2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

# Data Augmentation for Continual Learning of Fast and Smooth Imitation Motions Using Model Predictive Control

Akira Kanazawa[1], Hiroshi Ito[1], Hideyuki Ichiwara[1], Yoshiki Kanai[1], Takahiro Yoshida[1], Hiroyuki Yamada[1], and Naoaki Noguchi[1]

*Abstract*— **Research aimed at enabling robots to acquire not only repetitive and simple tasks but also complex and delicate tasks is receiving significant attention to expand the use of robots to support and replace workers. Imitation learning is one of the promising approaches to enable robots to learn complex human skills with minimal learning cost. However, since the motion data generated by human-operated robots serves as the reference motion for the robot, it is difficult for the robot to acquire fast and smooth motions that surpass human operation. In this paper, we propose a data augmentation method to generate faster and smoother motion data by adjusting the robot's motion output from the trained model. By utilizing model predictive control (MPC) for motion adjustment, it becomes possible to balance trackability to the original motion and smoothness of the motion through the design of the evaluation functions and constraints. The robot can perform tasks efficiently and robustly by continuously learning augmented motion data that has been optimized using MPC. We demonstrate through experiments on object picking and placing task that higher-quality motion data generated in the real-world.**

## I. INTRODUCTION

In the background of remarkable progress in learning-based robot motion generation technologies, researches to promote task support and automation with robots in the industrial fields are accelerating. Many research teams are working on the development of end-to-end methods that directly output robot motion commands from sensor information as input, enabling robots to acquire complex skills. [1], [2]. Specifically, through researches adopting imitation learning, robots are beginning to succeed in complex tasks that were difficult to achieve using conventional methods [3], [4].

The authors also propose an imitation-based robot motion generator, called deep predictive learning (DPL) [5]. DPL uses time-series data including the robot's joint angles and camera images as input data and trains a model to minimize the prediction error between the current and next step data. By introducing a spatial attention mechanism that automatically extracts important points from input images, DPL not only reduces the preprocessing costs that require annotation work but also performs robustly against background changes.

Imitation learning, including DPL, can generate complex robot motion similar to those performed by humans with lower learning costs compared to other learning methods such as reinforcement learning. However, since the motion data generated by human-operated robots become the reference motions for the robot, it is difficult for robots to acquire skills that surpass human operation. Due to limitations in the operability of the devices used to generate motion data, there are also limitations in improving data quality in terms of speed and smoothness of robot motions. As a result, there are concerns that the robot's task execution speed, accuracy, and reliability may fall below the inherent capabilities of the robot system. While leveraging the advantages of imitation learning to acquire complex skills with low learning costs, it is desirable to refine the acquired skills continuously.

In this paper, we propose a data augmentation strategy to generate higher quality motion data by adjusting the robot motions in the real-world. By utilizing model predictive control (MPC) for smoothing the speeded-up robot's motions, it becomes possible to balance trackability to the original motion and smoothness of the outputted motion through the design of the evaluation functions and constraints. This strategy allows the robot to move faster and smoother without decreasing the success rate of the originally acquired tasks. We conduct experiments where the actual robot performs the object picking and placing task at double speed, and demonstrate that efficient motion data can be generated by correcting the robot motions through the proposed data augmentation.

## II. RELATED WORKS

### A. Learning-Based Robot Motion Generation

Recent years have seen significant progress in learning-based methods in an end-to-end manner for robot motion generation. In the field of reinforcement learning, many researches are being conducted on the application of deep reinforcement learning to robot manipulation [6]. As representative approaches, methods for transferring skills acquired in simulation to real-world [7] and for directly applying asynchronous reinforcement learning in the real-world [8] have been proposed. However, efficiently exploring and acquiring motions for multi-degree-of-freedom manipulators still requires high learning costs.

Imitation learning that includes image information in the teaching data is currently a hot topic to acquire complex and delicate human skills. Action Chunking with Transformers (ACT) [3] and Diffusion Policy [4] are the most powerful approaches that have been successful in acquiring various manipulation skills using image and angle information as inputs. There are studies that utilize additional information

*This work was supported by JST [Moonshot R&D Program] Grant Number [JPMJMS2031].

[1]The authors are with the Hitachi Ltd., Hitachinaka-shi 312-0034, Ibaraki, Japan. e-mail: akira.kanazawa.qg@hitachi.com

such as force information [9], tactile information [10], and language communication [11] as input. Furthermore, in recent years, large-scale motion foundation models trained on vast amounts of motion data such as $\pi_0$ [12], GR00T [13] and Octo [14] have begun to be proposed. In this paper, we adopted DPL as imitation learning model because it can quickly generate robot motion and is robust to background changes in images. These advantages are useful for generating faster motions for motion data augmentation.

