IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 17, 2020, accepted April 4, 2020, date of publication April 13, 2020, date of current version April 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2987604

Motion Planning With Success Judgement Model Based on Learning From Demonstration

DAICHI FURUTA[1], KYO KUTSUZAWA[1], (Graduate Student Member, IEEE), SHO SAKAINO[2], (Member, IEEE), AND TOSHIAKI TSUJI[1], (Senior Member, IEEE)
[1]Graduate School of Science and Engineering, Saitama University, Saitama 338-8570, Japan
[2]Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba 305-8573, Japan

Corresponding author: Kyo Kutsuzawa (k.kutsuzawa.430@ms.saitama-u.ac.jp)

This work was supported in part by the New Energy and Industrial Technology Development Organization (NEDO), Japan, under Grant P15009.

ABSTRACT A technique named Learning from Demonstration allows robots to learn actions in a human living environment from the demonstrations directly. In a learning method from demonstrations directly, however, teaching actions cannot be reused between situations with different restrictions. In this study, we propose a method for training a success judgment model based on Learning from Demonstration and use this as a differentiable loss function of tasks. By formulating the constraints of the action in a manner in mathematical optimization and combining these constraints with the learned success judgment model into a loss function, an action generation model can be trained by the gradient method. This system was verified with the action of scooping up a pancake.

INDEX TERMS Force control, learning from demonstration, motion planning, robot learning.

I. INTRODUCTION

Advances in robotics are expanding the range of robot applications, from factories to human living spaces. In human living spaces, the conditions of the environment often vary. Therefore, it is important for robots to autonomously generate actions in response to these variations.

It is difficult to program such autonomous action generation; thus, Learning from Demonstration (LfD) has been introduced as an effective technique to learn from human actions [1], [2]. Recently, neural networks (NNs) are often used for learning due to their high representation ability. In many techniques using LfD, teaching data are collected by humans operating robots indirectly [3]–[5]. Yang *et al.* have demonstrated the use of LfD to generate actions to perform a folding task [3]. In that study, a human operates the robot using a head mont display and mouse, and the system collects human movement data from the action of folding a piece of cloth. Rahmatizadeh *et al.* have achieved generation of motion for multiple object manipulation tasks with a single NN using LfD [4]. These methods use position information as demonstration data.

The associate editor coordinating the review of this manuscript and approving it for publication was Xi Peng.

LfD has also been applied to operations using force information [6]–[9]. By performing tasks using force information adeptly as a human would, robots can flexibly respond to changes in the environment. For example, previous studies have involved the achievement of tasks such as erasing words on a whiteboard [6], scooping barley from a bowl [7], and drawing lines and arcs with a ruler and a protractor [8].

By teaching actions from humans through operating robots, such actions implicitly satisfy the constraints of the robot (*e.g.* velocity limitation and range of motion). In other words, the teaching action is tailored to the robot used for learning. Furthermore, by using remote control technology such as bilateral control, the characteristics of the control systems and actuators can also be considered [8]. These methods enable tasks in which constraints and success conditions are difficult to define; however, it is difficult to apply them directly to a different situation that will have different restrictions from those when data collection. For example, when robots are moving near humans, the robots should take a low velocity limit for safety; while, when robots should finish tasks rapidly, the robots are desired to act as high velocity as possible within a velocity limit.

It is relatively easy to formulate constraints such as velocity limitations and range of motion for a robot through mathematical optimization. Action generation methods based on mathematical optimization include model predictive control (MPC) [10]–[12] and rapidly-exploring random tree [13], [14]. It is also possible to add these restrictions in machine learning methods, such as NNs [15]. In the methods mentioned above, the robot can autonomously generate actions suited for the environment, considering constraints by treating them mathematically expressions. The conditions for success in a task, such as the target end position, can also be treated as mathematical formulae in the same way. Hirose *et al.* used MPC with constraints to stabilize the transport of luggage by a robot and achieved a luggage transportation task [11]. Kutsuzawa *et al.* achieved the task of flipping over a pancake by getting a robot to learn the conditions to prevent slipping as a loss function in NNs, based on a friction cone [15].

The above methods enable adaptation to changes in the robot or environmental conditions easily by giving constraints explicitly. However, for actions involving contact with the environment, it is difficult to model the environment and formulate success conditions. Thus, treating such tasks mathematically is challenging, making it difficult to formulate success conditions for many tasks performed in human living spaces.

One candidate approach is to combine machine learning and the use of mathematical formulae. One common way is to imitate the environment with machine learning and use the trained models for optimization methods [16]–[18]. Imitating the environment, however, is hard; the models that learned time development of the environment often behave incorrectly due to prediction errors accumulated by auto-regression [19]–[21]. There also exist ways that apply backpropagation to trained NNs for optimization [22]–[25]. In these methods, NNs are regarded as differentiable functions used for optimizing their input values. This approach seems to be effective to combine machine learning for complicated tasks and explicit expressions of restrictions into a loss function.

In this study, we propose a method that uses success/failure judgement for training action generation. This method uses two models—a *success judgment model* that predicts the success rate of action by learning from demonstration data, and an *action generation model*. The trained success judgment model is used as a loss function of the *action generation model* along with the constraints of the robot, such as velocity limitations and range of motion. Then, the action is learned, such that the success rate predicted by the success judgment model becomes 100%. Here, both the success judgment model and the action generation model consist of NNs; this allows learning by backpropagation and the gradient-based optimization to succeed in the task. Figure 1 shows a conceptual diagram of the proposed method. In this method, rather than learning the human demonstrations directly, they are used indirectly in the training of the success judgment model. It allows the action generation model to handle changes in the constraints without having to re-collect demonstration data.

