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

Learning Dynamic Non-Prehensile Object Reorientation via Reinforcement Learning

Abdullah Mustafa*[1], Ryo Hanai[1], Ixchel G. Ramirez-Alpizar[1], Floris Erich[1], Ryoichi Nakajo[1], Yukiyasu Domae[1] and Tetsuya Ogata[2]

***Abstract*— This work proposes a learning-based approach to dynamic non-prehensile object reorientation, enabling fast reorientation of large, grasp-infeasible objects using uni-manual manipulation. Our policy is trained in simulation via reinforcement learning, utilizing a carefully designed observation space, action space, and reward function to reorient randomly sized cuboids with varied physical properties. Given an object model and a target rotation direction, the policy plans offline trajectories suitable for both simulation and real-world deployment. Although the policy is sensitive to modeling uncertainties, experimental results show that accurate modeling enables successful sim-to-real transfer across different objects and rotation directions. Videos are available at `https://tinyurl.com/DNPR2L`.**

I. Introduction

On a conveyor belt, boxes of various sizes arrive with random orientations. The task is to reorient each box to a specific target orientation—quickly. However, many of these boxes are too large to be grasped and reoriented using two-fingered grippers or dexterous hands. Complicating matters further, the robot is equipped with only a single arm. How can it accomplish this task efficiently?

Our solution is dynamic non-prehensile object reorientation. For relatively large objects, prehensile manipulation methods such as grasping or suction may be infeasible. Meanwhile, traditional non-prehensile approaches—like planar pushing, exploiting external surfaces, or using bi-manual robots—often result in slow, long-horizon plans. In contrast, carefully planned dynamic trajectories can generate the necessary forces and moments for instantaneous reorientation of both large and small objects.

To generate such dynamic non-prehensile trajectories, model-based approaches such as those proposed by Lynch [1] can be employed. However, these methods often struggle to find feasible trajectories without good initial guesses and careful parameter tuning. Therefore, we propose formulating the problem as a Markov Decision Process (MDP) and applying reinforcement learning (RL) to train a policy capable of reorienting randomly sized cuboids in various directions, as illustrated in Figure 1. Our formulation does not rely on explicit object state feedback during control, enabling offline trajectory planning in tens of milliseconds. The resulting reorientation trajectories are fast, typically lasting between 1–2 seconds.

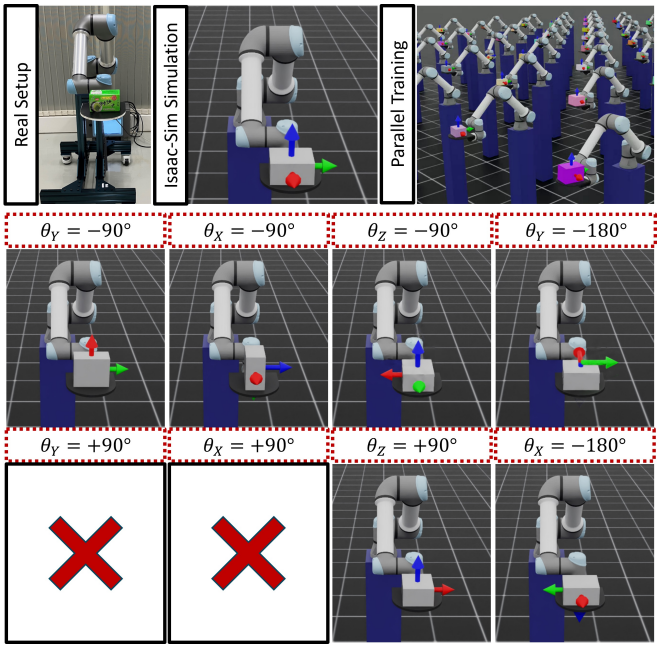

Fig. 1: Our system learns non-prehensile object reorientation via reinforcement learning. Through dynamic throwing and catching, our simulation-trained policy can zero-shot reorient various objects on real setup (see Fig. 5). Simulation and real videos are available at our project page: `https://tinyurl.com/DNPR2L`

This work introduces an MDP formulation with a carefully designed observation space, action space, and reward function for learning a state-based reorientation policy. We evaluate the policy's performance and sensitivity in simulation and successfully demonstrate real-world object reorientation across different objects, while discussing the limitations of sim-to-real transferability.