### B. Data Augmentation for Imitation Learning

In imitation learning, since the quantity and quality of training data directly affect the quality and robustness of robot motions, data augmentation is one of the important issues [15]. Augmenting image data to improve environmental robustness is the most common approach, and methods such as random shift and bilinear interpolation are well known for image data augmentation [16], [17]. One of these approaches involves utilizing generative models for data augmentation, and Yu et al. directly generate unseen training images from natural language [18]. Additionally, research has advanced on approaches that augment a small amount of demonstration data using simulation, including methods for generating robot trajectory candidates [19] and techniques for supplementing initiations through simulated transitions [20]. These approaches can generate large amounts of data with minimal effort, but ensuring data quality and adapting to the real world remain-challenging issues.

On the other hand, research has also been conducted on modifying motion data collected once or motions acquired by robots through human intervention. Xu et al. proposed a method in which a human directly intervenes to modify the robot's motions during operation [21]. Inami et al. proposed motion modification method that utilizes bilateral control to modify the force applied by a robot [22]. Additionally, alternative augmentation approaches such as modifying the data sampling of collected data [23] and introducing variable speed adjustments [24] have begun to be proposed.

In this paper, by utilizing MPC for motion correction, we generate high-speed and smooth robot motion data by smoothing motions accelerated by a certain factor from the original motion. We believe that the proposed method has the following advantages.

(i) By smoothing the accelerated motions, our augmentation method can generate motion data that the robot can follow more easily from a control perspective.
(ii) It is possible to maintain trackability to the original motion, thereby suppressing the divergence from learned behaviors, namely the distribution shift problem.
(iii) Since data augmentation is performed on the real-world, the consistency between the accelerated motion data and the vision data is maintained, making it less likely to adversely affect the task success rate and position generalization.

## III. METHOD

### A. Architecture for Fast and Smoothed Motion Data Augmentation

Fig.1 shows the overview of the proposed data autmentation architecture. The proposed architecture consists of three parts: a reference trajectory generator including the DPL as a learning model, MPC, and a position controller. The reference trajectory generator calculates a reference trajectory $\boldsymbol{X}_{ref}$ by recursively using the DPL. We used spatial attention recurrent neural network (SARNN) as DPL network. SARNN explicitly extracts the spatial coordinates of important points (such as target objects and a robot arm) from the input image and learns those points and the robot's joint angles. This inference strategy significantly improves the robustness of motion generation to changing environmental conditions. For more details on this learning model, see reference [25].

Let the state vector including the image and angle at discretized time step $t$ be $\boldsymbol{s}_t = [\boldsymbol{i}_t, \boldsymbol{\theta}_t]$, the predicted state $\hat{\boldsymbol{s}}_{t+1} = [\hat{\boldsymbol{i}}_{t+1}, \hat{\boldsymbol{\theta}}_{t+1}]$ at the next time step $t+1$ by the SARNN can be expressed as

$$\hat{\boldsymbol{s}}_{t+1} = f_{SARNN}(\hat{\boldsymbol{s}}_t) \tag{1}$$

where, $f_{SARNN}$ is the prediction function of an inference processing that encompasses a series of encode and decode processes from the initial input to the final output. By repeating this calculation, the reference trajectory is obtained as

$$\boldsymbol{X}_{ref} = [\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \cdots \hat{\boldsymbol{x}}_{N_r}] \tag{2}$$

$$\hat{\boldsymbol{x}}_t = [\hat{\boldsymbol{\theta}}_t, \hat{\dot{\boldsymbol{\theta}}}_t] \tag{3}$$

Where $N_r$ is the length of the reference trajectory, and $\hat{\dot{\boldsymbol{\theta}}}_t$ is the reference velocity at time step $t$. Since DPL does not predict angular velocity, a temporary constant is set to $\hat{\dot{\boldsymbol{\theta}}}_t$. The reference trajectory generator can arbitrarily set the speed multiplier, and can change the motion speed by adjusting the time step size $\Delta t$ to the desired value.

MPC takes the reference trajectory $\boldsymbol{X}_{ref}$ as input and outputs the optimal state sequence $\boldsymbol{X}_{opt}$ and input sequence $\boldsymbol{U}_{opt}$ that minimizes evaluation functions. In this study, the designed MPC works as a smoother for the reference trajectory $\boldsymbol{X}_{ref}$ and the optimal state sequence $\boldsymbol{X}_{opt}$ corresponds to the smoothed reference trajectory. The calculated optimal state sequence $\boldsymbol{X}_{opt}$ is output to position controller to calculate the control command $\boldsymbol{u}_{com}$ for the robot. In this architecture, we assume that position controller is a simple feedback controller such as PID controller. Finally, the joint angles of the motions reproduced by the robot and the observed vision data are stored as time-series data to use for continual learning.

