2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

DexCtrl: Sim-to-Real Dexterity with Adaptive Controller Learning

Shuqi Zhao, Ke Yang, Yuxin Chen, Chenran Li, Yichen Xie, †Xiang Zhang,
†Changhao Wang, †Masayoshi Tomizuka

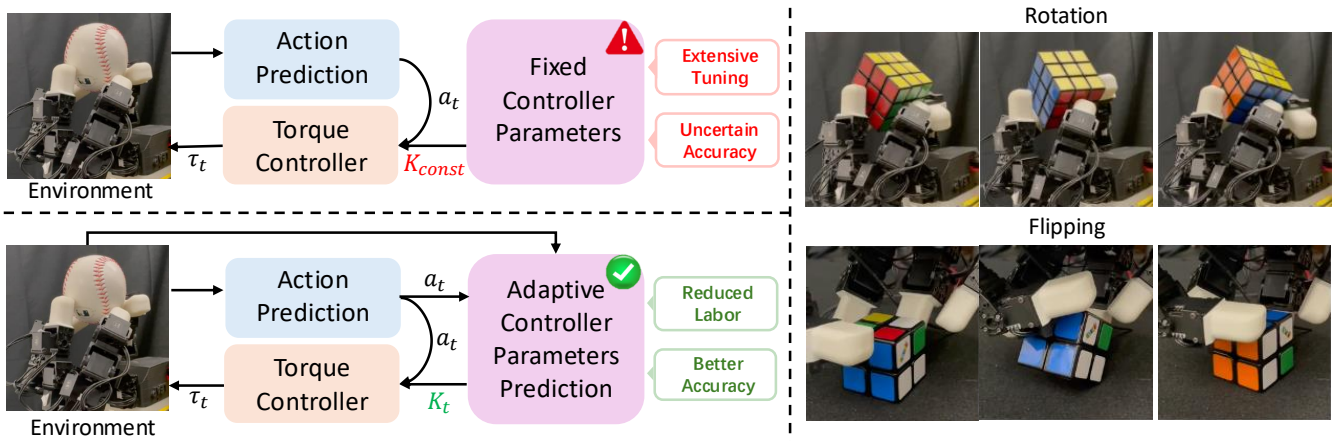

Fig. 1: Dexterous manipulation requires precise force control. While other works typically predict only actions (upper left), DexCtrl (lower left) jointly predicts both actions and control parameters. This joint prediction significantly reduces the need for manual tuning and achieves strong performance on two contact-rich manipulation tasks: rotation and flipping (right).

***Abstract*— Dexterous manipulation has advanced rapidly, with policies now capable of performing complex, contact-rich tasks in simulation. However, transferring these policies from simulation to real world remains a significant challenge. A key obstacle is the mismatch in low-level controller dynamics, where same trajectories can produce vastly different contact forces and behaviors when control parameters change. Existing solutions often rely on manual tuning or controller randomization, which can be labor-intensive, task-specific, and introduce substantial training difficulty. In this work, we propose DexCtrl, a novel framework that jointly learns actions and controller parameters by leveraging the historical information of both trajectory and controller. This adaptive controller adjustment mechanism enables the policy to automatically tune control parameters during execution, thereby mitigating severe sim-to-real gap without extensive manual tuning or excessive randomization. Moreover, by explicitly providing controller parameters as part of the observation, our approach facilitates better reasoning over force interactions and improves robustness in real-world scenarios. Experimental results demonstrate that our method achieves improved transfer performance across a variety of dexterous tasks involving variable force conditions.**

I. INTRODUCTION

Dexterity is a core component of human manipulation and has long posed a significant challenge in robotics research. Beyond their strong performance in grasping [1], [2], [3], [4], dexterous manipulation policies have shown capabilities in handling various contact-rich manipulation tasks such as rotating objects [5], [6], [7], [8], [9], playing the piano [10], [11], and using various tools [12], [13]. Despite the remarkable progress, transferring dexterous policies from simulation to the real world still remains an open challenge. Currently, many manipulation policies have attempted to address the sim-to-real gap from various aspects, such as introducing random noise to the observed proprioceptive information or applying random forces to objects to enhance output trajectory robustness.

However, in previous work, little attention has been paid to one extremely important issue: the discrepancies inside robot controllers between simulation and the real world. Since the final command sent to the robot is the motor torque derived from both the trajectory and control parameters, failing to explicitly consider robot controller gap results in a discrepancy between simulated and real-world performance. Nowadays, people try to bridge this gap by manually tuning controller parameters, which compares the robot's trajectory outputs between simulation and the real world to match the final performance [5], [14]. Additionally, mild randomization of controller parameters during training has also been widely used to enhance policy robustness against sim-to-real discrepancies [7], [12]. However, both manual tuning and randomization can ultimately cause task failure in real-world deployments: randomization leads to a substantial increase in training difficulty, and manually tuning control parameters often fails to achieve the necessary precision for successful task execution. Besides, these strategies require extensive effort in adjusting controller parameters or tuning randomization hyperparameters, significantly increasing human labor.

†Corresponding author. Ke Yang is with New York University, and other authors are with the Department of Mechanical Engineering, the University of California, Berkeley.

Fundamentally, current solutions to this problem have relied merely on extensive attempts based on prior human experiences, rather than a truly principled and automatic approach.