II. Related Works

Non-prehensile Reorientation is a specific application of non-prehensile manipulation [2], where an object is reoriented without achieving force-closure through an external gripper. Unlike two-fingered grippers, dexterous hands have enabled in-hand manipulation through reinforcement learning [3, 4, 5, 6], typically relying on vision feedback to carefully and slowly reorient small, graspable objects. Additionally, specially designed high degree-of-freedom (DOF) grippers have been developed to support certain modes of reorientation [7, 8]. To handle grasp-infeasible objects, non-prehensile

*Corresponding author, reachable at am-mustafa@aist.go.jp
[1]Abdullah Mustafa, Ryo Hanai, Ixchel Ramirez-Alpizar, Floris Erich, Ryoichi Nakajo, and Yukiyasu Domae are with the National Institute of Advanced Industrial Science and Technology (AIST), Japan
[2]Tetsuya Ogata is with the Graduate School of Fundamental Science and Engineering, Waseda University, Tokyo 169-8555, Japan and the National Institute of Advanced Industrial Science and Technology (AIST), Japan

approaches have been proposed, including planar pushing [9, 10, 11], bi-manual manipulation [12, 13], suction-based reorientation [14], and leveraging external surfaces [15]. For fast reorientation, dynamic non-prehensile strategies have been applied using simple one-DOF arms [1] and more complex platforms like the Barrett WAM 7-DOF robot [16], both demonstrating only 180° Y-axis reorientation of a single object. In contrast, our policy aims to generate instantaneous reorientation trajectories for randomly selected objects in various directions.

Reinforcement Learning (RL) has been successfully applied to a wide range of dynamic robotic tasks, including quadruped locomotion [17], dexterous manipulation [18], and agile flight [19]. By leveraging robust learning algorithms such as Proximal Policy Optimization (PPO) [20], high-fidelity vectorized simulators like IsaacSim [21], and scalable learning libraries such as RL-Games [22], RL can be extended to complex tasks where efficient learning occurs in simulation and policies are deployed zero-shot in the real world. While prior work has applied RL to non-prehensile pushing and reorientation, this work extends it to a dynamic formulation that enables faster planning and execution.

III. Method

This work addresses the problem of dynamic non-prehensile object reorientation through a learning-based solution. A state-based policy is trained via reinforcement learning (RL) by formulating the task as a Markov Decision Process (MDP). To overcome the sim-to-real transfer challenges, we perform system identification of the robot's dynamics and deploy the policy as an offline motion planner.

A. Markov Decision Process Formulation

To enable offline planning of reorientation trajectories, we carefully design the observation space, action space, and reward function. Once trained, the policy selects an optimal action $\mathcal{A}$ for a given state $\mathcal{S}$ to maximize the reward function $\mathcal{R}$. Stable learning further requires appropriate choices for the initial state distribution, termination conditions, and training hyper-parameters.

1) Initial State Distribution: Our policy was trained to re-orient randomized cuboids across different rotation directions. Table I lists the ranges used for uniformly sampling object dimensions (width, length, and height) and physical properties (mass, static/dynamic friction coefficients, and restitution coefficient). These ranges can be adjusted to accommodate specific operational setups, such as larger objects. The rotation direction is uniformly selected from six possible directions, as illustrated in Figure 1. While additional directions could be introduced, clockwise rotations around the X-axis (90° roll) and Y-axis (90° pitch) were found to be infeasible under the current robot dynamics, as the required forces and moments could not be generated or would cause self-collision. The robot configuration is initialized with a horizontal tray, as shown in Figure 1. To facilitate learning of X-axis rotation (-90° roll), the fifth robot joint, q_5, is randomized to achieve a perpendicular configuration, with the tray pointing along the Y-axis.

