sensory-motor profile changes structurally. Otherwise, they stay relatively unchanged. The sensory-motor sequences are segmented into chunks by utilizing these stepwise changes of the parametric bias. The higher level forward model learns to predict this discrete sequence of parametric bias changes in terms of P_T, which represents the chunking sequence of the sensory-motor profile. In the top-down process, an abstract action scenario is generated in the higher level as a sequence of P_T. An exact profile of the sensory-motor flow can then be generated by feeding the parametric bias into the lower level dynamics. The higher level plays the

role of a 2nd order forward model, as it learns to predict how the lower level forward model changes in terms of the parametric bias.

A diagram of the bottom-up process is shown in Figure 1 (b). The bottom-up process recognizes its own sensory-motor sequence from past steps to the current step. Now let us consider the case in which the system experiences a sensory-motor sequence while its arm is manually guided in a specific movement pattern. If the system has already learned this pattern, then the lower level RNN can regenerate the target sequence by adapting the parametric bias sequence to the corresponding sequence. The sequence of p_t is iteratively computed by solving the inverse problem of minimizing the errors between the target and output sequences with smoothness constraints on the p_t sequence. Once p_t is determined, its value is sent to the higher level as P_T through a time-mapping function. Following this, the guided behavior is recognized in an abstract manner in the higher level network through the sequential inputs of the parametric bias P_T determined by the lower level dynamics.

The bottom-up and the top-down interactions are explained by the diagram in Figure 1 (c). This process occurs as the system exhibits behavior while its future action plans are simultaneously computed in an on-line manner. In this case the values of the parametric biases (p_t and P_T) are iteratively computed by using both the top-down and the bottom-up processes. The top-down process attempts to predict the action scenario in terms of the sequence of the parametric biases from the past to the future. The past sequence of the parametric biases can be modified by the bottom-up process through the error signals generated by the difference between the sensory sequence predicted by $f()$ and the real outcome. In this way, the motor plan can be adjusted dynamically in response to environmental changes.

2.2 Neural network architecture

We describe a neural network implementation of the ideas described above. This neural network model is utilized in three different sensory-motor system processes: the learning process, the motor planning process, and the recall process.

Figure 2 shows our proposed neural network architecture. The main architecture on the left-hand side consists of two Jordan type RNNs (Jordan, 1986) which correspond to the lower and the higher level networks. These RNN networks utilize the working memory storage shown on the right-hand side of the figure. The working memory stores the sequences of the parametric biases and the sensory-motor inputs/outputs where the recall as well as planning processes take place. In the current study, the

working memory is not realized in the form of neural networks but is implemented by a simple computer program. In the main architecture, the lower level RNN receives two types of input. One type is the vector of the current sensory-motor values (s_t, m_t), and the other is the vector of the current parametric bias p_t. This RNN outputs the prediction of the sensory-motor values at the next time step $(\hat{s_{t+1}}, \hat{m_{t+1}})$. In contrast, the higher level RNN receives P_T (the parametric bias at the current time step T) as inputs, then outputs its prediction at the time step $T + 1$.

In Figure 2 the parametric bias units in the lower level network are bi-directionally connected to the higher level network. The values of the parametric bias units are dynamically determined by utilizing the bottom-up, the top-down, or bidirectional signals, depending on the operational processes. In the learning process, only bottom-up signals are utilized for determining the parametric bias in the past sequence. When the robot actually behaves, two iterative computational processes of behavior recall and planning for the future are simultaneously activated. The motor planning process generates the motor program (the sequence of motor values) for future steps, utilizing the parametric bias determined by top-down signals at each future step. The recall process re-evaluates the sequence of the parametric bias from the past to the current time step by utilizing both the top-down and the bottom-up signals. Thus, once generated in the top-down manner the motor program can be modulated by the bottom-up signal if errors are generated in the sensory prediction in the lower level network. This sort of top-down and bottom-up interaction scheme enables the robot to modulate the motor program by on-line adaptation in response to environmental changes. We will now describe each process in detail.

(1) The learning process

In learning a target sensory-motor sequence, the temporal profile of the parametric bias p_t is computed dynamically in the lower level, corresponding to the sensory-motor profile in the sequence, while the synaptic weights in the lower level network are modified. The higher level network, on the other hand, learns to predict how the parametric biases change in the sequence. Both learning processes in the lower and the higher levels are conducted in an off-line manner.

The temporal profile of p_t in the sequence is computed via the back-propagation through time (BPTT) algorithm (Rumelhart, Hinton, & Williams, 1986; Werbos, 1990), utilizing the sequence of the internal values of the parametric bias ρ_t, the teaching target, and the outputs of the sensory-motor sequences in the working memory storage. The total number of steps of these sequences in the working memory is L. For each learning iteration, the forward dynamics of the RNN are computed for L

steps by establishing closed sensory-motor loops. For the closed loop, denoted by the dotted line in the left-hand side of the lower level RNN in Figure 2, copies of the current sensory-motor prediction outputs are fed back to the next inputs which enables look-ahead prediction for an arbitrary number of future steps. Once the L steps of the prediction sequence are generated, the errors between the prediction outputs and the teaching targets are computed and then back-propagated through time in order to update both the values of the parametric bias at each step and the synaptic weights. The update equations for the ith unit of the parametric bias at time t in the sequence are:

$$\delta {\rho_t}^i = k_{bp} \cdot \sum_{t-l/2}^{t+l/2} {\delta_t^{bp}}^i \cdot 1/l + k_{nb}(\rho_{t+1}^i - 2\rho_t^i + \rho_{t-1}^i) \quad (1)$$

$$\triangle \rho_t^i = \epsilon \cdot \delta {\rho_t}^i + \eta \cdot \triangle \rho_{t-1} \quad (2)$$

$$p_t^i = sigmoid(\rho_t/\zeta) \quad (3)$$

In Eq. (1), $\delta {\rho_t}^i$, the update of the internal values of the parametric bias ρ_t^i, is obtained from the summation of two terms. The first term represents the delta error, δ_t^{bp}, back-propagated from the output nodes to the parametric bias nodes. It is summed over the period from $t - l/2$ to $t + l/2$. By summing the delta error, the local fluctuations of the output errors will not affect the temporal profile of the parametric bias significantly. The parametric bias should vary only corresponding to structural changes in the sensory-motor sequences. As will be discussed later, this summation step length l should be determined heuristically since its value affects the segmentation process significantly. The second term plays the role of a low pass filter through which frequent rapid changes of the parameter values are inhibited. ρ_t is updated by $\delta {\rho_t}^i$, obtained from the steepest descent method, as shown in Eq. (2). Then, the current parameters p_t are obtained by means of the sigmoidal outputs of the internal values ρ_t. A parameter ζ is employed such that the gradation of the parametric bias can be controlled. If ζ takes smaller values, the parametric bias tends to have more extreme values, either near 0 or 1. On the other hand, if ζ takes larger values, the parametric bias tends to take graded values. In the actual learning process, ζ varies from larger to smaller values as learning proceeds. At the end of learning, one can observe chunks of 1 or 0 bit representations of the parametric bias during the time steps in which the segmentation took place.

