Neural Networks, Vol. 6, pp. 485–497, 1993
Printed in the USA.

0893-6080/93 $6.00 + .00

ORIGINAL CONTRIBUTION

Recognition of Manipulated Objects by Motor Learning With Modular Architecture Networks

HIROAKI GOMI AND MITSUO KAWATO

ATR Auditory and Visual Perception Research Laboratories, Kyoto

(*Received* 20 *February* 1992; *revised and accepted* 27 *August* 1992)

Abstract—*For recognition and control of multiple manipulated objects, we present two learning schemes for neural-network controllers based on feedback-error-learning and modular architecture. In both schemes, the network consists of a recognition network and modular control networks. In the first scheme, a Gating Network is trained to acquire object-specific representations for recognition of a number of objects (or sets of objects). In the second scheme, an Estimation Network is trained to acquire function-specific, rather than object-specific, representations which directly estimate physical parameters. Both recognition networks are trained to identify manipulated objects using somatic and/or visual information. After learning, appropriate motor commands for manipulation of each object are issued by the control networks which have a modular structure. By simulation of simple examples, the potential advantages and disadvantages of the two schemes are examined.*

Keywords—Modular architecture, Object manipulation, Feedback-error-learning, Gaussian mixture, Multimodal control, Somatosensory information.

1. INTRODUCTION

In previous studies of the adaptive/learning motor control using a neural network model, Barto, Sutton, and Anderson (1983), Jordan (1988), and Psaltis, Sideris, and Yamamura (1987) addressed the problem of how to obtain the error signal for a neural network feedforward controller. In supervised learning (Barto, 1989), the difference between the desired response and the actual response cannot directly be used as the error for controller adaptation. The error for controller adaptation should not be trajectory error (plant performance error) but command error.

As one possible solution to this problem, Kawato, Furukawa, and Suzuki (1987) proposed a learning method to acquire a feedforward controller, which uses the output of a feedback controller as the error for training a neural network model. They called this learning method *feedback-error-learning*. Using this method, the neural network model for feedforward control acquires an inverse dynamics model of a controlled object. They successfully applied the feedback-error-learning scheme to several objects (Miyamoto, Kawato, Setoyama, & Suzuki, 1988; Kawato, 1990; Katayama & Kawato, 1991). Additionally, Kawato (1990) clarified the differences between several methods for training the neural network feedforward controller.

However, conventional neural network feedforward controllers (Barto et al., 1983; Psaltis et al., 1987; Kawato et al., 1987; Kawato, 1990; Jordan, 1988; Katayama & Kawato, 1991) cannot cope with multiple manipulated objects or disturbances because they cannot immediately adjust the control laws corresponding to several different objects. In interaction with manipulated objects, or, in more general terms, in interaction with an environment which contains unpredictable factors, feedback information is essential for control and object recognition. From these considerations, Gomi and Kawato (1990) have examined the adaptive-feedback-controller learning schemes using feedback-error-learning, from which *impedance control* (Hogan, 1985) can be obtained automatically. However, in that scheme, some higher system needs to supervise the setting of the appropriate mechanical impedance for each manipulated object or environment.

In this paper, we introduce semi-feedforward control schemes using neural networks which receive feedback and/or feedforward information for recognition of

We would like to thank Drs. E. Yodogawa and K. Nakane of ATR Auditory and Visual Perception Research Laboratories for their continuing encouragement. This work was supported by HFSP Grant to M.K.

Requests for reprints should be sent to Hiroaki Gomi, ATR Human Information Processing Research Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan.

multiple manipulated objects based on both feedback-error-learning and modular network architecture. These new schemes have two advantages over previous ones; (a) Learning is achieved without an exact target motor command vector, which is unavailable during supervised motor learning; and (b) Although somatic information alone was found to be sufficient for recognizing objects, object identification is predictive and more reliable when both somatic and visual information are used.

2. RECOGNITION OF MANIPULATED OBJECTS

The most important issues in object manipulation are (a) how to recognize the manipulated object, and (b) how to achieve uniform performance for different objects. For the first issue, there might be several ways to acquire helpful information for recognizing manipulated objects. Visual information and somatic information (i.e., signals obtained from motor commands and performance outcomes during execution of movements) are most informative for object recognition for manipulation. For the second issue, the motor commands should be changed in order to produce the same performance for different objects. For example, when I lift a cup from a table to my mouth, my muscle-motor commands are adjusted according to the quantity in the cup. If appropriate motor commands are not sent, I am likely to spill some of the contents. When manipulating a particular object, therefore, the manipulated object should be clearly recognized by integrating several kinds of information, and the appropriate motor command should be generated based on the characteristics of the recognized object (i.e., dynamical properties and kinematical properties, etc.) to achieve good performance.

How should the abilities to recognize objects and to generate commands be obtained in motor learning for object manipulation? The physical characteristics useful for object manipulation, such as mass, softness, slipperiness, and center of gravity, cannot be predicted without prior experience in manipulating similar objects. And we do not know what kind of information is important for a particular manipulation task for which there is no experience. In this respect, object recognition for manipulation should be learned through object manipulation. And likewise, the ability to generate proficient motor commands to achieve uniform performance for different objects should also be simultaneously acquired through object manipulation.

Manipulated object recognition has also been investigated in robotics research. In many cases, the object recognition problem is usually defined as a 3D-shape reconstruction of the object by using vision and/or tactile sensing (Allen, 1987). 3D-shape representation is heuristically adopted as the internal representation for manipulation planning. Of course 3D-shape representation of objects is sufficiently available for manipulation planning, but it is still an open question whether this internal representation is effective for human-like manipulation or not. Furthermore, object recognition problems in their statement are usually discussed separately from control problems (i.e., motor command generation problems) in which recognized properties should be utilized. In the most of these cases, straightforward calculations are done to get some control-parameters, such as grasping points, hand shape, and input forces for each grasping point, taking into consideration the dynamic and kinematic properties such as the object shape, center of gravity, and surface conditions, etc. For computational reasons it is difficult to effectively change the internal representations of objects by taking account of the final performance error.

We will now examine the computational models of manipulation learning as combined recognition and control problems in the following sections.

3. MODULAR ARCHITECTURE USING A GATING NETWORK

Jacobs, Jordan, and Barto (1990), Jacobs and Jordan (1991), Nowlan (1990), and Nowlan and Hinton (1991) have proposed a competitive modular network architecture which was applied to the task decomposition or classification problems. Jacobs and Jordan (1991) applied this network architecture to a multi-payload robotics task in which each expert network controller was trained for a different category of manipulated objects in terms of object mass. In their scheme, the payload's identity was fed to the gating network in order to select a suitable expert network which acts as a feedforward controller.