TABLE I: Sampling distributions for environment randomization

Property	Unit	Distribution
Object Width/Length/Height	cm	$\mathcal{U}(5, 15)$
Object Mass	kg	$\mathcal{U}(0.05, 0.5)$
Object/Tray Friction	-	$\mathcal{U}(0.1, 1.0)$
Object/Tray Restitution	-	$\mathcal{U}(0.0, 1.0)$
Target Rotation	-	$\mathcal{U}(range(6))$
Joint 5 Position	rad	$\mathcal{U}(0, \pi/2)$

2) Observation Space: The observation space is designed to ensure full observability, allowing the actor to estimate optimal actions for a given state. Table II lists the proposed observation spaces for both the actor and the critic. Since our policy is used for offline motion planning without access to the system model at inference time, the object state is not explicitly provided to the actor. Instead, the actor relies on the robot state (joint positions and velocities) to implicitly learn the system's kinematics and dynamics. Given knowledge of the object model—including its size and physical properties—and assuming a deterministic environment, the actor can implicitly infer the object state. The target rotation is specified using Euler angles (e.g., $\theta_{RPY} = [-\pi/2, 0, 0]$ for a -90° roll around the X-axis). In the absence of explicit object state information, the normalized relative step t (ranging from 0 to 1) is included to indicate the progression of the task. To support action regularization, the observation space also includes the previous action. Additional states are introduced to the critic to enhance value function estimation and stabilize learning, following an asymmetric actor-critic architecture. These additional states include the explicit end-effector and object poses and velocities.

TABLE II: Observation space for the actor and the critic

Input State	Symbol	Dim.	Actor	Critic
Relative Step	t	1	✔	✔
Last Action	a_{Last}	6	✔	✔
Target Euler	θ_{RPY}	3	✔	✔
Object Model				
Object Size	$[w, l, h]_O$	3	✔	✔
Object mass	m_O	1	✔	✔
Object/Tray Friction	$[\mu_s, \mu_k]$	2	✔	✔
Object Restitution	e	1	✔	✔
Robot State				
Joint Position	q	6	✔	✔
Joint Velocity	$\dot{q}$	6	✔	✔
End-Effector Position	P_{EE}	3	✗	✔
End-Effector Rot. Mat.	R_{EE}	6	✗	✔
End-Effector Velocity	v_{EE}	6	✗	✔
Object State				
Object Position	P_O	3	✗	✔
Object Rot. Mat,	R_O	6	✗	✔
Object Velocity	v_O	6	✗	✔
Total States			29	59

✔ and ✗ refers to state presence or absence, respectively.

3) Action Space: An acceleration-based action space was selected to generate smooth trajectories during control frequency upsampling. For dynamic tasks such as non-prehensile transportation, acceleration has also been used as a control input [23]. For UR5e control, the acceleration trajectory is integrated into position or velocity commands as shown in (1). We upsample the policy's 20 Hz output ($dt = 50$ms) to 120 Hz in simulation and 480 Hz on the

physical system by commanding a constant acceleration for 6 ($dt = \frac{50}{6}$ms) and 24 ($dt = \frac{50}{24}$ms) steps, respectively. Since we integrate (1) over a constant 50 ms period regardless of frequency, similar position and velocity profiles are attained in both simulation and real environments. To support our offline planning approach and mitigate the effects of signal noise and control latencies, only internal position and velocity estimates are used by the actor/critic networks, without relying on simulated or real ground-truth states. For the safety of the physical system, we limit the resulting jerk by clipping the change in acceleration to a maximum of 10 m/s^2.

$$q_t = q_{t-1} + \dot{q}_{t-1}dt + \frac{1}{2}\ddot{q}_t dt^2 \quad , \quad \dot{q}_t = \dot{q}_{t-1} + \ddot{q}_t dt \qquad (1)$$

4) Reward Formulation: To learn object reorientation, we designed a set of reward terms including a live reward r_L, dense rewards (r_{Ori}, r_{Rot}, r_p), a sparse reward r_{Succ}, and regularization rewards (r_q, $r_{\ddot{q}}$, r_a, $r_{\dot{a}}$), as listed in Table III. The total reward is computed as a weighted sum of these terms. These components were engineered through trial and error to achieve the desired policy behavior. Further ablations on reward design will be explored in future work.

Live Reward. To delay episode termination due to object drops, a constant survival reward is provided at every time step.

