Real-to-Sim Robot Policy Evaluation with Gaussian Splatting Simulation of Soft-Body Interactions

Kaifeng Zhang[1,2*], Shuo Sha[1,2*], Hanxiao Jiang[1], Matthew Loper[2], Hyunjong Song[2], Guangyan Cai[2], Zhuo Xu[3], Xiaochen Hu[2], Changxi Zheng[1,2], Yunzhu Li[1,2]

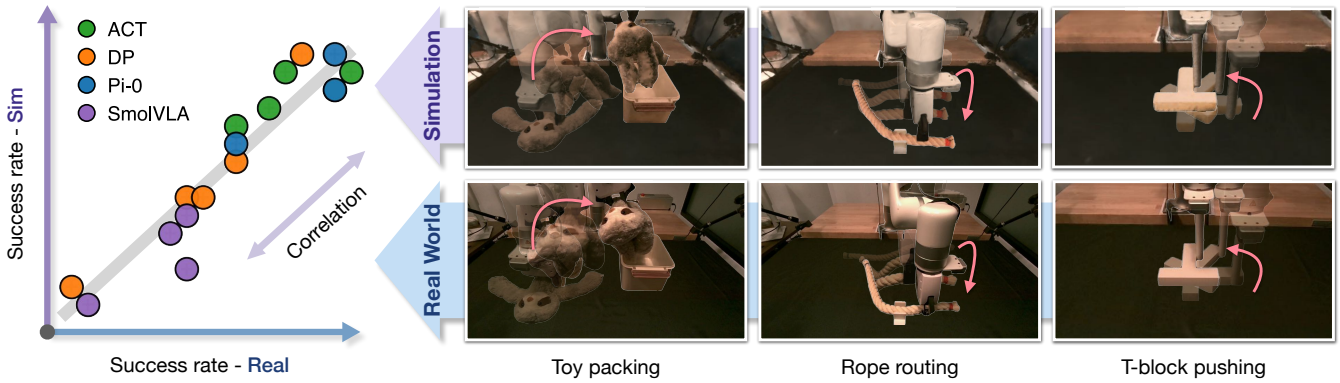


Fig. 1: **Real-to-sim policy evaluation with Gaussian Splatting simulation.** *Left:* Correlation between simulated and real-world success rates across multiple policies (ACT [1], DP [2], Pi-0 [3], SmolVLA [4]) shows that our simulation reliably predicts real-world performance. *Right:* Representative tasks used for evaluation, including plush toy packing, rope routing, and T-block pushing, are visualized in both real and simulated settings. Our framework reconstructs soft-body digital twins from real-world videos and achieves realistic appearance and motion, enabling scalable and reproducible policy assessment.

***Abstract*— Robotic manipulation policies are advancing rapidly, but their direct evaluation in the real world remains costly, time-consuming, and difficult to reproduce, particularly for tasks involving deformable objects. Simulation provides a scalable and systematic alternative, yet existing simulators often fail to capture the coupled visual and physical complexity of soft-body interactions. We present a real-to-sim policy evaluation framework that constructs soft-body digital twins from real-world videos and renders robots, objects, and environments with photorealistic fidelity using 3D Gaussian Splatting. We validate our approach on representative deformable manipulation tasks, including plush toy packing, rope routing, and T-block pushing, demonstrating that simulated rollouts correlate strongly with real-world execution performance and reveal key behavioral patterns of learned policies. Our results suggest that combining physics-informed reconstruction with high-quality rendering enables reproducible, scalable, and accurate evaluation of robotic manipulation policies. Website: `https://real2sim-eval.github.io/`**

I. Introduction

Robotic manipulation policies have advanced rapidly across a wide range of tasks [1, 2, 5–7]. However, their evaluation still relies heavily on real-world trials, which are slow, expensive, and difficult to reproduce. As the community shifts toward training foundation models for robotics [3, 8–12], whose development depends on rapid iteration and large-scale benchmarking, this reliance has become a significant bottleneck.

[1]Columbia University [2]SceniX Inc. [3]Google DeepMind
* Equal contribution. Work partially done while interning at SceniX Inc.