Finally, the higher level network learns to predict the sequences of the segmentations. More specifically, the higher level network learns to predict the bit patterns P_{T+1} and the interval length τ_{T+1} of the next chunk.

(2) The motor planning and recall processes

The sequence of the sensory-motor prediction for future steps is generated in a real time manner while the sensory-motor sequence of past steps is recalled such that the plans are generated appropriately for the current behavioral context. The prediction of the future steps from t_c to t_p is generated by means of both forward dynamics in the lower level RNN and in the higher level RNN. The higher level RNN supervises the lower level RNN in a top-down manner. The higher level RNN generates a predicted sequence of the parametric bias P_T and the interval length τ_T in a closed loop manner, which determines the parametric bias units p_t in the lower level RNN. Consequently, for a given P_t, a sequence of the sensory-motor prediction is generated as a future plan by means of the closed-loop forward dynamics in the lower level RNN. The p_t in future steps is computed using

$$\delta\rho_t{}^i = k_{top} \cdot (\hat{P_T}{}^i - 0.5) + k_{nb}(\rho^i_{t+1} - 2\rho^i_t + \rho^i_{t-1}) \tag{4}$$

The first term represents the top-down signal where $\hat{P_T}{}^i$ and k_{top} denote the parametric bias predicted in the higher level network and its coefficient, respectively. The second term is the same as the second term in Eq. (1). p_t is calculated by means of the steepest descent method with the sigmoid transformation applied to p_t, as in Eq. (2).

The computation of P_T in future steps by means of the forward dynamics in the higher level RNN requires the determination of P_T in the past. P_T in the past is continuously modified through recall of the past sequence in the lower level RNN, where the results of the bottom-up sensations are re-interpreted utilizing the top-down expectations for the sequence. More specifically, the past sequence of p_t from t_r to $t_c - 1$ is computed iteratively in the interaction between the expectation of the parametric bias in the higher level RNN and the signal back-propagated from the sensory prediction error in the lower level RNN. The update rule for the ith parametric bias at time t in the sequence is:

$$\delta\rho_t{}^i = k_{top} \cdot (\hat{P_T}{}^i - 0.5) + k_{bp} \cdot \sum_{t-l/2}^{t+l/2} \delta_t^{bp\,i} \cdot 1/l + k_{nb}(\rho^i_{t+1} - 2\rho^i_t + \rho^i_{t-1}) \tag{5}$$

The first and second terms represent the top-down and the bottom-up signals, respectively. Subsequently, P_T is determined by applying the segmentation rules to the p_t sequence obtained.

3 Robot Experiments

Our proposed model was examined through a set of behavior learning experiments using an arm robot. The robot used in our experiments has 4 degrees of freedom in its arm joints; a hand on the top of the arm can sweep over the task table horizontally as shown in Figure 3. A colored mark is attached to the top of the hand for the purposes of video image processing. A colored object is prepared in the task space which the robot can manipulate by pushing with its hand. The robot is equipped with a color video camera by which the plan view positions of the hand and the object are tracked using color filtering. A handle is attached to the hand so that a trainer can teach behavior to the arm manually.

To teach the robot, the arm is guided manually to generate various behavioral sequence patterns. The sensory-motor sequences associated with the generated sequence patterns are used for off-line learning of these behavioral patterns. In preparing the manual teaching patterns, a set of primitive behaviors is defined. The teaching sequence patterns are generated by combining the primitive behaviors in sequence. The robot then has to learn these given behavioral sequence patterns as articulated - i.e., a pattern to be learned is recognized as a sequential combination of primitive sequences that can be repeatedly utilized for representing other sequence patterns. The objective of this learning is not just to learn to regenerate taught patterns but also to seek the segmentation structures hidden in the patterns. We will closely observe the way such articulation structures self-organize through learning based on the hidden primitives. The advantage of this type of learning is that once the primitive representations are self-organized internally, the robot can easily adapt to diverse behavior patterns by combining these primitives.

We conduct two types of experiments. In the first experiment we examine how the robot learns to generate a set of behavior patterns by focusing on the ways of self-organizing primitive representations in the network. In the second experiment, we investigate how the motor plans can be dynamically modified in the course of their execution in response to environmental changes. The second experiment provides us with an opportunity to examine the roles of the bottom-up and the top-down interactions in on-line behavior generation.

These experiments are conducted using the proposed neural network model. The network size and parameters were determined by trial and error in order to find robust conditions for the robot experiments. Parameter sensitivity tests were then conducted for those parameters considered essential for the total system dynamics. The lower level

RNN has 8 input nodes which are allocated to the 4 motor positions of the arm and to the two dimensional Cartesian positions of the hand and object obtained through the video camera image processing for the current time step. The output nodes are allocated in the same way as the input nodes, but with the values for the next time step. All values are normalized to a range of 0.0 to 1.0. The lower level RNN has 20 hidden nodes and 8 context nodes. It also has 4 parametric bias nodes in the input layer. The higher level RNN has 4 input and output nodes which are allocated to the parametric bias of the current and the next time step respectively. It also has 10 hidden nodes and 6 context nodes.

3.1 Learning

In the learning experiments, 7 primitive behavior patterns are defined as illustrated in Figure 4(a). In Figure 4(a), a shaded circle represents the colored object. The arm is in the home position when it is touching the rectangular box on the right-hand side. Abbreviations for each primitive behavior are given. After determining the primitive behaviors, 7 training sequence patterns are generated through manual guidance of the arm by combining these primitives in sequences. The lower level RNN is then trained using the sequences. Figure 4(b) shows the 7 training sequence patterns in terms of sequences of the primitives. Note that the training sequences are carefully designed such that they do not include any deterministic sub-sequences of the primitives. For example, after AO either PO or HO can follow. If PO were always to follow after AO, the AO-PO sequence could be regarded as an alternative primitive.