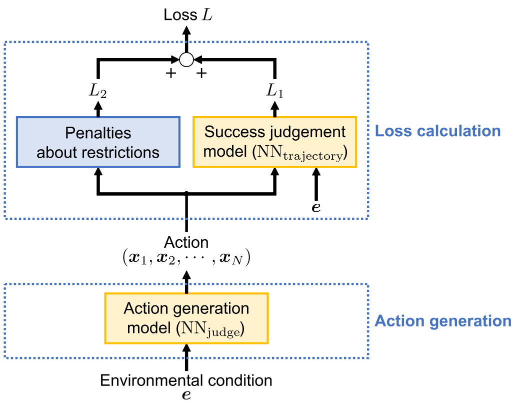
Loss L
$+$
$+$
L_2
L_1
Penalties about restrictions
Success judgement model ($\mathrm{NN_{trajectory}}$)
e
Loss calculation
Action
$(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$
Action generation model ($\mathrm{NN_{judge}}$)
Action generation
Environmental condition
e

FIGURE 1. Conceptual diagram of the proposed method. Details of each component and symbol are explained later.

The approach of learning human demonstrations indirectly has also been taken in inverse reinforcement learning (IRL) [26], [27]. The difference from it is that IRL uses only successful motions to imitate the expert's policy, while our method uses both success and failure motions to discriminate whether planned motions succeed in the task or not. Due to not imitating the policy, our method can be used even when constraints are different from that in demonstrations; note that different behaviors are required when constraints are different.

To verify the effectiveness of our proposed method, we used a task involving a scooping action. Scooping is a task that requires quick and accurate motion, which cannot be performed without considering the constraints of the robot. It is also a task requiring contact with the environment, which complicates the formulation of the success conditions of the task. These aspects make it a suitable task for verifying the effectiveness of our proposed method. This effectiveness is demonstrated through physical simulation and experimental results.

The structure of this paper is as follows. Section II explains the success judgment model, and Section III explains the action generation learning that constitutes our proposed method. Here, the model structures are to be designed as generic ones because the main proposal in this study is the training architecture. To verify the effectiveness of the proposed method, in Section IV, we describe verification by physical simulation, and in Section V, we describe verification by experiments with an actual manipulator. Finally, in Section VI, we summarize the study and discuss future prospects.

II. NN TO JUDGE THE SUCCESS OF THE TASK

In this section, we describe the collection of human action data and the structure of the success judgment model in order to create an NN model to judge the success of the task.

A. COLLECTION OF TEACHING DATA

In this study, a tool equipped with motion-capture markers and force sensors [28] is used to obtain data on the

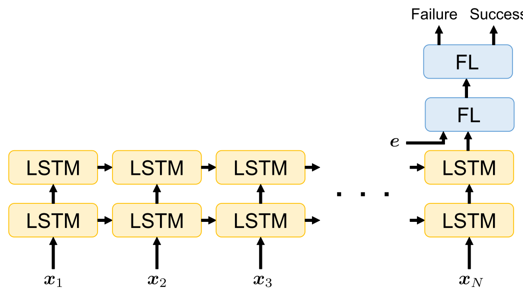

FIGURE 2. Structure of $\mathrm{NN_{judge}}$, which judges the success or failure of the task.

relationship between human action and the success or failure of the task. By teaching actions that involve using tools in a manner similar to that actually used by humans, pure human movements can be recorded without including robot restrictions. In addition, as the tool is easy to handle, the large amount of teaching data required for the success judgment model can be easily collected.

During data collection, trajectories of the tool position $\boldsymbol{q}_k$, velocity $\dot{\boldsymbol{q}}_k$, and contact force $\boldsymbol{f}_k$ are recorded; here, k indicates the time. We define $\boldsymbol{x}_k$ as a combination of these variables as follows:

$$\boldsymbol{x}_k = [\boldsymbol{q}_k^\top, \dot{\boldsymbol{q}}_k^\top, \boldsymbol{f}_k^\top]^\top. \tag{1}$$

Also, a label of success/failure is recorded for each trajectory. When the environment was changed during data collection, the environmental condition should also be recorded as a variable $\boldsymbol{e}$.

B. STRUCTURE OF THE NN

For success/failure judgement of human actions, we used long short-term memories (LSTMs) [29], [30]. Previous studies have already demonstrated LSTMs are effective for human-action processing [31], [32]. In contrast to other methods such as sliding window based methods [33], hidden Markov models [34], and Gaussian mixture models [35], LSTMs can process long-term sequences without designing sliding windows and pre-processing.

Figure 2 shows the structure of the NN used to judge the success or failure of a task from an action. $\mathrm{NN_{judge}}$ contains a total of four layers: two layers of LSTMs and two feed-forward layers (FLs) with ReLU activation. This model receives input regarding environmental conditions, as well as position, velocity, and force, over time and outputs the success or failure of the task as binary classification. A cross-entropy function was used as the loss function during training. Besides, dropout with the selection probability 0.3 was applied during training.

III. NN TO GENERATE ACTIONS

A. STRUCTURE OF THE NN

Figure 3 shows an overview of the action generation with $\mathrm{NN_{trajectory}}$, two-layers of FLs that is used to generate actions. This NN takes current position $\boldsymbol{q}_k$ and environmental condition $\boldsymbol{e}$ and generates the next samples of force command

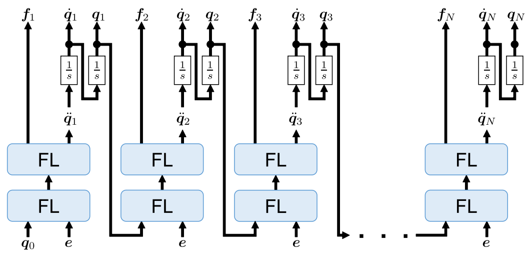

FIGURE 3. Action generation method of $\mathrm{NN_{trajectory}}$.

value $\boldsymbol{f}_{k+1}$ and acceleration command value $\ddot{\boldsymbol{q}}_{k+1}$. By integrating the generated acceleration command value, the velocity $\dot{\boldsymbol{q}}_{k+1}$ and position $\boldsymbol{q}_{k+1}$ of the next step can be determined. By feeding this position $\boldsymbol{q}_{k+1}$ back into the model at the next time, an action can be generated autoregressively.

B. TRAINING NN FOR ACTION GENERATION

The goal of the action generation model $\mathrm{NN_{trajectory}}$ is to generate an action that 1) satisfies the constraints in the situation and 2) successfully performs the task. The former can be learned by defining a differentiable penalty function for mathematical optimization. The latter can be achieved by using the success judgment model, $\mathrm{NN_{judge}}$, explained above; thanks to $\mathrm{NN_{judge}}$ differentiable, it is possible to find the gradient direction such that the action is classified as successful.