To this end, as shown in Figure 1, we propose a novel method, DexCtrl, that can automatically adjust the controller behavior in a closed-loop manner. Distinct from previous work, it avoids the intensive human labor to manually tune controller parameters or explore randomization ranges. Concretely, DexCtrl model adjusts the controller parameters by learning to output both actions and controller parameters for each time step based on previous desired and actual joint trajectories as well as the corresponding controller parameters within a time window, which captures the subtle force information and narrows the sim-to-real gap. Overall, the contributions of our work are summarized as follows:

- We propose a novel method to adjust the control parameters adaptively, and demonstrate that learning adaptive control parameters at each step can effectively bridge the sim-to-real gap in dynamic multi-contact and high-dimensional dexterous manipulation tasks, thereby extending method application to broader scenarios.
- We design a simple and elegant framework to jointly obtain actions and controller parameters based on historical information, which can offer better adaptivity to force variation.
- Extensive experiments on two different tasks show our method can significantly outperform baselines in both simulation and real world, along with thorough analysis.

II. Related Work

A. Sim-to-real Transfer in Dexterous Manipulation

Dexterous manipulation tasks typically involve complex interactions between robots and objects through contact [15], [16], [17], [18], [19], [20]. Simulation has proven to be an effective way to learn these behaviors [21], [22], [4], [23], as teleoperation [24], [25], [26], [27], [28], [29] is often not feasible due to the embodiment gap and the delicate nature of the tasks. However, the sim-to-real gap remains a significant challenge [21], [30]. To bridge this gap, various approaches have been explored, including system identification, policy fine-tuning, and domain randomization [8], [31]. However, previous methods have largely overlooked the sim-to-real controller gap, while our work focuses on narrowing this sim-to-real controller-level gap and significantly improves task performance.

B. Learning Adaptive Force Control

Learning adaptive force control has been shown to be beneficial for contact-rich manipulation tasks, as varying control parameters can regulate the robot's behavior during interaction. Several works in the literature demonstrate its effectiveness for various contact-rich tasks, such as robotic table wiping [32], [33], object pivoting [34], [35], and assembly [36], [35], [37]. However, this approach has received relatively little attention in the context of dexterous manipulation, and the question of whether and how such a method influences dexterous hand manipulation has not been specifically answered. In our work, we apply the idea of adaptive control to contact-rich dexterous manipulation and prove its efficiency in performance improvement with ample quantitative results, visualization, and discussions.

III. Problem Statement

Manipulation Tasks To validate our approach, we implement two challenging dexterous manipulation tasks that involve *dynamic contact between objects, hands, and environments*: in-hand object rotation and flipping. Goal of the in-hand rotation task is to rotate an object using the fingertips along a specific axis without dropping it, while goal of the flipping task is to flip an object on a table along a designated axis. Both tasks exemplify the contact-rich behaviors characteristic of dexterous manipulation, where the sim-to-real gap significantly impacts real-world performance.

Robot Controller Our method is primarily designed for policies operating under joint torque control. We use the LEAP hand as an example, which has 16 degrees of freedom (DOF). The torque controller takes in the robot hand desired joint trajectories $[q^d, \dot{q}^d] \in \mathbb{R}^{32}$, and robot current trajectories $[q^c, \dot{q}^c] \in \mathbb{R}^{32}$, and computes the corresponding joint torque $\tau \in \mathbb{R}^{16}$ as follows:

$$\tau = K_P(q^d - q^c) + K_D(\dot{q}^d - \dot{q}^c) \tag{1}$$

We assume K_P, K_D are diagonal for simplicity, and define $K = \{K_P, K_D\} \in \mathbb{R}^{32}$ as the collection of controller parameters, representing the robot stiffness and damping matrices, respectively. As shown in Eq. 1, the torque output is directly modulated by the choice of K, which necessitates careful tuning of these parameters. In particular, besides control parameters K that directly determine the actual torque values, increasing stiffness K_P reduces steady-state error but may induce oscillations, while increasing damping K_D suppresses overshoot but can amplify high-frequency noise from a dynamic perspective. In policies without adaptive control mechanisms, the controller parameters are fixed, and only the desired trajectory q_t^d is predicted by the policy. Desired velocities $\dot{q}^d$ are set as zero in both our method and previous work.

IV. Methods

Overall framework of our method is shown in Figure 2. During training (Figure 2 a), we collect sufficient data using an oracle policy trained in simulation with diverse object physical parameters. Then we distill our student policy into two separate models to predict desired action and control parameters respectively, leveraging historical information extracted from the collected dataset. During inference (Figure 2 b), we iteratively predict desired actions for the next step given current and historical observations, and predict control parameters for next step based on generated desired action.

In this section, we introduce detailed information of DexCtrl in following aspects: sim-to-real problem solving (Sec. IV-A), oracle policy for data collection (Sec. IV-B), and student policy training and inference (Sec. IV-C).