Simulation offers a scalable and systematic alternative and is widely used for data generation and training [13–18]. Yet it is far less common as a tool for policy evaluation, primarily due to poor sim-to-real correlation: a policy that performs well in simulation often fails to translate to similar real-world success. Narrowing this gap would allow simulation to serve as a trustworthy proxy for real-world testing, greatly accelerating development cycles. This raises the central question: *how can we design simulators that are sufficiently realistic to evaluate robot policies with confidence?* To answer this question, we propose a framework for building high-fidelity simulators and investigate whether they can predict real-world policy performance reliably.

We identify two key factors for aligning simulation with reality: *appearance* and *dynamics*. On the appearance side, rendered scenes must closely match real-world observations. This is particularly challenging for policies that rely on wrist-mounted cameras, where simple green-screen compositing [19] is insufficient. We address this by leveraging 3D Gaussian Splatting (3DGS) [20], which reconstructs photorealistic scenes from a single scan and supports rendering from arbitrary viewpoints. Beyond prior uses of 3DGS for simulation [21–24], we enhance it with automatic position and color alignment and object deformation handling, which are essential for closing the appearance gap.

Dynamics present another major source of sim-to-real discrepancy. Traditional simulators rely on low-dimensional parameter tuning, which is insufficient for deformable objects with many degrees of freedom. To address this challenge,

we adopt PhysTwin [25], a framework that reconstructs deformable objects as dense spring-mass systems optimized directly from object interaction videos. This approach yields efficient system identification while closely matching real-world dynamics.

We integrate these appearance and dynamics components into a unified simulator and expose it through a Gym-style interface [26]. We evaluate this framework on representative rigid- and soft-body manipulation tasks, including plush toy packing, rope routing, and T-block pushing, using widely adopted imitation learning algorithms: ACT [1], Diffusion Policy (DP) [2], SmolVLA [4], and Pi-0 [3]. By comparing simulated and real-world success rates and performing ablation studies, we observe a strong correlation and confirm that rendering and dynamics fidelity are both crucial to the trustworthiness of simulation-based evaluation.

In summary, our main contributions are: (1) A complete framework for evaluating robot policies in a Gaussian Splatting-based simulator using soft-body digital twins. (2) Empirical evidence that simulated rollouts strongly correlate with real-world success rates across representative tasks, using policies trained *exclusively* on real-world data (no co-training). (3) A detailed analysis of design choices that improve the reliability of simulation as a predictor of real-world performance, offering guidance for future simulation-based evaluation pipelines.

II. Related Works

A. Robot Policy Evaluation

Evaluating robot policies is essential for understanding and comparing policy behaviors. Most systems are still evaluated directly in the real world [11, 27–30], but such evaluations are costly, time-consuming, and usually tailored to specific tasks, embodiments, and sensor setups. To enable more systematic study, prior works have introduced benchmarks, either in the real world through standardized hardware setups [31–35] or in simulation through curated assets and task suites [16, 33, 36–44]. Real-world benchmarks offer high fidelity but lack flexibility and scalability, while simulators often suffer from unrealistic dynamics and rendering, which limits their reliability as proxies for physical experiments. This is widely referred to as the "sim-to-real gap" [45–48]. We aim to narrow this gap by building a realistic simulator that combines high-quality rendering with faithful soft-body dynamics. Compared to SIMPLER [19], which relies on green-screen compositing, and Real-is-sim [21], which focuses on rigid-body simulation, our method integrates Gaussian Splatting-based rendering with soft-body digital twins derived from interaction videos, eliminating the dependence on static cameras and providing more realistic appearance and dynamics.

B. Physical Digital Twins

Digital twins seek to realistically reconstruct and simulate real-world objects. Many existing frameworks rely on pre-specified physical parameters [49–53], which limits their ability to capture complex real-world dynamics or leverage data from human interaction. While rigid-body twins are well studied [54–57], full-order parameter identification for deformable objects remains challenging. Learning-based approaches have been proposed to capture such dynamics [58–61], but they often sacrifice physical consistency, which is critical for evaluating manipulation policies in contact-rich settings. Physics-based methods that optimize physical parameters from video observations [62–65] offer a more promising path. Among them, PhysTwin [25] reconstructs deformable objects as dense spring-mass systems directly from human-object interaction videos, achieving state-of-the-art realism and efficiency. Our work builds on PhysTwin and integrates its reconstructions with a Gaussian Splatting simulator to bridge the dynamics gap in policy evaluation.