We examined the modular network architecture using feedback-error-learning for simultaneous learning of object recognition and control task as shown in Figure 1. In our study cases, several physical properties, not only mass, but also viscosity and stiffness, are different for each object. In this learning scheme, the quasi-target vector for the combined output of the expert networks is employed, instead of the exact target vector. This is because it is unlikely that the exact target motor command vector can be provided in supervised motor learning (Kawato, 1990; Jordan, 1988). The quasi-target vector for feedforward motor command, u', is given by:

$$u' = u + u_{fb}, \tag{1}$$

where u denotes the motor command fed to the controlled object and u_{fb} denotes the feedback motor command. Because u_{fb} approximately represents the degree of error in the motor command vector, u, we represented the quasi-target motor command vector as shown in eqn (1). Using this quasi-target vector, the

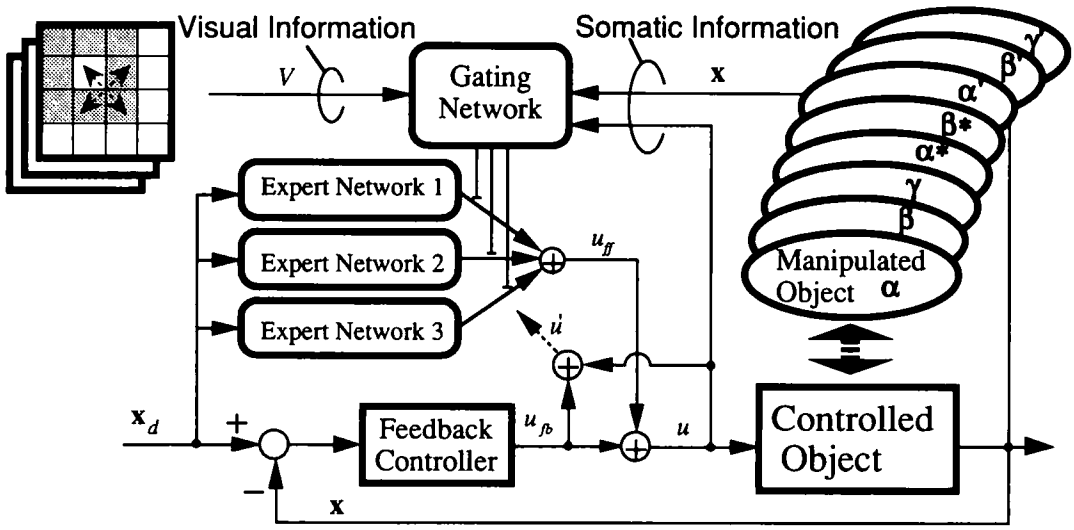

FIGURE 1. Configuration of the modular architecture using a Gating Network for object manipulation based on feedback-error-learning.

gating and expert networks are trained to maximize the log-likelihood function, ln L, by using back propagation.

$$\ln L = \ln \sum_{i=1}^{n} g_i e^{-\|u'-u_i\|^2/2\sigma_i^2} \tag{2}$$

Here, u_i is the i-th expert network output, σ_i is a variance scaling parameter for the i-th expert network, and g_i is the i-th output of the gating network, which is calculated by

$$g_i = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}}, \tag{3}$$

where s_i denotes the weighted input received by the i-th output unit. The total output of the modular network, u_{ff}, is

$$u_{ff} = \sum_{i=1}^{n} g_i u_i. \tag{4}$$

The adaptation rules for the weights in each network are derived from the partial derivative of eqn (2) with respect to each weight, and are represented as follows.

$$\frac{dw_{\text{gate}}}{dt} = \eta_{\text{gate}} \sum_{i=1}^{n} \frac{\partial s_i}{\partial w_{\text{gate}}} (g(i|X, u') - g_i)$$

$$\frac{dw_{\text{expert}\,i}}{dt} = \eta_{\text{expert}\,i} \frac{\partial u_i}{\partial w_{\text{expert}\,i}} g(i|X, u') \frac{(u' - u_i)}{\sigma_i^2} \tag{5}$$

where w_{gate} denotes the gating network weight, $w_{\text{expert}\,i}$ denotes the i-th expert network weight, η_{gate}, $\eta_{\text{expert}\,i}$ determine the learning rate in each network and $g(i|X, u')$ is the following equation corresponding to posterior probability.

$$g(i|X, u') = \frac{g_i e^{-\|u'-u_i\|^2/2\sigma_i^2}}{\sum_{j=1}^{n} g_j e^{-\|u'-u_j\|^2/2\sigma_j^2}}, \tag{6}$$

where X denotes the input vector of the gating network. By maximizing eqn 2 using the steepest ascent method (i.e., adaptation rule eqn (5)), the gating network learns to select the expert network whose output is closest to the quasi-target command, and each expert network is tuned correctly when it is chosen by the gating network. The desired trajectory is fed to the expert networks so as to make them work as feedforward controllers.

4. SIMULATION OF OBJECT MANIPULATION BY MODULAR ARCHITECTURE WITH A GATING NETWORK

We used simulation to demonstrate the efficacy of the learning schemes presented above. The configuration of a controlled and manipulated object is shown in Figure 2. M, B, K denote the mass, viscosity, and stiffness, respectively, of the coupled object (controlled- and manipulated-object). The manipulated object is changed every epoch (1 [sec]), while the coupled object is controlled to track the desired trajectory. Figure 3 shows (a) the selected object in each epoch, (b) the feedfor-

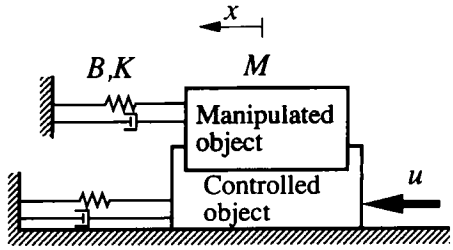

FIGURE 2. Configuration of a controlled object and a manipulated object.

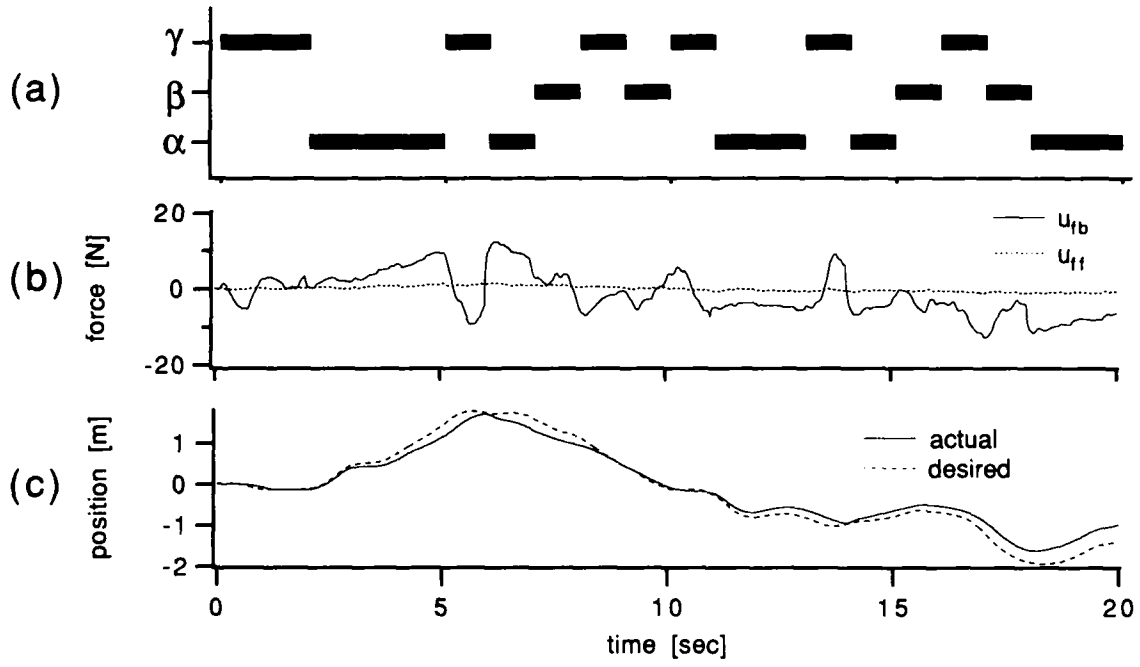

FIGURE 3. Temporal patterns of (a) objects, (b) motor commands, and (c) desired and actual trajectories before learning.

ward and feedback motor commands, and (c) the desired and actual trajectories, before learning.