Therefore, the action generation model should be trained using the following loss functions:

$$L = L_1 + L_2, \tag{2}$$

where

$$L_1 = -P \log y_{\mathrm{judge}} \tag{3}$$

and

$$\begin{aligned} L_2 = Q \sum_{k=1}^{N} \Big[& \| \max(\dot{\boldsymbol{q}}_k - \dot{\boldsymbol{q}}^{\max}, \boldsymbol{0}) \|^2 \\ & + \| \max(-(\dot{\boldsymbol{q}}_k - \dot{\boldsymbol{q}}^{\min}), \boldsymbol{0}) \|^2 \\ & + \| \max(\boldsymbol{q}_k - \boldsymbol{q}^{\max}, \boldsymbol{0}) \|^2 \\ & + \| \max(-(\boldsymbol{q}_k, -\boldsymbol{q}^{\min}), \boldsymbol{0}) \|^2 \\ & + \| \max(\boldsymbol{f}_k - \boldsymbol{f}^{\max}, \boldsymbol{0}) \|^2 \\ & + \| \max(-(\boldsymbol{f}_k - \boldsymbol{f}^{\min}), \boldsymbol{0}) \|^2 \Big] \\ & + R(\|\boldsymbol{q}_N - \boldsymbol{q}^{\mathrm{end}}\|^2 + \|\dot{\boldsymbol{q}}_N - \dot{\boldsymbol{q}}^{\mathrm{end}}\|^2), \end{aligned} \tag{4}$$

where max takes maximum values for each element of given vectors. Here, (3) represents the loss term that discriminates whether the trajectories succeeded or failed the task by using $\mathrm{NN_{judge}}$. L_1 guides learning so that the generated action achieves the task successfully. Here, P is a weighting factor, and y_{judge} is the success rate of the generated action predicted by $\mathrm{NN_{judge}}$. (4) is the loss term that gives mathematical penalties when the generated action did not satisfy the restrictions and constraints given by the user. By this term, the model learns in order that the generated actions satisfy the constraints of the robot. Here, Q is a weighting factor, and $\dot{\boldsymbol{q}}^{\min}$,

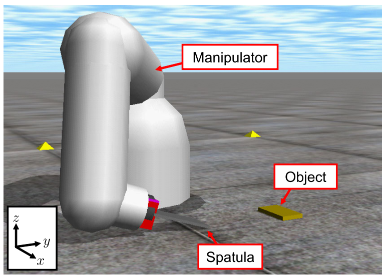

FIGURE 4. Manipulator in simulation.

$\dot{\boldsymbol{q}}^{\max}$, $\boldsymbol{q}^{\min}$, $\boldsymbol{q}^{\max}$, $\boldsymbol{f}^{\min}$, and $\boldsymbol{f}^{\max}$ indicate the minimum and maximum values of velocity, position, and force, respectively. The final term in L_2 evaluates whether the generated action finishes at the desired terminal position $\boldsymbol{q}^{\mathrm{end}}$ and velocity $\dot{\boldsymbol{q}}^{\mathrm{end}}$. Here, R is a weighting factor. Backpropagation is performed based on these loss functions, and finally, the desired action generation model is obtained by updating the parameters of $\mathrm{NN}_{\mathrm{trajectory}}$. Note that during the training of $\mathrm{NN}_{\mathrm{trajectory}}$, the parameters of $\mathrm{NN}_{\mathrm{judge}}$ are no longer updated.

In this way, our proposed method allows learning with loss functions to succeed in the task such that the loss functions simultaneously satisfy the constraints in the situation. Therefore, for a task in which the success condition is difficult to formulate, our proposed method enables the generation of actions that can achieve the task, taking into consideration the constraints of different situations.

IV. SIMULATION

We verified the generation of a scooping motion in a simulation environment considering the initial position of the object and the change in the coefficient of friction. In this simulation, we aim to verify the effectiveness of the proposed training method that uses the success/failure judgement of motions and environmental conditions.

A. SET-UP OF MANIPULATOR

For this verification, we used the physical simulator Open Dynamics Engine [36] to reproduce the six-axis robot manipulator shown in Fig. 4.

The block diagram in Fig. 5 shows the hybrid position/force control system [37], [38] used. In the control system, $\boldsymbol{K}_{\mathrm{p}}$, $\boldsymbol{K}_{\mathrm{v}}$, and $\boldsymbol{K}_{\mathrm{f}}$ indicate the position, velocity, and force control gains, respectively. Also, $\boldsymbol{\Lambda}$ and $\boldsymbol{M}$ indicate the mass matrices in the Cartesian coordinate system and joint coordinate system, respectively. $\boldsymbol{S}$ indicates the selection matrix between position and force control, and g indicates the cutoff frequency of pseudo-differentiators. Table 1 shows the parameters of the control system. This control system applies force control in the z-axis direction to press the spatula against the base, while performing position control in the y-axis direction to scoop up the object. Position control was applied in the other axial directions to maintain the position and orientation constant. The starting point of the motion was set as $(x, y, z) = (0.3\ \mathrm{m}, 0\ \mathrm{m}, 0\ \mathrm{m})$. The control interval was 1 ms.

