IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Object Recognition, Dynamic Contact Simulation, Detection, and Control of the Flexible Musculoskeletal Hand Using a Recurrent Neural Network with Parametric Bias

Kento Kawaharazuka[1], Kei Tsuzuki[1], Moritaka Onitsuka[1], Yuki Asano[1], Kei Okada[1]
Koji Kawasaki[2], and Masayuki Inaba[1]

Abstract— **The flexible musculoskeletal hand is difficult to modelize, and its model can change constantly due to deterioration over time, irreproducibility of initialization, etc. Also, for object recognition, contact detection, and contact control using the hand, it is desirable not to use a neural network trained for each task, but to use only one integrated network. Therefore, we develop a method to acquire a sensor state equation of the musculoskeletal hand using a recurrent neural network with parametric bias. By using this network, the hand can realize recognition of the grasped object, contact simulation, detection, and control, and can cope with deterioration over time, irreproducibility of initialization, etc. by updating parametric bias. We apply this study to the hand of the musculoskeletal humanoid Musashi and show its effectiveness.**

I. Introduction

Various robotic hands have been developed so far [1]–[5]. While there are many dexterous robotic hands with multiple rigid links [1], [2], flexible robotic hands [3]–[5] are prevailing in terms of the adaptable grasping and impact response. The rubber hand structure of [3] is driven by air pressure, and the hand of [4] is highly biomimetic with the elastic pulley system. In this study, we use the flexible and strong musculoskeletal hand [5] with machined spring fingers and a larger number of sensors than those hands. However, there are several problems concerning recognition of the grasped object, contact detection, and contact control with such a flexible hand [3]–[5].

First, its modeling is difficult, and therefore, controls directly using geometric models are challenging. This problem is widely known now, and various learning-based object recognitions and contact controls have been studied. [6] predicted the success rate of grasping and realized regrasping using reinforcement learning. [7] trained imitation learning-based deep visuomotor policies and realized various manipulation tasks with a simulated five-fingered hand. While these studies are applied to only rigid hands, some studies can handle flexible hands as explained below. [8] applied reinforcement learning of in-hand manipulation with the under-actuated two-fingered robotic hand. [9] realized the classification of the grasped objects using the pneumatic flexible robotic hand. However, these applications are limited,

[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@jsk.t.u-tokyo.ac.jp

[2] The author is associated with TOYOTA MOTOR CORPORATION.

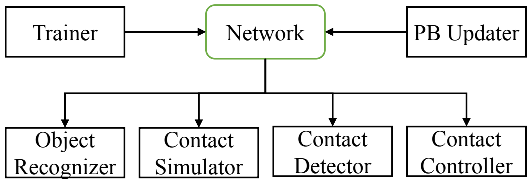

Fig. 1: Overview of the developed system.

such as control on a two-dimensional plane or only grasping classification.

Second, the modeling of the flexible hand is not only difficult but also can change constantly. Especially, soft materials such as rubber [3] and springs [4], [5] significantly deteriorate over time, and also other materials such as wire and metal deteriorate due to poor use. Also, joint angle sensors sometimes cannot be attached to the flexible hand [3]–[5], and it is difficult to initialize the joint angles of the fingers. In the case of the hand [5] used in this study, there are no joint angle sensors, and so the origin of the joint angle is initialized when we manually extend the fingers. However, the initialization is hard to reproduce, and the irreproducibility of initialization changes the model we construct. In addition, when we modelize a sensor state transition of the hand by control commands, the model depends on the surrounding environment (e.g. grasped objects). While online learning methods of intersensory networks [10], [11] have been developed for musculoskeletal humanoids to solve these problems, online learning becomes difficult if the dimension to be learned increases, and so these methods can be applied to only static movements. Therefore, it is necessary to develop a method stably modifying and adapting the network to the current hand dynamics, including the difference of initialization, deterioration, and grasped objects, by changing only a small part of the network parameters.

Third, as seen in the introduced studies, various components such as recognition of the grasped object, contact detection, and contact control have been developed individually [6]–[9]. While the individual network can specialize in each component, we consider that only one network representing sensor state transition is enough to handle these components in a relatively low layer. By using the integrated network, managing each network for each component is not necessary, and the parameter update making a certain component better can affect other components. Therefore, it is desirable to train only one integrated network for these components.

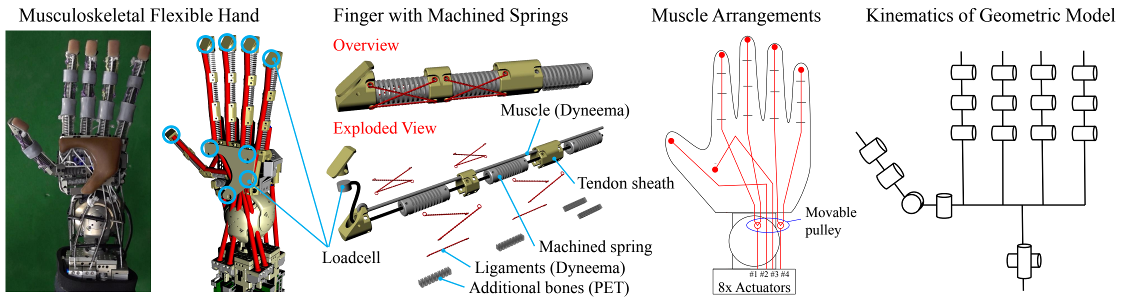

Fig. 2: The five-finger flexible musculoskeletal hand [5] of the musculoskeletal humanoid Musashi [12].

In this study, we construct a sensor state equation represented by a recurrent neural network with parametric bias [13] (Hand Dynamics Network, HADYNET), and use it for recognition of the grasped object, contact simulation, detection, and control (Fig. 1). We implicitly embed the information of deterioration, initialization, and grasped objects into the parametric bias (PB). By updating only PB, the network can adapt to the current hand dynamics and can cope with deterioration over time, irreproducibility of initialization, the change in grasped objects, etc. We apply this study to the flexible hand [5] of the musculoskeletal humanoid Musashi [12] and verify its effectiveness. The contributions of this study are shown below.