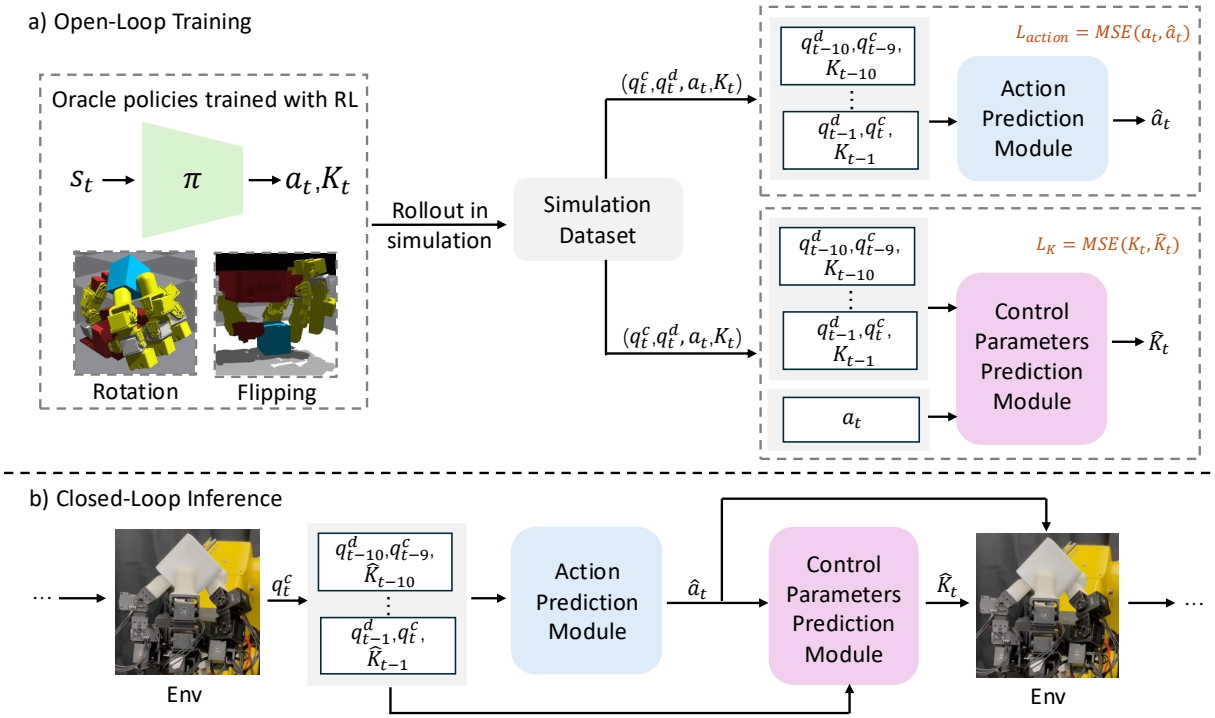

Fig. 2: Overview framework of DexCtrl, where $\hat{a}_t$ and $\hat{K}_t$ mean predicted joint position actions and predicted control parameters, respectively.

A. Solving Sim-to-Real Problem with Adaptive Controller

Previous work typically relies on fixed controller parameter K_{const} and only predicts desired joint trajectory q_t^d.

$$\tau = f(q_t^d | K_{const}), \quad q_t^d = g(s_t, o_t) \tag{2}$$

whose performance heavily depends on the manual tuning of controller parameters for each task separately.

In contrast, DexCtrl also includes controller parameters K_t as action and observation along with q_t^d at each time step:

$$\tau = f(q_t^d | K_t), \quad q_t^d = g_{\text{traj}}(s_t, o_t), \quad K_t = g_{\text{ctrl}}(s_t, o_t) \tag{3}$$

This enables synchronous controller adjustment at each time step to meet force requirements based on historical information, mitigating the sim-to-real transfer problem. Specifically, by leveraging historical tracking errors, the policy implicitly estimates underlying system properties such as mass, friction, or delay, which are difficult to model precisely in real world. This history-conditioned parameter adaptation prevents overfitting to simulated dynamics and provides online domain adaptation when encountering unseen real-world variations. Moreover, it works as an implicit system identification mechanism by analyzing the temporal dependency inside past observation, which can continuously refine the control law to align with the true dynamics in an online manner rather than merely reacting to errors.

Despite similar methods in gripper-based tasks, they only demonstrate adaptive control's ability to mitigate sim-to-real gaps with simple dynamic contact between object and external surfaces where gripper and object are treated as a whole. However, dexterous manipulation faces fundamentally different challenges because it involves complex, multiple dynamic concerns among robot hand, object, and environment, which introduce more severe sim-to-real discrepancies. DexCtrl proves that similar methodology can also solve multi-dynamic contact patterns, thereby extending method application to broader scenarios.

B. Oracle Policy for Data Collection

We train oracle policies in simulation with model-free Proximal Policy Optimization (PPO) reinforcement learning. Specifically, for each time step t, oracle policy $\pi(a_t, K_t | s_t)$ takes in state s_t, outputs joint action a_t and K_t simultaneously. The action a_t is executed using a controller with parameters K_t. The desired joint trajectories at t time step are obtained from $q_t^d = q_{t-1}^d + a_t$. Detailed task designs for two contact-rich manipulation tasks are described as follows:

State: The states $s_t \in \mathbb{R}^{219}$ contain observation of object and robot over the last three time steps. Robot information $s_t^r \in \mathbb{R}^{64}$ for each step includes current joint positions q_t^c, desired joint positions q_t^d and controller parameters K_t. Object information $s_t^{obj} \in \mathbb{R}^9$ contains object pose $p_t^{obj} \in \mathbb{R}^6$ and object property vector $\mu \in \mathbb{R}^3$, including scale, mass and friction. In short, the state can be presented as: $s_t \triangleq (s_{t-2:t}^r, s_{t-2:t}^{obj})$, $s_t^r \triangleq (q_t^c, q_t^d, K_t)$, $s_t^{obj} \triangleq (p_t^{obj}, \mu)$

Reward: The reward function r_t has four parts: $r_t = r_{rotation} + r_{contact} + r_{smoothness} + r_{terminate}$. Rotation speed reward $r_{rotation}$ encourages object to rotate faster along a certain axis until it reaches the targeted maximum

speed. Contact reward $r_{contact}$ encourages binary contacts between the object and the fingertips. Smoothness reward $r_{smoothness}$ penalizes sudden changes of robot joint positions and torques. Terminal reward $r_{terminate}$ penalizes objects when falling off the fingertips (Rotation) and moving too far from initial positions (Flipping).