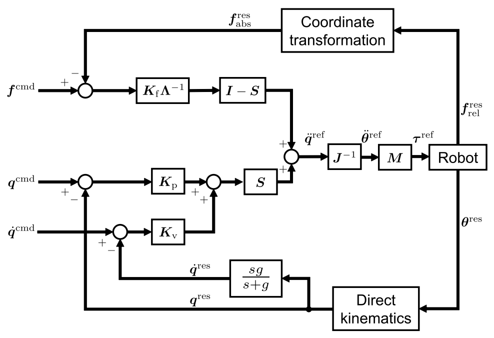
Coordinate transformation
$\boldsymbol{f}^{\mathrm{res}}_{\mathrm{abs}}$
$\boldsymbol{f}^{\mathrm{cmd}}$
$\boldsymbol{K}_{\mathrm{f}}\boldsymbol{\Lambda}^{-1}$
$\boldsymbol{I}-\boldsymbol{S}$
$\boldsymbol{f}^{\mathrm{res}}_{\mathrm{rel}}$
$\ddot{\boldsymbol{q}}^{\mathrm{ref}}$
$\ddot{\boldsymbol{\theta}}^{\mathrm{ref}}$
$\boldsymbol{\tau}^{\mathrm{ref}}$
$\boldsymbol{J}^{-1}$
$\boldsymbol{M}$
Robot
$\boldsymbol{q}^{\mathrm{cmd}}$
$\boldsymbol{K}_{\mathrm{p}}$
$\boldsymbol{S}$
$\dot{\boldsymbol{q}}^{\mathrm{cmd}}$
$\boldsymbol{K}_{\mathrm{v}}$
$\boldsymbol{\theta}^{\mathrm{res}}$
$\dot{\boldsymbol{q}}^{\mathrm{res}}$
$\frac{sg}{s+g}$
$\boldsymbol{q}^{\mathrm{res}}$
Direct kinematics

FIGURE 5. Block diagram of control system.

Symbol	Value
$\boldsymbol{K}_{\mathrm{p}}$	diag[1000, 1000, 1000, 500, 500, 500]
$\boldsymbol{K}_{\mathrm{v}}$	diag[300, 300, 300, 300, 300, 300]
$\boldsymbol{K}_{\mathrm{f}}\boldsymbol{\Lambda}^{-1}$	diag[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
$\boldsymbol{S}$	diag[1.0, 1.0, 0.1, 1.0, 1.0, 1.0]
g	10 Hz

TABLE 1. Control system parameters used in physical simulation.

For safety reasons, the experiment was performed with a slight position control added in the force control direction. However, as long as the contact is maintained properly, the deviation of the position control would be sufficiently small that the effect of adding this position control would be considered to be negligible.

B. COLLECTION OF TEACHING DATA

In this simulation, instead of actions performed by humans, actions performed randomly on the simulator were used as teaching actions. Random points were plotted in the space spanned by the position in the y-axis direction and the force in the z-axis direction, and a random scooping trajectory was generated by spline curves generated from these points. The success or failure of the scooping action was labeled based on the final position of the object when this trajectory was reproduced.

During the collection of teaching data, the position of the object on the y-axis, y^{obj}, and the friction coefficient between the spatula and the object, μ, were varied. As training data, three points of y^{obj} (0.1 m, 0.19 m, and 0.28 m) and three points of μ (1.0, 2.0, and 3.0) were combined, with 1000 actions obtained for each combination. As test data, 1000 actions were obtained at random positions and friction coefficients within the range $0.1\mathrm{m} \leq y^{\mathrm{obj}} \leq 0.28\mathrm{m}$ and $1.0 \leq \mu \leq 3.0$. The success/failure result of the final collected data was 25.6% success, 74.4% failure.

1) TRAINING OF NNS

The collected data (trajectories of position and force, environmental conditions, and success/failure label) were used for learning in the success judgment NN $\mathrm{NN}_{\mathrm{judge}}$, as shown in Fig. 2. Table 2 shows the parameters.

The NN for action generation, $\mathrm{NN}_{\mathrm{trajectory}}$, used the structure shown in Fig. 3; the parameters are shown in Table 3.

Item	Detail
Input to $\mathrm{NN}_{\mathrm{judge}}$, $\boldsymbol{x}_k$	$[y_k^{\mathrm{res}}, \dot{y}_k^{\mathrm{res}}, f_k^{\mathrm{res}}]$
Environmental variable $\boldsymbol{e}$	$[y^{\mathrm{obj}}, \mu]$
Number of neurons in each LSTM layer	100
Number of neurons in each FL layer	102
Training data	9000 actions
Test data	1000 actions
Learning epochs	500 epochs
Actions learned per epoch	9000 actions
Mini-batch size	10 actions

TABLE 2. Parameters of $\mathrm{NN}_{\mathrm{judge}}$ used for simulation.

Item	Detail
y^{obj}, μ	random, random
q_0	-0.03 m
$\dot{q}_0$	0.0 m/s
$q^{\min}$, $q^{\max}$	-0.2 m, 0.4 m
$\dot{q}^{\max}$, $\dot{q}^{\min}$	1.0 m/s, -0.05 m/s
$f^{\min}$, $f^{\max}$	7.5 N, 22.5 N
q^{end}	y^{obj}
$\dot{q}^{\mathrm{end}}$	0.0 m/s
Environmental variable $\boldsymbol{e}$	$[y^{\mathrm{obj}}, \mu]$
Number of neurons in each FL	150
Learning count (epoch)	5000 times
Actions learned per epoch	100 actions

TABLE 3. Variables and structural parameters of $\mathrm{NN}_{\mathrm{trajectory}}$ used for simulation.

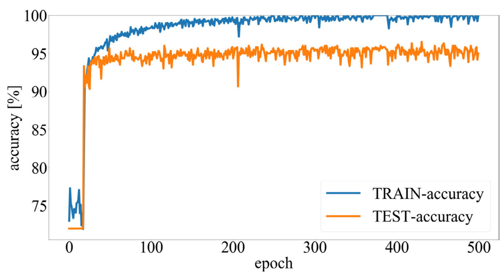

FIGURE 6. Accuracy rate per epoch of $\mathrm{NN}_{\mathrm{judge}}$ training data and test data used in simulation.

Throughout this paper, Adam [39] was used for parameter updates.

2) VERIFICATION RESULTS

The success judgment NN, $\mathrm{NN}_{\mathrm{judge}}$, was trained to determine the success or failure of the teaching data. Consequently, we obtained a model with a maximum accuracy rate of 96.5%. The progress of accuracy rate is shown in Fig. 6. Note that the model did not over-fit to the training dataset because the accuracy with the test dataset did not decrease. We confirmed that the model could determine success or failure with a high accuracy rate, even for values of y^{obj} and μ not used for training. In other words, the model demonstrated that could maintain high generalization and discrimination performance could be achieved, even with changes in the environmental conditions.