- Acquisition of a sensor state equation represented by a recurrent neural network with parametric bias for the flexible musculoskeletal hand.
- Coping with the change of the model due to deterioration over time, irreproducibility of initialization, the difference in grasped object, etc. by updating only parametric bias.
- Recognition of the grasped object, contact simulation, detection, and control using the trained network.

II. Musculoskeletal Hand

We show the musculoskeletal hand [5] used in this study, in Fig. 2. This hand attached to the musculoskeletal humanoid Musashi [12] has five fingers, and each finger joint is composed of a flexible machined spring. To make the machined springs anisotropic, Dyneema and PET plate imitating ligaments are attached to the side of the fingers.

In the forearm of Musashi, there are eight muscle actuators [14], and three and five of them are assigned to move the wrist and fingers, respectively. Two of the five tendons for fingers actuate the middle/index and ring/little fingers, respectively. Two tendons branched by a pulley control the two fingers at the same time. The other two of the five tendons actuate the thumb, and the last one can change the stiffness of fingers by pressing on the machined springs.

Nine loadcells are distributed in each fingertip and the palm, and these arrangements are shown in the middle figure of Fig. 2. Muscle tension and length can be measured from the muscle actuator [14]. We represent the loadcell value as $\boldsymbol{f}_{contact}$, the current muscle length as $\boldsymbol{l}$, and muscle tension as $\boldsymbol{f}$. The kinematics of the geometric model of this hand is shown in the right figure of Fig. 2. The joints of machined springs are simply represented by rotational joints. Although finger joint angles cannot be measured, the wrist joint angles $\boldsymbol{\theta}$ can be measured by a joint module [12] in the wrist joint.

The dimensions of $\boldsymbol{l}$ and $\boldsymbol{f}$ are 8, that of $\boldsymbol{f}_{contact}$ is 9, and that of $\boldsymbol{\theta}$ is 2. In the case of using only 4 muscle actuators #1–#4 directly involving finger joints, we represent these muscle tensions and lengths as $\boldsymbol{f}_{finger}$ and $\boldsymbol{l}_{finger}$.

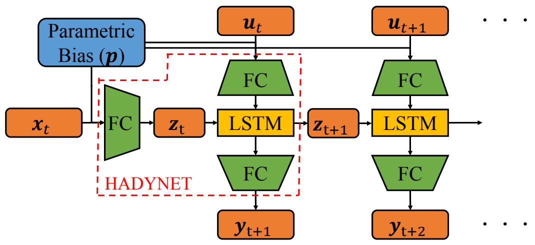

Fig. 3: Network structure of hand dynamics network (HADYNET).

III. Hand Dynamics Network

We show the detailed network structure in Fig. 3 and the whole system in Fig. 4.

A. Network Structure

HADYNET is the network making use of the recurrent neural network with parametric bias (RNNPB) proposed by J. Tani [13]. Several studies using RNNPB have been conducted [15], [16] so far, and RNNPB has been used to embed multiple dynamics of various motion sequences into one network. In this study, we make use of RNNPB as the sensor state equation with multiple dynamics caused by deterioration over time, irreproducibility of initialization, the difference in grasped objects, etc.

HADYNET is represented by functions $\boldsymbol{h}_{\{1,2\}}$ as below,

$$\boldsymbol{z}_t = \boldsymbol{h}_1(\boldsymbol{x}_t, \boldsymbol{p}) \tag{1}$$

$$\boldsymbol{y}_{t+1} = \boldsymbol{h}_2(\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{p}) \tag{2}$$

where $\boldsymbol{x}$ is the initial sensor state of the hand, $\boldsymbol{z}$ is the hidden state, $\boldsymbol{u}$ is the control command, $\boldsymbol{y}$ is the observed sensor state, $\boldsymbol{p}$ is the parametric bias (PB), t is the current time step, and $\bullet_t$ is the value at the time step t. In this study, $\boldsymbol{x}$ represents $(\boldsymbol{l}^T, \Delta\boldsymbol{l}^T, \boldsymbol{f}^T, \boldsymbol{f}^T_{contact}, \boldsymbol{\theta}^T, \Delta\boldsymbol{\theta}^T)^T$ ($\Delta\{\boldsymbol{l}, \boldsymbol{\theta}\}$ is the difference value from the previous time step, and the dimension of $\boldsymbol{x}$ is 37). Also, $\boldsymbol{u}$ represents $\Delta\boldsymbol{l}^{ref}$ ($\Delta\boldsymbol{l}^{ref}$ is the change of the target muscle length, and the dimension of $\Delta\boldsymbol{l}^{ref}$ is 8), $\boldsymbol{y}$ represents $(\boldsymbol{l}^T, \boldsymbol{f}^T, \boldsymbol{f}^T_{contact}, \boldsymbol{\theta}^T)^T$ (the dimension of $\boldsymbol{y}$ is 27). We call Eq. 1 and Eq. 2 HADYNET. Thus,

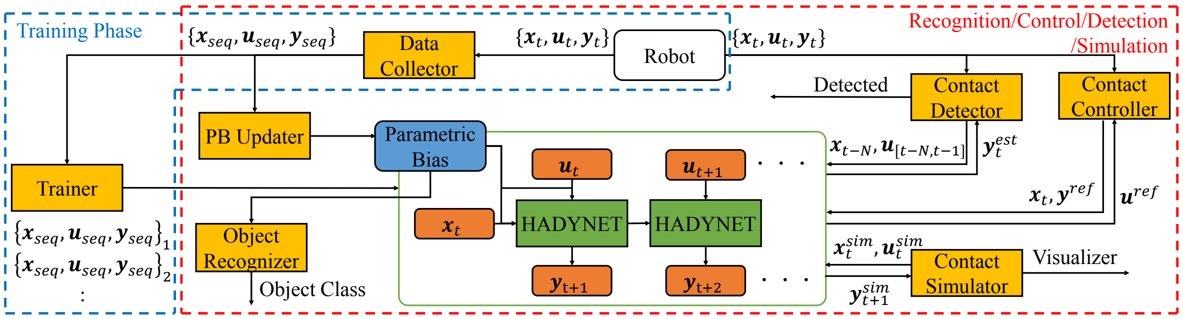

Fig. 4: The overall system using the hand dynamics network.