The learning experiments are conducted repeatedly for various integration step lengths l since this parameter is assumed to affect significantly the segmentation of the sensory-motor flow. Other parameters are determined in the pre-experiments such that the results of the learning are acceptable –i.e. the mean square error is reduced to less than 0.005 while generating stable segmentation structure. The parameters are set as follows for Equations (1) and (2): $k_{bp} = 0.1$, $k_{nb} = 0.005$, $\epsilon = 0.01$ and $\eta = 0.9$. We examined how the behavioral primitives are acquired as articulated in the training patterns by observing the relationship between the training error and the segmentation rate with parameter l variation. The segmentation rate is calculated as the average ratio of the actual number of the segments generated in the learning processes to the actual numbers of primitives combined in the training sequence patterns. The results are plotted in Figure 5 in which the mean square error and the segmentation rate (on a log scale) are plotted as a function of the integration step length. Observe that the

mean square error becomes higher and the segmentation rate becomes lower as the integration step length increases. This means that the learning error can be minimized if fragmentation of the segmentation is allowed, while the error typically increases if such fragmentation is not permitted through control of the parameter l.

We examine in detail the case in which l is set to 6. Figure 6 shows how the parametric bias is activated during learning for each training sequence. The plots in the top row of this figure show the activation of four parametric bias units as a function of the time step; the activation values from 0.0 to 1.0 are represented using a grayscale from white to black, respectively. The plots in the second and third rows represent the temporal profile of motor and sensor values for each training sequence. The vertical dotted lines indicate the occurrence of segmentation when the behavior sequence switches from one primitive to another in generating the training sequence. The capital letters associated with each segment denote the abbreviation of the corresponding primitive behavior. In this figure, observe that the switching of bit patterns in the parametric bias takes place mostly in synchronization with the segmentation points known from the training sequences, although some segments are fragmented. Observe also that the bit patterns in the parametric bias correspond uniquely to primitive behaviors in a one-to-one relationship in most cases.

Next, we examine how the robot can generate novel behavioral patterns made by combining the pre-learned primitive behaviors. Three behavioral patterns are prepared. When each behavioral pattern is taught, only the connective weights in the higher level RNN are allowed to adapt, while those in the lower level RNN remain unchanged. This scheme assumes that the internal representations for primitives are preserved in the lower level RNN. More specifically, the sequences of the parametric bias p_t are obtained by iterative computation using Eq. (1) for the lower level RNN in which the learning rate of the connective weights ϵ is set to 0.0. Subsequently, the higher level RNN is trained using the articulated sequences of the parametric bias P_T.

After the learning in the higher level RNN converges, the robot attempts to regenerate each behavioral pattern. The actual behavior of the robot is generated based on the motor planning and recall processes described previously. Figure 7 shows the comparison of the temporal profiles between the taught patterns (left-hand side) and the regenerated patterns, as the robot actually behaves (right-hand side), for each sequence. Observe that the sensory-motor profiles are successfully regenerated from the patterns taught without any significant discrepancies.

One may conclude that there exists a certain parameter range in which learning can be performed successfully using a self-organizing internal representation of primi-

tive behaviors. The primitives acquired turn out to be reusable in other instances of learning.

3.2 On-line adaptation of motor planning

The next robot experiment simulates possible cognitive behaviors of animals or humans when alternative behavior plans have to be generated in response to environmental changes. One difficulty is that behavior should be continuously generated even during the on-line modification of the motor program. We demonstrate that our scheme of both recalling the past and planning for the future through interactive bottom-up and the top-down processes can cope with the problems of on-line adaptation of behavior plans.

For the experiments, the robot is trained with two different behavioral patterns each of which is associated with a specific environmental situation. The test is then to examine how the behavioral patterns are switched when the environmental situation changes. This switching task is not as trivial as simple phase transitions in physics, where the system slips down smoothly from one energy state to another when certain external forces are applied. Although two behavior tasks have to be alternated, the top-down pathway cannot generate the sequential image of such switching situations since the two behavior patterns are learned as independent sequences. The top-down processes would force the system to stay in the current task context. On the other hand, the bottom-up processes would force the system to switch to another task context as the error back-propagated from the sensory prediction tends to modulate the parametric bias at the moment of the situational change. The question is how these two conflicting forces can be resolved. One might predict that the experimental results would be either (1) a smooth phase transition, (2) getting stuck in some local minimum, or (3) diverse transient dynamics.

In the experiments, two different behavior patterns are trained by adapting the synaptic weights only in the higher level RNN while those in the lower level RNN are preserved as acquired in the previous experiment. In the first behavior, the arm repeats a sequence of approaching an object and then returning home. The object is located in the center of the task space. The second behavior is that the arm repeats a sequence of centering, making a C-shape, and then returning home while the object is located to the left-hand side of the task space. Figure 8 shows the sensory-motor profiles associated with the parametric bias for these trained behaviors. After the learning converged, we examined how the behavioral patterns changed when the position of the object was

suddenly moved from the center to the left-hand side of the task space in the middle of executing the first behavior.

One may assume that the balance between the top-down and the bottom-up processes greatly affects the system's behavior. Therefore, the experiments on behavior switching were conducted repeatedly, changing the strength of the top-down signals by varying the coefficient k_{top} in Eq. (4) and Eq. (5) from 0.0 to 0.1 with a 0.01 increment. Other parameters in Eq. (4) and Eq. (5) were set to $k_{bp} = 0.1$, $k_{nb} = 0.005$ and $l = 6$ which are the same as those used in the previous learning experiment. In particular, we examined the smoothness of behavior switching by observing the time lag from the moment the object was moved to the moment when the second behavior was activated. The switching trial was repeated five times for each setting of k_{top}. Other conditions, such as the timing of the object movement, were the same for all the trials. Figure 9 shows the plot of the time lag versus the coefficient k_{top}.

It was observed that the first behavior pattern was not accomplished when k_{top} was set to less than 0.03. The parametric bias was not activated as learned since the top-down signal was too weak. When the top-down behavior plan, in terms of the parametric bias sequence, was executed on-line, the sequence was easily affected by even small prediction errors in the lower level, caused by noise in the robot's operation. When k_{top} was set from 0.03 to 0.07, switching from the first behavior to the second behavior took place. The time lag tended to increase as k_{top} was increased. This indicates that the motor plans tend to be less sensitive to environmental changes when the top-down effects become larger, as expected. However, an important observation is that there is a relatively large distribution of the time lag for each k_{top} value. This suggests that diverse transient behavior are generated during the switching. When k_{top} was set to larger than 0.07, no behavior switching was observed. The parametric bias was no longer affected by the bottom-up sensations since the top-down influence on the parametric bias was too strong.