Next, we trained the action generation NN, $\mathrm{NN}_{\mathrm{trajectory}}$, using the resulting $\mathrm{NN}_{\mathrm{judge}}$ and then verified whether the corresponding action could be generated, even if the environmental conditions (y^{obj} and μ) changed. First, we compared

Method	Success rate
Using success/failure judgement and all environmental conditions	100%
Using success/failure judgement and only changes in y^{obj}	90%
Using opposite success/failure labels with all environmental conditions	0%
Randomly-generated actions	28%

TABLE 4. Results of task success rate between different methods in simulation.

Method	Range of μ	Success rate
Interpolation within trained μ	$1.0 \leq \mu \leq 3.0$	100%
Extrapolation with larger μ	$3.0 \leq \mu \leq 3.5$	100%
Extrapolation with smaller μ	$0.5 \leq \mu \leq 1.0$	100%
Extrapolation with considerably larger μ	$3.5 \leq \mu \leq 4.0$	81%
Extrapolation with considerably smaller μ	$0.05 \leq \mu \leq 0.5$	45%

TABLE 5. Results of task success rate between different ranges of environmental conditions in simulation.

the task success rates between the following four methods: 1) the proposed method that uses success/failure judgement and all environmental conditions, 2) training only with changes in y^{obj}, 3) training where success/failure labels were inverted (*i.e.*, training $\mathrm{NN}_{\mathrm{trajectory}}$ toward failure), and 4) randomly-generated actions. Table 4 shows the results. The success rates were low for all cases except for the proposed method. This shows that the use of success/failure judgement and environmental conditions is essential to ensuring success in the task. Second, we verified the proposed method with different ranges of friction parameters. Table 5 shows the results. The model resulted in high success rates even though the environmental conditions were outside the trained range. However, the success rate decreased when the environmental conditions were far away from the training.

V. EXPERIMENTS

A. SET-UP OF MANIPULATOR

For this experiment, we used a six-axis manipulator MOTOMAN-MH3F supplied by Yaskawa Electric. Figure 7 shows the appearance of this manipulator. A force sensor for measuring the contact force is attached to the tip of the manipulator, and a metal spatula is attached to the force sensor. The force sensor was a six-axis force sensor WDF-6M200-3 supplied by WACOH-TECH Inc. A thin rubber block with 1.3 cm height, 6.2 cm length, and 7.0 cm width was used as the object to be scooped up; and a carpet on a mat was used for the floor.

We used the same control system described in Section IV-A to control the robot, but the control parameters were changed,

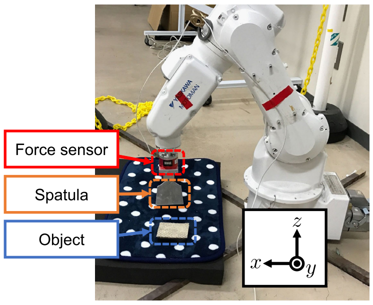

FIGURE 7. Appearance of six-axis manipulator.

Symbol	Value
$\boldsymbol{K}_\mathrm{p}$	diag[1000, 1000, 1000, 2000, 2000, 2000]
$\boldsymbol{K}_\mathrm{v}$	diag[40, 40, 40, 50, 50, 50]
$\boldsymbol{K}_f\boldsymbol{\Lambda}^{-1}$	diag[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
$\boldsymbol{S}$	diag[1.0, 1.0, 0.1, 1.0, 1.0, 1.0]
g	10 Hz

TABLE 6. Control system parameters used in experiments.

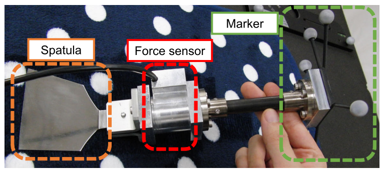

FIGURE 8. Device with force sensor on spatula.

as shown in Table 6. The starting point of motion was set as $(x, y, z) = (0.35\text{ m}, 0\text{ m}, 0\text{ m})$.

B. COLLECTION OF TEACHING DATA

To record demonstration data from a human performing a scooping action, we used a spatula equipped with a force sensor and a motion-capture marker. Figure 8 shows the appearance of this device. The human instructor performs the action holding this spatula, and the position and force over time is recorded during the action. A label of success or failure is added to each action. We used a six-axis force sensor FFS055YA101U6S manufactured by Leptrino Inc. and four Prime41 motion-capture systems manufactured by NaturalPoint, Inc. Figure 10 shows snapshots of an succeeded action during data collection. The success or failure of the action was determined visually. As a guideline, it was judged as success if 90% or more of the object was scooped onto the spatula.

This device enables an average of 700 scooping movements per hour to be obtained. The device is easy to handle, allowing a lot of data to be recorded in a short period of time.

C. TRAINING OF SUCCESS JUDGMENT NN

Success and failure data for the scooping task was collected as described in SectionV-B. The total data collected consisted of 1115 actions, which required approximately 1 hour and 15 minutes to collect. Of these actions, 1000 were used as training data and 115 as test data. The success/failure result of the collected data was 65.4% success, 34.6% failure. Figure 11 and 12 show examples of the trajectory of each motion. Figure 13 shows the distribution of the maximum velocity and maximum force in each trajectory of the 1115 actions obtained. As seen from the figure, success or failure cannot be determined from simple indicators such as maximum velocity and maximum force.

The collected data were used to train the NN $\mathrm{NN}_\mathrm{judge}$, whose structure is shown in Fig. 2. The parameters of the model are shown in Table 7. The initial position of the object on the y-axis, y^obj, was used as the environmental condition e; in the experiments, however, the initial position of the object was set to a fixed value: $y^\mathrm{obj} = 0.246\text{m}$.

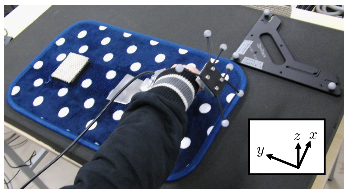

FIGURE 9. Collection of human action data.