The desired trajectory vector, $\mathbf{x}_d$, which consists of acceleration, velocity, and position, was produced by using the Ornstein-Uhlenbeck random process. As shown in Figure 3, the error between the desired trajectory and the actual trajectory was not eliminated because the feedback controller with fixed gains was used. The physical characteristics, M, B, K, of the objects used are listed in the second column of Figure 6.

4.1. Somatic Information for Gating Network

We call the actual trajectory vector, $\mathbf{x}$, (this vector consists of acceleration, velocity, and position) and the final motor command, u, "somatic information." Somatic information is most useful for on-line (feedback) recognition of the dynamical characteristics of manipulated objects. We used somatic information from the last four sampling time period (sampling period is 2 [msec]) as the gating network inputs for identification of the coupled object in this simulation. Then, s in eqn (3) is expressed as:

$$s(t) = \psi_1(\mathbf{x}(t), \mathbf{x}(t-1), \mathbf{x}(t-2), \mathbf{x}(t-3), u(t), u(t-1), u(t-2), u(t-3)), \quad (7)$$

where $\mathbf{x}(t)$ denotes the actual trajectory vector at time t and $u(t)$ denotes the final motor command at time t. The task of the gating network is to divide the 16-dimensional input space into three subsets for each object. The physical characteristics, M, B, K, of the coupled objects are listed in Figure 6. The object was changed every epoch (1 [sec]). The variance scaling parameter in eqn (2) was $\sigma_i = 0.8$ and the learning rates in eqn (5) were $\eta_{\text{gate}} = 1.0 \times 10^{-3}$, and $\eta_{\text{expert}\,i} = 1.0 \times 10^{-5}$. A three-layered feedforward neural network (input 16, hidden 30, output 3) was employed for the gating network, and two-layered linear networks (input 3, output 1) were used for the expert networks.

Figure 4 shows the time courses of the moving average of u_{fb} and u_{ff} squared, and Figures 5(a)–(c) show the time courses of weights in each expert network during the learning phase. The feedback motor command gradually decreased during learning and the expert network weights (i.e., gains for each input) approach asymptotic values. Comparing the expert network weights for each input after learning listed in the third row of Figure 6 and the actual physical characteristics of the coupled objects listed in the second column of Figure 6, we realize that expert networks No. 1, No. 2, and No. 3 obtained the inverse dynamics of coupled

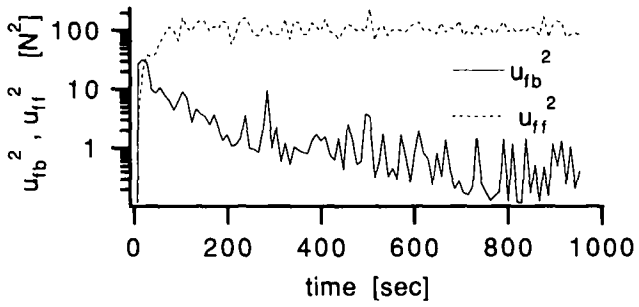

FIGURE 4. The moving average over time of the squared motor commands.

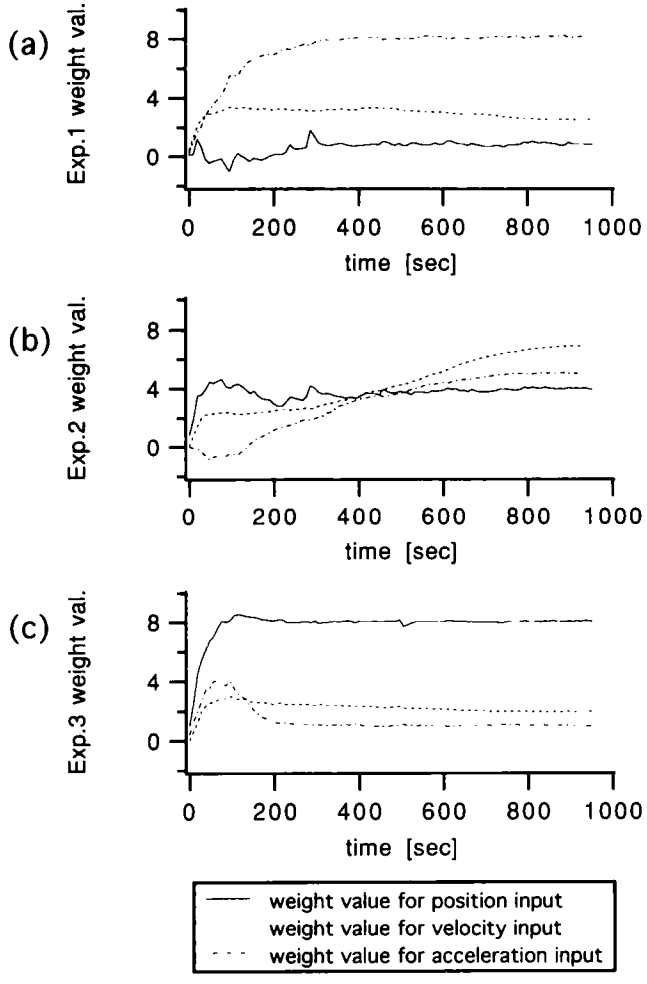

FIGURE 5. The time courses of Expert network weights during learning when somatic information was used as the gating network input.

objects γ, β, and α, respectively, which were employed during the learning phase.

Figure 7 shows (a) the time variation of the objects, (b) the gating network outputs, (c) the motor commands, and (d) the trajectories, after learning. The gating network outputs for the objects responded correctly throughout most of the test phase (compare Figure 7(a) with 7(b)). In the several parts of the test phase, however, results were poor. For example, even though the correct gating network outputs were obtained in the latter half of the test phase, gating network outputs g_1 and g_2 at around 1 [sec] and 4.5 [sec] were wrong. This is because the learning for the gating network was not completely successful. In other words, the correct gating network outputs were learned for the gating network input vector in the latter half of the test phase but were not learned for the input vectors in the first half of the test phase. The statistical correspondences are shown in Figure 6. Each bar height in Figure 6 denotes the averaged value of each gating network output while each object was selected during the test phase. We can see the ratio of the correct to the wrong gating network output during the test phase from each row in this figure.

In spite of the unsatisfactory discrimination results, the value of the feedback motor command, u_{fb}, was almost zero and the actual trajectory almost perfectly corresponded to the desired trajectory. In other words, the gating network successfully executed the "recognition task for manipulation" and each expert network acquired the "internal model of an object for manipulation." Moreover, learning generalization in a "compact state space" was almost ascertained because the simulation was done for a random desired trajectory (O-U process). In the object recognition process using the somatic information shown above, gating network required complicated calculations. By using a limited trajectory or by using a network which has more capacity, better discrimination results might be obtained. On the other hand, more satisfactory results were obtained by using visual information, as described below, because visual cues are very simple and there are only a few patterns.