Object Orientation Reward. This dense reward encourages object reorientation toward a target rotation matrix $R_{O_{goal}}$, derived from a given goal Euler angle rotation θ_{RPY}. For the current object orientation R_{O_t}, matrix multiplication is equivalent to computing the cosine similarity (dot product) per axis. The reward r_{Ori} reaches its maximum value of 3 when the target orientation is achieved.

Object Rotation Reward. When a 180° rotation is required, an equal r_{Ori} is awarded for both clockwise and counterclockwise rotations around the target rotation axis $\vec{U}$. However, such multi-modal rewards were hypothesized to be insufficient for learning 180° reorientation. To address this, an additional reward r_{Rot} encourages negative (clockwise) rotation around the specified axis. A cross-product between R_{O_t} and $R_{O_{t-1}}$ yields a normal vector, whose dot product with $\vec{U}$ is proportional to the clockwise rotation angle around $\vec{U}$. For negative roll, pitch, or yaw rotations (i.e., -90° or -180°), $\vec{U}$ corresponds to the positive X, Y, or Z axis.

End-Effector/Object Relative Position Reward. The r_p reward minimizes the relative distance between the end-effector (EE) and the object in the XY plan allowing for necessary separation during throwing and maintaining close proximity for catching. Discarding this term results in out-of-reach throws during early training phases.

Sparse Success Reward. The success reward combines a binary success indicator α_{Succ} with a continuous reward kernel $\mathcal{K}(r_{Ori}, r_p)$. A trajectory is considered successful if both the object orientation and the relative end-effector/object position error ($d_p = ||p_{O_t} - p_{EE_t}||$) fall within predefined thresholds: $\alpha_{Succ} = (r_{Ori} > r_{Ori_{th}}) \wedge (d_p < d_{p_{th}})$. The thresholds $r_{Ori_{th}}$ and $d_{p_{th}}$ are set to 2.0 and 0.1, respectively. Higher rewards are achieved by refining policy behavior using a logistic kernel $\mathcal{K}$ applied to r_{Ori} and d_p, where $\mathcal{K}(x) = (e^{ax} + b + e^{-ax})^{-1}$, as described in [4].

Regularization Penalties. To produce smoother trajectories, limit aggressive actions, and reduce robot movement relative to its nominal configuration, regularization terms penalize excessive joint positions (r_q), accelerations ($r_{\ddot{q}}$), action commands (r_a), and action rates ($r_{\dot{a}}$) using an L2-norm penalty. The weights of these penalties are scaled proportionally to the mean success reward r_{Succ}, allowing regularization to be relaxed during early training phases.

TABLE III: Proposed reward formulation and corresponding weights.

Reward Term	Symbol	Formulation	Weight
Live	r_L	-	1.0
Object Ori.	r_{Ori}	$R_{O_t} R_{O_{goal}}^T$	1.0
Object Rot.	r_{+Rot}	$(R_{O_t} \times R_{O_{t-1}}) \cdot \vec{U}$	1.0
Object Rel. Pos.	r_p	$-\|\|p_{O_t} - p_{EE_t}\|\|$	1.0
Success	r_{Succ}	$\alpha_{Succ}\mathcal{K}(3 - r_{Ori})\mathcal{K}(r_p)$	50
Joint Pos.	r_q	$-\|\|q_t - q_0\|\|$	1.0
Joint Acc. Reg.	$r_{\ddot{q}}$	$-\|\|\ddot{q}_t\|\|$	0.1
Action Reg.	r_a	$-\|\|a_t\|\|$	1.0
Action Rate Reg.	$r_{\dot{a}}$	$-\|\|a_t - a_{t-1}\|\|$	1.0

5) Episode Termination: An episode is terminated upon timeout (reaching the maximum length of 100 steps, equivalent to 5 seconds). To stabilize training and avoid collecting irrelevant transitions, early termination is triggered in cases of extreme separation—such as object drops or excessive throws—where the relative elevation between the end-effector and the object exceeds a 1-meter threshold.