### B. Formulate Optimization Problem for MPC

We use MPC to smooth out the vibrations and excessive acceleration/deceleration in speeded-up robot motion. Let the robot's state be $\boldsymbol{x}_k = (\boldsymbol{\theta}_k, \dot{\boldsymbol{\theta}}_k)^T$ and control input be $\boldsymbol{u}_k$ at

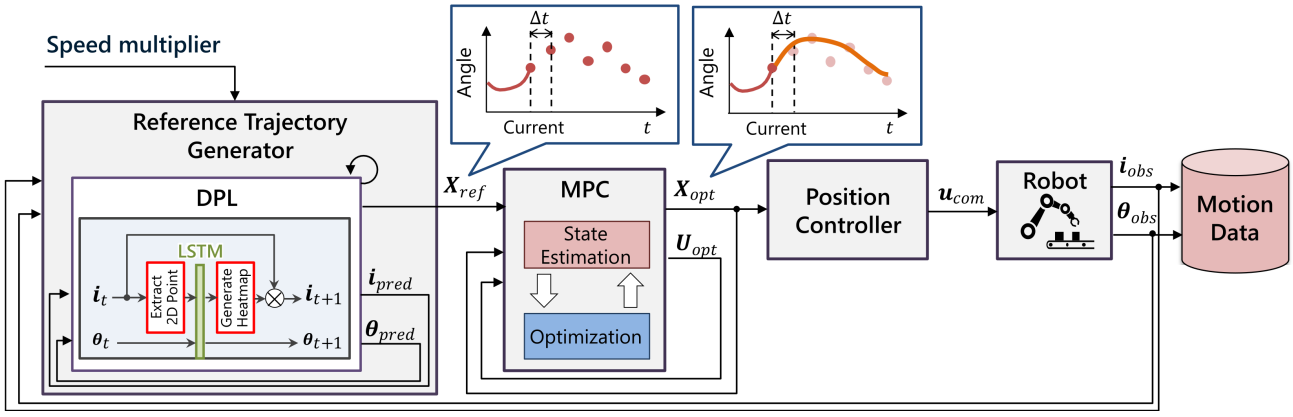



Fig. 1: Architecture to augment the robot motion data with MPC.

**Algorithm 1** Calculate Optimal State $\boldsymbol{X}_{opt}$

**Require:** $\boldsymbol{\theta}_{obs}, \boldsymbol{i}_{obs}$
**Ensure:** $\boldsymbol{X}_{opt}$
$\hat{\boldsymbol{s}}_0 \Leftarrow \boldsymbol{i}_{obs}, \boldsymbol{\theta}_{obs}$
**while** $t = 0$ *to* $N_m$ **do**
  $\hat{\boldsymbol{s}}_{t+1} \Leftarrow f_{SARNN}(\hat{\boldsymbol{s}}_t)$
**end while**
$\boldsymbol{X}_{ref} \Leftarrow [\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \cdots \hat{\boldsymbol{x}}_{N_r}]$
**if** MPC is used first time **then**
  $\boldsymbol{x}_0 \Leftarrow \boldsymbol{\theta}_{obs}, \dot{\boldsymbol{\theta}}_{obs}$
  $\boldsymbol{U} \Leftarrow \boldsymbol{0}$
**else**
  $\boldsymbol{x}_0 \Leftarrow \boldsymbol{x}_{0,opt,pre}$
  $\boldsymbol{U} \Leftarrow \boldsymbol{U}_{opt,pre}$
**end if**
Minimize cost function $J$ by OSQP
Obtain optimal solution $\boldsymbol{X}_{opt}$ and $\boldsymbol{U}_{opt}$
Output $\boldsymbol{X}_{opt}$ to the position controller

discretized time step $k$. We design the evaluation function $J$ to be minimized in MPC as

$$J = \varphi(\boldsymbol{x}_{N_m}) + \sum_{k=1}^{N_m-1} (L_1(\boldsymbol{x}_k) + L_2(\boldsymbol{u}_k, \boldsymbol{u}_{k-1})) \tag{4}$$