4.2. Visual Information for Gating Network

Consciously or unconsciously, we usually make some assumption about the manipulated object's characteristics by using visual information before we actually manipulate the object. Visual information might be quite helpful for feedforward recognition. Object discrimination tasks by using modular networks from visual images were investigated by Jacobs (1991) in which teacher signals (i.e., target object identities) were employed in training.

When visual information is available in the proposed scheme, s in eqn (3) is expressed as:

$$s(t) = \psi_2(V(t)), \tag{8}$$

where $V(t)$ denotes the retinal matrix values at time t. We used three simple visual cues corresponding to each coupled object in this simulation as shown in Figure 8. At each epoch in this simulation, one of three visual cues of the size 3×3 is selected randomly and then randomly placed at one of four possible locations on a 4×4 retinal matrix. Each black pixel of these cues takes the value one and each white pixel on the retinal matrix takes the value zero. The value set of $V(t)$ was fixed during each epoch. The visual cues for each object are different, but objects α and α^* have the same physical characteristics, M, B, K, as listed in the second column of Figure 8. The gating network should identify the object and select a suitable expert network for feedforward control by using this visual information. The learning coefficients were $\sigma_i = 0.7$, $\eta_{\text{gate}} = 1.0 \times 10^{-3}$, and $\eta_{\text{expert}\,i} = 1.0 \times 10^{-5}$. The same networks used in the above experiment were used in this simulation.

	Object physical characteristics M B K	retinal image	Expert Net. Weight values for each input, $\ddot{x}_d$ $\dot{x}_d$ x_d, after learning		
			No.1 $\ddot{x}_d$ $\dot{x}_d$ x_d	No.2 $\ddot{x}_d$ $\dot{x}_d$ x_d	No.3 $\ddot{x}_d$ $\dot{x}_d$ x_d
			8.1 2.5 0.87	5.0 6.9 4.0	0.97 1.9 8.0
α	1.0 2.0 8.0	none			
β	5.0 7.0 4.0	none			
γ	8.0 3.0 1.0	none			

FIGURE 6. Gating Network outputs vs. objects using Somatic information. The second column shows the physical characteristics of each object, and the third row shows the acquired parameter values for the inputs to each expert network. The bar height denote the averaged gating outputs in the test phase after learning.

After learning, expert network No. 2 acquired the inverse dynamics of objects α and α^* (which have the same physical characteristics), and expert network No. 3 accomplished this for object γ (compare the object physical characteristics with the expert network weights for each input listed in Figure 8). Figure 8 summarizes the statistical analysis of the correspondence between the objects and the gating network output. The gating network almost always selected expert network No. 2 for object α and α^*, and almost always selected expert

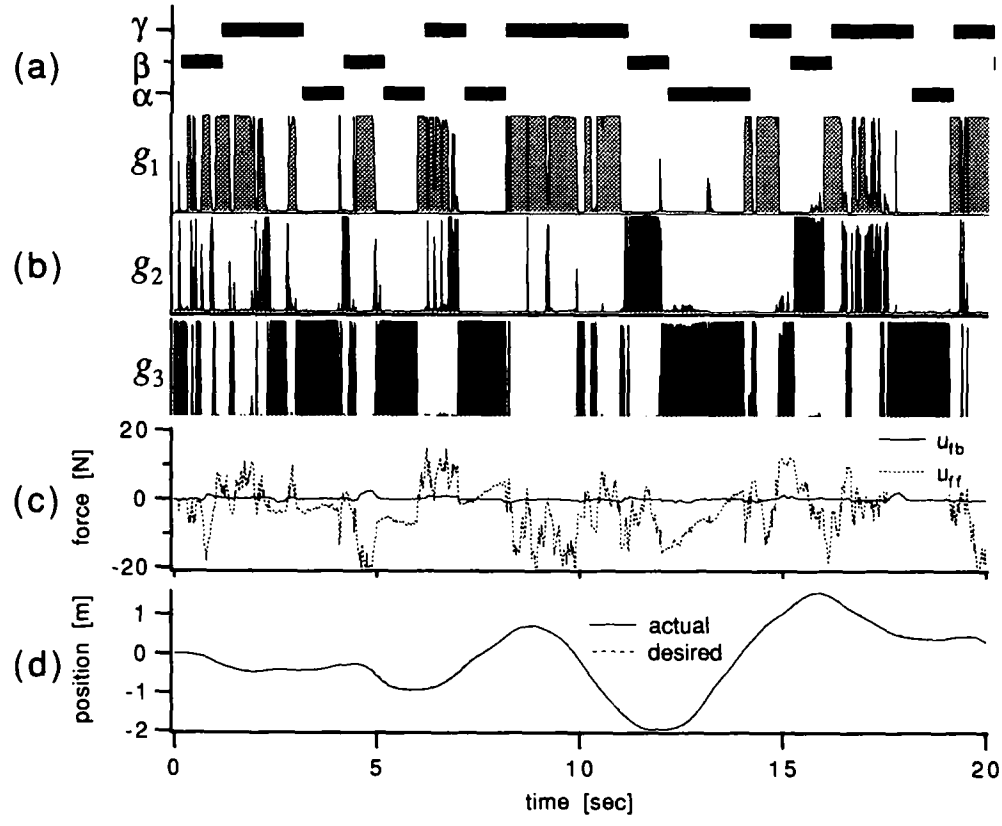

FIGURE 7. Temporal patterns of (a) objects, (b) gating outputs, (c) motor commands, and (d) trajectories after learning using Somatic information.

	Object physical characteristics M B K	retinal image (randomly moved)	Expert Net. Weight values for each input, $\ddot{x}_d$ $\dot{x}_d$ x_d, after learning		
			No.1 $\ddot{x}_d$ $\dot{x}_d$ x_d	No.2 $\ddot{x}_d$ $\dot{x}_d$ x_d	No.3 $\ddot{x}_d$ $\dot{x}_d$ x_d
			4.3 3.0 -0.34	1.2 2.0 8.0	8.0 3.0 0.99
α	1.0 2.0 8.0			■	
α^*	1.0 2.0 8.0			■	
γ	8.0 3.0 1.0				■

FIGURE 8. Gating Network outputs vs. objects using Visual information. (Same notation with Figure 6.)

network No. 3 for object γ, as shown in Figures 8 and 9. Expert network No. 1, which did not acquire inverse dynamics corresponding to any of the three objects, was not selected in the test phase after learning. That is to say, the redundant expert network did not contribute to control tasks by learning as mentioned by Jacobs (1990). The actual trajectory in the test phase corresponded almost exactly to the desired trajectory.