6) Implementation Details: The policy was trained using NVIDIA IsaacLab [24] (formerly ORBIT), a wrapper for NVIDIA Isaac-Sim that simplifies parallelized robot learning. A Universal Robots UR5e arm with a tray-like end-effector was used; however, extending to systems with more advanced kinematics and dynamics (e.g., additional degrees of freedom) could improve reorientation performance, particularly for positive roll and pitch motions. The policy was trained using the RL-Games [22] implementation of Proximal Policy Optimization (PPO [20]). Leveraging Isaac-Sim and RL-Games vectorized environments, data collection and training were conducted using 4096 parallel environments on a single NVIDIA RTX 4090 GPU. Training the policy for 1000 epochs (250 million steps) took approximately 50 minutes.

B. Sim-to-Real Transfer

To transfer simulation-trained policies to a real setup, system identification of both the robot and object dynamics is essential. Additionally. to mitigate potential control latencies and sensor noise, we deploy the policy as an offline motion planner.

1) System Identification: To minimize the gap between simulated and real robot dynamics, we execute a set of predefined sinusoidal trajectories on the real robot and identify a set of gains (stiffness and damping) that closely match the position and velocity trajectories. This matching is achieved by minimizing the L2-norm error across thousands of simulated environments with randomized gain parameters.

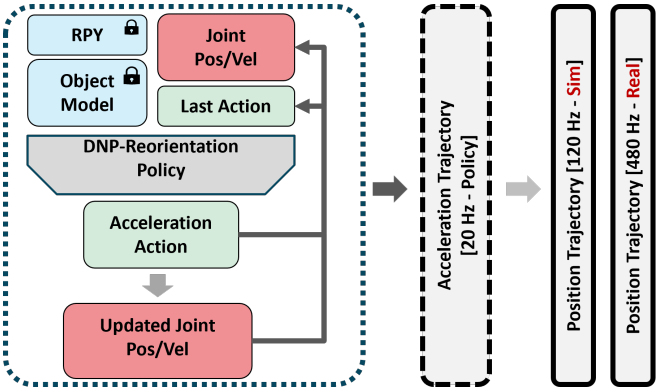

Fig. 2: Offline motion planning for the reorientation of a given object model toward a target rotation direction. The acceleration trajectories generated by the policy are upsampled for simulation and real-world deployment.

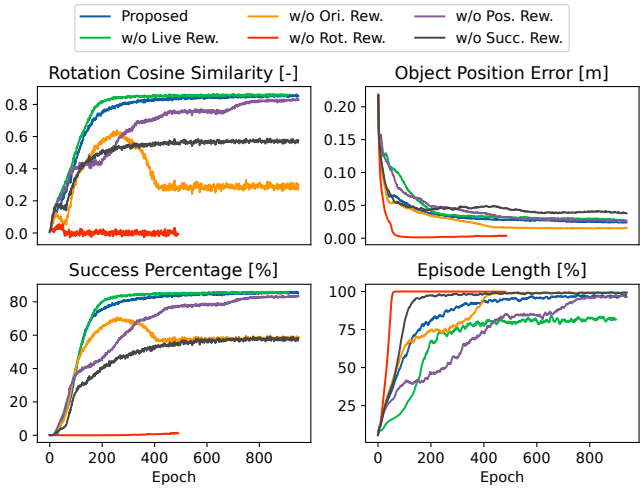

Fig. 3: The learning curve demonstrates smooth and stable progress of proposed formulation, with the target orientation reached and minimal object displacement, resulting in successful reorientation.

These gains are refined through iterative sampling and optimization of new populations.

For object handling, our policy uses an explicit model of the object to generate reorientation trajectories. However, due to discrepancies in contact, air resistance, and inertia modeling, simulated behavior often diverges from real-world performance. While manual tuning currently enables successful reorientation, incorporating sensory feedback for online system identification and control is expected to improve policy performance and reduce reliance on manual tuning.

2) Offline Motion Planning: Given an object model (size, mass, friction, and restitution) and a target rotation direction, the acceleration trajectory is generated offline by rolling out the policy and recursively updating joint position and velocity estimates, as illustrated in Fig. 2. The tuned actuator stiffness and damping in simulation, along with the "servoJ" command parameters on the UR5e in the real setup, ensure that the estimated joint position and velocity trajectories closely approximate ground-truth behavior. As a result, offline-generated trajectories can be reliably executed in both simulation and the physical system.