Figure 10 shows the temporal profile of the behavior generated in the case where k_{top} was set to 0.05. The profiles of the parametric bias, the motor outputs and the sensor inputs are plotted in the top, in the second and in the third rows, respectively. The vertical dotted line denotes the moment when the object is moved from the center to the left-hand side of the task space. Observe that it takes 20 steps until the second behavior pattern is initiated after the object is moved to the left-hand side. Observe also that the parametric bias, the motor outputs, and the sensory inputs fluctuate during this transition period. The fluctuation is initiated because of the gap generated between the top-down prediction of the parametric bias and the real inputs in the

bottom-up process. The fluctuations in the parametric bias result in the generation of complex motor patterns in the lower level. These patterns occur by means of the top-down pathway which generates the sensory prediction error that is again fed-back to the parametric bias by means of the bottom-up pathway. In the five repeated trials in this parameter setting, the profiles of the transient patterns were never repeated, as is suggested by the large distribution of the time lag. The observed fluctuation seems to play the role of a catalyst in searching for diverse transition paths from the steady attractor, associated with the first behavior pattern, to that for the second behavior pattern. Once the transition is complete, the second behavior pattern proceeds steadily.

The experimental results are summarized as follows. On-line motor plan modification can be conducted in the interactions between the top-down and the bottom-up processes. If the top-down signal is too strong and the bottom-up signal is too weak, then the behavior can be generated robustly by following the top-down commands, but it becomes insensitive to environmental changes. In the opposite case, the behavior generation becomes unstable since it becomes too sensitive to noise. There is a relatively large range of the parameter ($0.03 < k_{top} < 0.07$) where on-line motor plan modification can be successfully conducted. The transient behavior during switching is neither smooth nor trapped by certain local minima. It is complex and diverse since it is sensitive to slight differences in the initial conditions. (Note that we attempted to initiate the switching with the same timing as during the first task.) Where does this complexity come from? First, it may come from the dynamic mechanism of mutual interactions between multiple levels, from the sensory-motor level to the lower and higher RNN levels. Second, it may come from the context dependent nature in the recall of the past as well as future planning. The matching between the top-down representation of the sensory-motor pattern and the actual experience is made not only for the current moment, but for a past time period up to the current moment. The fact that future behavior patterns are generated based on this sort of context dependent matching (including the past history) makes the system's behaviors nontrivial. Finally, the complexity may come from the nonlinear characteristics of the proposed distributed representation. Figure 10 shows fragmented patterns of the parametric bias during switching. It is likely that this fragmentation contributes to the generation of diverse behaviors utilizing nonlinear characteristics of the system. This argument will be revisited in the discussion section.

4 Discussion and Summary

In this paper we have proposed the FF-model, which is characterized by a distributed representation scheme. Our experiment demonstrated that complex behaviors can be learned and generated in an articulated, segmented manner. Behavior primitives self-organize by utilizing the distributed representation scheme in the lower level RNN of the FF-model. It was also shown that the bottom-up and the top-down interactions are essential for the system to adapt to its environment in a robust and flexible manner.

There are continuing discussions about whether primitives should be represented locally or in distributed networks. In the localist view, complex behaviors can be decomposed into a set of reusable behavioral patterns which are each stored in a specific local neural network module. Our FF-model contrasts with this localist view in that multiple behavior primitives are represented in a distributed manner in a single RNN, where all neurons and synaptic weights participate in representing all trained patterns.

A specific difference between the two schemes is that the number of primitives in the local representation scheme is constrained by the number of local modules, while that in the distributed system is constrained by the number of possible bit combinations in the parametric bias, as we have described previously. However, it could be argued that an infinite number of behavior patterns can be generated in a localist mixture of experts scheme if the gating of local modules is allowed to take analog values. In fact, Haruno, Wolpert and Kawato (2001) showed in their MOSAIC model that novel behavior patterns can be generated by means of linear interpolations among outputs of local forward model networks by adjusting their gate values. In contrast with this result, our recent study (Tani, 2002) showed that nonlinear characteristics play essential roles in generating behavior patterns in the FF-model. In this study, multiple attractor dynamics were trained in the lower level RNN while the parametric bias was gradually changed for each attractor pattern. After the convergence of learning, we examined the type of mapping that was organized between the parametric bias and the corresponding dynamic patterns. It was shown that the attractor patterns bifurcated in various ways depending on the changes of the parametric bias. The patterns could be abruptly modulated in some regions of the parametric bias space by small changes of the parametric bias, while in other regions they could be smoothly modulated. The system was capable of generating diverse patterns that could not be accounted for by means of linear interpolations among trained patterns, since the mapping between the parametric bias and the sensory-motor outputs could be nonlinear. (Note that the parametric bias in the input layer can nonlinearly affect the sensory-motor values

in the output layer in three-layered networks, as employed in the current scheme.) Such nonlinear characteristics could be utilized for generating diverse, self-exploratory behaviors. However, it is also true that the proposed scheme could have stability problems in generating behaviors. Behavior control becomes more difficult when the mapping of the parametric bias to the behavior outputs becomes nonlinear.

These two types of motor control scheme differ also in their performance during incremental learning. In the localist network architecture, a novel pattern can be learned by allocating an additional local module, thereby minimizing interference between the novel patterns and previously memorized patterns. For example, Wang and Yuwono (1996) reported that their localist scheme does not suffer from so-called catastrophic interference (McClosky & Cohen, 1989), which is the complete loss of old memories during new learning. Their extensive simulations indicated that the number of damaged or forgotten patterns did not increase with the number of stored patterns. Also, in our prior studies (Tani & Nolfi, 1998) using a mixture of RNN experts, incremental learning was successfully performed without catastrophic interference. However, catastrophic interference did occur in the current distributed representation scheme. The learning processes can converge only through off-line batch processes.

Deciding whether to use a distributed or a local representation scheme seems to be a trade-off problem. The localist scheme has greater stability for generating and learning behavior patterns. On the other hand, if nonlinear effects in the distributed representation scheme are utilized, then one gains diversity in the generated patterns. at the cost of losing the stability of behavior generation and learning. It is likely that biological neural networks never sit in either of these two extremes, and that they find optimal locality/distributedness depending on the cognitive tasks involved. Our future studies will focus on possible adaptation mechanisms by which the degree of locality/distributedness can be self-determined.