4.3. Somatic and Visual Information for Gating Network

We show here the simulation results using both somatic and visual information as the gating network inputs. In this case, s of eqn (3) is represented as:

$$s(t) = \psi_3(\mathbf{x}(t), \ldots, \mathbf{x}(t-3), u(t), \ldots, u(t-3), V(t)). \quad (9)$$

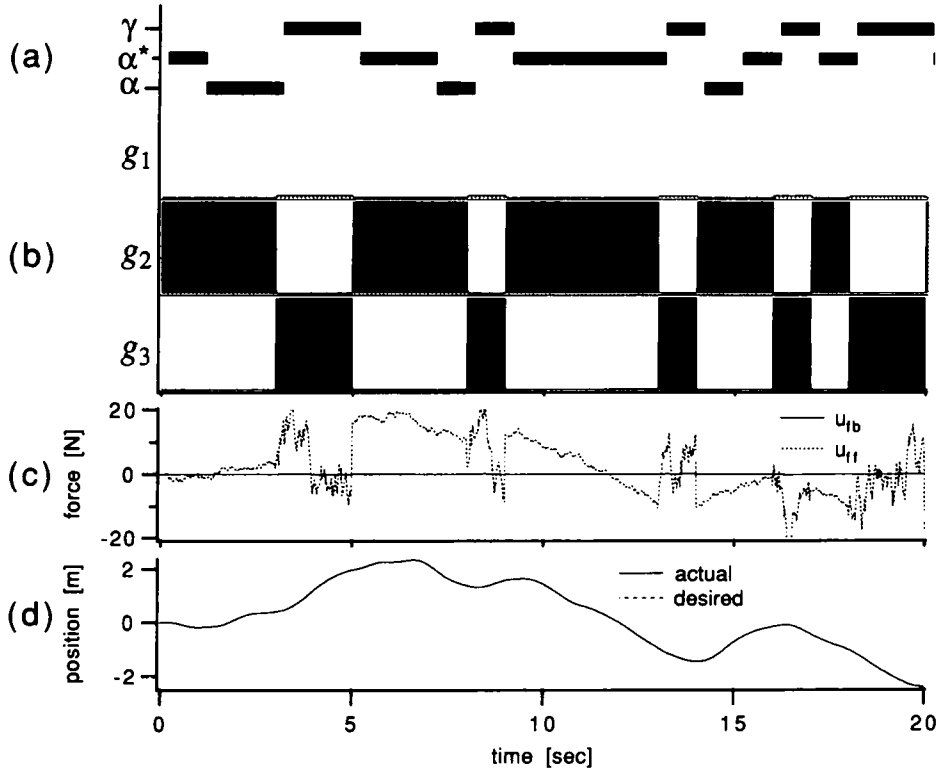

FIGURE 9. Temporal patterns of (a) objects, (b) gating outputs, (c) motor commands, and (d) trajectories after learning.

	Object physical characteristics M B K	retinal image (randomly moved)	Expert Net. Weight values for each input, $\ddot{x}_d$ $\dot{x}_d$ x_d, after learning		
			No.1 $\ddot{x}_d$ $\dot{x}_d$ x_d	No.2 $\ddot{x}_d$ $\dot{x}_d$ x_d	No.3 $\ddot{x}_d$ $\dot{x}_d$ x_d
			8.1 2.4 0.8	5.1 6.9 4.0	1.0 1.9 8.0
α	1.0 2.0 8.0				
β^*	5.0 7.0 4.0				
γ	8.0 3.0 1.0				

FIGURE 10. Gating Network outputs vs. objects using Somatic and Visual information. (Same notation with Figure 6.)

In this simulation, the objects α and β^* had different physical characteristics but shared the same visual cue as listed in Figure 10. Thus, to identify the coupled objects one by one, it is necessary for the gating network to utilize not only visual information but also somatic information. The learning coefficients were $\sigma_i = 1.0$, $\eta_{\text{gate}} = 1.0 \times 10^{-3}$, and $\eta_{\text{expert}\,i} = 1.0 \times 10^{-5}$. The gating network had 32 input units, 50 hidden units, and 3 output units, and the expert networks were the same as in the above experiment.

After learning, expert networks No. 1, No. 2, and No. 3 acquired the inverse dynamics of objects γ, β^*, and α, respectively, as listed in Figure 10. As shown in Figure 11, the gating network almost always identified the object correctly. As shown in the statistical analysis in Figure 10, the recognition performance (the average

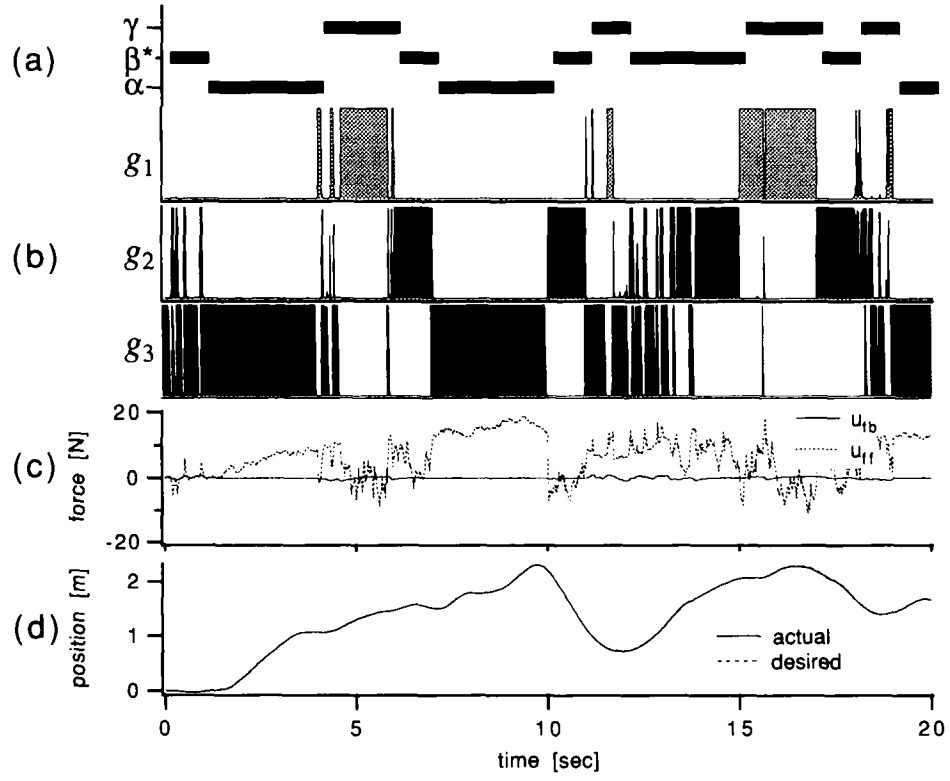

FIGURE 11. Temporal patterns of (a) objects, gating (b) outputs, (c) motor commands, and (d) trajectories after learning using Somatic and Visual information.

value of the gating network output to each expert network for each object) is better than that in Section 4.1. The recognition performance for object γ, however, was not as good as the result in Section 4.2, even though visual information was also available for recognizing object γ. This might be because the sensory signals, visual and somatic, were not normalized or slanted properly in this simulation.

4.4. Unknown Object Recognition by Using Somatic Information

It is possible that the scheme shown in Figure 1 might also be applied to unknown objects because the gating network switches between the expert networks not crisply but fuzzily. This is because the desired output of the gating network has Gaussian distribution, as shown in eqn (6). To examine this, we applied the modular network trained in the experiment described in Section 4.1 to unknown objects. Figure 13 shows the temporal responses for unknown objects whose physical characteristics were slightly different from known objects (the different parameters are shown in Figures 6 and 12), using somatic information as the gating network inputs. Even though each tested object was not exactly the same as any of the known (learned) objects, the gating network selected the expert network whose inverse dynamics model was the closest to the unknown object's (compare Figure 12 with Figure 6). For object α', expert network No. 3 was chosen with high probability and for object β', expert network No. 2 was selected with a higher probability than the other two expert networks. For object γ', expert network No. 1 was selected most often. As shown in the middle part of Figure 13, during some period in the test phase, the feedback command increased because of an inappropriate feedforward command.

