Deep Predictive Model Learning with Parametric Bias: Handling Modeling Difficulties and Temporal Model Changes

Kento Kawaharazuka[1], Kei Okada[1], and Masayuki Inaba[1]

***Abstract*— When a robot executes a task, it is necessary to model the relationship among its body, target objects, tools, and environment, and to control its body to realize the target state. However, it is difficult to model them using classical methods if the relationship is complex. In addition, when the relationship changes with time, it is necessary to deal with the temporal changes of the model. In this study, we have developed Deep Predictive Model with Parametric Bias (DPMPB) as a more human-like adaptive intelligence to deal with these modeling difficulties and temporal model changes. We categorize and summarize the theory of DPMPB and various task experiments on the actual robots, and discuss the effectiveness of DPMPB.**

I. INTRODUCTION

The purpose of a robot is to realize the target task state when given a certain task by manipulating its own body, target objects, and tools. This is possible by modeling and applying the relationships among its body, target objects, tools, and environment. For example, cloth manipulation can be achieved by knowing how to move the arm to change the state of the grasped cloth. Also, balance control can be achieved by knowing how to apply force to the ankle to change the zero moment point. However, there are two major problems here: modeling difficulties and temporal model changes.

First, when the relationship among the body, tools, target objects, and environment is complex, it is difficult to model using classical methods. Most classical methods in the past have dealt with robots that are rigid and easy to model, or tools and objects that can be approximated by rigid bodies [1], [2]. However, nowadays, robots with flexible bodies [3] and tasks that handle flexible objects [4] are attracting attention, and it is necessary to develop methods that can be applied to them. Second, when the relationship among the body, tools, target objects, and environment changes with time, it is necessary to deal with the temporal changes in the model. In the past, target objects, tools, and environment were mostly fixed, but now robots are expected to be deployed in more complex environments where they change dynamically [5], [6]. In addition, the nature of flexible robots makes them prone to physical changes due to aging and other factors, and they should be able to cope with the temporal model changes caused by these physical changes. Robots need a learning system resembling human adaptive intelligence that allows them to cope with these problems in the real world.

[1] The authors are with the Department of Mechano-Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan. [kawaharazuka, k-okada, inaba]@jsk.t.u-tokyo.ac.jp

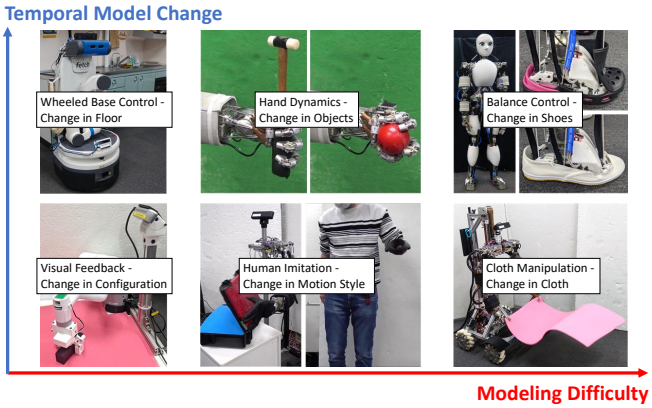

Fig. 1. The developed deep predictive model with parametric bias (DPMPB) can handle various modeling difficulties and temporal model changes.

Therefore, we have developed Deep Predictive Model with Parametric Bias (DPMPB) to cope with these modeling difficulties and temporal model changes [7]–[12]. This predictive model describes the correlation between sensors $\boldsymbol{s}$ and actuators $\boldsymbol{u}$ with a neural network. This learning-based modeling makes it possible to cope with modeling difficulties. By using this predictive model, the robot body is controlled by its forward propagation or by iterative backpropagation from an appropriate loss function to the network input. It is also possible to detect anomalies based on the prediction error of the model.

In order to cope with temporal model changes in such learning-based modeling, we apply parametric bias [13], which can implicitly embed multiple attractor dynamics. Parametric bias is a learnable input variable, and dynamics information is embedded in this variable so that various data transitions with different dynamics can be represented in a single model. This has been used mainly in the context of cognitive robotics with imitation learning [14]. By applying this approach to predictive model learning, it is possible to implicitly embed temporal changes in the body, tools, target objects, and environment. In addition, unlike [13], we develop a mechanism that autonomously updates the parametric bias online, after which the control changes accordingly. This will enable the system to recognize and adapt to the current state of the body, tools, target objects, and environment.

Reinforcement learning [15] and imitation learning [16] are commonly considered as control methods to deal with modeling difficulties. Reinforcement learning is a method that can acquire controllers through autonomous learning based on rewards. While it is mainly applied to simula-

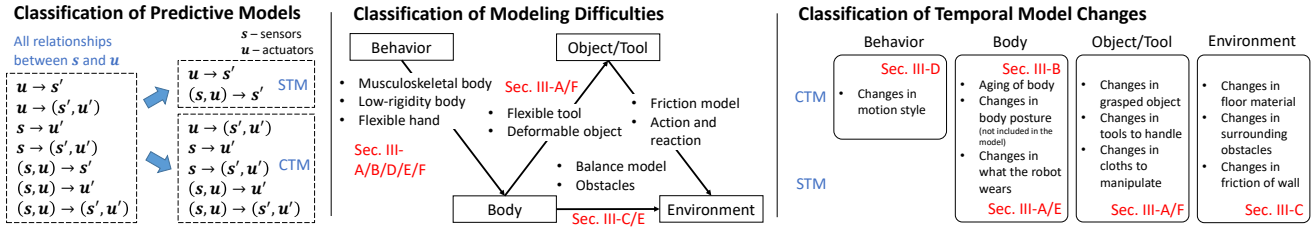

Fig. 2. The classification of predictive models, modeling difficulties, and temporal model changes. The predictive models are classified by the network input and output of the values of sensors $\boldsymbol{s}$ and actuators $\boldsymbol{u}$. The modeling difficulties are classified by the relationship among robot behavior, body, object/tool, and environment. The temporal model changes are classified by the network structure of CTM or STM and by robot behavior, body, object/tool, and environment.

tions where the number of trials can be increased, efficient reinforcement learning methods that can be learned on actual robots have also been developed [17]. Imitation learning is a method where a robot learns to imitate a human demonstration, and is a type of predictive model that is handled in this study. In addition, a predictive model representing state transition has been developed [18], which is also handled in this study. On the other hand, there are very few cases in which the problem of temporal model changes is explicitly addressed in the learning methods that deal with these modeling difficulties. All of reinforcement learning studies deal with the temporal model changes implicitly by learning in various environments, but few of them deal with the temporal changes of bodies, objects, and tools. Predictive model learning studies so far basically do not deal with the changes in bodies, tools, and environments [19], and the robot is controlled with the fixed dynamics from the time it was trained. There is also a method using online learning of neural networks to adapt to the current body and environment [20], but it requires a large amount of data to relearn the entire network, losing its applicability to other bodies and environments. By introducing parametric bias, changes in the body and environment can be embedded in small dimensional variables, which can be updated online to adapt to the current body, tool, and environment quickly without destroying the dynamics of the entire network. In addition, since the changes in the body and environment can be taken into account in the model as explicit variables, they can be applied to the recognition of the body and environment, thus expanding the range of possible tasks.

In this study, we introduce parametric bias in predictive model learning, and discuss how to cope with the modeling difficulties and temporal model changes based on this approach. We summarize the actual robot examples using predictive model learning that we have developed so far [7]–[12], and integrate the methods into a single unified theory, DPMPB. The predictive model learning is classified into state transition model type and control transition model type, and these types are identified with the collected dataset. Modeling difficulties and temporal model changes in the predictive model learning are classified and summarized. The theory of DPMPB includes data collection, network training, online adaptation, control, and anomaly detection in the form of forward and backward propagation of inputs and outputs of the constructed network. We also describe a concrete implementation of our system with necessary parameters specified. Based on this unified theory, we show that experiments on actual robot tasks with various modeling difficulties and temporal model changes can be comprehensively achieved by simply changing parameters and training data in exactly the same form (Fig. 1).

II. Deep Predictive Model with Parametric Bias

A. Classification of Predictive Models, Modeling Difficulties, and Temporal Model Changes

1) Classification of Deep Predictive Models: First, we classify the predictive models based on the relationship between sensors $\boldsymbol{s}$ and actuators $\boldsymbol{u}$ (the left figure of Fig. 2). There are seven possible time evolution relationships between $\boldsymbol{s}$ and $\boldsymbol{u}$: $\{\boldsymbol{u} \to \boldsymbol{s}', \boldsymbol{u} \to (\boldsymbol{s}', \boldsymbol{u}'), \boldsymbol{s} \to \boldsymbol{u}', \boldsymbol{s} \to (\boldsymbol{s}', \boldsymbol{u}'), (\boldsymbol{s}, \boldsymbol{u}) \to \boldsymbol{s}', (\boldsymbol{s}, \boldsymbol{u}) \to \boldsymbol{u}', (\boldsymbol{s}, \boldsymbol{u}) \to (\boldsymbol{s}', \boldsymbol{u}')\}$ ($\{\boldsymbol{s}', \boldsymbol{u}'\}$ represents $\{\boldsymbol{s}, \boldsymbol{u}\}$ at the next time step). The five that contain $\boldsymbol{u}$ in the output, $\{\boldsymbol{u} \to (\boldsymbol{s}', \boldsymbol{u}'), \boldsymbol{s} \to \boldsymbol{u}', \boldsymbol{s} \to (\boldsymbol{s}', \boldsymbol{u}'), (\boldsymbol{s}, \boldsymbol{u}) \to \boldsymbol{u}', (\boldsymbol{s}, \boldsymbol{u}) \to (\boldsymbol{s}', \boldsymbol{u}')\}$, have the network structure of control transition model (CTM) used in imitation learning, in which the motion of the next time step is output from the current state. Also, since $\boldsymbol{u} \to \boldsymbol{s}'$ and $(\boldsymbol{s}, \boldsymbol{u}) \to \boldsymbol{s}'$ contain only $\boldsymbol{s}$ in the output, they can be regarded as the network structure of state transition model (STM) that predicts the state of the next time step. In CTM, $(\boldsymbol{s}, \boldsymbol{u}) \to (\boldsymbol{s}', \boldsymbol{u}')$ contains all the information of the other four network structures, and in STM, $(\boldsymbol{s}, \boldsymbol{u}) \to \boldsymbol{s}'$ contains all the information of the other network structure. The predictive model learning with the network structures of $(\boldsymbol{s}, \boldsymbol{u}) \to \boldsymbol{s}'$ (STM) and $(\boldsymbol{s}, \boldsymbol{u}) \to (\boldsymbol{s}', \boldsymbol{u}')$ (CTM) are handled in this study, since these two network structures can show the basic functions of DPMPB.