References

[1] Bapi, S., & Doya, K. (2000). Multiple forward model architecture for sequence processing. In Sun, R., and Giles, C., eds., *Sequence Learning*. Berlin, Springer

[2] Gomi, H. & Kawato, M. (1993). Recognition of manipulated objects by motor learning with modular architecture networks. *Neural Networks* 6, 485–497.

[3] Haruno, M., Wolpert, D., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Computation* 13, 2201–2220.

[4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation* 9(8), 1735–1780.

[5] Jacobs, R., Jordan, M., Nowlan, S. & Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation* 3(1), 79–87.

[6] Jordan, M.I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. *Proc. of Eighth Annual Conference of Cognitive Science Society* 531–546., Hillsdale, NJ: Erlbaum

[7] Jordan, M.I. & Rumelhart, D.E. (1992). Forward models: supervised learning with a distal teacher. *Cognitive Science* 16, 307–354.

[8] Kawato, M., Furukawa, K. & Suzuki, R. (1987). A hierarchical neural network model for the control and learning of voluntary movement. *Biological Cybernetics* 57, 169–185.

[9] Kuniyoshi, Y., Inaba, M., & Inoue, H. (1994). Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance. *IEEE. Trans. on Robotics and Automation* 10(6), 799–822.

[10] McCloskey, M. & Cohen, N. (1989). Catastrophic interference in connectionist network. *Psych. Learning Motivat.* 24, 109–165.

[11] Pawelzik, K., Kohlmorgen, J. & Muller, K.-R. (1996). Annealed competition of experts for a segmentation and classification of switching dynamics. *Neural Computation* 8(2), 340–356.

[12] Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In Rumelhart, D., and McClelland, J., eds., *Parallel Distributed Processing*. Cambridge, MA: MIT Press.

[13] Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation* 4(2), 234–242.

[14] Sun, R., & Sessions, S. (2000). Automatic segmentation of sequences through hierarchical reinforcement learning. In Sun, R., and Giles, C., eds., *Sequence Learning*. Berlin, Springer

[15] Tani, J. & Nolfi, S. (1998). Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. In Pfeifer, R., Blumberg, B., Meyer, J. and Wilson, S.W., eds., *From animals to animats 5*. Cambridge, MA: MIT Press. later published in *Neural Networks* 12, 1131–1141.

[16] Tani, J. (2002). Self-organization of behavioral primitives as multiple attractor dynamics: a robot experiment. to be presented at *IJCNN'2002*, Honolulu

[17] Wang, D. & Arbib, M. (1990). Complex sequence learning based on short-term memory. *Proc. IEEE* 78, 1536–1543.

[18] Wang, D. & Yuwano, B. (1996). Incremental learning of complex temporal patterns. *IEEE Trans. on Neural Networks* 7(6), 1465–1480.

[19] Werbos, P. (1990). Backpropagation through time: what does it mean and how to Do it. *Proc. IEEE* 78, 1550–1560.

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

List of Figures

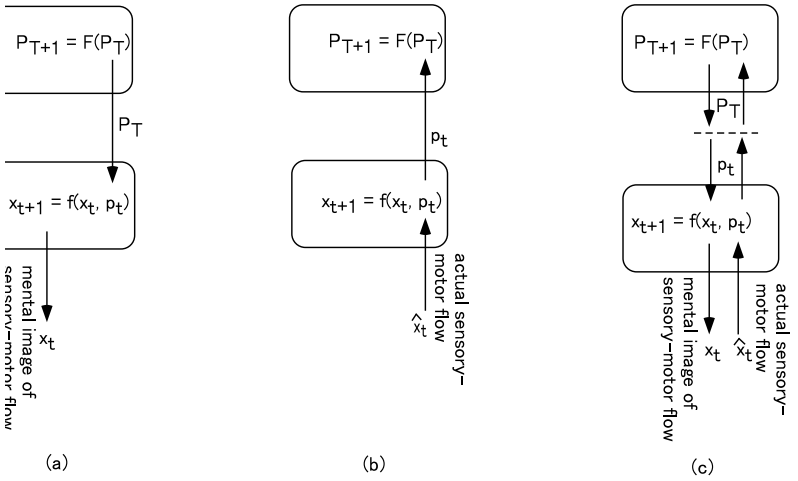


Figure 1: Schematic diagram of the model for (a) the top-down process, (b) the bottom-up process and (c) the top-down and the bottom-up interactive process.

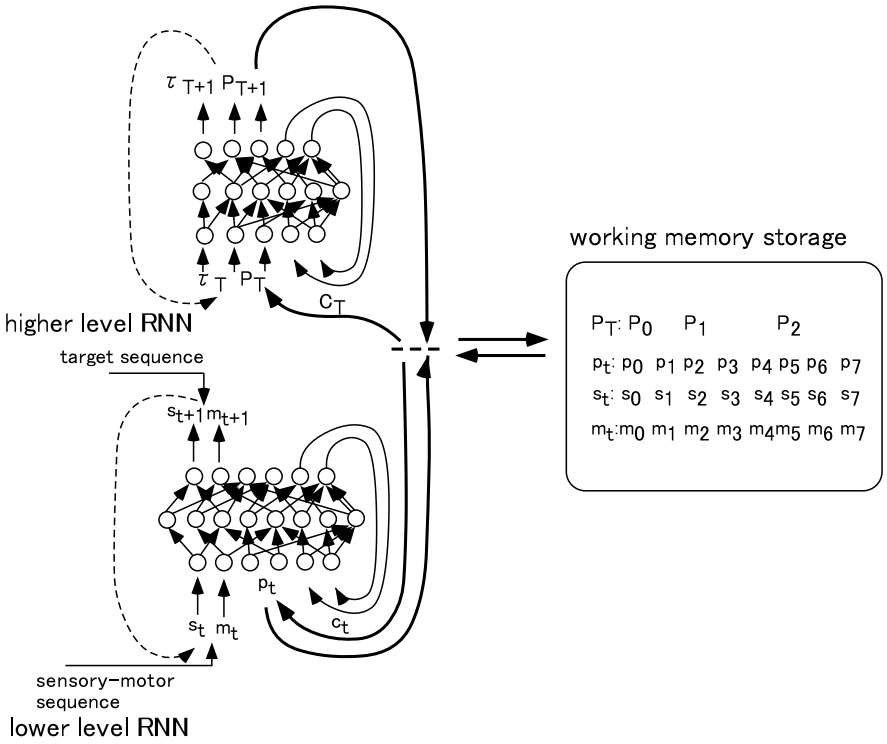


Figure 2: The complete architecture. Two levels of RNNs on the left-hand side and the working memory on the right-hand side.