IV. Experiments

In this work, we evaluate our reorientation policy in both simulated and real-world setups, addressing the following key questions:

1) How effective is the proposed MDP formulation for learning a reorientation policy?
2) How sensitive is the policy to uncertainties in object modeling?
3) Can the policy be successfully transferred from simulation to the real-world setup?

A. Learning Curves

Figure 3 presents the learning curves across multiple metrics for both the proposed and ablated reward formulations. Under the proposed formulation, the policy gradually learns to reorient objects, as evidenced by increasing rotation cosine similarity and decreasing relative object position error. The success rate is averaged across all asynchronous environments (4096). At convergence, the success rate exceeds 80%, indicating that successful reorientation is typically achieved within 20% of the trajectory length—under one second. Additionally, the policy converges to a 100% episode length, reflecting nearly zero early terminations due to object drops.

In contrast, the ablated reward designs exhibit suboptimal learning behavior. Removing the live reward led to frequent object drops and shorter episode lengths, as this failure mode was not explicitly penalized. Omitting orientation (R_{Ori}), rotation (r_{Rot}), or success (r_{Succ}) rewards resulted in poor reorientation behavior and lower success rates. When the relative position error penalty (r_p) was dropped, the policy slowly converged to good performance but required significantly more exploration.

B. Sensitivity to Object Modeling

While simulation-based evaluation provides access to ground-truth object states and eliminates the dynamics gap, policy sensitivity was evaluated by varying the object model inputs from the ground-truth configuration. This mismatch causes the offline-generated trajectories to induce unintended behaviors, highlighting the challenges of sim-to-real transfer under modeling uncertainty.

Figure 4 illustrates the policy's sensitivity to object model uncertainty. Regarding object mass, the policy demonstrates robustness to moderate variations, with only a slight performance drop for relatively large masses. Although induced changes in inertial and frictional forces can be tolerated to some extent, higher mass uncertainty would result in failure due to excessively small/large forces. While not depicted in Figure 4, variations in the coefficient of restitution had unexpectedly negligible impact on policy performance. Thus, the contact modeling in the current simulator (IsaacSim/IsaacLab) requires further examination.

For friction (assumed equal for both static and dynamic), only limited uncertainty was tolerable. Higher uncertainty led to a drastic drop in performance. Accurate contact modeling is critical for both throwing and catching phases. Assuming higher friction values resulted in more aggressive

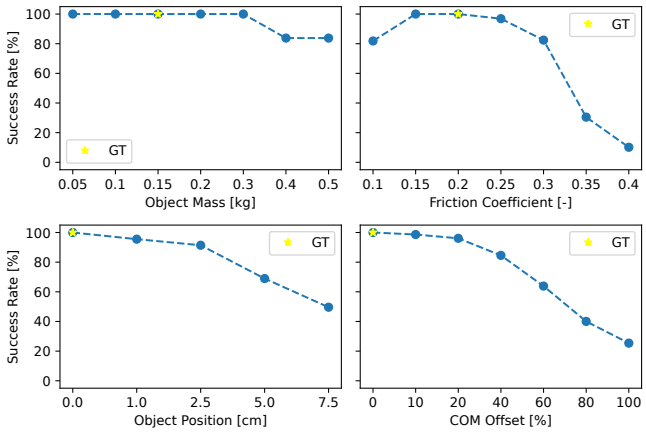

Fig. 4: The reorientation policy is sensitive to modeling errors, with success rates degrading due to inaccuracies in mass or friction estimation. Poor object placement and shifts in the center of mass can induce unintended motion. Ground-truth model is indicated by a yellow star.

trajectories to overcome static friction (break contact), which caused excessive sliding when the actual friction was lower. Conversely, assuming lower friction produced conservative trajectories that failed to induce necessary sliding.

During training, the object is assumed to be centered on the tray, but manual placement in real-world setups does not guarantee perfect centering. Even small deviations from the center can significantly affect in-flight trajectories due to altered force and moment distributions. As the object deviates further from the expected center, success rates decline. To improve robustness to placement uncertainty, future training procedures will incorporate randomly placed objects, and the real system will use visual pose estimation to determine the initial object pose.