5. MODULAR ARCHITECTURE USING ESTIMATION NETWORK

The modular architecture shown in the above section is competitive in the sense that expert networks compete with each other to occupy a niche in the input space. We propose here a new cooperative modular architecture where expert networks specialized for different functions (i.e., preprocessing for different physical parameters, not for different objects) cooperate to produce the required output. In this scheme, estimation networks are trained to recognize the physical parameters of manipulated objects by using feedback information. Using this method, an infinite number of manipulated objects in a limited domain can be handled by using a small number of estimation networks. This is because the values of each object's parameters is interpolated and estimated directly by the estimation networks, and the number of estimation networks is limited to the number of effective parameters necessary for controlling the target objects.

Figure 14 shows the configuration for manipulation control employing this idea. In this figure, the inverse dynamics model (IDM) network is prepared for the controlled object (i.e., manipulator etc.) and expert networks No. 1, 2, and 3 are trained to represent different functional forces which correspond to different physical characteristics of manipulated objects. Each output of the expert networks is multiplied by each output of the estimation network to produce the motor commands. Each expert network works cooperatively rather than competitively. The number of the estimation

	Object physical characteristics M B K	retinal image	Expert Net. Weight values for each input, $\ddot{x}_d$ $\dot{x}_d$ x_d, after learning		
			No.1	No.2	No.3
			$\ddot{x}_d$ $\dot{x}_d$ x_d	$\ddot{x}_d$ $\dot{x}_d$ x_d	$\ddot{x}_d$ $\dot{x}_d$ x_d
			8.1 2.5 0.87	5.0 6.9 4.0	0.97 1.9 8.0
α'	2.0 3.0 7.0	none			
β'	4.0 6.0 5.0	none			
γ'	9.0 2.0 2.0	none			

FIGURE 12. Gating Network outputs vs. objects during an unknown object recognition task using Somatic information. (Same notation with Figure 6.)

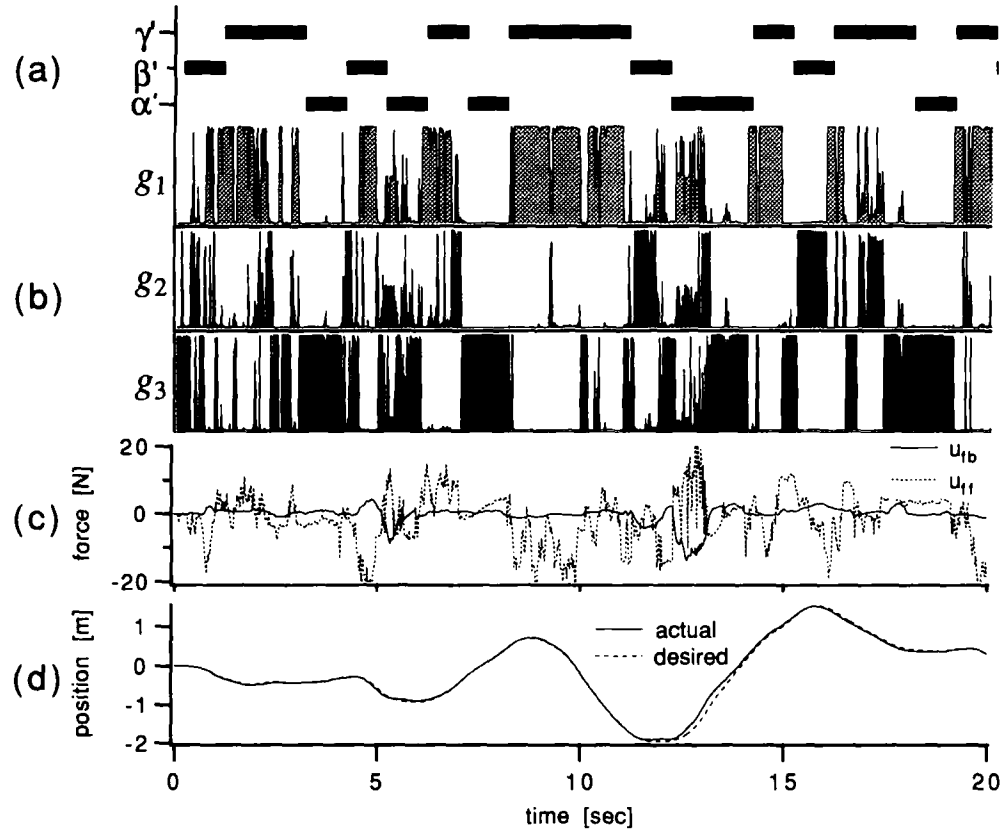

FIGURE 13. Temporal patterns of (a) objects, (b) gating outputs, (c) motor commands, and (d) trajectories during an unknown object recognition task using Somatic information.

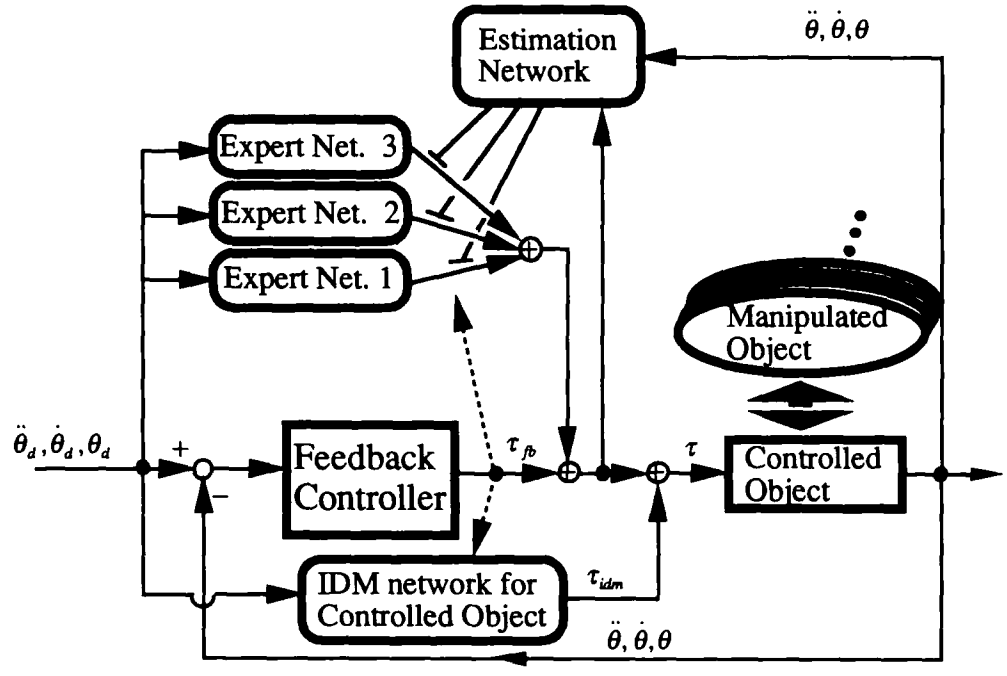

FIGURE 14. Configuration of the modular architecture using an Estimation Network for object manipulation by feedback-error-learning.