HADYNET represents the state equation of all the sensors equipped with the hand by the control command of target muscle velocity.

The detailed network structure is shown in Fig. 3. FC represents a fully-connected layer, and LSTM represents Long Short-Term Memory [17]. The number of units of LSTM is 128, and the dimension of PB is 8. The number of units of FC can be determined from the input and output dimensions, and after feeding the input value into FC, the value is fed into Batch Normalization [18] and ReLU [19]. The cell state of LSTM is initialized to 0.

In this study, the interval of the adjacent time steps is 0.2 seconds.

B. Training of HADYNET

To embed multiple dynamics into PB, we collect various data with various dynamics, and train HADYNET as each data corresponds to different PB.

First, we prepare the hand dynamics state k with different initializations, deteriorations, and grasped objects. We conduct various motions with the hand of each dynamics state k, and obtain the N steps motion data of $D_k = \{\boldsymbol{x}_{[0,N)}, \boldsymbol{u}_{[0,N)}, \boldsymbol{y}_{[0,N)}\}_k$. In Fig. 4, $\bullet_{[0,N)}$ is abbreviated to $\bullet_{seq}$.

Second, we convert the collected data. We determine N^{train}, how many time steps are predicted by HADYNET. We extract N^{train} steps of consecutive data from D_k every one time step, and we represent all the extracted data as $D'_k = \{\{\boldsymbol{x}_0, \boldsymbol{u}_{[0,N^{train})}, \boldsymbol{y}_{[1,N^{train}+1)}\}_k, \{\boldsymbol{x}_1, \boldsymbol{u}_{[1,N^{train}+1)}, \boldsymbol{y}_{[2,N^{train}+2)}\}_k, \cdots\}$. Regarding $\boldsymbol{x}$, only the initial $\boldsymbol{x}$ is necessary. We execute this process for all the collected data and prepare $D'_{k=\{1,2,\ldots,K\}}$ (K represents the number of the collected hand dynamics state).

Third, we train HADYNET using D'. We shuffle D' and train HADYNET by setting the batch size N^{train}_{batch} and the epoch N^{train}_{epoch}. We use $\boldsymbol{p}_k$ as PB of the hand dynamics state k (PB is the same among the same hand dynamics state but different among the different hand dynamics state), and this $\boldsymbol{p}_k$ is implicitly trained with HADYNET. All $\boldsymbol{p}_k$ are initialized to $\boldsymbol{0}$ before training.

In this study, N is different regarding each hand dynamics state, and we set $N^{train} = 10$, $N^{train}_{batch} = 100$, and $N^{train}_{epoch} = 200$.

C. Update of Parametric Bias

Although the network weight of HADYNET and PB are trained in Section III-B, we do not need the trained PB when using HADYNET except for in the object recognition task. When changing the grasped objects or initializing again, we collect the data D' regarding the current hand dynamics state, fix the network weight of HADYNET, and update only PB. As explained in Section I, by updating only PB rather than all the network weight, we can adapt HADYNET to be close to the current hand dynamics state, while avoiding an over-fitting problem. Because only the low dimensional space of PB is updated, only the hand dynamics of the network is adapted while keeping the overall sensor state equation. In Section III-B, if we train HADYNET with various grasped objects, initializations, and deterioration, the multiple dynamics states are constructed in PB. By searching the PB fitting to the current sensor transition, we can obtain the current hand dynamics state. We represent the batch size as N^{update}_{batch} and the number of epochs as N^{update}_{epoch}, and use MomentumSGD as the update rule.

Also, although the method to train HADYNET offline using the collected D' is more stable, the online update of PB is enabled because the over-fitting problem can be avoided. We determine the data size N^{online}_{thre} that starts the online update of PB, and the maximum accumulated data size N^{online}_{max}, by which the update can be executed online. We accumulate data as in Section III-B, and when the accumulated data size exceeds N^{online}_{thre}, PB starts to be updated by setting the batch size N^{online}_{batch} and the number of epochs N^{online}_{epoch}. When the accumulated data size exceeds N^{online}_{max}, the oldest data is removed.

In this study, $N^{\{update,online\}}_{batch} = N$ (N is the number of data in D'), $N^{update}_{epoch} = 20$, $N^{online}_{epoch} = 3$, $N^{online}_{thre} = 100$, and $N^{online}_{max} = 200$.

D. Recognition of Grasped Object

The robot can recognize the grasped object by making use of the PB updated in Section III-C. Regarding the K hand dynamics states used in Section III-B, we save the trained PB $\boldsymbol{p}_k$ and the name of each grasped object. We can visually plot these $\boldsymbol{p}_k$ by reducing the dimension of PB to two-dimensional space using principal component analysis (PCA). After updating PB in Section III-C, we convert the updated $\boldsymbol{p}$ by the converter of PCA as above and plot it in the two-dimensional space. We can visually obtain the current object name from the position of $\boldsymbol{p}$ in $\boldsymbol{p}_k$. Also, by the nearest neighbor method, the robot can recognize which object is the closest to the current grasped object. This object recognition is possible because the difference of hand dynamics caused by initialization and deterioration is smaller than that caused by the difference in grasped objects. This was found through subsequent experiments, as the difference of grasped objects appeared like in Fig. 9 and Fig. 10 even if we obtain PB while changing the initialization and grasped objects.

E. Contact Simulation

The contact simulation is executed only by forwarding HADYNET. We set the initial sensor state x^{sim}_{t}, and by feeding the control command u^{sim}_{t} into HADYNET, y^{sim}_{t+1} can be obtained. We can visually simulate the force and position of the hand by drawing $\boldsymbol{f}$ as color gradation, $\boldsymbol{f}_{contact}$ as the size of the force arrows, and $\boldsymbol{\theta}$ as the joint angle. Also, just by changing PB, we can observe the change of hand dynamics with various grasped objects and initializations. This simulation can be used for reinforcement learning.