2) Classification of Modeling Difficulties: Next, we classify the modeling difficulties handled in this study (the center figure of Fig. 2). The behavior of a robot propagates its effects in the form of its body, the object/tool it handles, and the environment in which it operates. For this reason, four categories are defined: behavior, body, object/tool, and environment. Predictive models represent the relationships among multiple sensors and actuators. Therefore, modeling difficulties can be categorized into the modeling between behavior and body, between body and object/tool, between body and environment, and between object/tool and environment. The modeling difficulty between behavior and body occur when the body is complex, especially with flexible and redundant musculoskeletal systems, low-rigidity plastic bodies, flexible hands, etc. (handled in Section III-A, Section III-B, Section III-D, Section III-E, and Section III-F). The

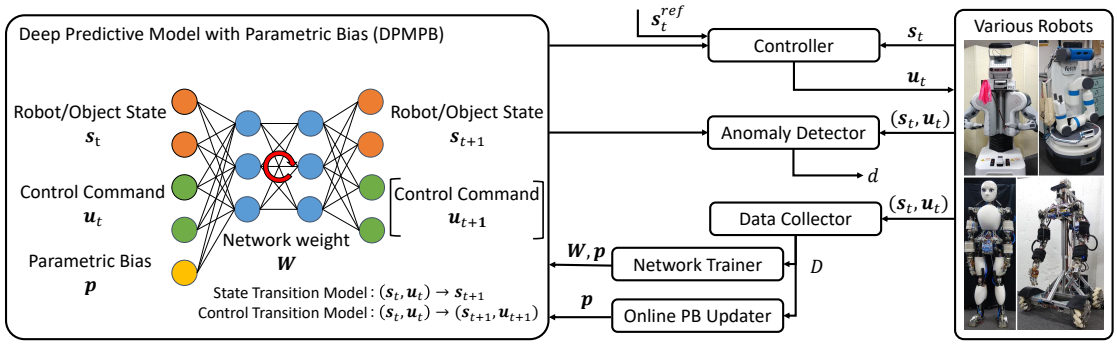

Fig. 3. The system overview of deep predictive model with parametric bias (DPMPB). DPMPB has the network input of robot/object state s_t, control command u_t, and parametric bias p, and the network output of s_{t+1} and u_{t+1} depending on the network structure of state transition model (STM) or control transition model (CTM). Controller, Anomaly Detector, Data Collector, Network Trainer, and Online PB Updater of various robots can be executed through DPMPB by only changing the network input/output and a few parameters.

modeling difficulty between body and object/tool occurs when the robot body handles flexible tools or flexible objects (handled in Section III-A and Section III-F). The modeling difficulty between body and environment occurs when the body interacts with various undefined environments such as unknown floor material and obstacles (handled in Section III-C and Section III-E). Although not directly handled in this study, modeling difficulties between object/tool and environment, such as friction and action-reaction, are also likely to occur.

3) Classification of Temporal Model Changes: Finally, we classify the temporal model changes handled in this study (the right figure of Fig. 2). We can classify them by STM or CTM, and also by the robot behavior, body, object/tool, or environment. Here, temporal model changes can be detected because the model can predict the changing values of sensors and actuators. Therefore, CTM can detect temporal model changes for behavior, body, object/tool, and environment, but STM cannot consider temporal model changes for behavior because it does not predict behavior. We will raise examples of temporal model changes. Regarding behavior, we can consider the changes in behavior for the same state, i.e., the change in motion style (handled in Section III-D). Regarding body, we can consider the changes in body state due to aging, changes in parts of the body not directly included in the model, changes in what the robot wears, etc. (handled in Section III-B for CTM, and in Section III-A and Section III-E for STM). For object/tool, we can consider changes in grasped objects, handled tools, manipulated cloths, etc. (handled in Section III-A and Section III-F). For environment, we can consider changes in floor materials, surrounding obstacles, wall friction, etc. (handled in Section III-C).

DPMPB can realize various tasks by combining its structure (STM or CTM), the handled modeling difficulties among behavior, body, object/tool, and environment, and the handled temporal model changes in behavior, body, object/tool, and environment. Specifically, this combination corresponds to changes in the parameters of whether $\boldsymbol{u}$ is included in the network output, what $\boldsymbol{s}$ and $\boldsymbol{u}$ are used, and what changes occur when $\boldsymbol{s}$ and $\boldsymbol{u}$ are collected. In this study, we show that it is possible to realize a comprehensive set of possible task examples based on these classifications by simply changing these parameters in DPMPB.

B. Network Structure

The network structure of DPMPB proposed in this study (Fig. 3) is shown in the following equations,

$$\begin{aligned} \boldsymbol{y}_{t+1} &= \boldsymbol{h}(\boldsymbol{x}_t, \boldsymbol{p}) \\ \boldsymbol{x}_t &= \left(\boldsymbol{s}_t^T, \boldsymbol{u}_t^T\right)^T \\ \boldsymbol{y}_{t+1} &= \begin{cases} \boldsymbol{s}_{t+1} & (type: \text{STM}) \\ \left(\boldsymbol{s}_{t+1}^T, \boldsymbol{u}_{t+1}^T\right)^T & (type: \text{CTM}) \end{cases} \end{aligned} \tag{1}$$

where t is the current time step, $\boldsymbol{s}$ is the sensor state of the robot body, target objects, etc., $\boldsymbol{u}$ is the control input representing the motion, $\boldsymbol{x}$ is the network input, $\boldsymbol{y}$ is the network output, $\boldsymbol{p}$ is parametric bias (PB), and $\boldsymbol{h}$ is the predictive model containing the network weight $\boldsymbol{W}$. The information contained in $\boldsymbol{y}$ differs depending on whether the predictive model type is STM or CTM. STM outputs only $\boldsymbol{s}$, while CTM outputs $\boldsymbol{u}$ in addition. In the case of STM, $\boldsymbol{u}$ is optimized to make $\boldsymbol{s}$ closer to the target state, while in the case of CTM, the next control input can be calculated directly from the current state. Note that the network structure of STM or CTM can be automatically determined from the collected data. $\boldsymbol{p}$ is a low-dimensional input variable that can embed implicit differences in dynamics by collecting data while changing the physical state of the robot, target objects, tools, and environment.

Here, we briefly describe the applications of this network structure. The network basically contains $\boldsymbol{s}$, $\boldsymbol{u}$, $\boldsymbol{p}$, and $\boldsymbol{W}$ as values. The only operations possible here are to compute these values by forward propagation, or to update them from the loss function by back propagation and gradient descent methods. In Fig. 3, the former operation is used for Simulator to update the current $\boldsymbol{s}$ (not directly handled in this study), and for Anomaly Detector to take prediction errors for $\boldsymbol{s}$. The latter operation is used for Network Trainer to update $\boldsymbol{W}$ and $\boldsymbol{p}$ simultaneously, for Online PB Updater to update only $\boldsymbol{p}$, and for Controller to update only $\boldsymbol{u}$.

The basic network structure is described below. Our DPMPB is a 10-layer recurrent neural network consisting of four FC layers (fully-connected layers), two LSTM layers (long short-term memory layers [21]), and four FC layers. The activation function is hyperbolic tangent and the update

rule is Adam [22]. In addition, all values of $\boldsymbol{s}$ and $\boldsymbol{u}$ are normalized and used as the network input and output. The control frequency of $Eq.1$ is basically 5 Hz (1 Hz only for Section III-B). The dimension of $\boldsymbol{p}$ should be slightly smaller than the expected changes in the body state, because too small a dimensionality will not represent the change in dynamics properly, and too large a dimensionality will make self-organization of $\boldsymbol{p}$ difficult. Regardless of the dimensionality of the changes in dynamics, they will be compressed to the point where they can be represented by the set dimension of PB.

C. Data Collection and Training of DPMPB

First, we collect the time series data of $\boldsymbol{s}$ and $\boldsymbol{u}$. For this purpose, we mainly use two types of data collection methods: (1) random motion and (2) teaching motion. Random motion is a motion in which the minimum and maximum values of the control input are determined and the body is moved randomly within these values. Teaching motion is a motion in which a human moves the robot body using a VR device, GUI, gaming controller, etc. While random motion allows the robot to explore space widely and evenly, teaching motion is effective for tasks that are difficult to succeed with random motion. Since CTM requires that the next control input can be computed from the current state, STM is most likely used when handling the data collected by (1), but the network structure will be determined automatically.

Using the obtained time series data D of $\boldsymbol{s}$ and $\boldsymbol{u}$, DPMPB is trained. In this process, by collecting data while changing the states of the body, target objects, tools, and environment, this information can be implicitly embedded into the space of PB. In order to allow the transition of each time series data with different dynamics to be represented by a single model, the differences in dynamics are self-organized in a low-dimensional space of $\boldsymbol{p}$. The data collected for a given identical state k is represented as $D_k = \{(\boldsymbol{s}_1, \boldsymbol{u}_1), (\boldsymbol{s}_2, \boldsymbol{u}_2), \cdots, (\boldsymbol{s}_{T_k}, \boldsymbol{u}_{T_k})\}$ $(1 \leq k \leq K$, where K is the total number of states, and T_k is the number of time steps for the trial in the state k), and the data used for training $D_{train} = \{(D_1, \boldsymbol{p}_1), (D_2, \boldsymbol{p}_2), \cdots, (D_K, \boldsymbol{p}_K)\}$ is obtained. Here, $\boldsymbol{p}_k$ is PB that represents the dynamics with respect to the state k. Since $\boldsymbol{p}_k$ is a variable, the initial value can be anything, just like the bias term in a neural network, and $\boldsymbol{p}_k$ does not require a specific value for training. PB is a common variable for a particular state but is another variable for another state. Using this D_{train}, we train DPMPB with the number of network expansions as N_{step}^{train}, the number of batches as N_{batch}^{train}, and the number of epochs as N_{epoch}^{train}. In the usual training, only the network weight $\boldsymbol{W}$ is updated, but here $\boldsymbol{W}$ and $\boldsymbol{p}_k$ for each state are updated simultaneously. Note that the mean squared error for the network output $\boldsymbol{y}$ is used as the loss function in the learning process, and $\boldsymbol{p}_k$ is optimized with an initial value of $\boldsymbol{0}$ for all. In this way, the differences in the dynamics of each body state are embedded in $\boldsymbol{p}_k$.

Here, we also describe the automatic determination of the network structure of STM or CTM. First, we train the network as $\boldsymbol{y}_{t+1} = \left(\boldsymbol{s}_{t+1}^T, \boldsymbol{u}_{t+1}^T\right)^T$. In this case, we calculate the loss L_n separately for each value of $\boldsymbol{y}_n$ $(1 \leq n \leq N_{sensor}$, where N_{sensor} represents the number of sensors). We set a threshold L_{thre} and adopt the value with L_n smaller than L_{thre} as $\boldsymbol{y}$. If $\boldsymbol{y}$ includes the value of $\boldsymbol{u}$, it is CTM, and if not, it is STM. Thus, by removing the values that are difficult to infer from the network output, the network structure is defined and the possible tasks in the network are changed. The network is then re-trained with the adopted $\boldsymbol{y}$.