Finally, manipulated objects may not have their center of mass (COM) aligned with their geometric centroid. For example, a partially filled box may experience COM shifts during motion. Although the current policy is trained assuming a centroidal COM, we evaluate its performance under COM uncertainty. Moderate deviations are tolerable, but excessive COM shifts can significantly degrade performance. Future policies will be trained with randomized COM configurations to enhance robustness. Moreover, integrating closed-loop visual feedback can further improve performance under inertial uncertainty.

C. Real-World Evaluation

For real-world evaluation, four objects with six configurations were selected and reoriented in six different directions, as shown in Table IV. Figure 5 visualizes samples of successful trajectories. Starting with the "Choco-box" object, we manually tuned the friction and restitution coefficients used by the policy, resulting in a high success rate. These contact parameters were then fixed for the remaining objects ($\mu_s = \mu_k = e = 0.25$). Overall, the sim-to-real transferability of the current reorientation policy remains limited due to the dynamics gap and modeling inaccuracies.

Analysis of average real-world performance in Table IV reveals that 90° rotations around the X and Y axes achieved higher success rates compared to 180° rotations, which required higher throws. This suggests that a more robust policy should achieve 180° rotations through two consecutive 90° rotations. For the Z-axis, clockwise and counter-clockwise rotations were achieved through distinct behaviors. However, counter-clockwise rotations (-90°) had lower success rates, as they relied on two consecutive rotational motions, making them more susceptible to compounding errors.

In the real-world setup, common failure modes included sliding/sticking, rolling/bouncing, and in-flight trajectory mismatches. As discussed in Fig. 4, poor contact modeling often led to unintended object sticking or sliding on the tray. Mismatches in restitution behavior caused objects to bounce or roll during the catching phase, particularly for high throws associated with 180° rotation targets. Inaccurate inertia modeling resulted in sim-to-real mismatches of in-flight motion, affecting the anticipated landing pose. These failure modes highlight the importance of online system identification and closed-loop control for improving real-world performance.

TABLE IV: Real-world quantitative evaluation of the reorientation policy is conducted across various object and direction combinations. Each object is reoriented 10 times per combination, and success is approximately estimated.

	Target Reorientation					
	θ_Y		θ_X		θ_Z	
Object	-90°	-180°	-90°	-180°	-90°	+90°
Choco-Box	7	10	10	10	10	10
Choco-Box-R	10	2	10	4	10	10
Tissue-Box	7	10	6	9	4	9
Wood-Blocks	6	0	8	0	1	3
Tea-Box [40g]	10	5	1	-	-	-
Tea-Box [200g]	8	2	10	-	-	-
Average [%]	80	48	75	58	63	80

V. Conclusion

In this work, we demonstrated a dynamic, non-prehensile reorientation policy trained via reinforcement learning. The offline-generated trajectories were successfully deployed in both simulated and real-world environments. However, the current performance exhibits limited sim-to-real transferability, which we aim to improve through several strategies. First, visual feedback can be used to approximate object properties such as friction by executing predefined trajectories (e.g., forward-backward motions to induce sliding). Second, a visuomotor policy incorporating closed-loop pose estimation and inertia randomization could enhance catching behavior under modeling uncertainty. Third, penalizing aggressive throws and high-impact catches may improve transferability. Finally, more accurate contact modeling in simulation could help bridge the existing dynamics gap.