C. Gaussian Splatting Simulators

Building simulators that closely match the real world requires high-quality rendering and accurate physics. Gaussian Splatting (3DGS) [20] has recently emerged as a powerful approach for scene reconstruction, enabling photorealistic, real-time rendering from arbitrary viewpoints [51, 56]. Several studies have demonstrated its potential in robotics, showing that 3DGS-based rendering can improve sim-to-real transfer for vision-based policies [22, 66, 67], augment training datasets [23, 24, 68, 69], and enable real-to-sim evaluation [21, 70]. We extend this line of work by supporting soft-body interactions, incorporating PhysTwin [25] for realistic dynamics, and introducing automated position and color alignment, resulting in a complete and evaluation-ready simulator.

III. Method

A. Problem Definition

We study the policy evaluation problem: *Can a simulator reliably predict the real-world performance of visuomotor policies trained with real data?* In a typical evaluation pipeline [11, 71], multiple policies are executed across controlled initial configurations in both simulation and the real world, and performance is measured through rollout-based metrics, typically expressed as scalar scores $u \in [0,1]$. The objective is to establish a strong correlation between simulated and real-world outcomes, represented by the paired set $\{(u_{i,\text{sim}}, u_{i,\text{real}})\}_{i=1}^{N}$, where $u_{i,\text{sim}}$ and $u_{i,\text{real}}$ denote the performance of the i-th policy in simulation and reality, respectively, and N is the number of evaluated policies.

To achieve better performance correlation, one promising way is to build a simulator that yields consistent results with the real world. Formally, let $\{(s_t, o_t, a_t)\}_{t=1}^{T}$ denote the sequence of environment states s_t, robot observations o_t, and robot actions a_t over a time horizon T. A simulator for policy evaluation should contain two core components: (1) *Dynamics model:* $s_{t+1} = f(s_t, a_t)$, which predicts future states given the current state and robot actions. (2) *Appearance model:* $o_t = g(s_t)$, which renders observations in the input modality required by the policy (e.g., RGB images). Accordingly, the fidelity of simulation can be assessed along