D. Online Update of Parametric Bias

$\boldsymbol{p}_k$ computed at the time of training represents the dynamics corresponding to each data D_k and does not represent the current dynamics. Therefore, we update PB online using the data D obtained from the current state of the body, target objects, tools, and environment. If the network weight $\boldsymbol{W}$ is updated, DPMPB may overfit to the data, but if only the low-dimensional PB $\boldsymbol{p}$ is updated, overfitting is less likely to occur. This online learning allows us to constantly recognize the current robot state and to obtain a model that adapts to it.

Let the number of data obtained be N_{data}^{online}, and start online learning when the number of data exceeds the threshold N_{thre}^{online}. Whenever new data is received, PB is updated with the number of batches as N_{batch}^{online}, the number of epochs as N_{epoch}^{online}, and the update rule as MomentumSGD. Data exceeding the maximum number of data N_{max}^{online} are deleted from the oldest ones. The smaller N_{max}^{online} is, the faster the model may adapt to the current state, but the learning may become unstable due to the decrease in the number of data.

E. Control Using DPMPB

We describe a control method using DPMPB. This is very different depending on whether the network structure is CTM or STM. In the case of CTM, it is very simple; since $\boldsymbol{u}$ is in the output, we only need to calculate $\boldsymbol{u}_t$ from $\boldsymbol{s}_{t-1}$ and $\boldsymbol{u}_{t-1}$ of the previous time step through the forward propagation of the network, and send it to the robot.

On the other hand, in the case of STM, since $\boldsymbol{u}$ is not in the output, we need to optimize $\boldsymbol{u}$ from a loss function. This is a kind of learning-based model predictive control. First, we give the initial value of time-series control input $\boldsymbol{u}_{seq}^{init}$ for $\boldsymbol{u}_{seq} = \boldsymbol{u}_{t:t+N_{step}^{control}-1}$ ($\boldsymbol{u}_{t_1:t_2}$ is $\boldsymbol{u}$ in $[t_1, t_2]$). $N_{step}^{control}$ is the number of DPMPB network expansions representing control horizon (the expansion operation will be explained later). Let $\boldsymbol{u}_{seq}^{opt}$ be $\boldsymbol{u}_{seq}$ to be optimized, and repeat the following calculation at time t to obtain the optimal $\boldsymbol{u}_t^{opt}$,

$$\boldsymbol{s}_{seq}^{pred} = \boldsymbol{h}_{expand}(\boldsymbol{s}_t, \boldsymbol{u}_{seq}^{opt}, \boldsymbol{p}) \tag{2}$$

$$L = h_{loss}(\boldsymbol{s}_{seq}^{pred}, \boldsymbol{u}_{seq}^{opt}) \tag{3}$$

$$\boldsymbol{u}_{seq}^{opt} \leftarrow \boldsymbol{u}_{seq}^{opt} - \gamma \partial L / \partial \boldsymbol{u}_{seq}^{opt} \tag{4}$$

where $\boldsymbol{s}_{seq}^{pred}$ is the predicted $\boldsymbol{s}_{t+1:t+N_{step}^{control}}$, $\boldsymbol{h}_{expand}$ is the function of $\boldsymbol{h}$ expanded $N_{step}^{control}$ times, h_{loss} is the loss function, and γ is the learning rate. Note that the network expansion in $\boldsymbol{h}_{expand}$ is a function that represents the operation of inputting $\boldsymbol{u}^{opt}$ from $\boldsymbol{s}_t$ and predicting $\boldsymbol{s}^{pred}$ in $N_{step}^{control}$ steps. In other words, the future $\boldsymbol{s}$ is predicted from $\boldsymbol{s}_t$ by $\boldsymbol{u}_{seq}^{opt}$, and $\boldsymbol{u}_{seq}^{opt}$

is optimized by backpropagation and gradient descent to minimize the loss function.

Here, $\boldsymbol{u}_{seq}^{init}$ is set to $\boldsymbol{u}_{\{t+1,\cdots,t+N_{step}^{control}-1,t+N_{step}^{control}-1\}}^{prev}$, by using $\boldsymbol{u}_{t:t+N_{step}^{control}-1}^{prev}$ ($\boldsymbol{u}_{seq}$ optimized in the previous step), and by shifting the time step of the value by one and duplicating the last term. Faster convergence can be obtained by using the previous optimization results. For γ, we set the maximum learning rate γ_{max}, prepare $N_{batch}^{control}$ of γ by dividing $[0, \gamma_{max}]$ exponentially, run Eq. 4 on each γ, calculate the loss by Eq. 3, and select $\boldsymbol{u}_{seq}^{opt}$ with the smallest loss. This procedure is repeated $N_{epoch}^{control}$ times. Faster convergence can be obtained by trying various γ and always choosing the best learning rate.

There are various possible forms of the loss function h_{loss}. For each value of $\boldsymbol{s}$ and $\boldsymbol{u}$, the main possible forms are minimization of the value, maximization of the value, minimization of the error with a certain target value, and minimization of the change from the value of the previous time step. These are set appropriately for each experiment.

F. Anomaly Detection Using DPMPB

Anomaly detection can be performed from the prediction error of DPMPB. This is executed for the magnitude of the error between the current value $\boldsymbol{s}_t$ and the predicted value $\boldsymbol{s}_t^{pred}$ for the output of the network. First, for the data D collected in the normal state without any anomaly, we collect the measured value $\boldsymbol{s}_t^{data}$ and the predicted value $\boldsymbol{y}_t = \boldsymbol{h}(\boldsymbol{x}_{t-1}, \boldsymbol{p})$ calculated from the previous time step. For this data, the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ of the error $\boldsymbol{s}_t^{data} - \boldsymbol{s}_t^{pred}$ is calculated. For anomaly detection, $\boldsymbol{e}_t = \boldsymbol{s}_t^{pred} - \boldsymbol{s}_t$ (which is the difference between the current state value $\boldsymbol{s}_t$ and the estimated value $\boldsymbol{s}_t^{pred}$) is always obtained, and the Mahalanobis distance $d = \sqrt{(\boldsymbol{e}_t - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{e}_t - \boldsymbol{\mu})}$ is calculated for them. When d exceeds a threshold value, an anomaly has been detected. For this threshold, the variance of d can be calculated for the data used to calculate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, and 3σ value can be used. Anomaly can be calculated for each sensor or for all sensors at once.

G. Detailed Implementation

We describe the implementation of DPMPB in more detail (Fig. 4). First of all, we use ROS (Robot Operating System) for all sensor communication and Chainer as the deep learning framework in this study. The configuration file describes which ROS topic is used for the input and output of DPMPB, the number of PB and hidden layer units in the network, and the model file name. The model file of the network contains the network weight $\boldsymbol{W}$, the class name for each state obtained during training and the corresponding PB $\boldsymbol{p}_k$, the current PB $\boldsymbol{p}$, the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ of the prediction error of the network. Note that by storing the class names, explicit object and environment recognition can be performed from $\boldsymbol{p}_k$, which is closest to the current $\boldsymbol{p}$. Although the sensory and motion data of the robot are collected as is, images are processed in a special way. Since the data size of an image is too large as it is, we train Convolutional AutoEncoder in advance and use the value converted into the latent space as a ROS Topic. Data Collector in Fig. 4 is drawn differently from Fig. 3 for convenience, but it summarizes the obtained data for Network Trainer, Online PB Updater, Controller, and Anomaly Detector. For Network Trainer, Online PB Updater, and Controller, the number of batches, the number of epochs, and the learning rate need to be given as parameters, and for Controller, the form of the loss function and the target value need to be given in addition. Network Trainer stores $\boldsymbol{W}$ and $\boldsymbol{p}_k$, and Online PB Updater updates only $\boldsymbol{p}$.

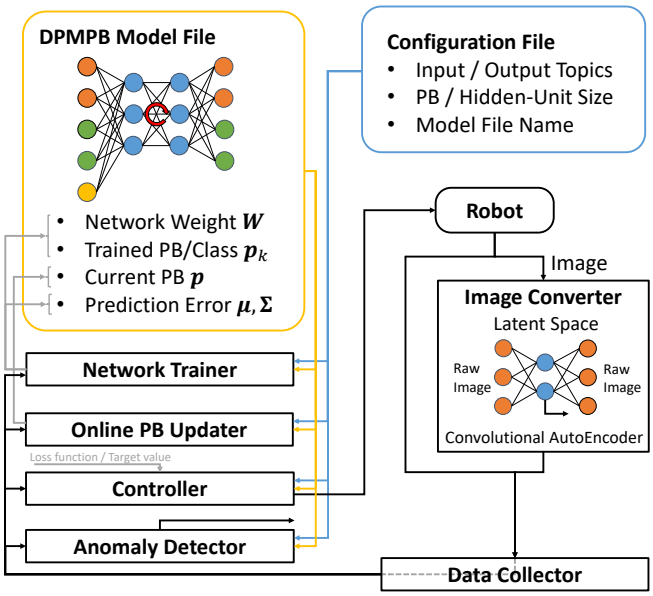

Fig. 4. The detailed implementation of deep predictive model with parametric bias (DPMPB). The model file includes the information of network weight $\boldsymbol{W}$, trained parametric bias (PB) $\boldsymbol{p}_k$ and class, current PB $\boldsymbol{p}$, and the average μ and variance Σ of the prediction error calculated at training. The configuration file includes input/output topics of Robot Operating System (ROS), the dimension of PB and hidden unit of DPMPB, and the model file name. The image is compressed by AutoEncoder and Data Collector gathers and sends all data to each component of Network Trainer, Online PB Updater, Controller, and Anomaly Detector. Network Trainer updates $\boldsymbol{W}$ and $\boldsymbol{p}_k$, and Online PB Updater updates $\boldsymbol{p}$.