F. Contact Detection

From the prediction error of HADYNET, we can execute contact detection (anomaly detection) of the hand. First, before contact detection, we collect D' and update PB as in Section III-C without grasping anything. We determine the time steps N^{detect} ($N^{detect} \leq N^{train}$) to be predicted by HADYNET, and regarding D', we calculate the average and covariance matrix of the prediction error of HADYNET after N^{detect} time steps. In detail, we extract $\boldsymbol{x}_{i-N^{detect}}$ and $\boldsymbol{u}_{[i-N^{detect},i)}$ when setting i as the last time step of the N^{detect} steps sequence, calculate the error between $\boldsymbol{y}^{est}_{i}$ predicted by using the extracted data with HADYNET and $\boldsymbol{y}_{i}$ in D', and obtain the average μ_{e} and the covariance matrix Σ_{e} of the errors regarding all the sequence in D'.

Second, we will explain the actual contact detection phase. We always accumulate $\boldsymbol{x}$ and $\boldsymbol{u}$ from the current time step t to $t-N^{detect}$. At t, we extract $\boldsymbol{x}_{t-N^{detect}}$ and $\boldsymbol{u}_{[t-N^{detect},t)}$, and predict $\boldsymbol{y}^{est}_{t}$ through HADYNET. Then, we obtain $\boldsymbol{y}_{t}$ and calculate Mahalanobis distance as shown below,

$$d = \sqrt{(\boldsymbol{e}_{t}-\mu_{e})^{T}\Sigma^{-1}_{e}(\boldsymbol{e}_{t}-\mu_{e})} \tag{3}$$

where $\boldsymbol{e}_{t} = \boldsymbol{y}^{est}_{t} - \boldsymbol{y}_{t}$. When d exceeds the threshold C^{detect}, we regard that a motion different from the prediction occurs, and so the unpredicted contact is detected. Although we can use 3σ of d in D' as C^{detect}, as too many contacts should not be detected, we set a higher constant value.

In this study, we set $C^{detect} = 100$.

G. Contact Control

By optimizing the control command to make the predicted sensor state close to the target value, the robot can realize the target contact state. Here, the contact state means the values of $\boldsymbol{f}$ and $\boldsymbol{f}_{contact}$. This control is a method applying [20] to HADYNET.

First, we determine the number of time steps $N^{control}$ ($N^{control} \leq N^{train}$) to be predicted (control horizon) and the target contact state $\boldsymbol{y}^{ref}$. Second, we determine the control command $\boldsymbol{u}^{opt}_{[t,t+N^{control})}$ before optimization (we represent it as $\boldsymbol{u}^{opt}_{seq}$ below). By obtaining $\boldsymbol{x}_{t}$ at the current time step and feeding it with $\boldsymbol{u}^{opt}_{seq}$ into HADYNET, the predicted sensor state of $\boldsymbol{y}^{est}_{[t+1,t+N^{control}+1)}$ is obtained. Then, the loss of L is calculated by the loss function h_{loss}, and $\boldsymbol{u}^{opt}_{seq}$ is optimized as shown below,

$$L = h_{loss}(\boldsymbol{y}^{est}_{seq}, \boldsymbol{y}^{ref}_{seq}) \tag{4}$$

$$\boldsymbol{g} = \partial L / \partial \boldsymbol{u}^{opt}_{seq} \tag{5}$$

$$\boldsymbol{u}^{opt}_{seq} \leftarrow \boldsymbol{u}^{opt}_{seq} - \alpha \boldsymbol{g}/||\boldsymbol{g}||_{2} \tag{6}$$

where $\boldsymbol{y}^{est}_{seq}$ represents $\boldsymbol{y}^{est}_{[t+1,t+N^{control}+1)}$, $\boldsymbol{y}^{ref}_{seq}$ represents a vector vertically arranging $N^{control}$ vectors of $\boldsymbol{y}^{ref}$, $||\bullet||_{2}$ represents L2 norm, and α is an update rate. $\boldsymbol{u}^{opt}_{seq}$ is updated using $N^{control}_{batch}$ kinds of α, $\boldsymbol{u}^{opt}_{seq}$ with the minimum L in the batch is adopted, and the gradient g is calculated again. This is repeated $N^{control}_{epoch}$ times. Also, we use $\boldsymbol{u}^{opt}_{\{t,\cdots,t+N^{control}-1,t+N^{control}-1\}}$, in which $\boldsymbol{u}^{opt}_{[t-1,t+N^{control}-1)}$ optimized at the previous time step is shifted and its last term is replicated, as $\boldsymbol{u}^{seq}_{control}$. After optimization, $\boldsymbol{u}^{opt}_{t}$ is sent to the actual robot.

Here, we consider the design of the loss function h_{loss}. In this study, we mainly consider the stabilization of the grasp and aim to keep the initial contact state at all times when grasping the tool object. Therefore, $\boldsymbol{y}^{ref}$ is the initial value $\boldsymbol{y}^{init}$ when grasping the tool. For example, although the contact state and the grasp condition gradually change when grasping a hammer and hitting with it, we aim to inhibit the change. In this case, we design the loss function as shown below,

$$h_{loss}(\boldsymbol{y}^{est}_{seq}, \boldsymbol{y}^{ref}_{seq}) = h_{loss,1}(\boldsymbol{y}^{est}_{seq}, \boldsymbol{y}^{ref}_{seq}) + h_{loss,2}(\boldsymbol{y}^{est}_{seq}, \boldsymbol{y}^{ref}_{seq}) \tag{7}$$

$$h_{loss,1}(\boldsymbol{y}^{est}_{seq}, \boldsymbol{y}^{ref}_{seq}) = ||\boldsymbol{w}^{T}_{1}(\boldsymbol{F}^{est}_{seq} - \boldsymbol{F}^{ref}_{seq})||^{2}_{2} \tag{8}$$

$$\boldsymbol{w}_{1}[i] = \begin{cases} 1.0 & (\boldsymbol{F}^{est}_{seq}[i] \geq \boldsymbol{F}^{ref}_{seq}[i]) \\ \beta & (otherwise) \end{cases}$$

$$h_{loss,2}(\boldsymbol{y}^{est}_{seq}, \boldsymbol{y}^{ref}_{seq}) = w_{2}||\boldsymbol{\theta}^{est}_{seq} - \boldsymbol{\theta}^{ref}_{seq}||^{2}_{2} + w_{3}||\boldsymbol{l}^{est}_{seq} - \boldsymbol{l}^{ref}_{seq}||^{2}_{2} \tag{9}$$