network outputs is restricted to the number of effective parameters of the manipulated objects. For example, to properly control the manipulated object which is attached at the top of the two-link manipulator shown in Figure 15, the feedforward motor command, τ_{ff}, should be expressed as follows.

$$\tau_{ff} = \mathbf{J}^T(\mathbf{M}\ddot{x}_d + \mathbf{B}\dot{x}_d + \mathbf{K}x_d) + \tau_{\text{idm}}, \quad (10)$$

where

$$\mathbf{M} = \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} B & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} K & 0 \\ 0 & 0 \end{bmatrix}.$$

$\mathbf{J}^T$ denotes a transposed Jacobian matrix, and x_d is the desired trajectory position vector in Cartesian space. If the IDM network perfectly compensates only for the controlled object, the appropriate feedforward motor command for the manipulated object (i.e., one that is produced by expert networks No. 1, 2, and 3 and the estimation network shown in Figure 14) is expressed as:

$$\tau_{mi} = M\Psi_{1i}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) + B\Psi_{2i}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) + K\Psi_{3i}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d), \quad (11)$$

where τ_{mi} denotes the feedforward motor command for the i-th link and M, B, K, respectively, denote mass, viscosity, and stiffness of the manipulated object and $\ddot{\theta}_d$, $\dot{\theta}_d$, and θ_d, respectively, denote the angular acceleration, angular velocity, and angular position vector of the manipulator. Ψ_{ji} is a preprocessing nonlinear function realized by each expert network according to the following equations for producing the appropriate feedforward motor command for the manipulated object.

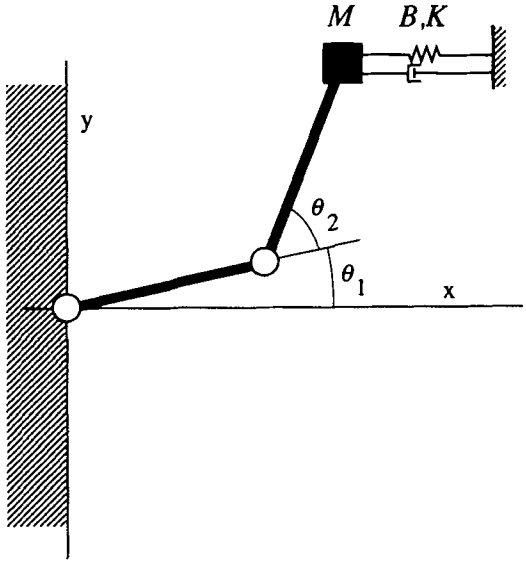

FIGURE 15. Configuration of the two-link manipulator and a manipulated object.

$$\Psi_{11}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) = \ddot{\theta}_{d1}(l_1^2 + l_2^2 + 2l_1l_2\cos\theta_{d2}) + \ddot{\theta}_{d2}(l_2^2 + l_1l_2\cos\theta_{d2}) - (2\dot{\theta}_{d1} + \dot{\theta}_{d2})\dot{\theta}_{d2}l_1l_2\sin\theta_{d2}. \quad (12)$$

$$\Psi_{21}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) = [\dot{\theta}_{d1}(l_1\sin\theta_{d1} + l_2\sin(\theta_{d1} + \theta_{d2})) + \dot{\theta}_{d2}l_2\sin(\theta_{d1} + \theta_{d2})](l_1\sin\theta_{d1} + l_2\sin(\theta_{d1} + \theta_{d2})). \quad (13)$$

$$\Psi_{31}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) = (l_1\cos\theta_{d1} + l_2\cos(\theta_{d1} + \theta_{d2})) \times (l_1\sin\theta_{d1} + l_2\sin(\theta_{d1} + \theta_{d2})). \quad (14)$$

$$\Psi_{12}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) = \ddot{\theta}_{d1}(l_2^2 + 2l_1l_2\cos\theta_{d2}) + \ddot{\theta}_{d2}l_2^2 - \dot{\theta}_{d1}^2l_1l_2\sin\theta_{d2}. \quad (15)$$

$$\Psi_{22}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) = \dot{\theta}_{d1}l_2(l_1\sin\theta_{d1} + l_2)\sin(\theta_{d1} + \theta_{d2}) + \dot{\theta}_{d2}l_2^2\sin(\theta_{d1} + \theta_{d2}). \quad (16)$$

$$\Psi_{32}(\ddot{\theta}_d, \dot{\theta}_d, \theta_d) = l_2(l_1\cos\theta_{d1} + l_2\cos(\theta_{d1} + \theta_{d2}))\sin(\theta_{d1} + \theta_{d2}). \quad (17)$$

Here, l_1, l_2, θ_{d1}, and θ_{d2}, respectively, denote the first link length, second link length, first link desired angle, and second link desired angle of the manipulator. We expect that these preprocessings are obtained for each expert network and that the parameters, M, B, K, will be compellingly represented as the estimation network outputs by using the constraint of the limited number of estimation-network outputs. This expectation comes from previous studies in which efficient and compressed representations were obtained in the hidden layer of the network when the number of hidden units was kept small (Cottrell, Monroe, & Zipser, 1987; Bourlard & Kamp, 1988; Irie & Kawato, 1990).

We applied this idea to recognizing the mass of manipulated objects in one-dimensional movement, with configurations similar to that in Figure 2. Generally, expert networks in this learning scheme are expected to acquire the preprocessing as shown in the example for the two-link manipulator control shown above. In this preliminary case, however, nonlinear preprocessing was not necessary for the expert network, which meant that we only confirmed the capability of estimation network learning in this simulation. (That is, the feedforward signal was directly multiplied by the estimation network output without any calculation at the expert network.) Figure 16(a) shows the output of the estimation network compared to actual masses. The realized trajectory almost coincides with the desired trajectory as shown in Figure 16(b). This learning scheme can be applied not only to estimating mass, but also to other physical characteristics, such as softness or slipperiness.

6. DISCUSSION

6.1. Acquisition of Task-Based Representation

In the simulation of manipulation learning using visual information (Section 4.2), the internal models for ob-

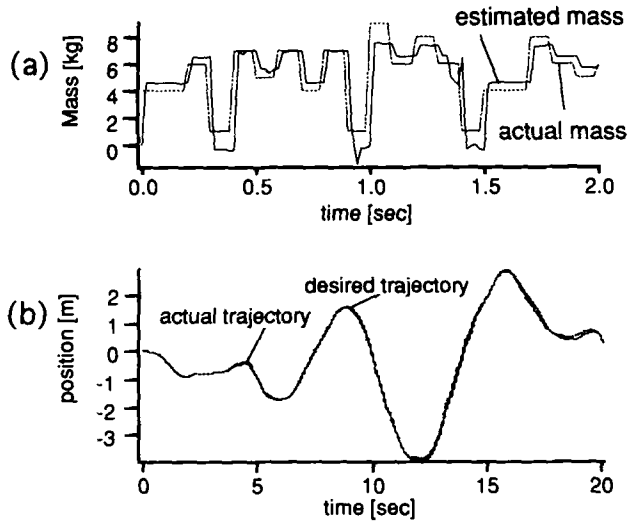

FIGURE 16. (a) Comparison of actual and estimated mass; (b) Desired and actual trajectories.

ject manipulation (in this case, inverse dynamics) were represented not in terms of visual information, but rather, of motor commands. That is to say, the same internal representation is acquired when the same motor command is required for obtaining the final performance, even if the input visual cues are different. This is good not only for saving the internal representation, but also for deriving the common features between objects in terms of motor control. Although the current simulation is preliminary, it indicates the very important issue that task-based internal representations of objects (or environments), rather than declarative ones, are automatically acquired by motor learning. This observation supports the "functional representation" notion in Artificial Intelligence research as a promising strategy in high-level recognition processes. For example, the abstract idea of "chair" cannot simply be acquired from many image examples of "chair," but task-based (i.e., functional) representation, i.e., a "chair is an object on which people can sit," makes the many different shapes of chairs comprehensible.