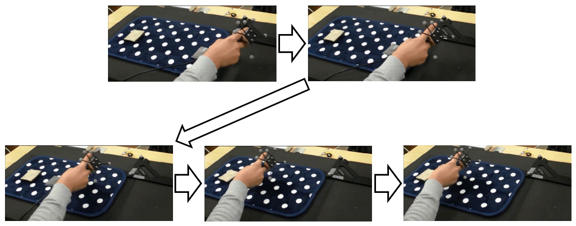

FIGURE 10. Snapshots during collection of scooping action data (success).

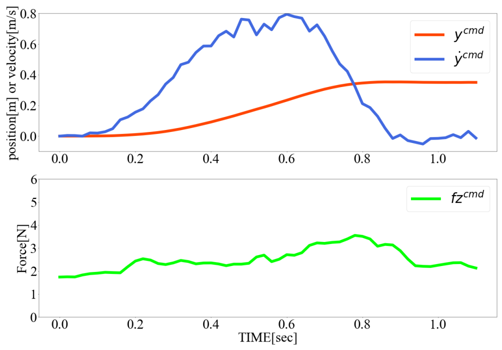

FIGURE 11. Example of motion (success) when performing scooping task.

D. TRAINING OF ACTION GENERATION NN

The action generation NN, $\mathrm{NN}_\mathrm{trajectory}$, was trained with the structure shown in Fig. 1. In this experiment, two $\mathrm{NN}_\mathrm{trajectory}$ models with different velocity restrictions were trained: $\dot{q}^\mathrm{max} = 1.0\text{ m/s}$ and $\dot{q}^\mathrm{max} = 0.4\text{ m/s}$. The other parameters are shown in Table 8. This evaluation with the two restrictions confirms that the proposed method allows models to learn actions satisfying different restrictions by using the same demonstration data.

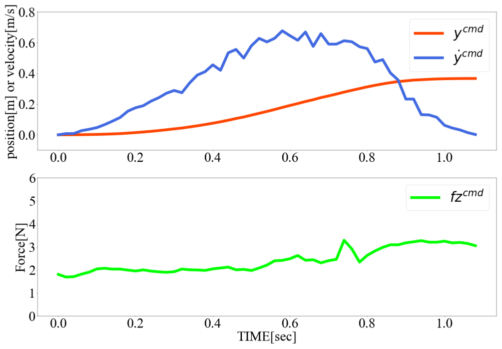

FIGURE 12. Example of motion (failure) when performing scooping task.

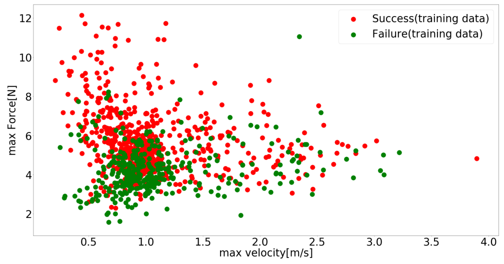

FIGURE 13. Distribution of maximum velocity and maximum force in human action data.

Item	Details
$\mathrm{NN_{judge}}$ input $\boldsymbol{x}_k$	$[y_k^{\mathrm{res}}, \dot{y}_k^{\mathrm{res}}, fz_k^{\mathrm{res}}]$
Environmental variable e	y^{obj}
LSTM number of neurons	100 each
FL number of neurons	101 each
Training data	1000 actions
Test data	115 actions
Learning count (epoch)	300 times
Actions learned per epoch	1000 actions
Mini batch	10 actions

TABLE 7. Parameters of $\mathrm{NN_{judge}}$.

Item	Details
y^{obj}	0.246 m
$q_0, \dot{q}_0$	-0.1 m, 0.0 m/s
$q^{\min}, q^{\max}$	-0.2 m, 0.4 m
$\dot{q}^{\min}$	-0.05 m/s
$f^{\min}, f^{\max}$	1.0 N, 10.0 N
$q^{\mathrm{end}}, \dot{q}^{\mathrm{end}}$	y^{obj}, 0.0 m/s
Environmental variable e	y^{obj}
Number of neurons in each FL	150
Learning count (epoch)	3000 times
Actions learned per epoch	100 actions

TABLE 8. Parameters of $\mathrm{NN_{trajectory}}$ in experiments.

E. EXPERIMENTAL RESULTS

First, we evaluated the performance of the success judgment NN, $\mathrm{NN_{judge}}$. Figure 14 shows the progress in the accuracy rate during training. Although a slight decrease was observed in the test-dataset accuracy, it is negligible for the performance. We eventually obtained a model with a maximum accuracy rate of 96.52% with the test data, indicating that $\mathrm{NN_{judge}}$ maintains a high discrimination performance.

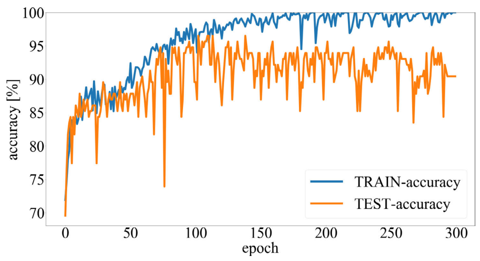

FIGURE 14. Accuracy rate for each epoch of training data and test data of $\mathrm{NN_{judge}}$ used in the experiment.

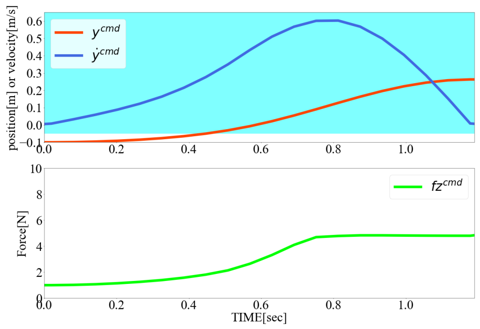

FIGURE 15. Motion trajectory of $\mathrm{NN_{trajectory}}(\dot{q}^{\max} = 1.0$ m/s).

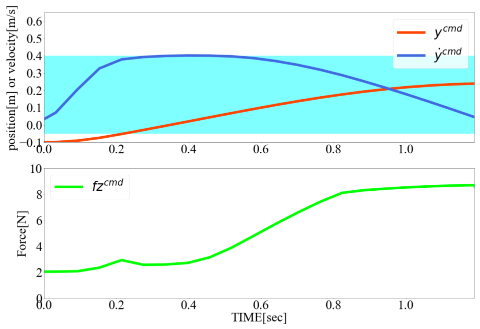

FIGURE 16. Motion trajectory of $\mathrm{NN_{trajectory}}(\dot{q}^{\max} = 0.4$ m/s).