where, $\boldsymbol{F}^{\{est,ref\}}_{seq}$ represents a vector vertically arranging $\boldsymbol{f}$ and $\boldsymbol{f}_{contact}$ extracted from $\boldsymbol{y}^{\{est,ref\}}_{seq}$, and $\{\boldsymbol{\theta},\boldsymbol{l}\}^{\{est,ref\}}_{seq}$ represents a vector vertically arranging $\{\boldsymbol{\theta},\boldsymbol{l}\}$ extracted from $\boldsymbol{y}$. Also, $w_{\{2,3\}}$ is a constant weight, $\boldsymbol{w}_{1}$ is a weight vector, $\boldsymbol{w}_{1}[i]$ is the i^{th} element of $\boldsymbol{w}_{1}$, and β is a constant weight. The design of w_{1} considers the characteristics of the contact sensors. Although the sensor values of the contact and muscle tension sensors change just until 0 in the minus direction, the values can largely change until the rated values in the plus direction. When setting $\beta = 1.0$, as the result of optimization, the contact state tends to change in the minus direction and the initial contact state cannot be kept. Therefore, by setting $\beta > 1.0$, we generate the control command to keep the initial contact state. At the same time, by limiting joint angles and muscle lengths by $h_{loss,2}$, we can obtain the control command which does not largely change the muscle length and joint angle while keeping the contact state.

In this study, we set $N^{control} = 8$, $N^{control}_{batch} = 4$, $N^{control}_{epoch} = 3$, $\beta = 3.0$, $w_{2} = 1.0$, and $w_{3} = 1.0$.

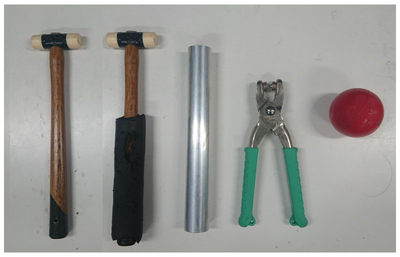
Fig. 5: Grasped objects in this study: Hammer, Hammer-S (hammer with soft cover), Cylinder, Gripper, and Ball.

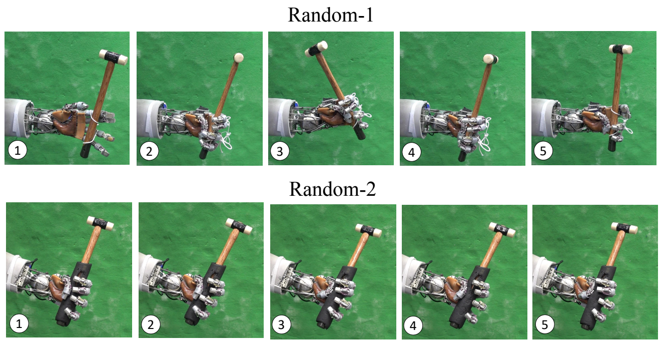

Fig. 6: Motion Sequences of the data collection phase of Random-1 and Random-2.

IV. Experiments

A. Training of HADYNET

The grasped objects used in this study are Hammer, Hammer-S (hammer with soft cover), Cylinder, Gripper, and Ball, as shown in Fig. 5. We handle the six objects, including None without grasping anything. In this study, we conduct two random motions of Random-1 and Random-2 to collect training data. Random-1 is a repeating motion that sends random target joint angles over random intervals by converting the target joint angle to the target muscle length using a geometric model as in Fig. 2. Random-2 is a motion randomly changing finger muscle lengths $\boldsymbol{l}_{finger}$ while grasping the object. Especially, Random-2 is important for the contact control explained in Section III-G, and consecutive contact changes by different grasps can be obtained.

Holding these six objects by the hand, Random-1 and Random-2 were each conducted 500 time steps (100 seconds). The data collection experiment is shown in Fig. 6. Although we must lightly tie up the object to the hand by wires or tapes so as not to drop it regarding Random-1, regarding Random-2, such a deal is unnecessary because the grasping shape of the hand does not largely change. Random-1 and Random-2 were each conducted three times regarding one object while changing the initialization, and 36 experiments were conducted with all the objects. The number of the collected data was about 18000, and we trained HADYNET using it. We compare the loss transition when fixing PB and training PB implicitly as different values (this study) in Fig. 7. The loss of the latter was smaller by about 20 %. However, because many data without grasping, which is the same data with None, was included regarding

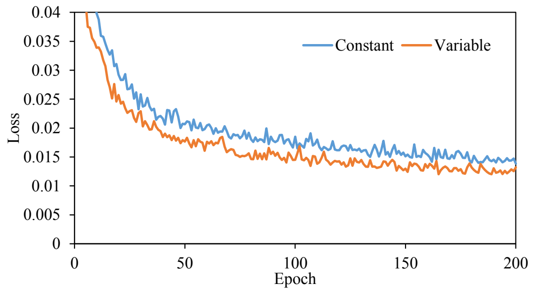

Fig. 7: Comparison of the training results of HADYNET between with fixed parametric bias (Constant) and with variable parametric bias (Variable, this study).

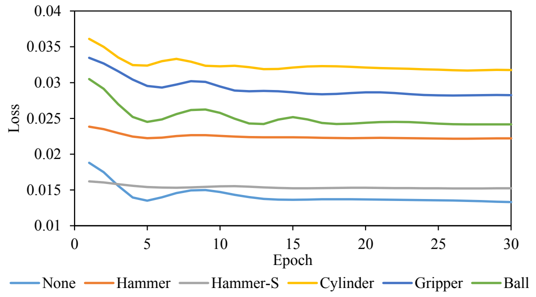

Fig. 8: Transition of loss when updating only parametric bias.

each grasped object, the actual difference is considered to be larger. Thus, by adding parametric bias, the network can implicitly train the difference of the grasped objects and initializations. In this study, the trained network can cope with the difference of the grasped objects and initializations, and by collecting data after a while (e.g. one month), the network becomes able to cope with deterioration over time. The deterioration over time can mainly affect the elongation of muscles and the change in the original length of the spring, and this is about the same degree of difference that can be seen due to the irreproducibility of initialization. Therefore, we consider that deterioration over time can be taken into account if irreproducibility of initialization can be taken into account. Also, it is difficult to determine whether the difference in PBs is due to irreproducibility of initialization or deterioration over time, and so we do not perform any direct experiments on deterioration over time.