C. Action Prediction and Control Parameters Prediction

We separately train our student policy into two modules, *i.e.*, an action prediction module for trajectory generation and a control parameter prediction module for adapting control parameters. They encode fundamentally different aspects of the task, and also prevent control parameter prediction from affecting action prediction. Moreover, such design potentially enables generalization across tasks with few-shot fine-tuning on control parameter module in the future and allows the reuse of trajectory-only datasets without retraining via reinforcement learning. Also, it allows different frequencies of parameter adaptation and trajectory prediction, which is particularly important for fine-grained manipulation.

Historical Information for Distillation Though feasible for task completion, oracle policies cannot be directly transferred to the real world because some primitive information such as object information is not directly accessible in real-world settings. To solve this problem, our method utilizes historical states of robot proprioception to distill primitive information used in oracle policy, enabling the estimation of rotated object properties [6]. Concretely, we use the last ten steps of the current and desired joint trajectories, along with corresponding controller parameters as historical information for policy input. By conditioning on the temporal trajectory of control parameters, the policy can be equipped with awareness of their temporal trends, facilitating inference of interaction forces and requirements of smooth trajeetories.

Module Design To better leverage historical information, we use self-attention to model temporal historical input for action prediction. For control parameter prediction, we use cross-attention where current action serves as query and historical input serves as key and value, modeling the relationship between the current action and historical input. Although the historical input formulations of the two modules remain the same, their meaning is completely different. In action prediction, the input mainly indicates the trend of joint trajectory variation, similar to the previous work [7], [8]. In control parameter prediction, it mainly indicates an approximate relation between joint actions and control parameters, which can be analogized to how humans infer current control parameters decisions based on historical trajectories and previous parameters.

Training and Inference The two modules are trained in an open-loop manner, where all input data is directly retrieved from the collected simulation dataset. However, during both simulation and real-world inference, our method performs closed-loop behaviors. In this case, the values of current trajectories are obtained from actual robot sensors. We linearly map the control parameters from simulation to their corresponding values on the real-world system, with only an approximate estimate of upper and lower bounds instead of careful parameter tuning. Furthermore, we find that adding Gaussian noise to the current trajectory values during student policy training is enough for sim-to-real transfer, even though the training dataset is not collected by an oracle policy with input noise randomization. This finding indicates that certain randomization on observation can be reduced to ease the oracle policy training procedure.

V. Experiments

We conducted experiments in both simulated and real-robot environments to evaluate our proposed method. Specifically, we present results on contact-rich dexterous manipulation tasks, examining two key aspects: in-hand rotation, which emphasizes **object-hand interaction**, and object flipping, which highlights **dynamic contact among hand, robot and environment**. Our investigation primarily addresses key questions through in-hand rotation experiments, with in-hand flipping serving as a supporting task to provide additional insights: **Q1:** Does our method improve original oracle performance? **Q2:** Does our method narrow the sim-to-real gap? **Q3:** How does our method perform across objects with varying physical parameters? **Q4:** How do controller parameters changes impact results?

A. Experimental Setup

Baselines We compare DexCtrl with two main baselines:

- *Fixed$_T$+Fixed$_S$:* Similar to [5], [7], this baseline adopts fixed controller parameters for both oracle and student policies. The controller parameters are tuned carefully by comparing the trajectory outputs between simulation and real world. A small range of randomization is also applied to the controller.
- *Ada$_T$+Fixed$_S$:* This baseline only adopts adaptive controller parameters learning in the oracle policy, while fixed controller parameters same as the first baseline is applied to the student policy during inference.

Table I further explains the differences between DexCtrl and two baselines. Overall, *Fixed$_T$+Fixed$_S$* stands for the basic trajectory-based method with manually tuned control parameters and mild randomization, while *Ada$_T$+Fixed$_S$* is served as an intermediate method to investigate whether DexCtrl can also enhance the trajectory sim-to-real performance.

TABLE I: Comparison between DexCtrl and two baselines, where ✓ indicates policy includes learning control parameters during training phase.

Methods	Oracle Policy	Student Policy
Fixed_T+Fixed_S	×	×
Ada_T+Fixed_S	✓	×
Ada_T+Ada_S (**Ours**)	✓	✓

Metrics We quantitatively evaluate the performance of all methods with four metrics [6], [7]:

- *Rotation Reward/Radians (RotR)*: This metric represents the rotation speed of the targeted object around the desired axis. In simulation, it is calculated as the average rotation reward over a trajectory. In the real world, it is calculated by the net rotation of object in radians over a trajectory.
- *Time To Fail (TTF):* This metric represents the average trajectory length before the object falls off the hand or moves too far from its initial positions in rotation and flipping tasks, respectively.
- *Object linear Velocity (ObjVel):* This metric represents the average magnitude of object linear velocity per action step. It reflects the stability of the targeted object. This metric is only calculated in the simulation because real-world object velocity can not be easily measured in rotation. It is only evaluated in rotation because such behavior is inevitable in flipping.
- *Torque Penalty (Torque):* This metric computes the torque penalty reward per time step to measure the energy cost and it is only measured in simulation.