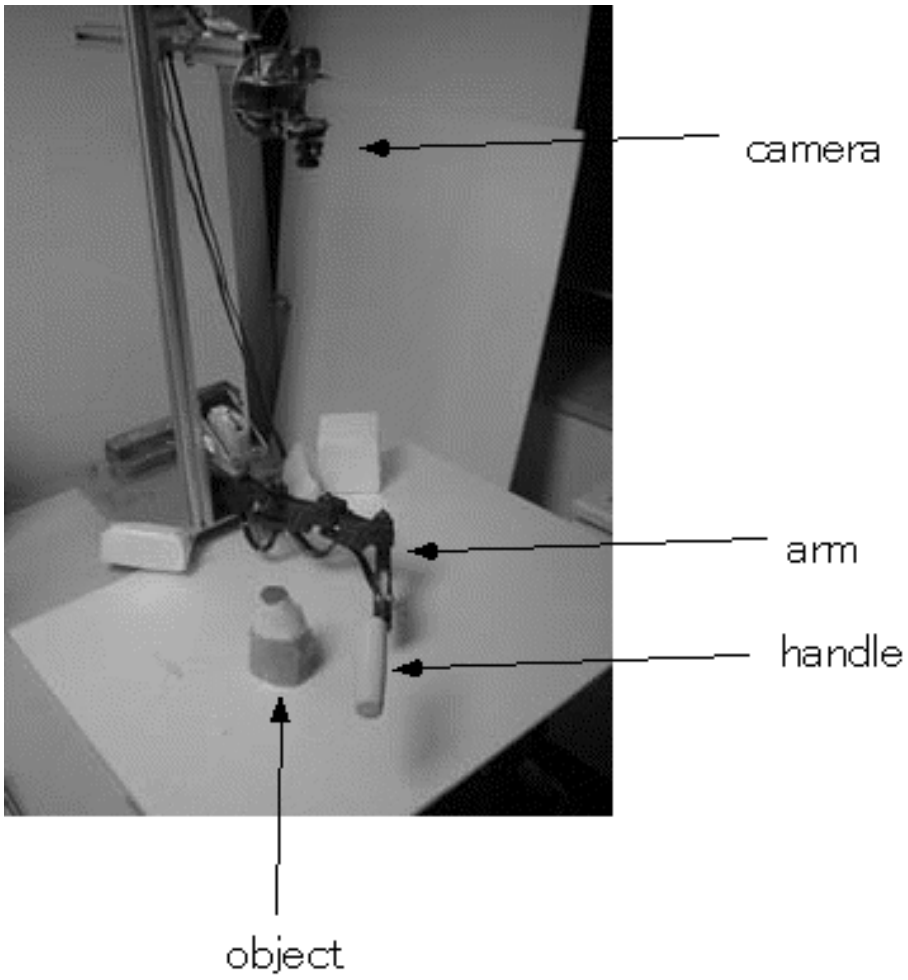
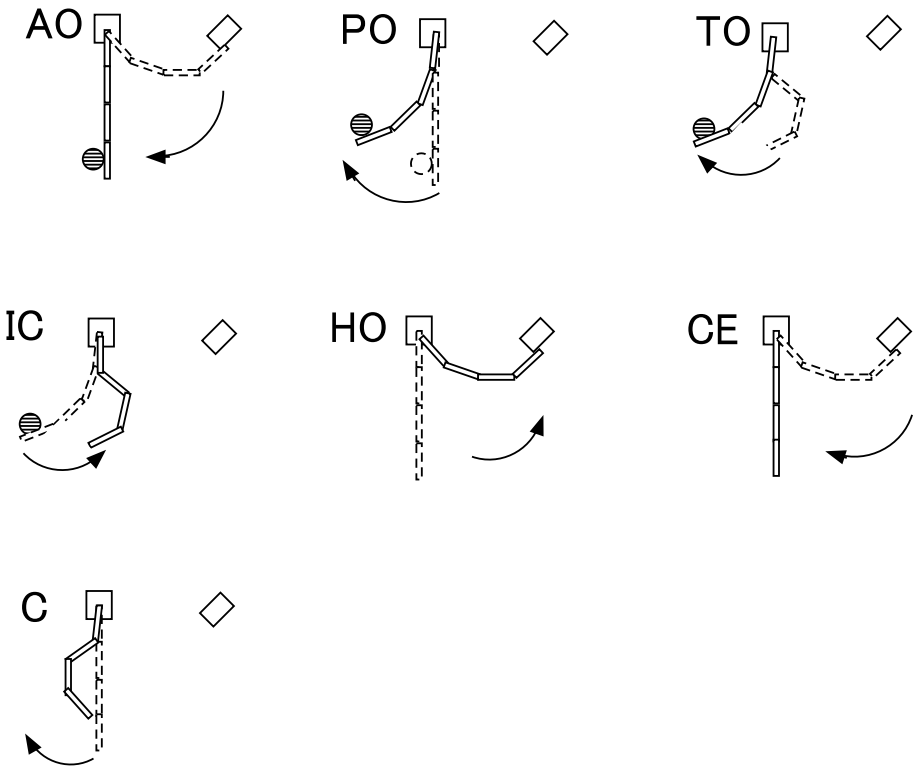


Figure 3: The arm robot used in the imitation learning experiments.



(a)

(1) AO−PO−IC−TO−HO
(2) AO−PO−IC−TO−IC−TO−HO
(3) AO−HO−AO−HO−AO−HO
(4) AO−PO−HO
(5) CE−C−CE−HO
(6) CE−C−CE−C−HO
(7) CE−HO−CE−HO

(b)

Figure 4: (a) The primitive behaviors defined for imitation learning, abbreviated as follows. AO: approach to object in the center from the right-hand side. PO: push object from the center to the left-hand side. TO: touch object. IC: perform inverse C shape. HO: go back to home position. CE: go to the center from the right-hand side. C: perform C shape. (b) The seven sequences used.