B. Recognition of Grasped Objects

First, we conducted an experiment on the update of PB. We initialized the hand and collected new data regarding six objects. We show the loss transition when updating only PB in Fig. 8. Regarding all the grasped objects, the loss decreased by about 6–28% by updating only PB. Thus, by changing PB, the hand dynamics states with various grasped objects and initializations can be reproduced.

In Fig. 9, we plot PBs trained in Section IV-A and PBs updated using the newly collected data in the two-dimensional space as explained in Section III-D. Each PB was distinctly grouped according to the names of grasped objects. The data surrounded by a square is a new data, and regarding each new data, the correct name of the new grasped object can be recognized with the nearest neighbor method.

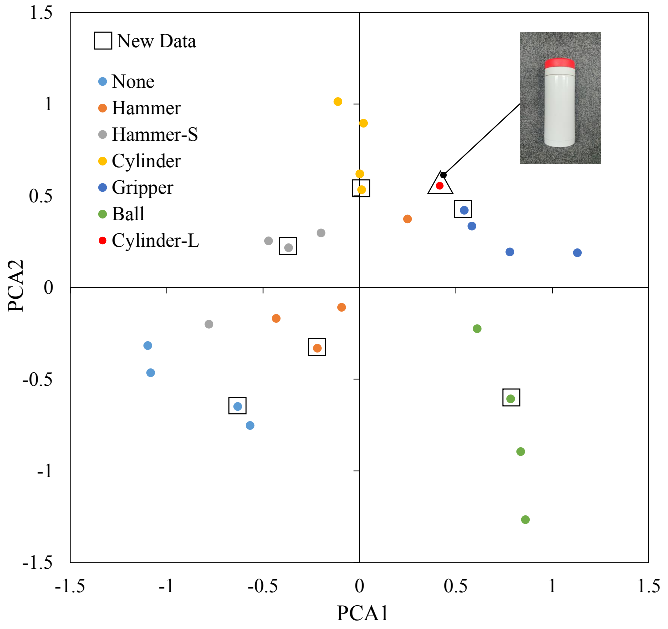

Fig. 9: Results of the object recognition. The marks surrounded by squares are new data to recognize what the hand is grasping. The mark surrounded by a triangle is a new object, the large cylinder (Cylinder-L).

Thus, by grasping the target object and moving the hand randomly, the robot can recognize the object without visual information.

We collected data regarding a new object, Cylinder-L, and updated PB. The diameter of Cylinder-L is 68 mm, while the diameter of Cylinder is 32 mm. Also, the width of Gripper, which is the largest among objects used for the training, is 60 mm. We show the updated PB of Cylinder-L surrounded by a triangle in Fig. 9. This PB is almost at the halfway point of PBs of the Cylinder and Gripper. This is considered to be valid because Cylinder-L has the shape characteristics of Cylinder and the size characteristics of Gripper. Thus, regarding such a new object, by referring to the trained parametric bias, the robot can recognize the characteristics of the grasped object.

Finally, we tried an online learning experiment of PB explained in the latter half of Section III-C. We executed only Random-2 in Section III-B over about one and a half minutes each with objects Hammer-S, Ball, Cylinder, None, Gripper, and Hammer-S, in order. We show the transition of PB plotted in the two-dimensional space while updating PB online, in Fig. 10. The PBs moved in the direction of the arrows, and the transitions of the updated PBs are shown in the same colors as the PBs trained in Section IV-A. From the figure, although the accuracy decreased compared with Fig. 9, we can see that the current PB moved in the direction of PBs of the current grasped object. While PB when grasping Ball moved accurately, regarding the other objects, although the current PBs moved near PBs of the grasped objects, they could not completely reach the correct values. This is considered to be because only the data up to N_{max}^{online} is used, the data used to update PB always changes, and the loss does

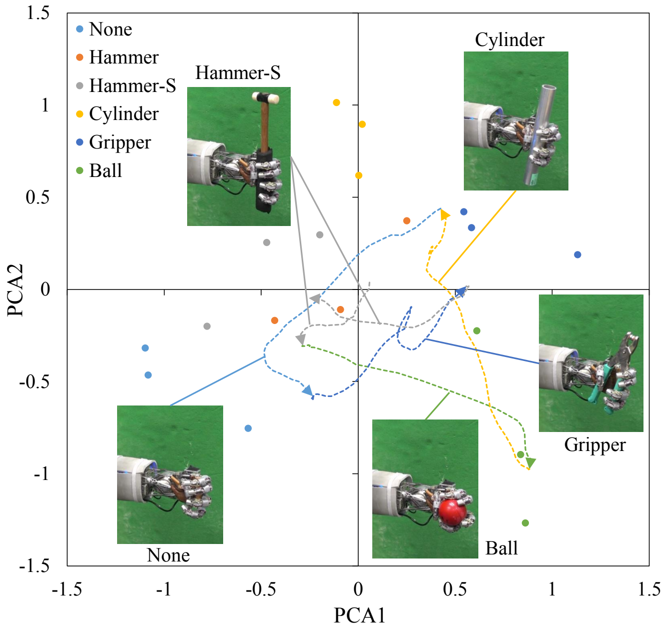

Fig. 10: Online update experiment of parametric bias. The graph shows the transition of parametric bias when updating it online.