III. Experiments

In this section, we describe several examples of robot tasks using DPMPB, focusing on what kind of modeling difficulties and temporal model changes are handled, what kind of sensors and actuators are handled, and what kind of loss functions are used in training, online update, and control. First, we show the basic experiments of STM and CTM in Section III-A and Section III-B, respectively, to demonstrate their functions. Next, in Section III-C, we show an experiment of variance minimization control by introducing mean-variance representation to the network output as an application of STM, and in Section III-D, we show an experiment of actively changing the motion style in imitation learning as an application of CTM. Finally, Section III-E and Section III-F show examples of highly difficult modeling, specifically a balance control experiment considering change in shoes and a dynamic cloth manipulation considering change in cloth material by musculoskeletal humanoids. We

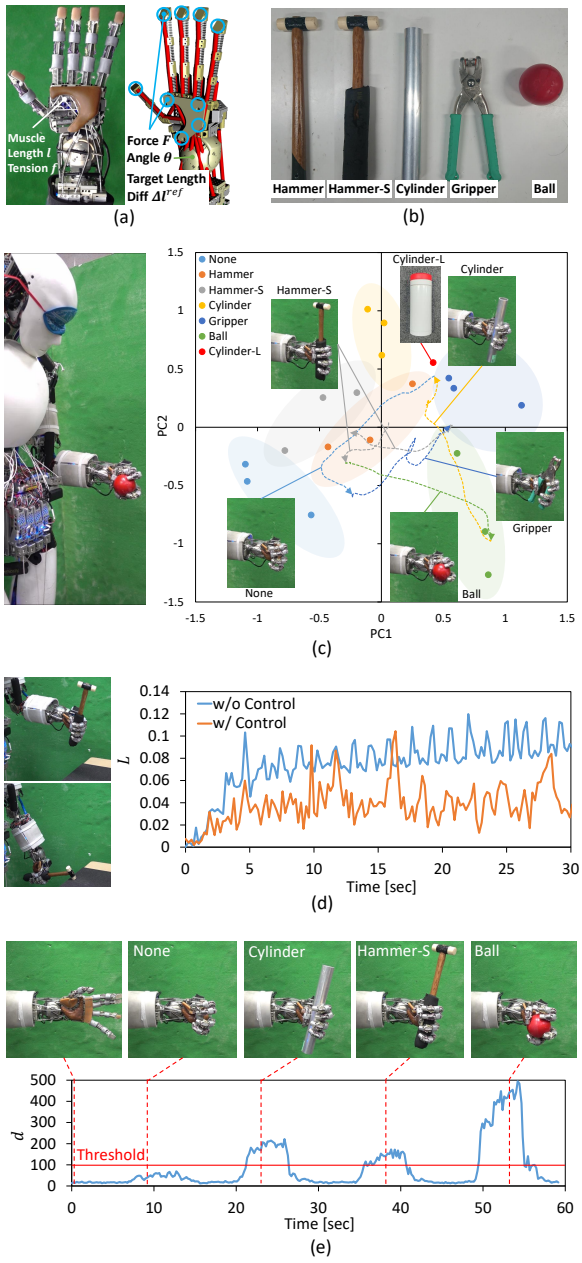

Fig. 5. Experiment of grasping object recognition and contact control of the flexible hand. (a) shows the sensors and actuators of the flexible musculoskeletal hand, (b) shows the grasped objects and tools, (c) shows the trained parametric bias and its trajectory when conducting online update of PB, (d) shows the transition of L when using or not using a grasping stabilization control, and (e) shows the transition of d for contact detection when grasping various objects.

summarize [7]–[12] in a new angle from the viewpoint of modeling difficulties and temporal model changes. Note that the same symbols are used with different definitions in each experiment. Note also that the results of control with the wrong parametric bias are shown for the case where the model at the time of training is different from the current model in the general predictive model, and that these are the results of comparison between DPMPB and the general predictive model.

A. Grasping Object Recognition and Contact Control of Flexible Hand

In this section, we deal with the sensory-motor model of a musculoskeletal flexible hand [23] as a modeling difficulty. As a temporal model change, we deal with the change in grasped objects/tools. For sensor state and control input, we set $\boldsymbol{s} = \{\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{\theta}, \boldsymbol{F}\}$ and $\boldsymbol{u} = \Delta \boldsymbol{l}^{ref}$. Note that, as shown in (a) of Fig. 5, $\{\boldsymbol{l}, \boldsymbol{f}, \Delta \boldsymbol{l}^{ref}\}$ is {muscle length, muscle tension, change in target muscle length} of the muscles related to the wrist and hand $\in \mathbb{R}^8$, $\boldsymbol{\theta}$ is the wrist joint angle $\in \mathbb{R}^2$, and $\boldsymbol{F}$ is the contact sensor value $\in \mathbb{R}^9$. In (b) of Fig. 5, we show the grasped objects/tools: Hammer, Hammer-S, Cylinder, Gripper, and Ball. None refers to the state without grasping any objects/tools. This experiment is difficult in that it is necessary to acquire the relationship between five different sensors and actuators in a flexible body, and to recognize the changes in the dynamics of the sensors and actuators and control them accordingly. We collected data by randomly changing $\boldsymbol{u}$ while changing the grasped objects/tools. DPMPB with 8-dimensional PB was trained using 36 datasets with about 500 steps each (about 18000 steps in total). Here, $L_1 = 0.093$, $L_2 = 0.184$, $L_3 = 0.212$, $L_4 = 0.036$, and $L_5 = 0.468$, and since $\boldsymbol{y}_5 = \boldsymbol{u}$ is removed from the output by setting $L_{thre} = 0.3$, the network structure is STM. The final loss when training was 0.014.

(c) of Fig. 5 shows the arrangement of the trained PBs $\boldsymbol{p}_k$. Note that PBs are shown after being compressed to two dimensions by Principle Component Analysis (PCA) in all of the following experiments. Also, shading in (c) of Fig. 5 is manually generated for the set of PBs for better understanding (the same with subsequent experiments). It can be seen that the space of PB is self-organized by the dynamics of grasped objects/tools. When we collect data for new object Cylinder-L (upper part of (c) in Fig. 5), which is the same shape as Cylinder and has the same radius as Gripper, and update only PB, we can see that PB of Cylinder-L is at about halfway between the PBs of Cylinder and Gripper, and that the space of PB is generalized even for untrained objects/tools. In addition, from the trajectory of PB shown as the dotted line in (c), when we perform random motion while changing the grasped objects/tools and simultaneously executing online learning of PB, the current PB $\boldsymbol{p}$ gradually approaches the trained PB $\boldsymbol{p}_k$ for the current grasped object/tool. In other words, it is possible to perform recognition of grasped object/tool from its dynamics. Here, the average number of data steps used for online update is 500, which indicates that the adaptation takes about 100 seconds.

In (d) of Fig. 5, we show a grasping stabilization control experiment using DPMPB. It is possible to maintain the initial grasping state against external forces by setting h_{loss} in Section II-E as follows,

$$h_{loss}(\boldsymbol{s}_{seq}^{pred}) = ||\boldsymbol{w}_1 \otimes (\boldsymbol{F}_{seq}^{pred} - \boldsymbol{F}_{seq}^{init})||_2 + w_2||\boldsymbol{\theta}_{seq}^{pred} - \boldsymbol{\theta}_{seq}^{init}||_2 + w_3||\boldsymbol{l}_{seq}^{pred} - \boldsymbol{l}_{seq}^{init}||_2 \quad (5)$$

where $\{\boldsymbol{F}, \boldsymbol{\theta}, \boldsymbol{l}\}_{seq}^{init}$ denotes the initial $\{\boldsymbol{F}, \boldsymbol{\theta}, \boldsymbol{l}\}$ when grasping

the object/tool, $\{w_1, w_2, w_3\}$ denotes the constant weight, $||\cdot||_2$ denotes L2 norm, and $\otimes$ denotes element-wise product. Considering the anisotropy, where the values of the contact sensor vary only up to 0 in the negative direction, but vary significantly up to the rated value in the positive direction, we set $\boldsymbol{w}_1[i] = 1.0$ when $\boldsymbol{F}_{seq}^{pred}[i] \geq \boldsymbol{F}_{seq}^{ref}[i]$), otherwise $\boldsymbol{w}_1[i] = w_4 (w_4 > 1.0)$. The graph of (d) shows that when hitting a desk with a hammer, L expressed by h_{loss} gradually increases without this grasping stabilization control, while with it, it is possible to resist large external forces and keep L small.

In (e) of Fig. 5, we show an application experiment of anomaly detection using DPMPB. With the current $\boldsymbol{p}$ as $\boldsymbol{p}_k$ trained at None, grasping an object/tool changes the dynamics of the hand and causes prediction errors in the network. This is captured by the anomaly detection of Section II-F, which enables us to perform contact detection. Note that the farther the grasped object is from None in the space of PB, the larger the anomaly state d is likely to be.

B. Visual Feedback of Low-Rigidity Robot Considering Temporal Body Change

In this section, we deal with a visual feedback model of low-rigidity plastic-made axis-driven robot MyCobot as a modeling difficulty. As a temporal model change, we deal with the physical changes in robot state such as aging and configuration changes of the robot. For sensor state and control input, we set $\boldsymbol{s} = \boldsymbol{z}$ and $\boldsymbol{u} = \boldsymbol{\theta}^{ref}$. Note that, as shown in (a) of Fig. 6, $\boldsymbol{z}$ is the image compressed by AutoEncoder, and $\boldsymbol{\theta}^{ref}$ is the target joint angle $\in \mathbb{R}^7$ (the grasping state is represented by a binary value of 0 or 1). We use four objects {L-25, L-15, S-25, S-15} to be grasped, which are combinations of the length of the object (L or S) and the width of the object (15 mm or 25 mm). In (b) of Fig. 6, the aging of the joints is represented by {j0, j1} and the configuration change of the camera position is represented by {c1, c2, c3}. Since it is difficult to quantitatively evaluate the changes in the robot state over time, we artificially create and experimentally test these changes by assuming that j1 is the case where all joints are offset by 2 deg with respect to j0, and that {c1, c2, c3} is the case where the camera position is shifted by 10 mm. This experiment is difficult in that the robot with a flexible body needs to perform imitation behaviors while responding to time-varying body states. We collected data by a series of motions (approaching, grasping an object, and returning) while changing the robot state. In this process, since a robot with low rigidity cannot grasp an object as intended if it simply reaches for the recognized object, the robot itself first randomly places the object and then reaches out to the same position to grasp it. By repeating this procedure, data is autonomously collected for visual feedback without human instruction. DPMPB with 2-dimensional PB was trained using 120 datasets with about 17 steps each (about 2040 steps in total) Here, $L_1 = 0.088$ and $L_2 = 0.022$, and since both values are low and $\boldsymbol{u}$ can be inferred, the network structure is CTM. The final loss when training was 0.013.

(c) of Fig. 6 shows the placement of the trained PBs $\boldsymbol{p}_k$.