Next, we trained two $\mathrm{NN_{trajectory}}$ models with different velocity restrictions. One was $\mathrm{NN_{trajectory}}(\dot{q}^{\max} = 1.0$ m/s), while the other was $\mathrm{NN_{trajectory}}(\dot{q}^{\max} = 0.4$ m/s). Figures 15 and 16 show the trajectories generated by each model. The light blue area in these figures indicates that it is within the velocity constraint of the robot. The results showed that actions could be generated that satisfied the constraints given by the action generation model. We also confirmed that the pressing force increased when the velocity limitation $\dot{q}^{\max}$ was small. Qualitatively, this is thought to be because a stronger downward pressing force was required to successfully perform the task at the lower velocity.

Next, we provided these motion trajectories to the robot, and attempted the scooping task 10 times. It succeeded 10 times out of 10 with both of the actions. Figures 17 and 18 show the command values and response values. In these figures, the parts shaded in light green indicate the region of

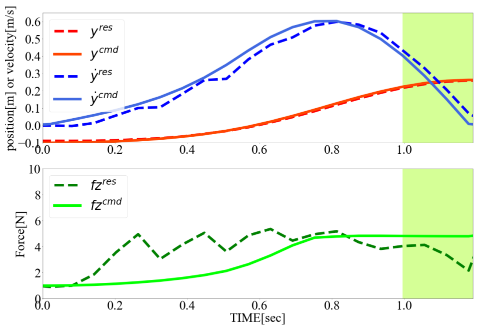

FIGURE 17. **Example of control response values ($\dot{q}^{\max} = 1.0$ m/s).**

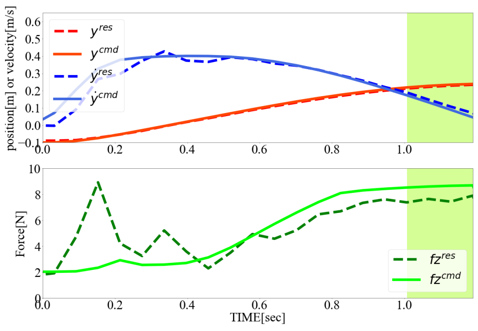

FIGURE 18. **Example of control response values ($\dot{q}^{\max} = 0.4$ m/s).**

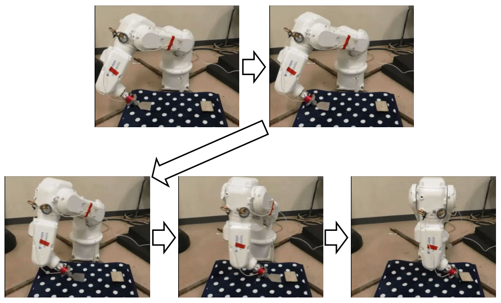

FIGURE 19. **Example of snapshots of scooping task during experiment ($\dot{q}^{\max} = 0.4$ m/s).**

$y \geq 0.215$m where the spatula and the object were believed to be in contact. The results show that the scooping task was achieved successfully with both actions. The graphs also show that position and velocity followed the command values accurately; besides, in the second half of the motion, which is important for the scooping action, force followed the command value, even though some errors remained. Also, Fig. 19 shows snapshots of the action by $\mathrm{NN}_{\mathrm{trajectory}}(\dot{q}^{\max} = 0.4$ m/s. These results show that with our proposed method, actions can be generated to successfully achieve the task, even when the constraints are changed.

VI. CONCLUSION

The purpose of this study was to develop a method to generate actions to achieve a task for which the success conditions are difficult to formulate mathematically, taking into consideration the constraints of different situations. To achieve this, we proposed an action generation method using a success judgment model. By using the success judgment model as a differentiable loss function, motion generation under different constraints can be learned without having to re-collect teaching actions. We verified the effectiveness of this proposed method through simulation and experiments.

In this paper, only kinematic constraints such as velocity limits were considered; however, being able to specify such constraints explicitly for tasks that are not easy to model is expected to be beneficial. Besides, although further verifications are necessary, it is expected that the proposed method can also be extended to reuse human demonstrations between robots with different constraints.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auto. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[2] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, p. 17, Apr. 2018.

[3] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable folding task by humanoid robot worker using deep learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 397–403, Apr. 2017.

[4] R. Rahmatizadeh, P. Abolghasemi, L. Boloni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3758–3765.

[5] 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 *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5628–5635.

[6] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3970–3975.

[7] H. Ochi, W. Wan, Y. Yang, N. Yamanobe, J. Pan, and K. Harada, "Deep learning scooping motion using bilateral teleoperations," in *Proc. 3rd Int. Conf. Adv. Robot. Mechatronics (ICARM)*, Jul. 2018, pp. 118–123.

[8] T. Adachi, K. Fujimoto, S. Sakaino, and T. Tsuji, "Imitation learning for object manipulation based on position/force information using bilateral control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3648–3653.

[9] L. Rozo, P. Jiménez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Intell. Service Robot.*, vol. 6, no. 1, pp. 33–51, Jan. 2013.

[10] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, May 1989.

[11] N. Hirose, R. Tajima, and K. Sukigara, "Personal robot assisting transportation to support active human life—Posture stabilization based on feedback compensation of lateral acceleration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Nov. 2013, pp. 659–664.

[12] T. Tsuji, K. Kutsuzawa, and S. Sakaino, "Optimized trajectory generation based on model predictive control for turning over pancakes," *IEEJ J. Ind. Appl.*, vol. 7, no. 1, pp. 22–28, Jan. 2018.

[13] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[14] P. Lertkultanon and Q.-C. Pham, "Dynamic non-prehensile object transportation," in *Proc. 13th Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, Dec. 2014, pp. 1392–1397.

[15] K. Kutsuzawa, S. Sakaino, and T. Tsuji, "Sequence-to-sequence models for trajectory deformation of dynamic manipulation," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2017, pp. 5227–5232.