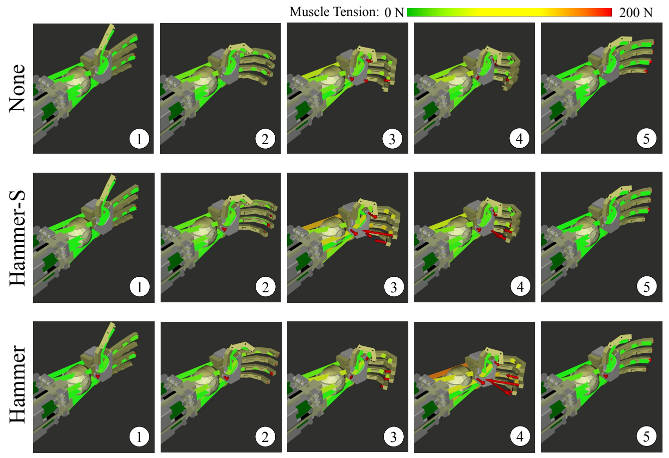

Fig. 11: Contact simulation using parametric biases of None Hammer-S, and Hammer.

not decrease completely. However, by updating parametric bias online, we can see that the robot can gradually obtain the characteristics of the grasped object.

C. Contact Simulation

We show the simulation results of Section III-E with None, Hammer-S, and Hammer in Fig. 11. Regarding each, PBs trained in Section IV-A were used. Here, the longer the arrow is, the larger the contact force is, and the redder the muscle color is, the higher the muscle tension is. The joint angles of fingers are calculated by using a geometric model and Kalman Filter. Regarding None, contact force and muscle tension did not largely change when grasping. When grasping Hammer-S, large contact forces were exerted at the loadcells of the fingertips and the palm. Also, the muscle tensions were high and some muscle colors became orange.

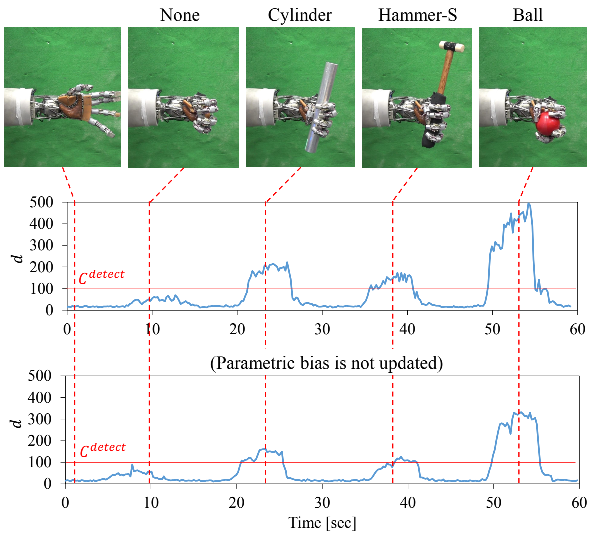

Fig. 12: Contact detection experiment when grasping various objects. The middle graph shows d when updating parametric bias and the lower shows d when using parametric bias trained previously.

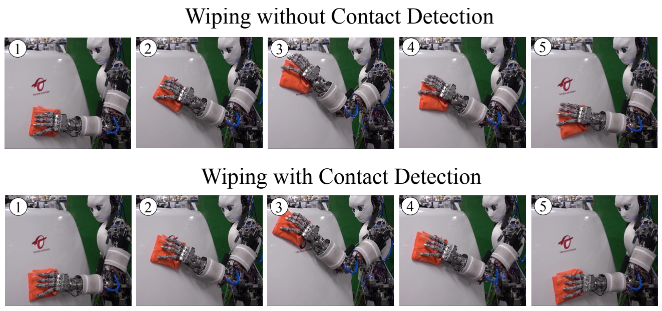
Wiping without Contact Detection
Wiping with Contact Detection

Fig. 13: Wiping experiments with and without contact detection.

Comparing Hammer-S and Hammer, the contact force and muscle tension of Hammer-S increased at an early stage of grasping because the diameter of Hammer-S is larger than that of Hammer due to the soft cover. Also, because of the lack of the soft cover, the impact of the grasping is not absorbed regarding Hammer. Therefore, in Hammer, it is observed that higher contact forces are suddenly exerted than in Hammer-S at the moment of contact. Thus, we can change the hand dynamics only by changing PB. This can be used for reinforcement learning of soft robotic hands which are difficult to modelize.

D. Contact Detection

First, we observed how the value d in Section III-F for contact detection changes when grasping various objects. We show the transition of d when updating PB at the current hand dynamics state of None (without grasping anything) and when grasping None, Cylinder, Hammer-S, and Ball, in order, in the middle graph of Fig. 12. In this study, because $C^{detect} = 100$, we can see correct transitions in that None was not detected and Cylinder, Hammer-S, and Ball were detected. When referring to Fig. 9, Ball and Cylinder, whose PBs are distant from None, showed larger d than Hammer-S, whose PB is near None. On the other hand, we show the result of contact detection when using PB of None trained in Section IV-A without updating PB at the current state of initialization, in the lower graph of Fig. 12. $d = 89$, which is near C^{detect}, was measured regarding None, and $d = 114$, which is near C^{detect}, was measured regarding Hammer-S at most. This causes the contact to be detected on and off due to minor errors. Thus, PB includes the hand dynamics information of initialization, and so we should update PB again when the hand is initialized, even if the grasped object is the same.

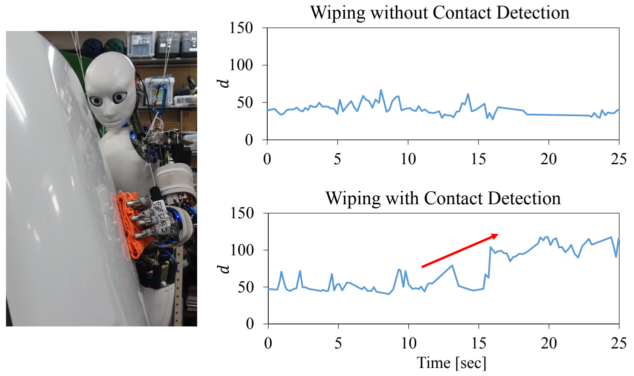

Fig. 14: The transitions of d when wiping with and without contact detection. The left figure shows the inclination of the target surface to be wiped.