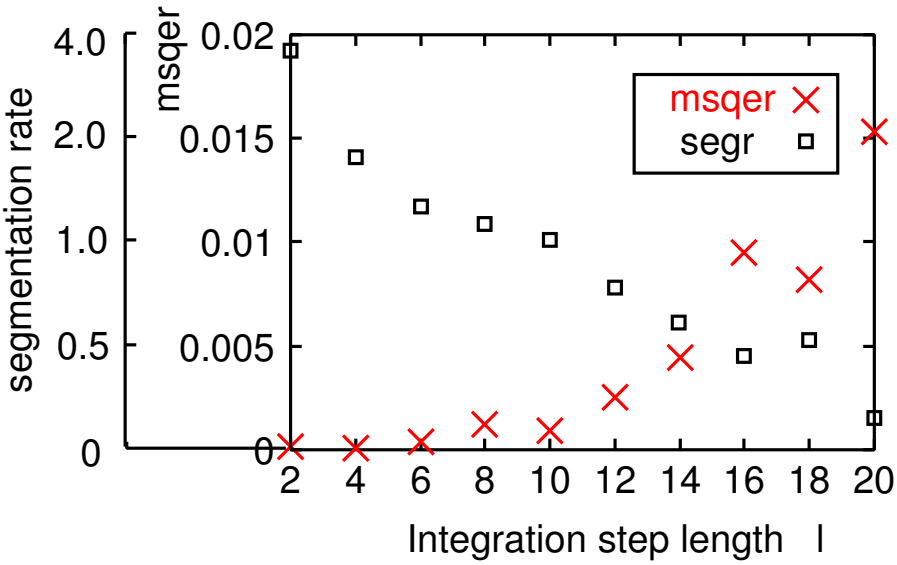


Figure 5: The mean square error (msqer) and the segmentation rate (segr, log-scale) plotted as a function of l in the repeated learning trials.

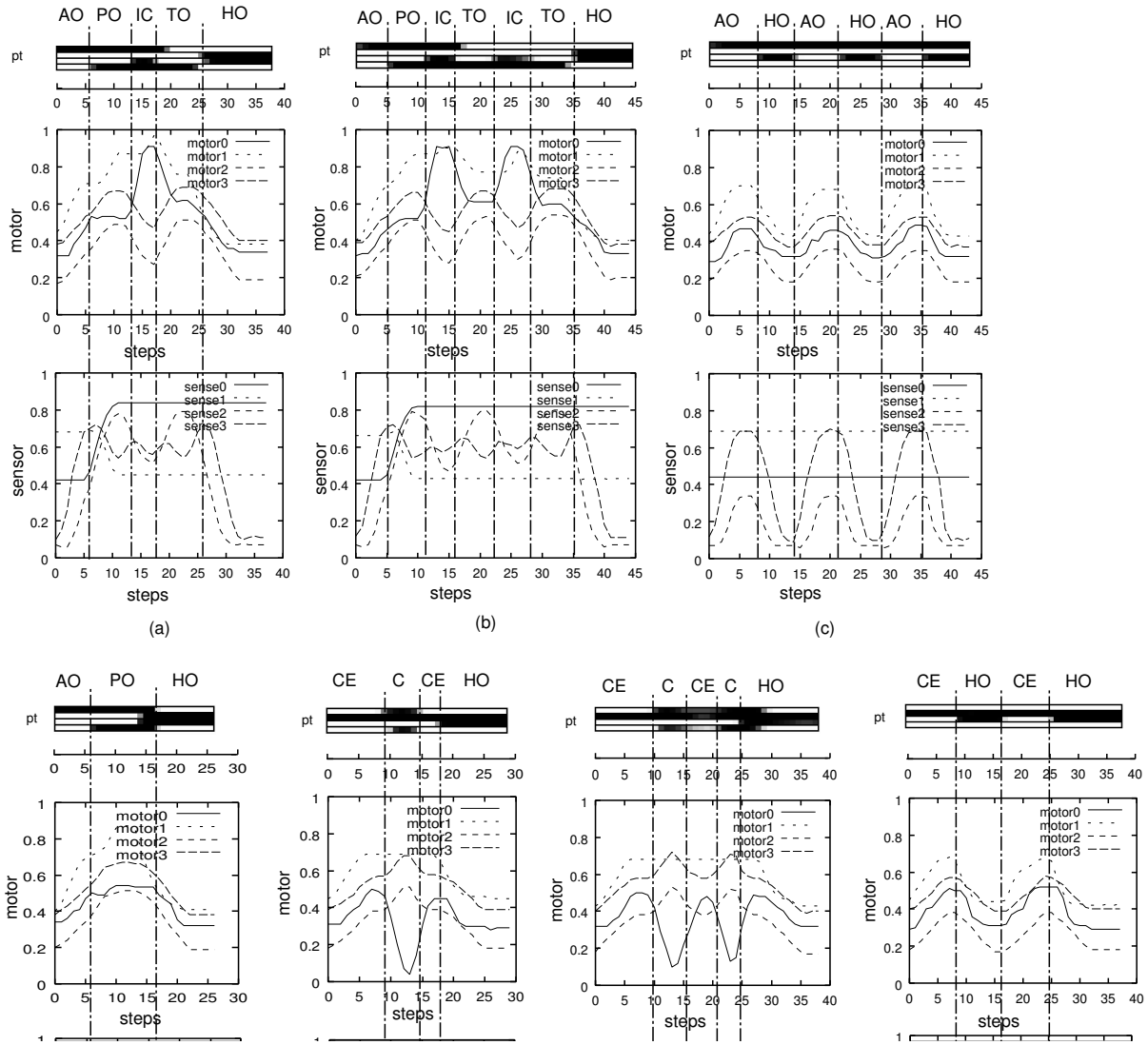


Figure 6: For the seven training sequences (a)-(g), the temporal profiles of the parametric bias which resulted from learning are plotted in the top row, the motor outputs are plotted in the second row and the sensor inputs are plotted in the third row. The vertical dotted lines denote the occurrence of segmentation when the primitive behaviors switched in the training sequences. The capital letters associated with each segment denote the abbreviation of the corresponding primitive behavior.