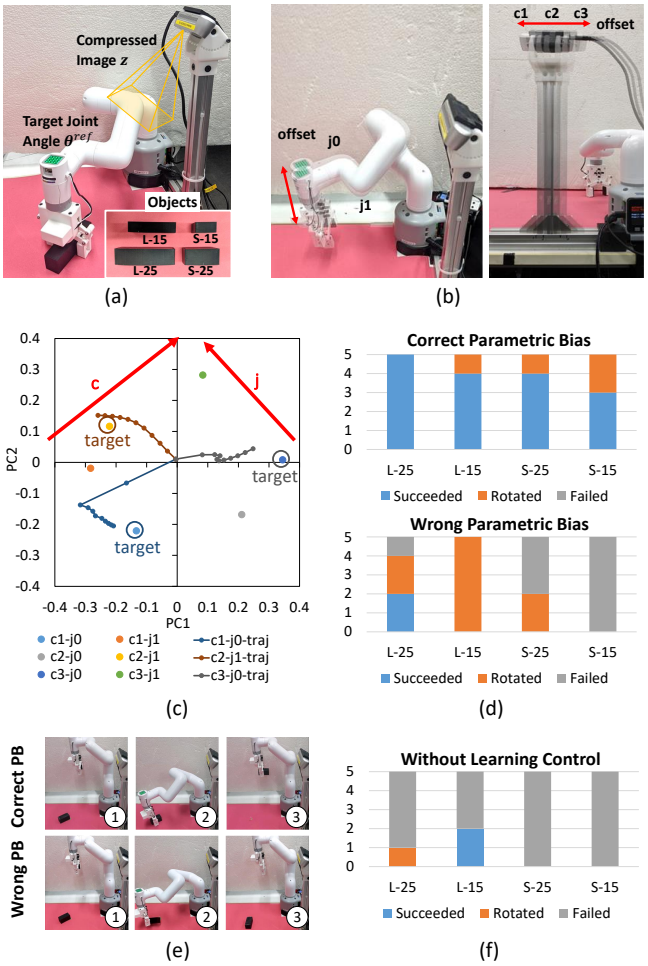

Fig. 6. Experiment of visual feedback of low-rigidity robot considering temporal body change. (a) shows the sensors and actuators of the low-rigidity robot MyCobot, (b) shows the changed offset values of joint angle and camera arrangement to express temporal change in robot configuration, (c) shows the trained parametric bias and its trajectory when conducting online update of PB, (d) shows the result of visual feedback experiment when using correct or wrong parametric bias, (e) shows examples of the visual feedback motion of (d), and (f) shows the result of general grasping control using point cloud and inverse kinematics.

We can see that the space of PB is self-organized along the axes of {c1, c2, c3} and {j0, j1}. In addition, from the trajectory of PB represented as "-traj" in (c), when the online learning of PB is performed with the robot state set to {c1-j0, c2-j1, c3-j0}, the current robot state can be accurately recognized from its dynamics. Here, the average number of data steps used for online update is 17, which indicates that the adaptation takes about 17 seconds.

In (d) and (e) of Fig. 6, we show the results of visual feedback experiments using DPMPB. Four objects are placed at different positions and the robot grasps them five times each by using visual feedback. Here, "Succeeded" means that the robot succeeded in grasping the object, "Rotated" means that the robot succeeded in grasping the object but the parallel gripper hit the edge of the object and the object rotated, and "Failed" means that the robot failed to grasp the object. The robot state is c2-j0, and we compare

the case where the trained PB for the same state is used (Correct) and the case where the trained PB for the wrong state c3-j1 is used (Wrong). As shown in (e), we can see that the visual feedback motion changes depending on the value of PB. From (d), we can see that in the case of Correct, all grasps Succeeded or Rotated, while in the case of Wrong, the probability of failure increases, indicating that the recognition of the current robot state is important. In both cases, the larger object is more likely to be grasped successfully even if its position is slightly misaligned.

In (f) of Fig. 6, we show the results of a general object grasping experiment without any learning control. A point cloud is obtained from a depth camera, the object is extracted from plane detection and Euclidean clustering, and the object is grasped by solving inverse kinematics. The robot body is approximated as a rigid body and the camera position on CAD (Computer-Aided Design) is used. The success rate of grasping is very low for all objects, and most of the movements do not even touch the object.

C. Stable Control of Wheeled Robot Considering Floor Change

In this section, we deal with the sensory-motor model of a wheeled robot Fetch as a modeling difficulty. As a temporal model change, we deal with the change in the operating environment of the wheeled robot, i.e., the floor. For sensor state and control input, we set $\boldsymbol{s} = \boldsymbol{w}$ and $\boldsymbol{u} = \boldsymbol{w}^{ref}$. Note that $\boldsymbol{w}$ is the current robot velocity obtained by Visual Odometry $\in \mathbb{R}^2$ (the translational direction is w_{trans} and the rotational direction is w_{rot}), and $\boldsymbol{w}^{ref}$ is the target robot velocity. As shown in (a) of Fig. 7, the wheeled base of Fetch has two active wheels and four passive casters. When moving backwards, the motion of the base may change stochastically depending on the direction of the casters, which is characteristic of the old Fetch model (it is possible to move forward accurately due to the suspension). In order to deal with the stochastic state transitions in this experiment, the network output in STM is changed to the mean $\boldsymbol{s}$ and variance $\boldsymbol{v}$ of the states, and is trained by a loss function based on the following maximum likelihood estimation,

$$P(s^k_{i,t}|D_{k,1:t-1},W,\boldsymbol{p}_k) = \frac{1}{\sqrt{2\pi\hat{v}_{i,t}}}\exp\left(-\frac{(\hat{s}^k_{i,t}-s^k_{i,t})^2}{2\hat{v}_{i,t}}\right)$$

$$L_{likelihood}(W,\boldsymbol{p}_{1:K}|D_{train}) = \prod_{k=1}^{K}\prod_{t=1}^{T_k}\prod_{i=1}^{N_s} P(s^k_{i,t}|D_{k,1:t-1},W,\boldsymbol{p}_k)$$

$$L_{train} = -\log(L_{likelihood}) \quad (6)$$

where P is the probability density function, $\{s,v\}_i$ is $\{s,v\}$ of the i-th sensor, $D_{k,1:t-1}$ is the data of D_k in $[1,t-1]$, $\{\hat{s},\hat{v}\}$ is the predicted value by using $D_{k,1:t-1}$, $\boldsymbol{W}$, and $\boldsymbol{p}_k$, $\boldsymbol{p}_{1:K}$ denotes a vector summarizing $\boldsymbol{p}_k$ in $1 \le k \le K$, and N_s is the dimension of $\boldsymbol{s}$. $L_{likelihood}$ represents the likelihood function for $\boldsymbol{W}$ and $\boldsymbol{p}$ given D_{train}, and we consider maximizing it and transform it to the problem of minimizing the loss L_{train} by taking -log. In (b) of Fig. 7, we prepare two kinds of environments, Room with high friction and Corridor with low friction. This experiment is difficult in that it

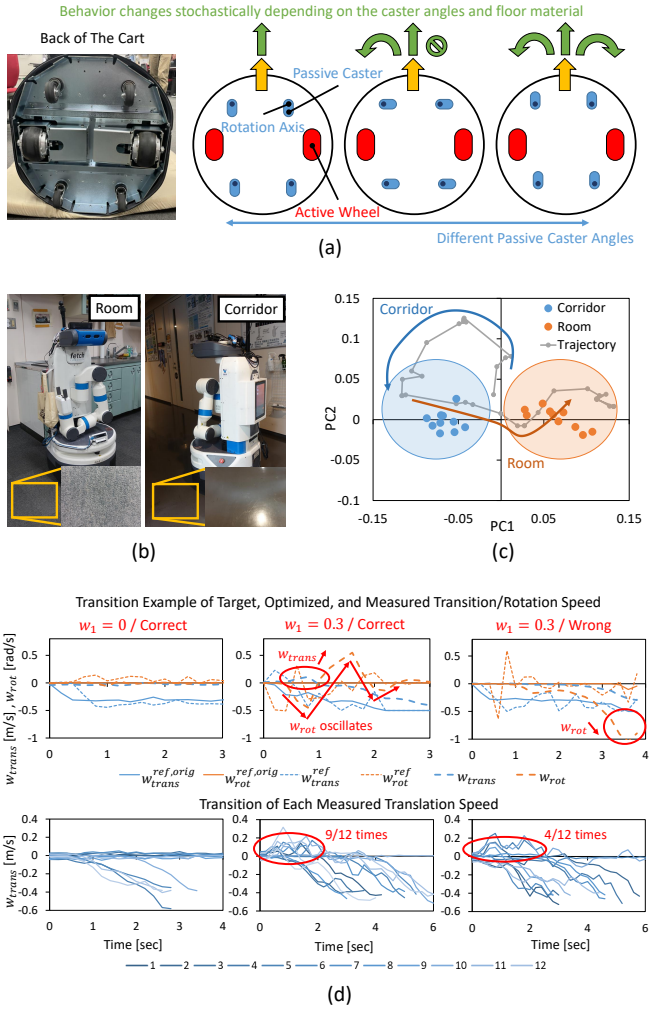

Fig. 7. Experiment of stable control of wheeled robot considering floor change. (a) shows the problem of probabilistic movement of the wheeled robot Fetch, (b) shows the changed floor environment of Room and Corridor, (c) shows the trained parametric bias and its trajectory when conducting online update of PB, (d) shows the result of stable control with variance minimization when setting $w_1 = 0$ / Correct, $w_1 = 0.3$ / Correct, and $w_1 = 0.3$ / Wrong.

is necessary to learn a stochastic and difficult-to-modelize behavior depending on the operating environment, and to control the body while recognizing the environment and considering the variance of the motion. We collected data by random control input while changing the floor. DPMPB with 2-dimensional PB was trained using 9 datasets with about 600 steps each (about 5400 steps in total). Here, $L_1 = 0.112$ and $L_2 = 0.949$, and since $\boldsymbol{y}_2 = \boldsymbol{u}$ is removed from the output by setting $L_{thre} = 0.3$, the network structure is STM. The final loss when training was -0.040 (which can be negative because of the use of Eq. 6).

In (c) of Fig. 7, we show the arrangement of the trained PBs $\boldsymbol{p}_k$. We can see that PBs are neatly divided into Room and Corridor, and that the space of PB is self-organized. From the trajectory of PB represented as "Trajectory" in (c), by updating PB online, we can see that when the robot is in Corridor, the current $\boldsymbol{p}$ approaches $\boldsymbol{p}_k$ trained in Corridor, and when the robot is in Room, the current $\boldsymbol{p}$ approaches $\boldsymbol{p}_k$

trained in Room, indicating that the operating environment can be correctly recognized from its dynamics. Here, the average number of data steps used for online update is 150, which indicates that the adaptation takes about 30 seconds.