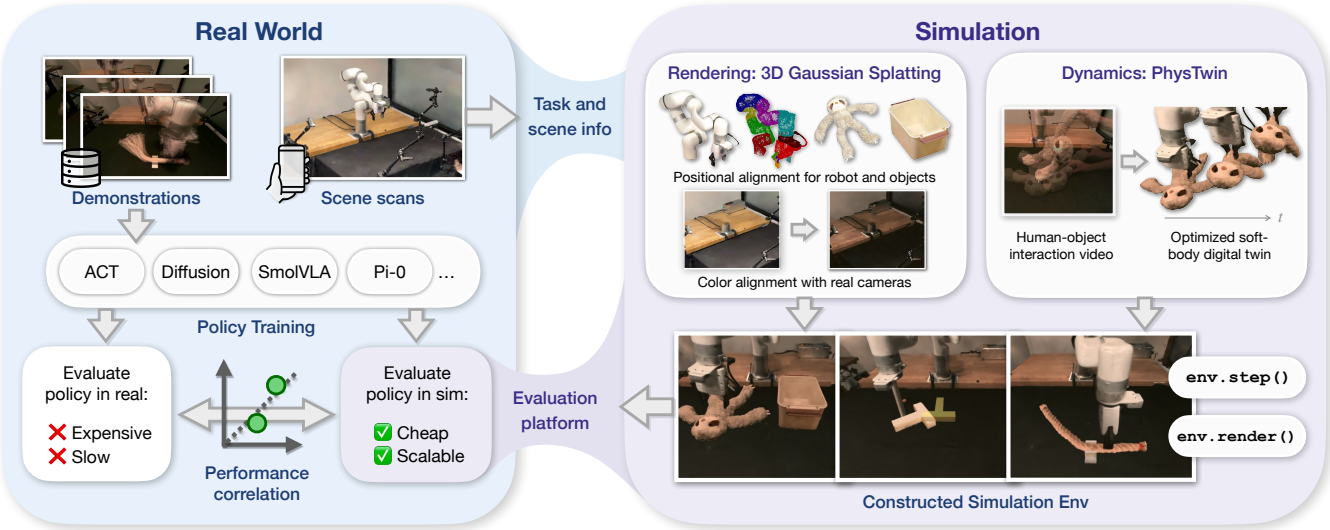


Fig. 2: **Proposed framework for real-to-sim policy evaluation.** We present a pipeline that evaluates real-world robot policies in simulation using Gaussian Splatting-based rendering and soft-body digital twins. Policies are first trained on demonstrations collected by the real robot, and a phone scan of the workspace is used to reconstruct the scene via Gaussian Splatting. The reconstruction is segmented into robot, objects, and background, then aligned in position and color to enable photorealistic rendering. For dynamics, we optimize soft-body digital twins from object interaction videos to accurately reproduce real-world behavior. The resulting simulation is exposed through a Gym-style API [26], allowing trained policies to be evaluated efficiently. Compared with real-world trials, this simulator is cheaper, reproducible, and scalable, while maintaining strong correlation with real-world performance.

two axes: (i) the accuracy of simulated dynamics, and (ii) the realism of rendered observations.

In this work, we address both axes by jointly reducing the visual gap and the dynamics gap. We employ physics-informed reconstruction of soft-body digital twins to align simulated dynamics with real-world object behavior, and use high-resolution Gaussian Splatting as the rendering engine to generate photorealistic observations. The following sections describe these components in detail, and an overview of the full framework is shown in Figure 2.

B. Preliminary: PhysTwin

We adopt the PhysTwin [25] digital twin framework, which reconstructs and simulates deformable and rigid objects from video using a dense spring-mass system. Each object is represented as a set of mass nodes connected by springs, with springs formed between each pair of nodes within a distance threshold d. The node positions evolve according to Newtonian dynamics.

To capture the behavior of diverse real-world deformable objects with varying stiffness, friction, and other material properties, PhysTwin employs a real-to-sim pipeline that jointly optimizes a set of physical parameters, including the spring threshold d and per-spring stiffness coefficients Y. The optimization is performed from a single video of a human interacting with the object by hand: human hand keypoints are tracked and attached to the spring-mass system as kinematic control points, and system parameters are tuned to minimize the discrepancy between tracked object motions in the video and their simulated counterparts. For rigid bodies, Y is fixed to a large value to suppress deformation. We adopt this same real-to-sim process for system identification of the objects that interact with the robot (plush toy, rope, and T-block).

C. Real-to-Sim Gaussian Splatting Simulation

We now describe the construction of our Gaussian Splatting-based simulator. Our approach addresses two complementary goals: (i) closing the visual gap through GS scene reconstruction, positional alignment, and color alignment, and (ii) closing the dynamics gap through physics-based modeling and deformation handling.

1) GS Construction: We begin by acquiring the appearance of each object of interest using Scaniverse [72], an iPhone app that automatically generates GS reconstructions from video recordings. In a tabletop manipulation scene, we first scan the static robot workspace, including the robot, table, and background, then scan each experimental object individually. The resulting reconstructions are segmented into robot, objects, and background using the SuperSplat [73] interactive visualizer. This reconstruction step is required only once per task.

2) Positional Alignment: After obtaining GS reconstructions of the static background, robot, PhysTwin object, and other static objects, we align all components to the reference frames: the robot base frame and canonical object frames. PhysTwin objects and static meshes are aligned to their corresponding PhysTwin particle sets and object 3D models by applying a relative 6-DoF transformation. For the robot, we automatically compute the transformation between the reconstructed GS model and ground truth robot points (generated from its URDF) using a combination of Iterative Closest Point (ICP) [74] and RANSAC [75]. We use 2,000 points per link to ensure sufficient coverage of link geometry. Because the background GS is in the same frame as the robot GS, we apply the same transformation estimated by ICP.