$$\varphi(\boldsymbol{x}_{N_m}) = (\boldsymbol{x}_{N_m} - \hat{\boldsymbol{x}}_{N_m})^T S(\boldsymbol{x}_{N_m} - \hat{\boldsymbol{x}}_{N_m}) \tag{5}$$

$$L_1(\boldsymbol{x}_k) = (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)^T Q(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k) \tag{6}$$

$$L_2(\boldsymbol{u}_k, \boldsymbol{u}_{k-1}) = (\boldsymbol{u}_k - \boldsymbol{u}_{k-1})^T R_1(\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) + \boldsymbol{u}_k^T R_2 \boldsymbol{u}_k \tag{7}$$

where, $N_m$ is the length of the prediction horizon. $\varphi$ is the terminal cost to ensure convergence to a reference trajectory at the final state. $L_1$ is the stage cost to track each waypoint in the reference trajectory. $L_2$ is the stage cost related to the input, the first term is an evaluation function for suppressing excessive input, and the second term is for suppressing excessive input changes. $S$, $Q$, $R_1$, and $R_2$ are the weight coefficient matrix for each cost. By increasing the weight coefficient $R_1$, MPC can correct the reference trajectory to be smoother. Note that the weight coefficients $S$ and $Q$ must be sufficiently large to prevent significant deviation of the corrected motion from the original motion. If these weight coefficients are set too small, the motion trajectories may be excessively corrected, which could prevent the learning model from generating proper motions.

The reference trajectory $\boldsymbol{X}_{ref} = [\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \cdots \hat{\boldsymbol{x}}_{N_r}]$ and the state sequence $\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots \boldsymbol{x}_{N_m}]$ can be set to different time steps. Here, let the time step size of the reference trajectory be $\Delta t$ and the time step size of the state sequence be $\Delta k$, the relation between $\Delta t$ and $\Delta k$ becomes as $\Delta k = \alpha \Delta t$, where $\alpha$ is the scaling factor. $\Delta t$ corresponds to the sampling period of the imitation motion, and $\Delta k$ corresponds to the sampling period of the controller, so typically $0 < \alpha \leqq 1$.

Then, the reference trajectory $\boldsymbol{X}_{ref}$ is rediscretized with a time step $k$ and rewritten as $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{\lfloor \alpha t \rfloor}$, where $\lfloor \cdot \rfloor$ is the floor function to extract the integer value. This means that the discretization is done to obtain a zero-order hold of the original reference trajectory. It is also possible to derive the reference waypoints before and after the target point by linear interpolation.

Finally, the optimization problem to minimize these costs is defined as follows:

$$\min_{\boldsymbol{X},\boldsymbol{U}} J = \varphi(\boldsymbol{x}_{N_m}) + \sum_{k=1}^{N_m} \{L_1(\boldsymbol{x}_k) + L_2(\boldsymbol{u}_k, \boldsymbol{u}_{k-1})\} \tag{8}$$

$$s.t. \quad \boldsymbol{x}_0 = \boldsymbol{x}_{cur} \tag{9}$$

$$\boldsymbol{x}_{k+1} = A_d \boldsymbol{x}_k + B_d \boldsymbol{u}_k \tag{10}$$

$$\boldsymbol{x}_{min} \leqq \boldsymbol{x}_k \leqq \boldsymbol{x}_{max} \tag{11}$$

$$\boldsymbol{u}_{min} \leqq \boldsymbol{u}_k \leqq \boldsymbol{u}_{max} \tag{12}$$

where, $\boldsymbol{x}_{cur}$ is the current joint angles and angular velocities, either the observed states $\boldsymbol{x}_{obs}$ or the states $\boldsymbol{x}_{0,opt,pre}$ optimized in the previous step are used. $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$ are the minimum and maximum state, $\boldsymbol{u}_{min}$ and $\boldsymbol{u}_{max}$ are the minimum and maximum control input. $A_d$ and $B_d$ is the linear state equation of the robot. This optimization problem is a quadratic programming problem and can be solved by a quadratic programming solver. We used an

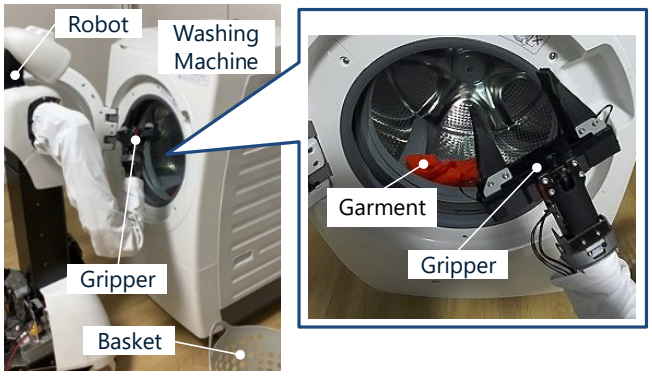



Fig. 2: Experimental environment and robot system used in the experiment.

operator splitting quadratic program (OSQP) which can solve quadratic programming problems quickly [26]. In this study, we assume that the state equation of the robot $A_d, B_d$ is linear to use a quadratic programming solver like OSQP. As in reference [27], we adopted a constant acceleration model with joint angle acceleration as the control input $\boldsymbol{u}$.