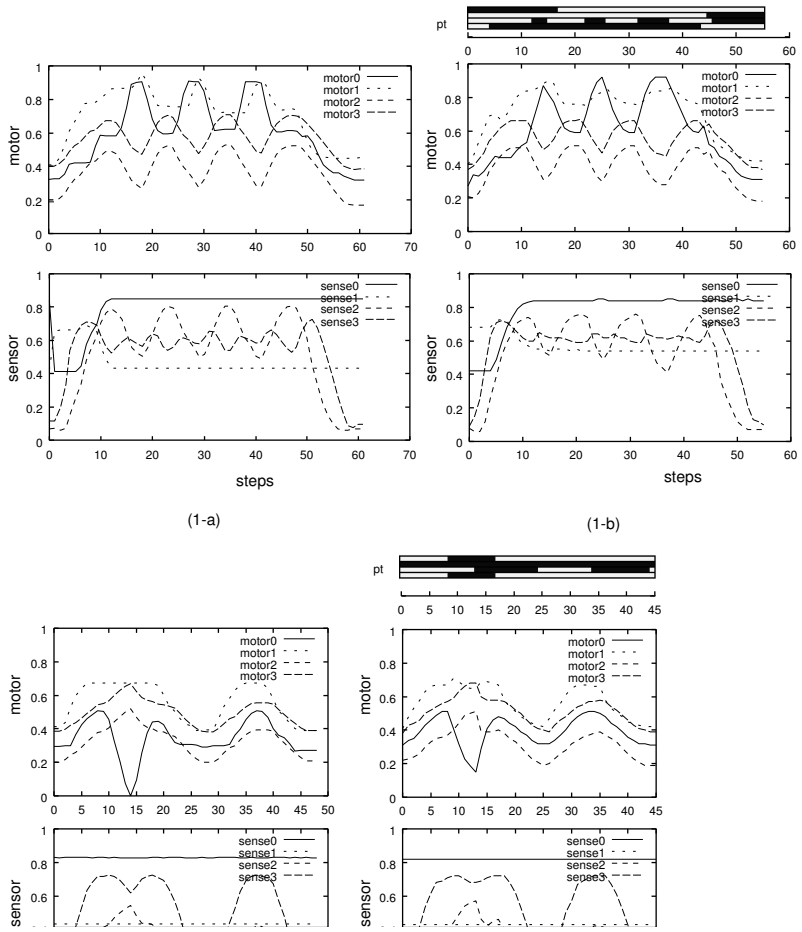


Figure 7: Comparison between the taught sensory-motor patterns on the left-hand side (1-a, 2-a, 3-a) and the regenerated patterns on the right-hand side (1-b, 2-b, 3-b) and their associated parametric bias, for 3 sequences.

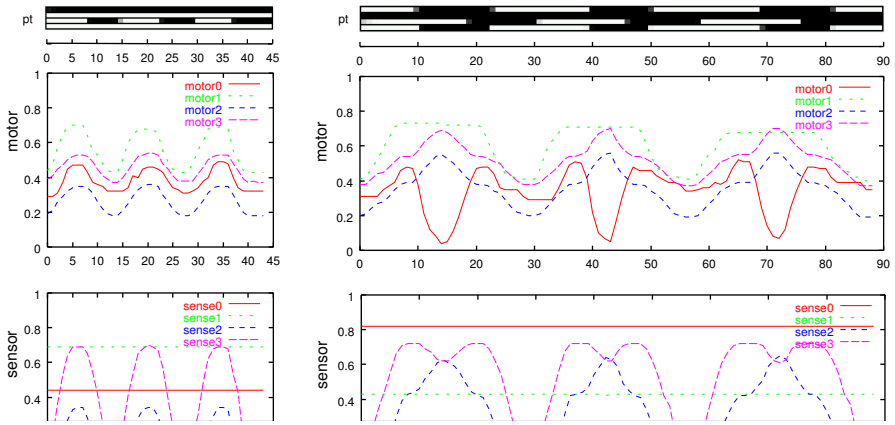


Figure 8: Two behavioral patterns (a) and (b) as trained in preparation for the experiments on on-line behavior switching. The temporal profiles of the parametric bias, the motor outputs, and the sensor inputs are shown in the top, the second, and the third rows, respectively.

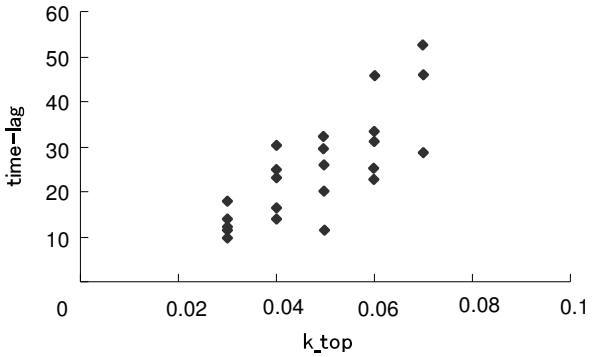


Figure 9: The time lag in behavior switching, plotted as a function of the top-down coefficient, k_{top}.

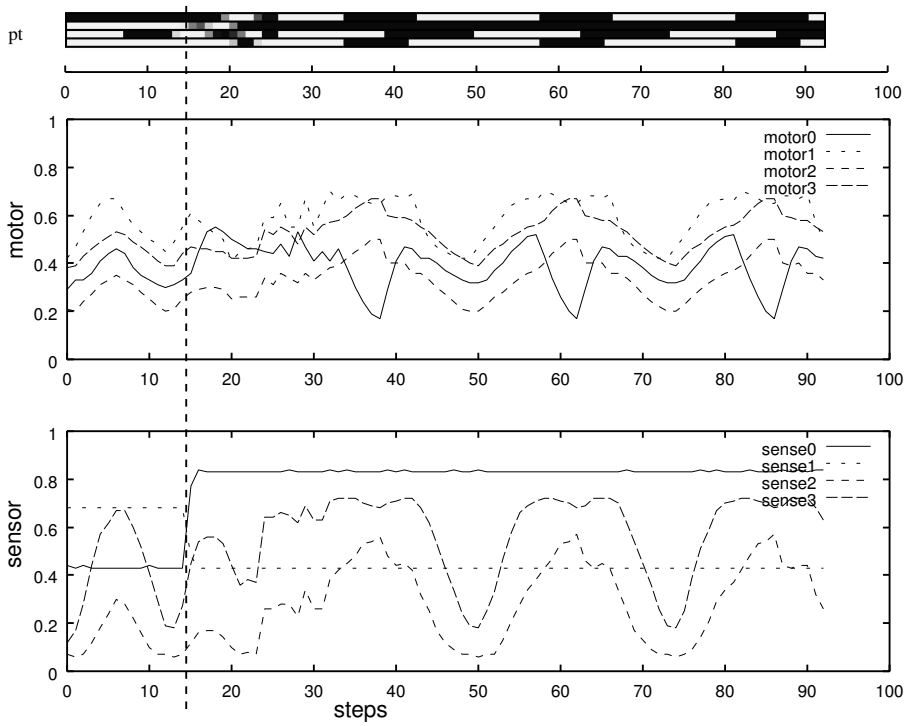


Figure 10: A behavioral switching trial for which k_{top} was set to 0.05. The temporal profiles of the parametric bias, the motor outputs, and the sensor inputs are shown in the top, the second, and the third rows, respectively. The vertical dotted line denotes the moment when the object was moved from the center to the left-hand side of the task space.