B. Does our Method Improve Original Oracle Performance?

We first evaluate DexCtrl in a simulation environment with and without randomly applying force disturbances on objects. During validation, we randomize 1024 different initial robot poses and set the simulation controllers exactly the same as those used during training, meaning no controller gap is involved. This experiment aims to show whether performance can be improved by simultaneously adjusting action and control parameters even under the same controllers. Table II and Table III present the quantitative results of baselines and DexCtrl in simulation validation. Compared to $Fixed_T$+$Fixed_S$, DexCtrl improves performance significantly, especially in RotR and TTF. This demonstrates that our method can effectively stabilize and accelerate the task process with or without disturbance. It's worth noticing that Ada_T+$Fixed_S$ also reaches relatively good performance, indicating the robustness of trajectories generated by our oracle policy. However, in the flipping tasks with results in Table III, Ada_T+$Fixed_S$ does not outperform the baseline. We believe it is because flipping involves rich contact between both the floor and the dexterous hands, making it more sensitive to controller parameters' variance. In conclusion, our method outperforms the baseline in both task performance and training speed, even in the absence of the controller gap.

C. Does our Method Narrow the Sim-to-real Gap?

We directly deploy DexCtrl in real-world scenarios. For the in-hand rotation task, we use 12 different real-world unseen objects with varying masses and frictions, and validate rotation performance based on the average metrics over 10 randomly sampled initial robot poses per object. Table IV and Figure 4 present quantitative results and visualizations of DexCtrl along with baselines across different objects, respectively. Compared to the simulation results, the performance gap among different methods is more pronounced in the real world, where DexCtrl significantly outperforms the baseline under zero-shot sim-to-real transfer. Also, Ada_T+$Fixed_S$ achieves relatively strong performance, demonstrating that trajectories generated by DexCtrl can be more robust when transferred to real-world scenarios. It's worth noticing that the performance gap between DexCtrl and Ada_T+$Fixed_S$ is much larger in the real world than in simulation, which highlights the necessity of adaptively adjusting control parameters at every step in real-world robots. We also conduct real-world experiments on the flipping task with visualizations shown in Figure 3, demonstrating the real-world task generalizability of our method.

D. How does our Method Perform across Objects with Varying Physical Parameters?

To further evaluate our policy, we conduct additional experiments on objects with different masses and friction coefficients. To ensure better control over confounding factors, we use a hollow cube for mass experiments and vary its mass by inserting different internal objects, and use cubes of different textures with the same mass for friction experiments. As shown in Table V and Figure 5, our method significantly outperforms both baselines, especially on heavy objects, indicating that it can better adapt to objects with different physical parameters. Also, the pattern of our method broadly aligns with our force-based predictions, namely, the lightest and smoothest objects exhibit the highest speed and lowest stability, respectively.

E. How do Controller Parameters Changes Impact Results?

In this section, we investigate the fundamental question: how do the learned controller parameters affect dexterous manipulation performance? To simplify the analysis, we focus solely on the pattern of learned stiffness, as it contributes more than damping to the task performance. Theoretically, changes in stiffness directly influence the resulting joint torques, thereby modulating the contact forces between the robot and the manipulated object. Given that objects with different physical properties (e.g., mass and friction) require distinct contact force profiles, we hypothesize that variations in stiffness are closely related to object-specific configurations, aiming to provide better adaptation to varying force requirements.

To validate this hypothesis, we first analyze data collected in the simulation. Figure 6 illustrates the relationship between the average stiffness observed across trajectories and object mass as well as surface friction. The results reveal that stiffness increases monotonically with mass (left of Figure 6), consistent with the intuitive physical principle that heavier objects require greater force for manipulation. However, the relationship between stiffness and friction is more nuanced: in some cases, stiffness increases with friction (middle of Figure 6), while in others it decreases (right of Figure 6). We attribute this inconsistency to task-dependent dynamics. For instance, in rotation tasks, friction may primarily resist angular motion at certain joints, while at others it may act more like a supporting or pushing force. This suggests that the role of friction—and consequently the required

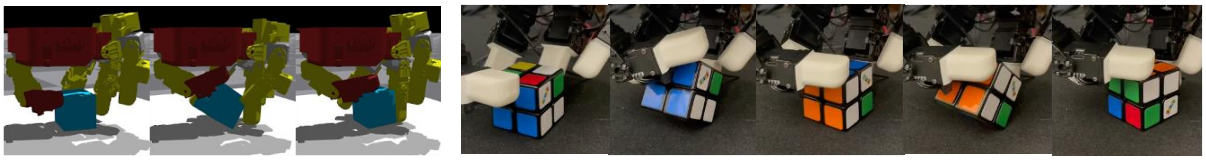

Fig. 3: Flipping task performance in simulation and real world.

Fig. 4: Real-world results of object rotation with different physical parameters.