Second, the contact detection is used for not only detecting but also keeping appropriate contact. We conducted an experiment of wiping a surface while keeping d at C^{detect}. In detail, the robot moves the hand away from the surface when contact is detected and moves toward the surface when contact is not detected. We show the cleaning experiments with and without contact detection in Fig. 13, and show the transitions of d in Fig. 14. In Fig. 13, while the cloth slipped from the hand without contact detection, the surface was traced well with contact detection. As shown in the left figure of Fig. 14, the surface to be cleaned has an inclined structure in which the hand leaves the surface as it goes up. Thus, the hand moved straight up without contact detection, the hand left the surface, and the cloth slipped from the hand. Without contact detection, d did not largely change. Compared with this case, when using contact detection, the hand not only moves straight up but also gets close to the surface, and the hand can move along the surface. When moving the hand up, because the inclination of the surface and the trajectory of the hand getting close to the surface were almost the same, we could not see a large change in d. However, when moving the hand down, as the hand and surface got closer, d became large, and d finally reached near C^{detect}. Thus, we can make use of the contact detection passively and actively.

E. Contact Control

We applied the strategy of grasping stabilizer explained in Section III-G to the hammer hitting operation, and evaluated the contact stability. First, we obtained D' regarding Hammer-S, and updated PB. Second, regarding the cases

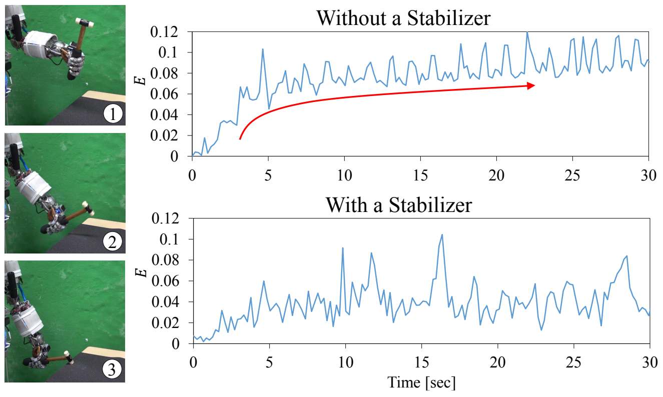

Fig. 15: Hammer hitting experiments with and without the grasping stabilizer. The graph shows the transition of the evaluation value E.

with and without the grasping stabilizer, we verified the transition of evaluation value E (explained subsequently), when hitting a table using Hammer-S over 30 seconds as shown in Fig. 15. $\boldsymbol{y}^{ref}$ is $\boldsymbol{y}^{init}$, which is the initial value of $\boldsymbol{y}$ when grasping Hammer-S. In this study, because the main purpose is keeping the initial contact state, we use $h_{loss,1}$ in Section III-G as E. The smaller the E, the more the grasping is stabilized. We show the experimental results in Fig. 15. In the case without grasping stabilizer, E gradually became large, and the initial contact state collapsed. Compared with this case, when using grasping stabilizer, E did not largely change from near 0.3, and even when E became large, the value reverted rapidly. Thus, by using HADYNET, the hand can realize the target contact state.

V. Conclusion

In this study, for the flexible musculoskeletal hand, we developed a method of the recognition of grasped objects, contact simulation, detection, and control, coping with its deterioration over time, irreproducibility of initialization, and the difference in grasped objects. By constructing the sensor state equation using a recurrent neural network with parametric bias, the different dynamics of the hand is implicitly learned. The hand can recognize the grasped object by the difference of parametric bias and its contact states can be simulated by the forwarding of the network. The hand can conduct the contact detection using the prediction error of the network, and can conduct the contact control by making the current sensor state close to the target value through backpropagation technique to the control input. We integrated these various components into one network, and verified the realization of various dynamics by changing only the parametric bias of the network.

In future works, we would like to focus on the task realization using these components.

References

[1] A. Kochan, "Shadow delivers first hand," *Industrial Robot*, vol. 32, no. 1, pp. 15–16, 2005.

[2] Y. Kim, Y. Lee, J. Kim, J. Lee, K. Park, K. Roh, and J. Choi, "RoboRay hand: A highly backdrivable robotic hand with sensorless contact force measurements," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*, 2014, pp. 6712–6718.

[3] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *The International Journal of Robotics Research*, vol. 35, no. 1–3, pp. 161–185, 2016.

[4] Z. Xu and E. Todorov, "Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation*, 2016, pp. 3485–3492.

[5] S. Makino, K. Kawaharazuka, M. Kawamura, A. Fujii, T. Makabe, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Five-Fingered Hand with Wide Range of Thumb Using Combination of Machined Springs and Variable Stiffness Joints," in *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4562–4567.

[6] Y. Chebotar, K. Hausman, Z. Su, G. S. Sukhatme, and S. Schaal, "Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning," in *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 1960–1966.

[7] D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar, and E. Todorov, "Learning Deep Visuomotor Policies for Dexterous Hand Manipulation," in *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, 2019, pp. 3636–3643.

[8] H. V. Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, 2015, pp. 121–127.

[9] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus, "Robust proprioceptive grasping with a soft robot hand," *Autonomous Robots*, vol. 43, no. 3, pp. 681–696, 2019.

[10] K. Kawaharazuka, S. Makino, M. Kawamura, Y. Asano, K. Okada, and M. Inaba, "Online Learning of Joint-Muscle Mapping using Vision in Tendon-driven Musculoskeletal Humanoids," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 772–779, 2018.

[11] K. Kawaharazuka, K. Tsuzuki, S. Makino, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Long-time Self-body Image Acquisition and its Application to the Control of Musculoskeletal Structures," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2965–2972, 2019.

[12] 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.

[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] K. Kawaharazuka, S. Makino, M. Kawamura, Y. Asano, Y. Kakiuchi, K. Okada, and M. Inaba, "Human Mimetic Forearm Design with Radioulnar Joint using Miniature Bone-muscle Modules and its Applications," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 4956–4962.

[15] J. Tani, M. Ito, and Y. Sugita, "Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using RNNPB," *Neural Networks*, vol. 17, no. 8, pp. 1273–1289, 2004.

[16] 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.

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

[18] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, pp. 448–456.

[19] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[20] K. Kawaharazuka, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Stable Tool-Use with Flexible Musculoskeletal Hands by Learning the Predictive Model of Sensor State Transition," in *Proceedings of the 2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 4572–4578.