(d) of Fig. 7 shows a stabilization control experiment based on variance minimization using DPMPB. The control can be achieved by setting h_{loss} in Section II-E as follows,

$$h_{loss}(\boldsymbol{s}_{seq}^{pred}, \boldsymbol{v}_{seq}^{pred}) = ||\boldsymbol{s}_{seq}^{ref} - \boldsymbol{s}_{seq}^{pred}||_2 + w_1||\boldsymbol{v}_{seq}^{pred}||_2 \quad (7)$$

where w_1 denotes the constant weight. This corresponds to the control that makes the current state closer to the target state while minimizing the variance of the state. In this experiment, $\boldsymbol{s}^{ref}$ is represented as $\boldsymbol{w}^{ref,orig}$, and the optimized value $\boldsymbol{u}^{opt}$ is represented as $\boldsymbol{w}^{ref}$. The effectiveness of this control is verified by moving the robot backward after one rotation. The passive caster faces perpendicularly to the active wheel after one rotation, and when the robot moves backward in this state, the motion often gets stuck. This experiment is conducted in Room, and three cases are compared: $w_1 = \{0, 0.3\}$ for the case of using $\boldsymbol{p}$ updated in Room (Correct), and $w_1 = 0.3$ for the case of using $\boldsymbol{p}$ updated in Corridor (Wrong). For each condition, the transitions of $\boldsymbol{w}^{ref,orig}$, $\boldsymbol{w}^{ref}$, and $\boldsymbol{w}$ are shown in the upper figure of (d) of Fig. 7. When $w_1 = 0$, there is no significant difference between $\boldsymbol{w}^{ref,orig}$ and $\boldsymbol{w}^{ref}$, but w_{trans} remains 0 and the motion is completely stucked. In contrast, in the case of $w_1 = 0.3$ / Correct, $\boldsymbol{w}^{ref,orig}$ and $\boldsymbol{w}^{ref}$ differ significantly, and w_{trans} starts to move after about 1.5 seconds. A characteristic of this condition is that w_{trans} first moves forward to change the direction of the caster, and $\boldsymbol{w}_{rot}$ is moved while oscillating to prevent it from getting stuck. In the case of $w_1 = 0.3$ / Wrong, w_{trans} also moves. In this condition, compared to $w_1 = 0.3$ / Correct, w_{trans} is not moved forward, but w_{trans} is directly moved backward. In addition, there are many cases where w_{rot} moves significantly in the same direction as that of the previous rotation; that is, it moves backward while rotating. w_{trans} for 12 trials is shown in the lower figure of (d) in Fig. 7. In the case of $w_1 = 0$, the motion is stuck in more than half of the trials, while in the case of $w_1 = 0.3$, the motion in the backward direction succeeds in all but one or two trials. In the case of $w_1 = 0.3$, the number of trials in which the robot moves backward after moving forward once is 9 out of 12 in the case of Correct, and 4 out of 12 in the case of Wrong.

D. Imitation Learning Considering Motion Style

In this section, we deal with a human imitation model of a musculoskeletal humanoid MusashiLarm with a flexible body as a modeling difficulty. As a temporal model change, we deal with the variability of human motion, i.e., change in motion style. This represents, for example, motion variations as different elbow positions or different movement speeds when performing a certain task. For sensor state and control input, we set $\boldsymbol{s} = \{\boldsymbol{z}, \boldsymbol{f}\}$ and $\boldsymbol{u} = \boldsymbol{l}^{ref}$. Note that, as shown in (a) of Fig. 8, $\boldsymbol{z}$ is the image compressed by AutoEncoder, and $\{\boldsymbol{f}, \boldsymbol{l}^{ref}\}$ is {muscle tension, target muscle length} of the muscles related to the arm $\in \mathbb{R}^{10}$ (5 DOFs of the shoulder and elbow are used). As shown in (b) of Fig. 8, a human moves MusashiLarm to close a box on the desk using a VR device, and the difference in the motion style is embedded in PB. This experiment is difficult in that it is necessary for a robot with a flexible body to learn to imitate human behaviors while changing its motion style. We collected data by performing a box-closing motion while changing the position and angle of the box. DPMPB with 3-dimensional PB was trained using 30 datasets with about 100 steps each (about 3000 steps in total) Here, $L_1 = 0.067$, $L_2 = 0.050$, and $L_3 = 0.013$, and since all values are low and $\boldsymbol{u}$ can be inferred, the network structure is CTM. The final loss when training was 0.017.

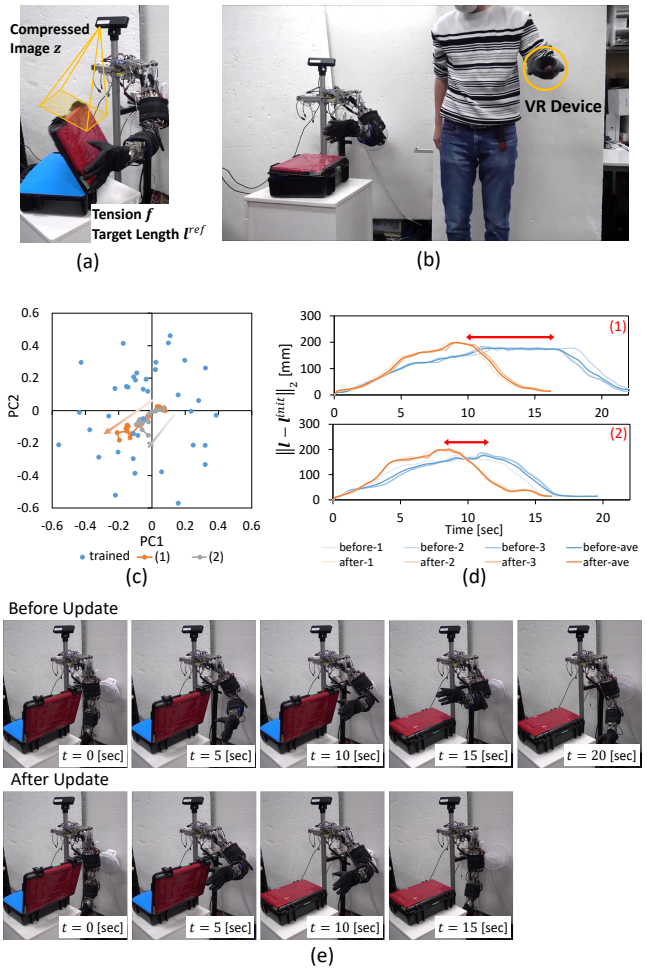

Fig. 8. Experiment of imitation learning considering motion style. (a) shows the sensors and actuators of the musculoskeletal humanoid MusashiLarm, (b) shows the data collection setting using VR device, (c) shows the trained parametric bias and its trajectory when conducting online update of PB, (d) shows the trajectory of $||\boldsymbol{l} - \boldsymbol{l}^{init}||_2$ when maximizing or not maximizing the muscle length velocity, and (e) shows the imitated motion of closing a box before and after the update of PB.

It is possible to realize the desired motion style by actively changing the motion style embedded in PB. In this study, we update$\boldsymbol{p}$ online to maximize the motion speed using the obtained data D, by setting the loss function as follows,

$$L_{update} = ||\boldsymbol{s}_{2:T}^{data} - \boldsymbol{s}_{2:T}^{pred}||_2 + w_1||\boldsymbol{u}_{3:T}^{pred} - \boldsymbol{u}_{2:T-1}^{pred}||_2 \quad (8)$$

where $\{\boldsymbol{s},\boldsymbol{u}\}_{1:T}$ is $\{\boldsymbol{s},\boldsymbol{u}\}$ in $[1,T]$ (T represents the number of time steps in D), $\{\boldsymbol{s},\boldsymbol{u}\}^{data}$ is the data contained in D, and w_1 is the constant weight (in this study, $w_1 < 0$ for the speed maximization). $\{\boldsymbol{s},\boldsymbol{u}\}_1^{data}$ is fed into the network and $\{\boldsymbol{s},\boldsymbol{u}\}^{pred}$ is inferred in an autoregressive manner. We maximize the velocity of $\boldsymbol{u}$ while imposing a slight constraint on $\boldsymbol{s}$.

In (c) of Fig. 8, we show the arrangement of the trained PBs $\boldsymbol{p}_k$ and the trajectories (1) and (2) of $\boldsymbol{p}$ during online learning based on this speed maximization. Although (1) and (2) show results for different box angles, $\boldsymbol{p}$ transitions in the same direction in both cases, indicating that the space of PB is self-organized to represent the motion style. Here, the average number of data steps used for online update is 100, which indicates that the adaptation takes about 20 seconds. In (d) of Fig. 8, the transition of the change in $\boldsymbol{l}$ from the initial muscle length $\boldsymbol{l}^{init}$ is shown. before-$\{1, 2, 3\}$ denotes the three trials before the update of $\boldsymbol{p}$, after-$\{1, 2, 3\}$ denotes the three trials after the update of $\boldsymbol{p}$, and $\{$before, after$\}$-ave denotes their average. It can be seen that the behavior after the update of $\boldsymbol{p}$ converges faster than before, and at the same time, the behavior is reproducible. (e) of Fig. 8 shows snapshots of the motion in (1), where the robot completes the box-closing operation more than 5 seconds faster after updating $\boldsymbol{p}$.

E. Balance Control of a Musculoskeletal Humanoid Considering Change in Shoes

In this section, we deal with a balance model of a full-body musculoskeletal humanoid Musashi [23] as a modeling difficulty. As a temporal model change, we deal with the physical changes in body state that are not included in the balance model, such as changes in shoes and upper body posture. For sensor state and control input, we set $\boldsymbol{s} = \{\boldsymbol{z},\boldsymbol{f},\boldsymbol{l}\}$ and $\boldsymbol{u} = \boldsymbol{l}^{ref}$. Note that, as shown in (a) of Fig. 9, $\boldsymbol{z}$ is the zero moment point (zmp) $\in \mathbb{R}^2$ ($\{x,y\}$ direction is denoted as $\{z_x,z_y\}$), $\{\boldsymbol{f},\boldsymbol{l},\boldsymbol{l}^{ref}\}$ is $\{$muscle tension, muscle length, target muscle length$\}$ of the muscles related to both ankles $\in \mathbb{R}^{12}$. This experiment is difficult in that it is necessary to control the balance of a bipedal humanoid with a flexible body while adapting to changes in its body state. Experiments are conducted on the simulation and the actual robot. In the simulation, the spine pitch joint θ_{s-p} and the offset of the ankle pitch joint angle θ_{a-p}^{offset}, which represents calibration deviation, are treated as changes in the body state. We collected data while changing the body state to nine combinations of $\theta_{s-p} = \{-5.0, 0.0, 5.0\}$ [deg] and $\theta_{a-p}^{offset} = \{-5.0, 0.0, 5.0\}$ [deg]. DPMPB with 2-dimensional PB was trained using 9 datasets with about 300 steps each (about 2700 steps in total). Here, $L_1 = 0.045$, $L_2 = 0.018$, $L_3 = 0.047$, and $L_4 = 0.296$, and since $\boldsymbol{y}_{\{4\}} = \boldsymbol{u}$ is removed from the output by setting $L_{thre} = 0.2$, the network structure is STM. The final loss when training was 0.051. In the actual robot, we handle which shoes in (d) of Fig. 9 are used and the posture change of the upper body as the changes in body state. We collected data in three types of shoes (Hard-Bare, Soft-Pink, and Soft-Navy) and four types of upper body