TABLE II: Rotation oracle policy in simulation.

	Method	RotR ↑	TTF ↑	Torque ↓	ObjVel ↓
With Disturbance	Fixed_T+Fixed_S	35.05	239.4	0.398	0.154
	Ada_T+Fixed_S	41.60	252.3	0.152	**0.140**
	Ada_T+Ada_S (**Ours**)	**43.51**	**255.9**	**0.099**	0.144
Without Disturbance	Fixed_T+Fixed_S	37.64	247.5	0.264	0.148
	Ada_T+Fixed_S	47.87	275.8	0.144	**0.131**
	Ada_T+Ada_S (**Ours**)	**52.33**	**287.7**	**0.092**	0.132

TABLE III: Flipping oracle policy in simulation.

	Method	RotR ↑	TTF ↑	Torque ↓
With Disturbance	Fixed_T+Fixed_S	91.07	295.2	0.376
	Ada_T+Fixed_S	82.23	295.6	0.299
	Ada_T+Ada_S (**Ours**)	**172.50**	**296.9**	**0.140**
Without Disturbance	Fixed_T+Fixed_S	92.24	295.0	0.446
	Ada_T+Fixed_S	82.90	295.4	0.328
	Ada_T+Ada_S (**Ours**)	**184.00**	**296.9**	**0.127**

TABLE IV: Quantitative results of rotation performance for objects with different masses and frictions.

		Cube(94g)	Bottle(150g)	Apple(221g)	Yogurt(164g)	Baseball(144g)	Average
Fixed_T+Fixed_S	RotR ↑	1.963	2.875	1.914	2.943	2.745	2.431
	TTF ↑	286.5	266.9	242.7	**300**	239.6	272.4
Ada_T+Fixed_S	RotR ↑	5.498	4.285	4.492	5.424	4.681	4.986
	TTF ↑	**297.7**	281.7	291.6	**300**	271.6	287.2
Ada_T+Ada_S (**Ours**)	RotR ↑	**9.386**	**14.006**	**9.676**	**15.017**	**11.342**	**11.041**
	TTF ↑	289.3	**300**	**300**	**300**	**292.2**	**292.6**

TABLE V: Quantitative results for same-shape objects rotation with different masses and frictions.

Methods	Metrics	Mass			Friction		
		Light	Medium	Heavy	Small	Medium	Large
Fixed_T+Fixed_S	RotR ↑	2.042	1.963	0.628	1.963	1.374	2.231
	TTF ↑	286.5	286.5	243	286.5	239.9	288
Ada_T+Fixed_S	RotR ↑	4.998	5.498	2.356	5.498	4.355	4.855
	TTF ↑	**298.8**	**297.7**	264.9	**297.7**	278.6	286.6
Ada_T+Ada_S (**Ours**)	RotR ↑	**10.414**	**9.386**	**5.998**	**9.386**	**10.211**	**10.681**
	TTF ↑	258.2	289.3	**300**	289.3	**300**	**290.4**

stiffness—depends on both the object's properties and the specific joint-task interaction.

To further investigate this relationship in real-world settings, we isolate the controller parameters prediction module

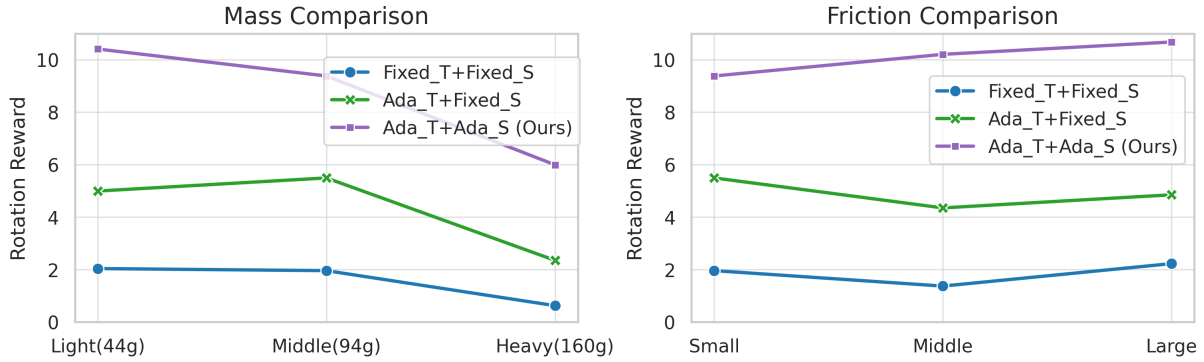

Fig. 5: Visualization for same-shape objects rotation with different masses and frictions.

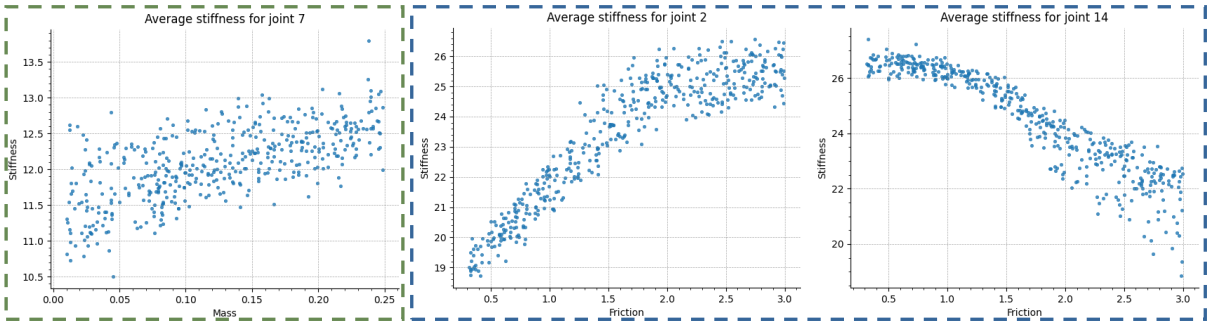

Fig. 6: Average stiffness curve under mass (left) and friction change (middle and right).