The procedure of calculating optimal state sequence $\boldsymbol{X}_{opt}$ and input sequence $\boldsymbol{U}_{opt}$ is summarized in Algorithm 1. First, the observed image $\boldsymbol{i}_{obs}$ and angles $\boldsymbol{\theta}_{obs}$ are used as the initial predicted state $\hat{\boldsymbol{s}}_0$, and recursive calculation of state prediction is performed by the SARNN. Then the optimization problem is solved by OSQP. Here, initial guess solution of input sequence $U$ is set to zero and observed joint angles $\boldsymbol{\theta}_{obs}$ and angular velocities $\dot{\boldsymbol{\theta}}_{obs}$ are used for initial state $\boldsymbol{x}_0$ in case that MPC calculation is the first time. Otherwise, the optimal solution from the previous MPC calculation $X_{opt,pre}$ and $\boldsymbol{U}_{opt,pre}$ used as the initial guess.

## IV. EXPERIMENT

### A. Experimental Setup

We conducted experiments to confirm that the proposed data augmentation method can generate high-speed and smooth motion data in real-world. Fig.2 shows the experimental environment and the robot hardware used in the experiment. In this experiment, a humanoid robot equipped with a parallel gripper at the end of an 8DoF arm was used to perform a picking and placing task. The robot performs a task of picking up one garment from inside the washing machine and placing it into a basket nearby. The entire process of picking and placing task was continuously measured by a camera mounted on the robot's head. This operation was performed 20 times by manual operation and joint angles and image time-series data were collected. The collected motion data was downsampled to 10Hz and used as training data. The proposed control architecture was implemented on a mini-PC equipped with an Intel 13th Gen Intel® Core™ i7-1360P processor and 32 GB of RAM. The quadratic programming solver OSQP is implemented using the open source optimization library CasADi [28].

When performing data augmentation, two approaches were taken: executing the learned original motion at the same speed of 10Hz and at twice that speed of 20Hz. For each motion speed, the generated motion data were compared in two scenarios: one where the motion was smoothed using simple linear interpolation and another where smoothing was performed using MPC. The smoothness of the motion data was evaluated based on motion speed, acceleration, and jerk. The weight coefficients $S, Q, R_1, R_2$ were also tuned in advance using a double-speed motion data. Note that, to speed up the processing time of MPC, MPC was applied only to the five joints starting from the robot's arm base (shoulder), where inertial effects are more pronounced during motion. On the other hand, linear interpolation was applied to the remaining three joints near the end-effector. By applying MPC only to the joints that require high-level smoothing, it became possible to operate MPC at a cycle close to 200Hz.

### B. Experimental Results and Discussions

Fig.3 shows an example of the data augmentation when the robot moves at double speed using MPC. The top row of the figure shows a sequence of the robot's motions: from 0 to 6 seconds, the robot grabs and takes out the garment; around 8 seconds, it places the garment into the basket nearby; finally, the robot's right arm returns to its initial posture. The bottom row of the figure shows the inside of the washing machine, where it can be seen that the garment was picked out by the robot from 0 to 6 seconds.

Fig.4 shows two graphs comparing the original motion data and the double-speed motion. Here, the joint angle trajectories are plotted, and it can be confirmed that the motion time for the double-speed data is approximately half that of the original motion data. Additionally, the solid lines in the graphs represent the measured actual trajectories, while the dashed lines indicate the target trajectories smoothed by MPC. It can be confirmed that even at double speed, the actual trajectories follow the target trajectories closely.

Fig.5 illustrates the comparison of acceleration and jerk that occurred in the generated motion data when using linear interpolation versus MPC. Compared to smoothing with linear interpolation, the graphs show that smoothing with MPC results in smaller magnitudes of acceleration and jerk. Table I summarizes the average absolute velocity, acceleration, and jerk of the generated motion data under four conditions with varying motion speed multipliers and smoothing methods. It can be confirmed that, for both the original speed and double speed operations, the acceleration and jerk are smaller when using MPC compared to linear interpolation. Regarding the double-speed operation, it is confirmed that using MPC reduces the magnitude of acceleration by approximately 30% and the magnitude of jerk by approximately 48%. On the other hand, when using MPC, the average speed is about 4% lower compared to using linear interpolation, but this difference is considered to have a smaller impact relative to the reductions in acceleration and jerk.