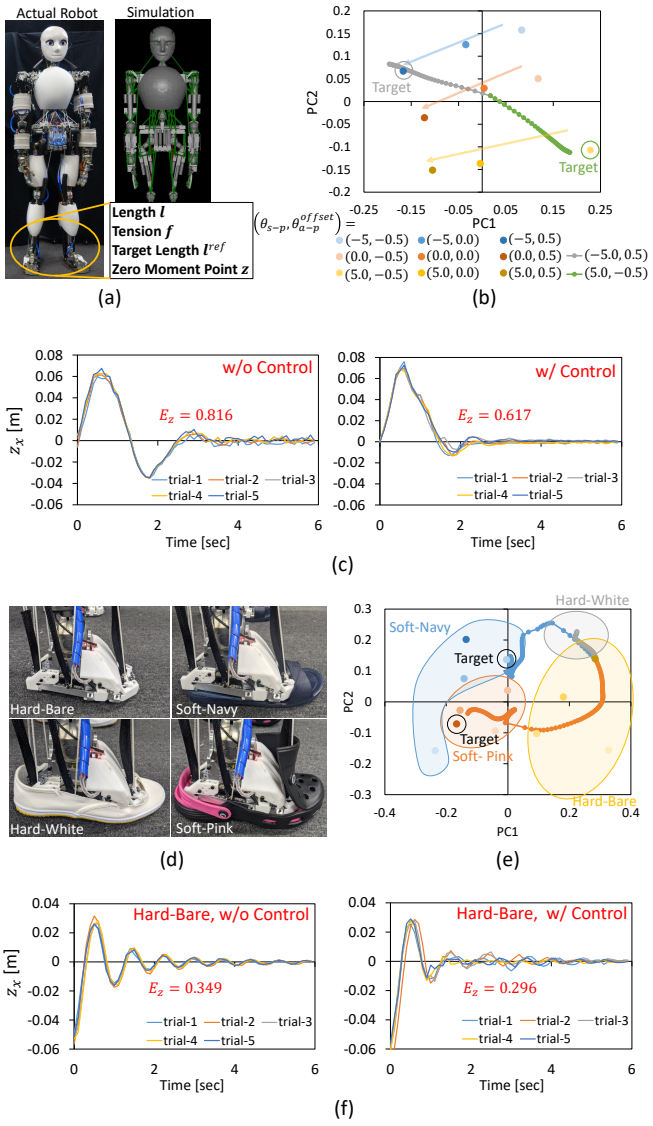

Fig. 9. Experiment of balance control of a musculoskeletal humanoid considering change in shoes. (a) shows the sensors and actuators of the musculoskeletal humanoid Musashi, (b) shows the trained parametric bias and its trajectory when conducting online update of PB in simulation, (c) shows the transition of z_x when using balance control or not in simulation, (d) shows the shoes worn by the robot, (e) shows the trained parametric bias and its trajectory when conducting online update of PB in the actual robot, (f) shows the transition of z_x when using balance control or not in the actual robot.

postures (Hard-White is not used for learning). Since it is necessary to move the robot within the range where it will not fall down, we gradually increase the random displacement of the control input and periodically change the random width of the input to obtain useful data for balance control. DPMPB with 2-dimensional PB was trained using 12 datasets with about 300 steps each (about 3600 steps in total), and the loss when training was 0.074.

(b) of Fig. 9 shows the arrangement of PBs $\boldsymbol{p}_k$ trained in the simulation, and we can see that the space of PB is neatly self-organized along the axes of θ_{s-p} and θ_{a-p}^{offset}. This is also the case in the actual robot. (e) of Fig. 9 shows the

arrangement of PBs $\boldsymbol{p}_k$ trained in the actual robot, where $\boldsymbol{p}_k$ is grouped for each pair of shoes and the space of PB is self-organized. When online learning is performed for both (b) and (e), we can see that the current $\boldsymbol{p}$ gradually approaches $\boldsymbol{p}_k$ trained in the current body state, which is represented by "Target". In addition, the PB of Hard-White, which is not used in the training, is above PBs of Hard-Bare, which may reflect the fact that Hard-White and Hard-Bare, Soft-Navy and Soft-Pink have similar sole hardness. Here, the average number of data steps used for online update is 280, which indicates that the adaptation takes about 56 seconds.

In (c) and (f) of Fig. 9, we show the results of the balance control experiment using DPMPB. After updating PB online, we can perform the balance control by setting h_{loss} in Section II-E as follows,

$$h_{loss}(\boldsymbol{s}_{seq}^{pred}, \boldsymbol{u}_{seq}^{opt}) = ||\boldsymbol{z}_{seq}^{pred} - \boldsymbol{z}_{seq}^{ref}||_2 + w_1||\boldsymbol{f}_{3:T}^{pred} - \boldsymbol{f}_{2:T-1}^{pred}||_2 + w_2||\boldsymbol{l}_{3:T}^{pred} - \boldsymbol{l}_{2:T-1}^{pred}||_2 + w_3||\boldsymbol{u}_{seq}^{opt}||_2 \quad (9)$$

where $w_{\{1,2,3\}}$ denotes the constant weight. (c) shows the results of five transitions of z_x after an external force of 30 N is applied to the waist link for 0.2 s in the simulation. It can be seen that the convergence of z_x to the external force becomes faster by the balance control. In addition, the average E_z of the total error of $|z_x|$ for 6 seconds (for 30 steps) drops from 0.816 to 0.617 with the balance control. Note that the usual PD control is slow in convergence due to the delay caused by the flexibility of the body. The result was $E_z = 0.791$ when setting the PD gains as (0.03, 0.1), and it was not much different from the result without any control, even after the parameters are manually tuned. (d) shows the results of five transitions of z_x after applying 15 N force to the waist link and then releasing it in the actual robot. It can be seen that the convergence of z_x to the external force becomes faster by the balance control. Also, E_z dropped from 0.349 to 0.296 with the balance control.

F. Dynamic Cloth Manipulation Considering Material Change

In this section, we deal with a dynamic cloth manipulation model by a musculoskeletal wheeled robot Musashi-W as a modeling difficulty. As a temporal model change, we deal with the change of cloth material. For sensor state and control input, we set $\boldsymbol{s} = \{\boldsymbol{z}, \boldsymbol{f}, \boldsymbol{l}\}$ and $\boldsymbol{u} = \{\boldsymbol{\theta}^{ref}, \boldsymbol{k}^{ref}\}$. Note that, as shown in (a) of Fig. 10, $\boldsymbol{z}$ is the image compressed by AutoEncoder, $\{\boldsymbol{f}, \boldsymbol{l}\}$ is {muscle tension, muscle length} of the muscles related to both arms $\in \mathbb{R}^{20}$ (5 DOFs of the shoulder and elbow are used), $\boldsymbol{\theta}^{ref}$ is the target joint angle of the arm in the sagittal plane $\in \mathbb{R}^2$ (the pitch joints of the shoulder and elbow, the same for both arms), and $\boldsymbol{k}^{ref}$ is the target value for the stiffness of the arm $\in \mathbb{R}^1$. The cloth is composed of soft or hard foam sheets and is varied by stacking {1, 2, 3} sheets (denoted as {soft-1, soft-2, hard-1, hard-2, hard-3}). This experiment is difficult in that it is necessary for a robot with a flexible body to dynamically manipulate a flexible object, taking into account the change of its material. We collected data from random control inputs and human operations using GUI while changing the cloth material to {soft-1, soft-2, hard-1, hard-3}. DPMPB with 2-dimensional PB was trained using 16 datasets with about 400 steps each (about 6400 steps in total). Here, $L_1 = 0.122$, $L_2 = 0.193$, $L_3 = 0.037$, $L_4 = 0.233$, and $L_5 = 0.288$, and since $\boldsymbol{y}_{\{4,5\}} = \boldsymbol{u}$ is removed from the output by setting $L_{thre} = 0.2$, the network structure is STM. The final loss when training was 0.082.

In (b) of Fig. 10, we show the arrangement of the trained PBs $\boldsymbol{p}_k$. We can see that the space of PB is neatly self-organized according to the hardness and thickness of the cloth. The trajectory of PB, represented as "-traj" in (b), shows that the current $\boldsymbol{p}$ approaches $\boldsymbol{p}_k$ trained in the current

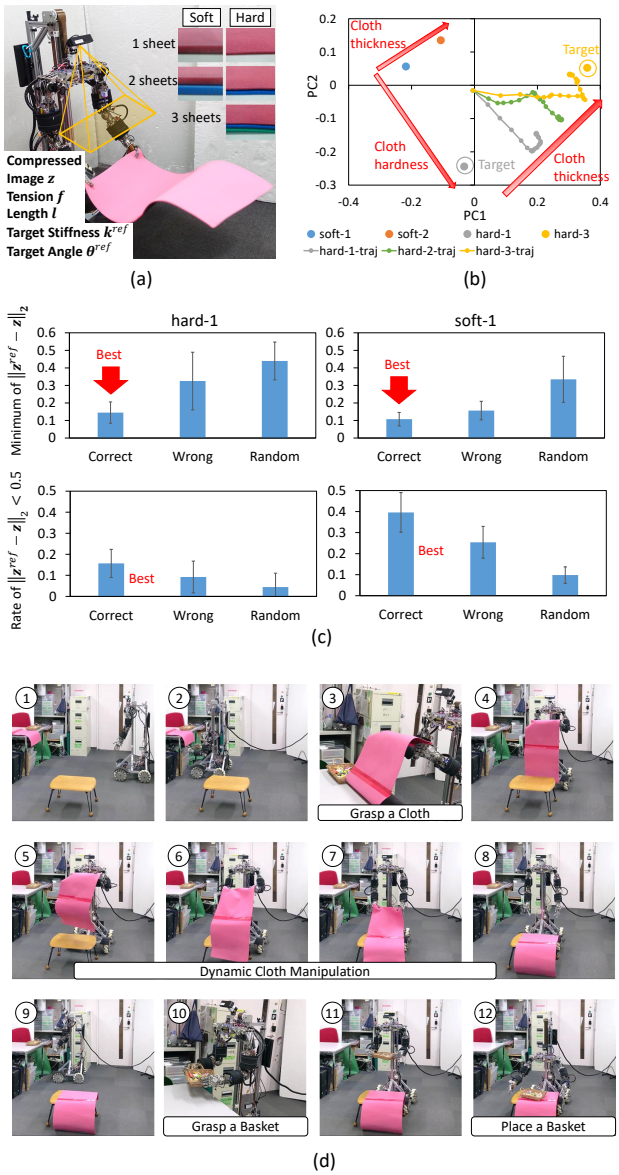

Fig. 10. Experiment of cloth manipulation considering material change. (a) shows the sensors and actuators of the musculoskeletal wheeled robot Musashi-W and the various cloths used in this experiment, (b) shows the trained parametric bias and its trajectory when conducting online update of PB, (c) shows the minimum value of $||\boldsymbol{z}^{ref} - \boldsymbol{z}||_2$ and the rate of $||\boldsymbol{z}^{ref} - \boldsymbol{z}||_2 < 0.5$ when conducting cloth manipulation with Correct / Wrong / Random settings, and (d) shows an integrated table setting experiment using the developed dynamic cloth manipulation.

cloth when the online learning of PB is performed. For hard-2, which is not used for training, PB is at about the middle of hard-1 and hard-3. Here, the average number of data steps used for online update is 120, which indicates that the adaptation takes about 24 seconds.