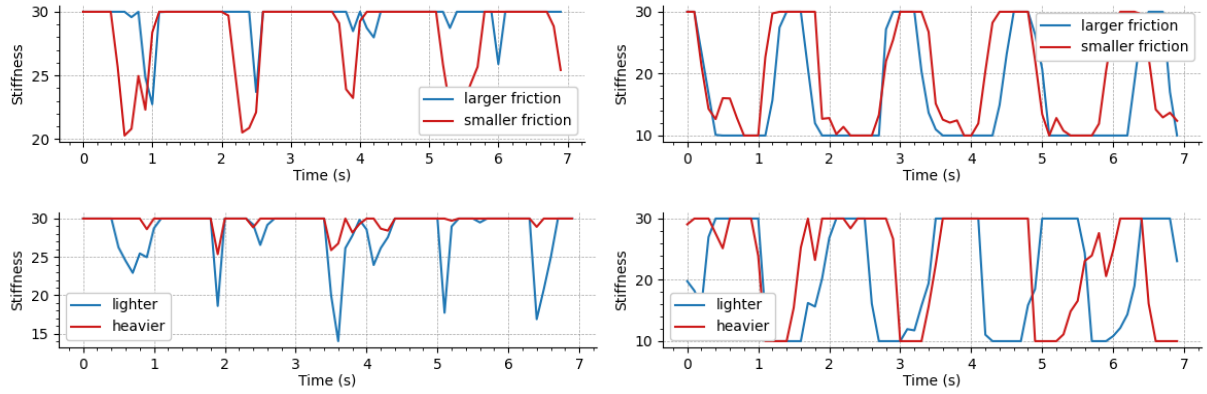

Fig. 7: Stiffness over time under varying mass and friction.

and provide ground-truth actions to only predict controller parameters. This allows us to observe the influence of object properties without the confounding effects of policy action noise. Figure 7 presents the trends across objects with varying mass and friction. Two consistent patterns emerge: (1) for heavier objects, stiffness tends to increase at specific time steps or remain at its maximum value for longer durations; (2) for smoother objects, stiffness exhibits similar patterns at some joints but opposite trends at others, again indicating task- and joint-specific behavior.

In summary, our results show that the learned stiffness adapts systematically to object mass and friction, validating our hypothesis that controller parameters encode variations in required contact forces and thereby enhance manipulation performance through better adjustment to force requirements.

VI. Conclusion

In this work, we address the challenge of narrowing the sim-to-real gap in dexterous manipulation by applying adaptive control parameters, and propose a novel method DexCtrl that jointly outputs actions and control parameters based on historical information. To validate DexCtrl, we conduct comprehensive experiments in both simulation and real-world scenarios, and analyze how control parameters contribute to performance improvements through in-hand rotation tasks. We also apply DexCtrl to the flipping task, demonstrating its generalizability across dexterous manipulation settings. In the future, we plan to expand the applicability of our policy so that multiple dexterous tasks can share a single control parameters prediction module. Also, if supported by hardware, another promising direction is to perform online adaptation based on real-time force feedback.

VII. Acknowledgement

Material in this paper is based upon work supported by FANUC. We gratefully acknowledge the FANUC Advanced Research Laboratory for their support.

References

[1] J. Zhang, H. Liu, D. Li, X. Yu, H. Geng, Y. Ding, J. Chen, and H. Wang, "Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes," in *8th Annual Conference on Robot Learning*, 2024.

[2] Z.-H. Yin, C. Wang, L. Pineda, K. Bodduluri, T. Wu, P. Abbeel, and M. Mukadam, "Geometric retargeting: A principled, ultrafast neural hand retargeting algorithm," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 17 376–17 382.

[3] J. Ye, J. Wang, B. Huang, Y. Qin, and X. Wang, "Learning continuous grasping function with a dexterous hand from human demonstrations," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2882–2889, 2023.

[4] M. Yang, A. Church, Y. Lin, C. J. Ford, H. Li, E. Psomopoulou, D. A. Barton, N. F. Lepora *et al.*, "Anyrotate: Gravity-invariant in-hand object rotation with sim-to-real touch," in *Conference on Robot Learning*. PMLR, 2025, pp. 4727–4747.

[5] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand reorientation of novel and complex object shapes," *Science Robotics*, vol. 8, no. 84, p. eadc9244, 2023. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adc9244

[6] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-hand object rotation via rapid motor adaptation," in *Conference on Robot Learning*. PMLR, 2023, pp. 1722–1732.

[7] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik, "General in-hand object rotation with vision and touch," in *Conference on Robot Learning*. PMLR, 2023, pp. 2549–2564.