Based on the above results, we believe that utilizing MPC enables the robot to generate motion data that allows for more efficient task performance. Note that it is important that the robot successfully completes the task during data

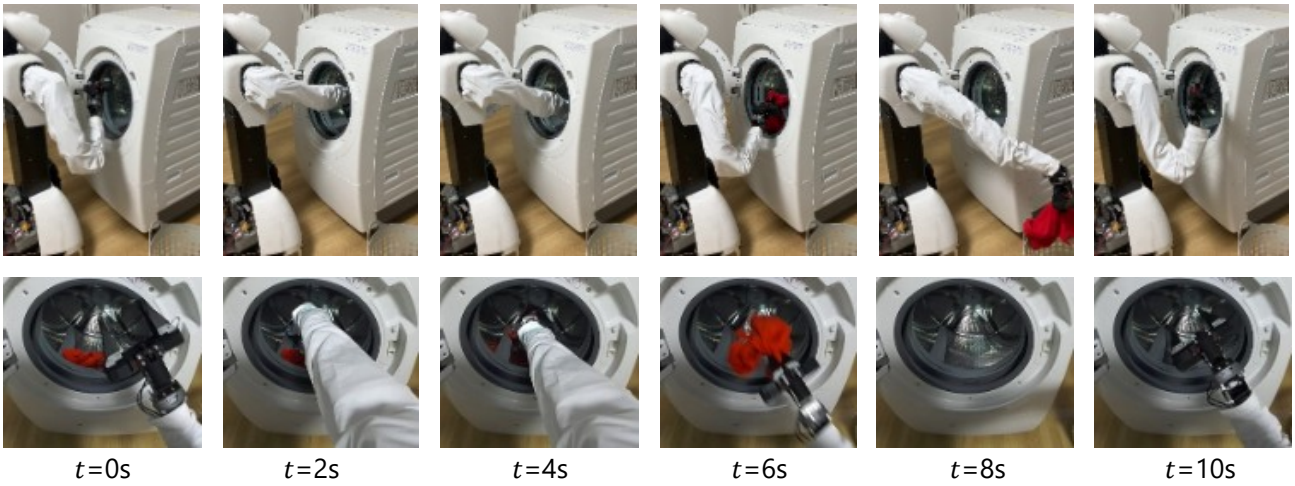



Fig. 3: Snapshot of the picking and placing task performed by the robot.

TABLE I: Comparison of velocity, acceleration, and jerk under each condition.

| Condition | Average absolute velocity [rad/s] | Average absolute acceleration [rad/$s^2$] | Average absolute jerk [rad/$s^3$] |
|---|---|---|---|
| Linear interpolation (10Hz) | 1.614 | 4.988 | 55.823 |
| MPC (10Hz) | 1.581 | 3.975 | 39.757 |
| Linear interpolation (20Hz) | 3.129 | 15.858 | 169.560 |
| MPC (20Hz) | 2.982 | 12.120 | 114.799 |

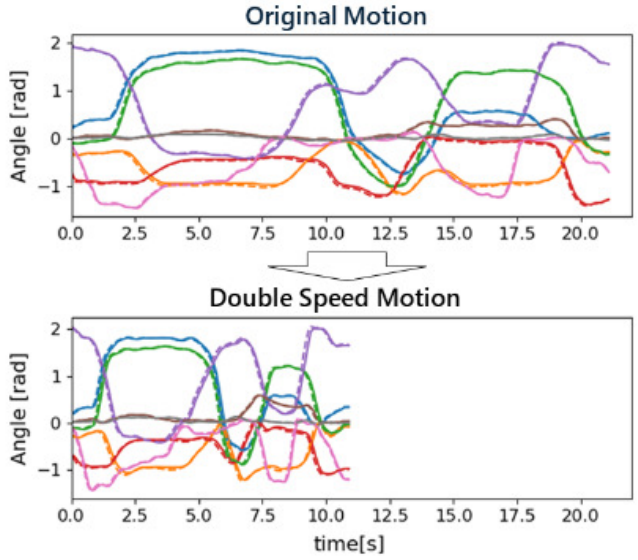



Fig. 4: An example of motion data accelerated to double speed generated using MPC.

augmentation. Even if the robot can generate high-speed and smooth motions, using motion data that resulted in task failure for training can lead to a decrease in the task success rate. This is essentially a distribution shift problem, which makes tuning the weight coefficients of MPC more difficult. We believe that performing data augmentation that captures the dynamical features of robot motions, as in reference [29], enables motion correction to prevent the learning model from encountering unknown states.