In (c) of Fig. 10, we show the results of dynamic cloth manipulation experiment using DPMPB. After updating PB online, we can transition the cloth to the desired state by setting h_{loss} in Section II-E as follows,

$$h_{loss}(\boldsymbol{s}_{seq}^{pred}) = ||\boldsymbol{m}_t \otimes (\boldsymbol{z}_{seq}^{ref} - \boldsymbol{z}_{seq}^{pred})||_2 + w_1||\boldsymbol{f}_{seq}^{pred}||_2 \quad (10)$$

where $\boldsymbol{z}^{ref}$ is the compressed latent value of the given target image and w_1 is the constant weight. This control realizes the target image while suppressing the muscle tension. Note that $\boldsymbol{m}_t$ ($\in \{0,1\}^{N_{step}^{control}}$) is a vector in which 1 appears at every $N_{periodic}^{control}$ step and is otherwise 0 ($N_{periodic}^{control}$ is a constant value). At each step, the vector is shifted to the left and 0 or 1 is inserted from the right according to $N_{periodic}^{control}$. This makes it possible to make the cloth state $\boldsymbol{z}$ closer to the target value for each $N_{periodic}^{control}$ step, and enables the control to handle the dynamic state of the cloth that can be realized only for a moment. (c) shows the minimum value of $||\boldsymbol{z}^{ref} - \boldsymbol{z}||_2$ and the ratio of $||\boldsymbol{z}^{ref} - \boldsymbol{z}||_2 < 0.5$, which refers to the state where the current image is close to the target image. In addition, the following cases are compared: when PB is correctly set to the value of hard-1 for hard-1 and soft-1 for soft-1 (Correct), when PB is incorrectly set to the value of soft-1 for hard-1 and hard-1 for soft-1 (Wrong), and when the robot is moved randomly (Random). The minimum value is smaller in the order of Correct, Wrong, and Random, and the ratio of $||\boldsymbol{z}^{ref} - \boldsymbol{z}||_2 < 0.5$ is larger in the order of Correct, Wrong, and Random, indicating that the target state can be realized accurately by the control, and that the recognition of PB is important for this control.

In (d) of Fig. 10, we show a series of table-setting experiments using this dynamic cloth manipulation model. Musashi-W picked up a cloth, laid it on the table as a tablecloth using this method, and placed a basket of sweets on the cloth.

IV. Discussion

We summarize the results obtained from our experiments. In this study, we found that DPMPB can learn a dynamics model between a flexible hand and objects, a visual feedback model of a low-rigidity body, a probabilistic relationship between a wheeled base and a floor, an imitation model of human motion, a balance model of a musculoskeletal humanoid, and a dynamic manipulation model of flexible cloth. DPMPB can learn not only the dynamics inside the robot body, but also the relationship it has with tools, objects, and environment. The parametric bias can embed differences in grasped objects, robot configurations, floor friction, human motion styles, shoes and cloth materials. DPMPB can recognize and adapt to the current state by updating only the parametric bias so that the network prediction matches the current state. It is also possible to recognize objects and states that are not used in training as intuitively correct dynamics in the space of PB. In the case of control using STM, the desired behavior can be achieved by constructing a loss function with a combination of minimization and maximization of a value, minimization of the error with a certain target value, and minimization of the change from the value of the previous time step. As an application of DPMPB, it was found that the stabilization control to minimize variance is possible by introducing the mean-variance representation to the network output, and that it is possible not only to adapt to the current state but also to actively change the motion style depending on the update rule of parametric bias in imitation learning. A series of behaviors using DPMPB is also possible, and DPMPB is expected to realize further behaviors to overcome modeling difficulties and temporal model changes.

A. Limitations

There are three main limitations in this study. First, the control period cannot be increased. The basic control period of the experiments on STM presented so far is 5 Hz, which is not fast. This is because the optimization process using iterative backpropagation and gradient descent methods in the control takes a long time. Note that the control period of CTM can be increased because the control input can be calculated from only the forward propagation. With the current network configuration, forward propagation takes about 10 msec and backward propagation takes about 40 msec. On the other hand, the optimization process can be accelerated by fixing the number of LSTM expansions in the control and learning the network as fully-connected layers [24]. We would like to continue development so that the robot can handle dynamic behaviors that require a control period of about 100 Hz.

Second, in this study, tasks can only be controlled within the range that can be expressed by the loss function. Although the tasks handled in this study were successful, it is difficult to apply this method to tasks whose loss functions are more complex and difficult to be described by humans. It needs to be modified to be able to handle more abstract input/output variables.

Third, we focus on the size of state/control space and data collection. In this study, we mainly trained the dynamics that is close to the robot body and easy to handle, but it is not suitable for motion planners with discrete states or task settings with large dimensions of control inputs and sensor states. It is also not suitable for tasks such as balance control or walking control where data collection itself is difficult. In order to handle discrete states and large state/control space in the real world, efficient data collection, definition of primitives, etc. would be needed. We also believe that it will be important to develop a method in which the robot itself determines whether the data collection is sufficient to perform the task and what the prediction error of the current network is. Although it should be theoretically possible to handle any kind of state transition, it is necessary to continue to verify the prediction accuracy and control performance when dealing with more sensors and nonlinearities.

V. CONCLUSION

In this study, we generalize a theory of deep predictive model learning with parametric bias, which can overcome modeling difficulties and temporal model changes in the relationship among the robot body, tools, target objects, and the environment. We constructed a predictive model using a neural network that overcomes modeling difficulties and an online update method of parametric bias that overcomes temporal model changes. In the network structure of the state transition model, the control input that makes the predicted state closer to the target state is calculated by repeating back-propagation and gradient descent methods for the network input, while in the network structure of the control transition model, the control input is calculated only from the forward propagation. The parametric bias, which can implicitly embed differences in dynamics into the network input, can be updated to make the predicted sensor state closer to the current sensor state in order to adapt to the current body and environment. Based on this predictive model learning, we have succeeded in the following tasks: grasping control and object recognition for a flexible hand, visual feedback for a low-rigidity robot, environmentally adaptive control with variance minimization for a wheeled robot, imitation learning for a musculoskeletal robot considering its motion style, balance control considering change in shoes for a full-body musculoskeletal humanoid, and dynamic cloth manipulation for a musculoskeletal wheeled robot considering cloth material change, and confirmed the effectiveness of DPMPB for various robot tasks. In the future, we would like to continue development so that the robot can autonomously collect data in the real world, acquire models of its body, tools, objects, and the environment, and perform various continuous tasks by coping with modeling difficulties and temporal model changes.

REFERENCES

[1] H. Kobayashi, K. Hyodo, and D. Ogane, "On Tendon-Driven Robotic Mechanisms with Redundant Tendons," *The International Journal of Robotics Research*, vol. 17, no. 5, pp. 561–571, 1998.

[2] C. C. Kemp and A. Edsinger, "Robot manipulation of human tools: Autonomous detection and control of task relevant features," in *Proceeding of the 2006 International Conference on Development and Learning*, 2006, pp. 1–6.

[3] C. Lee, M. Kim, Y. J. Kim, N. Hong, S. Ryu, H. J. Kim, and S. Kim, "Soft robot review," *International Journal of Control, Automation and Systems*, vol. 15, no. 1, pp. 3–15, 2017.

[4] D. Tanaka, S. Arnold, and K. Yamazaki, "EMD Net: An Encode-Manipulate-Decode Network for Cloth Manipulation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1771–1778, 2018.

[5] Y. Kuniyoshi and S. Suzuki, "Dynamic emergence and adaptation of behavior through embodiment as coupled chaotic field," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2042–2049.

[6] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[7] K. Kawaharazuka, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Object Recognition, Dynamic Contact Simulation, Detection, and Control of the Flexible Musculoskeletal Hand Using a Recurrent Neural Network With Parametric Bias," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4580–4587, 2020.

[8] K. Kawaharazuka, N. Kanazawa, K. Okada, and M. Inaba, "Self-Supervised Learning of Visual Servoing for Low-Rigidity Robots Considering Temporal Body Changes," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7881–7887, 2022.

[9] K. Kawaharazuka, K. Shinjo, Y. Kawamura, K. Okada, and M. Inaba, "Environmentally Adaptive Control Including Variance Minimization Using Stochastic Predictive Network with Parametric Bias: Application to Mobile Robots," in *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 8381–8387.

[10] K. Kawaharazuka, Y. Kawamura, K. Okada, and M. Inaba, "Imitation Learning with Additional Constraints on Motion Style using Parametric Bias," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5897–5904, 2021.

[11] K. Kawaharazuka, Y. Ribayashi, A. Miki, Y. Toshimitsu, T. Suzuki, K. Okada, and M. Inaba, "Learning of Balance Controller Considering Changes in Body State for Musculoskeletal Humanoids (in press)," in *Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[12] K. Kawaharazuka, A. Miki, M. Bando, K. Okada, and M. Inaba, "Dynamic Cloth Manipulation Considering Variable Stiffness and Material Change Using Deep Predictive Model With Parametric Bias," *Frontiers in Neurorobotics*, vol. 16, pp. 1–16, 2022.

[13] J. Tani, "Self-organization of behavioral primitives as multiple attractor dynamics: a robot experiment," in *Proceedings of the 2002 International Joint Conference on Neural Networks*, 2002, pp. 489–494.

[14] T. Ogata, H. Ohba, J. Tani, K. Komatani, and H. G. Okuno, "Extracting multi-modal dynamics of objects using RNNPB," in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 966–971.

[15] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3D biped," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2849–2854.

[16] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, 2018, pp. 5628–5635.

[17] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data Efficient Reinforcement Learning for Legged Robots," in *Proceedings of the 2019 Conference on Robot Learning*, 2019, pp. 1–10.

[18] K. Schmeckpeper, A. Xie, O. Rybkin, S. Tian, K. Daniilidis, S. Levine, and C. Finn, "Learning Predictive Models from Observation and Interaction," in *Proceedings of 2020 European Conference on Computer Vision*, 2020, pp. 708–725.

[19] K. Kawaharazuka, T. Ogawa, J. Tamura, and C. Nabeshima, "Dynamic Manipulation of Flexible Objects with Torque Sequence Using a Deep Neural Network," in *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, 2019, pp. 2139–2145.

[20] K. Kawaharazuka, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Musculoskeletal AutoEncoder: A Unified Online Acquisition Method of Intersensory Networks for State Estimation, Control, and Simulation of Musculoskeletal Humanoids," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2411–2418, 2020.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015, pp. 1–15.

[23] K. Kawaharazuka, S. Makino, K. Tsuzuki, M. Onitsuka, Y. Nagamatsu, K. Shinjo, T. Makabe, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Component Modularized Design of Musculoskeletal Humanoid Platform Musashi to Investigate Learning Control Systems," in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 7294–7301.

[24] K. Kawaharazuka, K. Tsuzuki, S. Makino, M. Onitsuka, K. Shinjo, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Task-specific Self-body Controller Acquisition by Musculoskeletal Humanoids: Application to Pedal Control in Autonomous Driving," in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 813–818.