ACKNOWLEDGEMENT

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

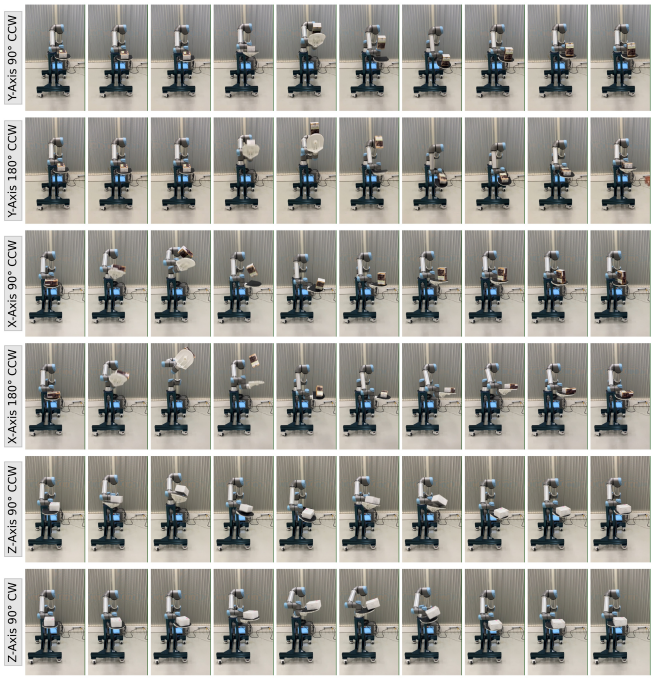

Fig. 5: Real-world reorientation trajectories for different objects in different directions (see `https://tinyurl.com/DNPR2L`).

REFERENCES

[1] K. Lynch and M. Mason, "Dynamic underactuated nonprehensile manipulation," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, vol. 2, 1996, pp. 889–896 vol.2.

[2] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.

[3] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020. [Online]. Available: https://doi.org/10.1177/0278364919887447

[4] A. Allshire, M. MittaI, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2022. [Online]. Available: http://dx.doi.org/10.1109/IROS47612.2022.9981458

[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] A. Patel and S. Song, "GET-Zero: Graph embodiment transformer for zero-shot embodiment generalization," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[7] W. G. Bircher, A. M. Dollar, and N. Rojas, "A two-fingered robot gripper with large object reorientation range," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3453–3460.

[8] W. G. Bircher, A. S. Morgan, and A. M. Dollar, "Complex manipulation with a simple robotic hand through contact breaking and caging," *Science Robotics*, vol. 6, no. 54, p. eabd2666, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abd2666

[9] J. Del Aguila Ferrandis, J. Moura, and S. Vijayakumar, "Nonprehensile planar manipulation through reinforcement learning with multimodal categorical exploration," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 5606–5613.

[10] M. Oller, D. Berenson, and N. Fazeli, "Tactile-Driven Non-Prehensile Object Manipulation via Extrinsic Contact Mode Control," in *Proceedings of Robotics: Science and Systems (RSS)*, Delft, Netherlands, July 2024.

[11] M. Kim, J. Han, J. Kim, and B. Kim, "Pre-and post-contact policy decomposition for non-prehensile manipulation with zero-shot sim-to-real transfer," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 10 644–10 651.

[12] L. Yang, H. J. T. Suh, T. Zhao, B. P. Graesdal, T. Kelestemur, J. Wang, T. Pang, and R. Tedrake, "Physics-driven data generation for contact-rich manipulation via trajectory optimization," in *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[13] L. Yan, T. Stouraitis, J. Moura, W. Xu, M. Gienger, and S. Vijayakumar, "Impact-aware bimanual catching of large-momentum objects," *IEEE Transactions on Robotics*, vol. 40, pp. 2543–2563, 2024.

[14] K. Wada, S. James, and A. J. Davison, "Reorientbot: Learning object reorientation for specific-posed placement," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8252–8258.

[15] Y. Wang, Y. Li, Y. Yang, and Y. Chen, "Dexterous non-prehensile manipulation for ungraspable object via extrinsic dexterity," 2025. [Online]. Available: https://arxiv.org/abs/2503.23120

[16] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3232–3237.

[17] E. Chane-Sane, J. Amigo, T. Flayols, L. Righetti, and N. Mansard, "Soloparkour: Constrained reinforcement learning for visual locomotion from privileged experience," in *Conference on Robot Learning (CoRL)*, 2024.

[18] A. Petrenko, A. Allshire, G. State, A. Handa, and V. Makoviychuk, "Dexpbt: Scaling up dexterous manipulation for hand-arm systems with population based training," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[19] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, "Demonstrating agile flight from pixels without state estimation," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024. [Online]. Available: https://arxiv.org/abs/2406.12505

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[21] NVIDIA, "Isaacgym," 2020. [Online]. Available: https://developer.nvidia.com/isaac-gym

[22] D. Makoviichuk and V. Makoviychuk, "Rl games," 2021. [Online]. Available: https://github.com/Denys88/rl_games/

[23] A. Heins and A. P. Schoellig, "Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 7986–7993, 2023.

[24] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.