## V. CONCLUSION

We proposed a data augmentation strategy that uses MPC to smooth double-speed motion data for continuously learning more efficient robot motions. In the proposed architecture, the DPL generates a reference trajectory to achieve the specified speed multiplier, while MPC corrects the target trajectory to minimize the generated acceleration and jerk, maintaining trackability of the reference trajectory. By executing the task while reproducing the corrected target trajectory, motion data is augmented in the real-world while maintaining consistency between vision data and motion data. We validated the effectiveness of our method on a task where the robot picks up a garment in the washing machine, and quantitatively confirmed that utilizing MPC enables the generation of motion data with smaller acceleration and jerk.

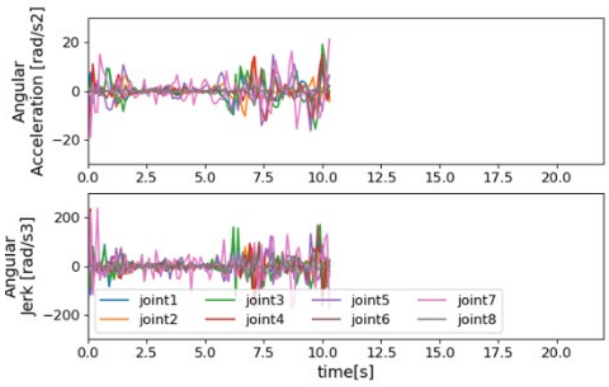
We believe that the proposed data augmentation strategy contributes to the robot acquiring more efficient skills through imitation learning. It is expected that enabling the robot to move faster and more smoothly enables it to complete tasks in a shorter time without decreasing the success rate. In future work, we verify that the robot can perform tasks more efficiently through continual learning using the augmented motion data.

## ACKNOWLEDGMENT

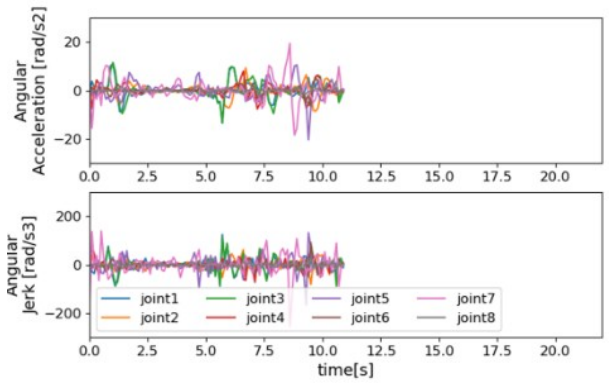
This work was supported by JST [Moonshot R&D Program] Grant Number [JPMJMS2031].

## REFERENCES

[1] S. Levine, C. Finn, T. Darrell, P. Abbeel, "End-to-end training of deep visuomotor policies," Journal of Machine Learning Research, vol. 17, no. 39, pp. 1–40, 2016.




(a) Using linear interpolation




(b) Using MPC

Fig. 5: Comparison of acceleration and jerk in the generated motion data.


[2] H. Ravichandar, A. S. Polydoros, S. Chernova, A. Billard, "Recent advances in robot learning from demonstration," Annual review of control, robotics, vol. 3, no. 1, pp. 297–330, 2020.

[3] T. Z. Zhao, V. Kumar, S. Levine, C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," Preprint at https://arxiv.org/pdf/2304.13705, 2023.

[4] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, S. Song, "Diffusion policy: visuomotor policy learning via action diffusion," Preprint at https://arxiv.org/pdf/2303.04137, 2023.

[5] H. Ito, K. Yamamoto, H. Mori, T. Ogata, "Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control," Science Robotics, vol. 7, no. 65, eaax8177, 2022.

[6] H. Nguyen, H. La, "Review of deep reinforcement learning for robot manipulation," Paper presented at the 2019 Third IEEE international conference on robotic computing (IRC), Naples, Italy, pp. 590–595, 25-27 Feb., 2019.

[7] J. Matas, S. James, A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," Paper presented at the 2nd IEEE international conference on robot learning (CoRL), Zürich, Switzerland, pp. 734–743, 29-31 Oct., 2018.

[8] S. Gu, E. Holly, T. Lillicrap, S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," Paper presented at the 2017 IEEE international conference on robotics and automation (ICRA), Singapore, pp. 3389–3396, 29 May - 4 June, 2017.