[8] J. Wang, Y. Yuan, H. Che, H. Qi, Y. Ma, J. Malik, and X. Wang, "Lessons from learning to spin "pens"," in *Conference on Robot Learning*. PMLR, 2025, pp. 3124–3138.

[9] H. Qi, B. Yi, M. Lambeta, Y. Ma, R. Calandra, and J. Malik, "From simple to complex skills: The case of in-hand object reorientation," *arXiv preprint arXiv:2501.05439*, 2025.

[10] K. Zakka, P. Wu, L. Smith, N. Gileadi, T. Howell, X. B. Peng, S. Singh, Y. Tassa, P. Florence, A. Zeng *et al.*, "Robopianist: Dexterous piano playing with deep reinforcement learning," in *7th Annual Conference on Robot Learning*.

[11] C. Qian, J. Urain, K. Zakka, and J. Peters, "Pianomime: Learning a generalist, dexterous piano player from internet demonstrations," in *8th Annual Conference on Robot Learning*.

[12] Z.-H. Yin, C. Wang, L. Pineda, F. Hogan, K. Bodduluri, A. Sharma, P. Lancaster, I. Prasad, M. Kalakrishnan, J. Malik *et al.*, "Dexteritygen: Foundation controller for unprecedented dexterity," *arXiv preprint arXiv:2502.04307*, 2025.

[13] X. Liu, J. Adalibieke, Q. Han, Y. Qin, and L. Yi, "Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references," in *The Thirteenth International Conference on Learning Representations*.

[14] M. Yu, B. Liang, X. Zhang, X. Zhu, L. Sun, C. Wang, S. Song, X. Li, and M. Tomizuka, "In-hand following of deformable linear objects using dexterous fingers with tactile sensing," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 13 518–13 524.

[15] T. Lin, Z.-H. Yin, H. Qi, P. Abbeel, and J. Malik, "Twisting lids off with two hands," in *Conference on Robot Learning*. PMLR, 2025, pp. 5220–5235.

[16] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[17] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[18] C. B. Teeple, B. Aktaş, M. C. Yuen, G. R. Kim, R. D. Howe, and R. J. Wood, "Controlling palm-object interactions via friction for enhanced in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2258–2265, 2022.

[19] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang, "Rotating without seeing: Towards in-hand dexterity through touch," *arXiv preprint arXiv:2303.10880*, 2023.

[20] H.-S. Fang, B. Romero, Y. Xie, A. Hu, B.-R. Huang, J. Alvarez, M. Kim, G. Margolis, K. Anbarasu, M. Tomizuka *et al.*, "Dexop: A device for robotic transfer of dexterous human manipulation," *arXiv preprint arXiv:2509.04441*, 2025.

[21] D. Guo, Y. Xiang, S. Zhao, X. Zhu, M. Tomizuka, M. Ding, and W. Zhan, "Phygrasp: Generalizing robotic grasping with physics-informed large multimodal models," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 14 915–14 922.

[22] S. Zhao, X. Zhu, Y. Chen, C. Li, Y. Xie, X. Zhang, M. Ding, and M. Tomizuka, "Dexh2r: Task-oriented dexterous manipulation from human to robots," *IEEE/ASME Transactions on Mechatronics*, 2025.

[23] F. Lan, S. Wang, Y. Zhang, H. Xu, O. O. Oseni, Z. Zhang, Y. Gao, and T. Zhang, "Dexcatch: Learning to catch arbitrary objects with dexterous hands," in *Conference on Robot Learning*. PMLR, 2025, pp. 2965–2981.

[24] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu, "Dexcap: Scalable and portable mocap data collection system for dexterous manipulation," *arXiv preprint arXiv:2403.07788*, 2024.

[25] K. Shaw, Y. Li, J. Yang, M. K. Srirama, R. Liu, H. Xiong, R. Mendonca, and D. Pathak, "Bimanual dexterity for complex tasks," in *Conference on Robot Learning*. PMLR, 2025, pp. 5166–5183.

[26] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," *arXiv preprint arXiv:2401.02117*, 2024.

[27] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto, "Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation," in *2023 ieee international conference on robotics and automation (icra)*. IEEE, 2023, pp. 5954–5961.

[28] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang, "Dexmv: Imitation learning for dexterous manipulation from human videos," in *European Conference on Computer Vision*. Springer, 2022, pp. 570–587.

[29] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, S. K. S. Ghasemipour, C. Finn, and A. Wahid, "Aloha unleashed: A simple recipe for robot dexterity," in *Conference on Robot Learning*. PMLR, 2025, pp. 1910–1924.

[30] K. Xu, S. Zhao, Z. Zhou, Z. Li, H. Pi, Y. Zhu, Y. Wang, and R. Xiong, "A joint modeling of vision-language-action for target-oriented grasping in clutter," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 597–11 604.

[31] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang, "Understanding domain randomization for sim-to-real transfer," in *International Conference on Learning Representations*.

[32] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 1010–1017.

[33] C. Wang, X. Zhang, Z. Kuang, and M. Tomizuka, "Safe online gain optimization for cartesian space variable impedance control," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2022, pp. 751–757.

[34] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[35] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka, "Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 1621–1639.

[36] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," *Applied Sciences*, vol. 10, no. 19, p. 6923, 2020.

[37] X. Zhang, M. Tomizuka, and H. Li, "Bridging the sim-to-real gap with dynamic compliance tuning for industrial insertion," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4356–4363.