6.2. Convergence Rate by Feedback-Error-Learning

The quasi-target motor command in the first scheme and the motor command error in the second scheme are not exactly correct because the proposed learning schemes are based on the feedback-error-learning method. Thus, the learning rates in the proposed schemes should be slower than in those schemes in which exact target commands are employed (Gomi & Kawato, 1990). In our preliminary simulation, it was about five times slower. We emphasize that exact target motor commands, however, are not available in supervised motor learning.

6.3. Limitation of the Number of Manipulated Objects and the Number of Estimation Parameters

The limited number of controlled objects which can be dealt with by the modular architecture with a gating network is a considerable problem (Jacobs & Jordan, 1991; Nowlan, 1990; Nowlan & Hinton, 1991). This problem depends on choosing an appropriate number of expert networks and an appropriate value for the variance scaling parameter, σ. Once this is done, the expert networks can interpolate the appropriate output for a number of unknown objects. The simulation results in Section 4.4 show a successful example for unknown objects. Our second scheme provides a more satisfactory solution to this problem.

On the other hand, one possible drawback of the second scheme is that it may be difficult to estimate many physical parameters for complicated objects, even though the learning scheme which directly estimates the physical parameters can handle any number of objects.

6.4. Modular Architecture as a Structural Constraint for Network Design

Modular architecture is one of the structural constraints in neural network design for a particular task. As Jacobs et al. (1990) described, modular architecture might be a good constraint for many kinds of underconstrained learning problems. If a single neural network is used for the object recognition problem shown here, learning will not be successful because the network has too many degrees of freedom. In particular, the importance of modular architecture will increase for problems in which many kinds of tasks should be learned together.

7. CONCLUSION

We presented basic examinations of two types of modular architecture using neural networks—a gating network and a direct estimation network. Both networks can use feedback and/or feedforward information for recognition of multiple manipulated objects. In order to examine the fundamental performance of modular networks, we did not take into account the geometrical properties of manipulated objects, which are important in deciding the grasping points and forming the hand shape. In the future, we will attempt to integrate the two proposed network architectures and to advance this combined architecture with sophisticated visual processing in order to model tasks involving skilled motor coordination and high-level recognition.

REFERENCES

Allen, P. K. (1987). *Robotic object recognition using vision and touch.* New York: Kluwer Academic Publishers.

Barto, A. G. (1989). Connectionist learning for control. In T. Miller, R. S. Sutton, & P. J. Werbos (Eds.), *Neural networks for control* (pp. 5–58). Cambridge, MA: The MIT Press.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, **13**, 834–846.

Bourlard, H., & Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, **59**, 291–294.

Cottrell, G. W., Munro, P., & Zipser, D. (1987). *Image compression by back propagation: An example of extensional programming* (ICS No. 8702). Institute for Cognitive Science, University of California. San Diego, CA.

Gomi, H., & Kawato, M. (1990). Learning control for a closed loop system using feedback-error-learning. *Proceedings of the 29th IEEE Conference on Decision and Control*, Hawaii, 3289–3294.

Hogan, N. (1985). Impedance control: An approach to manipulation: Part I–Theory, Part II–Implementation, Part III–Applications. *ASME Journal of Dynamic Systems, Measurement, and Control*, **107**, 1–24.

Irie, B., & Kawato, M. (1990). *Extraction of the nonlinear global coordinate system of a manifold by a five-layered hour-glass network* (Tech. Rep. No. TR-A-0094). Kyoto, Japan: ATR Auditory and Visual Perception Research Laboratories.

Jacobs, R. A., & Jordan, M. I. (1991). A competitive modular connectionist architecture. In R. P. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing Systems* 3 (pp. 767–773). San Mateo, CA: Morgan Kaufmann Publishers.

Jacobs, R. A., Jordan, M. I., & Barto, A. G. (1990). *Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks* (COINS Tech. Rep. No. 90-27).

Jordan, M. I. (1988). *Supervised learning and systems with excess degrees of freedom* (COINS Tech. Rep. No. 88-27). Amherst, MA: University of Massachusetts.

Katayama, M., & Kawato, M. (1991). Learning trajectory and force control of an artificial muscle arm by parallel-hierarchical neural network model. In R. P. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems* 3 (pp. 436–442). San Mateo, CA: Morgan Kaufmann Publishers.

Kawato, M. (1990). Computational schemes and neural network models for formation and control of multijoint arm trajectory. In T. Miller III, R. S. Sutton, & P. J. Werbos (Eds.), *Neural networks for control* (pp. 192–228). Cambridge, MA: The MIT Press.

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

Miyamoto, H., Kawato, M., Setoyama, T., & Suzuki, R. (1988). Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural Networks*, **1**, 251–265.

Nowlan, S. J. (1990). *Competing experts: An experimental investigation of associative mixture models* (Tech. Rep. No. CRG-TR-90-5). Toronto, Canada: University of Toronto.

Nowlan, S. J., & Hinton, G. E. (1991). Evaluation of adaptive mixtures of competing experts. In R. P. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems* 3 (pp. 774–780). San Mateo, CA: Morgan Kaufmann Publishers.

Psaltis, D., Sideris, A., & Yamamura, A. (1987). Neural controllers. In *Proceedings of the IEEE International Conference of Neural Networks*, **4**, 551–557.

NOMENCLATURE

u'	quasi-target vector of feedforward motor command
u	motor command
u_{fb}	feedback motor command
u_i	i-th expert network output
σ_i	variance scaling parameter of the i-th expert network
g_i	i-th output of the gating network
s_i	weighted input received by the i-th output unit
u_{ff}	total output of the modular network
X	input vector of the gating network
w_{gate}	gating network weight
$w_{\text{expert } i}$	i-th expert network weight
η_{gate}, $\eta_{\text{expert } i}$	learning rate in each network
$g(i \mid X, u')$	posterior probability
M, B, K	mass, viscosity and stiffness, of the object
Ψ_i, Ψ_{ji}	nonlinear function
$\mathbf{x}_d$	desired trajectory vector which consists of acceleration, velocity, and position
$\mathbf{x}$	actual trajectory vector which consists of acceleration, velocity, and position
V	retinal matrix value set
α, α^*, α', β, β^*, β', γ, γ'	object name
$\mathbf{J}^T$	transposed Jacobian matrix
x_d	desired trajectory position vector in Cartesian space
$\mathbf{M}$, $\mathbf{B}$, $\mathbf{K}$	inertia matrix, viscosity matrix, and stiffness matrix
τ_{mi}	feedforward motor command for i-th link
$\ddot{\theta}_d$, $\dot{\theta}_d$, θ_d	angular acceleration, angular velocity, and angular position vector of the manipulator
l_1, l_2	first link length, and second link length of the manipulator
θ_{d1}, θ_{d2}	first link desired angle, and second link desired angle of the manipulator.