[9] J. J. Liu, Y. Li, K. Shaw, T. Tao, R. Salakhutdinov, D. Pathak, "FACTR: Force-attending curriculum training for contact-rich policy learning," Preprint at https://arxiv.org/pdf/2502.17432, 2025.

[10] I. Guzey, Y. Dai, B. Evans, S. Chintala, L. Pinto, "See to touch: Learning tactile dexterity through visual incentives," Paper presented at the 2017 IEEE international conference on robotics and automation (ICRA), Yokohama, Japan, pp. 13825–13832, 13-17 May, 2024.

[11] J. Sun, Q. Zhang, Y. Duan, X. Jiang, C. Cheng, R. Xu, "Prompt, plan, perform: Llm-based humanoid control via quantized imitation learning," Paper presented at the 2017 IEEE international conference on robotics and automation (ICRA), Yokohama, Japan, pp. 16236–16242, 13-17 May, 2024.

[12] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, et al., "$\pi_0$: A vision-language-action flow model for general robot control," Preprint at https://arxiv.org/pdf/2410.24164, 2024.

[13] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, et al., "GR00T N1: An open foundation model for generalist humanoid robots," Preprint at https://arxiv.org/pdf/2503.14734, 2025.

[14] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, et al., "Octo: An open-source generalist robot policy," Preprint at https://arxiv.org/pdf/2405.12213, 2024.

[15] S. Belkhale, Y. Cui, D. Sadigh, "Data quality in imitation learning," Advances in neural information processing systems (NeurIPS), vol.36, pp. 80375–80395, 2023.

[16] D. Yarats, R. Fergus, A. Lazaric, L. Pinto, "Mastering visual continuous control: Improved data-augmented reinforcement learning," Preprint at https://arxiv.org/pdf/2107.09645, 2021.

[17] D. Yarats, I. Kostrikov, R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," Paper presented at the 9th International conference on learning representations (ICLR), 3-7 May, 2021.

[18] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, F. Xia, "Scaling robot learning with semantically imagined experience," Preprint at https://arxiv.org/abs/2302.11550, 2023.

[19] L. Ankile, A. Simeonov, I. Shenfeld, P. Agrawal, "Juicer: Data-efficient imitation learning for robotic assembly," Presented at the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp.5096–5103, 14-18 Oct., 2024.

[20] M. Jia, D. Wang, G. Su, D. Klee, X. Zhu, R. Walters, R. Platt, "Seil: Simulation-augmented equivariant imitation learning," Preprint at https://arxiv.org/pdf/2211.00194, 2022.

[21] X. Xu, Y. Hou, Z. Liu, S. Song, "Compliant Residual DAgger: Improving real-world contact-rich manipulation with human corrections," Preprint at https://arxiv.org/pdf/2506.16685, 2025.

[22] K. Inami, S. Sakaino, T. Tsuji, "Motion ReTouch: Motion modification using four-channel bilateral control," Paper presented at the 2025 IEEE International Conference on Mechatronics (ICM), Wollongong, Australia, 28 Feb-2 Mar., pp. 1–6, 2025.

[23] M. Kobayashi, T. Buamanee, Y. Uranishi, "DABI: Evaluation of data augmentation methods using downsampling in bilateral control-based imitation learning with images," Preprint at https://arxiv.org/pdf/2410.04370, 2024.

[24] N. Masuya, H. Sato, K. Yamane, T. Kusume, S. Sakaino, T. Tsuji, "Variable-speed teaching–playback as real-world data augmentation for imitation learning," Advanced Robotics, pp. 1–16, 2025.

[25] H. Ichiwara, H. Ito, K. Yamamoto, H. Mori, T. Ogata, "Contact-rich manipulation of a flexible object based on deep predictive learning using vision and tactility," Paper presented at the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, pp.5375–5381, 23-27 May 2022.

[26] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd, "OSQP: an operator splitting solver for quadratic programs," Mathematical Programming Computation, vol. 12, no. 4, pp. 637–672, 2020.

[27] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," Paper presented at the 6th International Conference on Robot Learning (CoRL), Auckland, New Zealand, pp. 750–759, 14-18 Dec., 2022.

[28] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," Mathematical Programming Computation, vol. 11. pp. 1–36, 2019.

[29] L. Ke, Y. Zhang, A. Deshpande, S. Srinivasa, A. Gupta, "CCIL: Continuity-based data augmentation for corrective imitation learning," Preprint at https://arxiv.org/pdf/2310.12972, 2023.