[16] I. Lenz, R. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Proc. 11th Robot., Sci. Syst. Conf.*, Jul. 2015, pp. 1–9.

[17] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2407–2414, Jul. 2019.

[18] N. Hirose, F. Xia, R. Martin-Martin, A. Sadeghian, and S. Savarese, ''Deep visual MPC-policy learning for navigation,'' *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3184–3191, Oct. 2019.
[19] E. Talvitie, ''Model regularization for stable sample rollouts,'' in *Proc. 30th Conf. Uncertain. Artif. Intell.*, 2014, pp. 780–789.
[20] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, ''Scheduled sampling for sequence prediction with recurrent neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1171–1179.
[21] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, ''Sequence level training with recurrent neural networks,'' Nov. 2015, *arXiv:1511.06732*. [Online]. Available: https://arxiv.org/abs/1511.06732
[22] D. Tanaka, S. Arnold, and K. Yamazaki, ''EMD net: An Encode–Manipulate–Decode network for cloth manipulation,'' *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1771–1778, Jul. 2018.
[23] A. Byravan, F. Leeb, F. Meier, and D. Fox, ''SE3-Pose-nets: Structured deep dynamics models for visuomotor control,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3339–3346.
[24] K. Kawaharazuka, T. Ogawa, J. Tamura, and C. Nabeshima, ''Dynamic manipulation of flexible objects with torque sequence using a deep neural network,'' in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 2139–2145.
[25] K. Kutsuzawa, S. Sakaino, and T. Tsuji, ''Trajectory adjustment for nonprehensile manipulation using latent space of trained sequence-to-sequence model,'' *Adv. Robot.*, vol. 33, no. 21, pp. 1144–1154, Nov. 2019.
[26] P. Abbeel and A. Y. Ng, ''Apprenticeship learning via inverse reinforcement learning,'' in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, pp. 1–8.
[27] J. Ho and S. Ermon, ''Generative Adversarial Imitation Learning,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4565–4573.
[28] N. Totsu, S. Sakaino, and T. Tsuji, ''A cooking support system with force visualization using force sensors and an RGB-D camera,'' in *Haptic Interaction*, S. Hasegawa, M. Konyo, K.-K. Kyung, T. Nojima, and H. Kajimoto, Eds. Singapore: Springer, 2018, pp. 297–299.
[29] S. Hochreiter and J. Schmidhuber, ''Long short-term memory,'' *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Dec. 1997.
[30] F. Gers, ''Long short-term memory in recurrent neural networks,'' Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2001.
[31] J. Liu, A. Shahroudy, D. Xu, and G. Wang, ''Spatio-temporal LSTM with trust gates for 3D human action recognition,'' in *Proc. Comput. Vis. (ECCV)*, 2016, pp. 816–833.
[32] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu, ''Online human action detection using joint classification-regression recurrent neural networks,'' in *Proc. Comput. Vis. (ECCV)*, 2016, pp. 203–220.
[33] L. Wang, Y. Qiao, and X. Tang. (2014). *Action Recognition and Detection by Combining Motion and Appearance Features*. [Online]. Available: http://crcv.ucf.edu/THUMOS14/
[34] L. Xia, C.-C. Chen, and J. K. Aggarwal, ''View invariant human action recognition using histograms of 3D joints,'' in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 20–27.
[35] G. Evangelidis, G. Singh, and R. Horaud, ''Skeletal quads: Human action recognition using joint quadruples,'' in *Proc. 22nd Int. Conf. Pattern Recognit.*, Aug. 2014, pp. 4513–4518.
[36] R. Smith. (2004). *Open Dynamics Engine*. [Online]. Available: http://ode.org/
[37] M. H. Raibert and J. J. Craig, ''Hybrid position/force control of manipulators,'' *J. Dyn. Syst., Meas., Control*, vol. 103, no. 2, pp. 126–133, Jun. 1981.
[38] O. Khatib, ''A unified approach for motion and force control of robot manipulators: The operational space formulation,'' *IEEE J. Robot. Autom.*, vol. 3, no. 1, pp. 43–53, Feb. 1987.
[39] D. P. Kingma and J. L. Ba, ''Adam: A method for stochastic optimization,'' in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–13.

DAICHI FURUTA received the B.E. degree in electrical and electronic systems and the M.E. degree in science and engineering from Saitama University, Japan, in 2017 and 2019, respectively.

His research interests include robotics, motion generation, and neural networks.

KYO KUTSUZAWA (Graduate Student Member, IEEE) received the B.E. degree in electrical and electronic systems and the M.E. degree in science and engineering from Saitama University, Japan, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree with the Graduate School of Science and Engineering.

He has been a Research Fellow (DC2) with the Japan Society for the Promotion of Science, since 2018. His research interests include robotics, motion generation, force signal processing, and neural networks.

Mr. Kutsuzawa received the Excellent Presentation Awards from the Institute of Electrical Engineers of Japan, in 2019 and the Young Investigation Excellence Award from the Robotics Society of Japan, in 2018.

SHO SAKAINO (Member, IEEE) received the B.E. degree in system design engineering and the M.E. and Ph.D. degrees in integrated design engineering from Keio University, Yokohama, Japan, in 2006, 2008, and 2011, respectively.

He was an Assistant Professor with Saitama University, from 2011 to 2019. Since 2019, he has been an Associate Professor with the University of Tsukuba. His research interests include mechatronics, motion control, robotics, and haptics.

Dr. Sakaino received the IEEJ Industry Application Society Distinguished Transaction Paper Award, in 2011.

TOSHIAKI TSUJI (Senior Member, IEEE) received the B.E. degree in system design engineering and the M.E. and Ph.D. degrees in integrated design engineering from Keio University, Yokohama, Japan, in 2001, 2003, and 2006, respectively.

He was a Research Associate with the Department of Mechanical Engineering, Tokyo University of Science, from 2006 to 2007. He is currently an Associate Professor with the Department of Electrical and Electronic Systems, Saitama University, Saitama, Japan. His research interests include motion control, haptics, and rehabilitation robots.

Dr. Tsuji received the FANUC FA and Robot Foundation Original Paper Award, in 2007 and 2008, respectively.