To enable the simulation of the static robot GS, we associate each Gaussian kernel with its corresponding robot link

through a link segmentation process. After ICP alignment, each kernel is assigned to a link by finding its nearest neighbor in the sampled robot point cloud and inheriting that point's link index. This process is applied to all links, including the gripper links, allowing us to render continuous arm motion as well as gripper opening and closing. The same procedure generalizes naturally to other robot embodiments with available URDF models.

3) Color Alignment: A major contributor to the visual gap in GS renderings is that reconstructed scenes often lie in a different color space from the policy's training data, leading to mismatched pixel color distributions, which can affect policy performance. In our setting, GS reconstructions inherit the color characteristics of iPhone video captures, while policies are trained in the color space of the robot's cameras (e.g., Intel RealSense, which is known to introduce color shifts). To close this gap, we design a color transformation that aligns GS colors to the real camera domain.

We perform this alignment directly in RGB space. First, we render images from the scene GS at the viewpoints of the fixed real cameras, using the original Gaussian kernel colors and opacities. Next, we capture real images from the same viewpoints, forming paired data for optimization. We then solve for a transformation function f that minimizes the pixel-wise color discrepancy:

$$f^* = \arg\min_{f \in \mathscr{F}} \frac{1}{N} \sum_{i=1}^{N} \|f(p_i) - q_i\|_2, \quad p_i \in I_{GS}, \quad q_i \in I_{RS}, \quad (1)$$

where I_{GS} and I_{RS} denote GS renderings and real camera captures, N is the number of pixels, p_i and q_i are corresponding RGB values, and $\mathscr{F}$ is the function space. We parameterize $\mathscr{F}$ as the set of degree-d polynomial transformations:

$$f = \{f_i\}_{i=1}^{d}, \quad f_i \in \mathbb{R}^3, \quad (2)$$

$$f(p_i) = [f_0 \ f_1 \ \cdots \ f_d] \cdot [1 \ p_i \ \cdots \ p_i^d]^T, \quad (3)$$

which reduces the problem to a standard least-squares regression. We solve it using Iteratively Reweighted Least Squares (IRLS) [76] to improve robustness to outliers. Empirically, we find that a quadratic transform ($d = 2$) offers the best trade-off between expressivity and overfitting.

4) Physics and Deformation: With GS reconstruction and alignment mitigating the rendering gap, the physics model must accurately capture real-world dynamics. We use a custom physics engine built on NVIDIA Warp [77], extending the PhysTwin [25] spring-mass simulator to support collisions with both robot end-effectors and objects in the environment. For grasping soft-body digital twins, we avoid the common but unrealistic practice of fixing object nodes to the gripper. Instead, we model contact purely through frictional interactions between gripper fingers and the object. The gripper closing motion halts automatically once a specified total collision-force threshold is reached, yielding more realistic and stable grasps.

At each simulation step, the updated robot and environment states from the physics engine are propagated to the Gaussian kernels. For rigid bodies, including objects and robot links, kernel positions and orientations are updated using the corresponding rigid-body transformations. For deformable objects, following PhysTwin [25], we apply Linear Blend Skinning (LBS) [78] to transform each kernel based on the underlying soft-body deformation.

Overall, with GS rendering, the physics solver, and LBS-based deformation being the major computational steps, our simulator runs at 5 to 30 FPS on a single GPU, depending on the robot-object contact states. By eliminating the overhead of real-world environment resets and leveraging multi-GPU parallelization, we empirically achieve evaluation speeds several times faster than real-world execution.

D. Policy Evaluation

To evaluate visuomotor policies in our simulator, we first design tasks and perform real-world data collection and policy training. Demonstrations are collected through human teleoperation using GELLO [79], after which we scan the scene to construct the corresponding simulation environments. All policies are trained *exclusively* on real data (i.e., no co-training between simulation and reality). To improve consistency and reduce variance, we follow the practice of Kress-Gazit et al. [71] by defining a fixed set of initial object configurations for each task and performing evaluations in both simulation and the real world. In the real world, we use a real-time visualization tool that overlays simulated initial states onto live camera streams, enabling operators to accurately and consistently reproduce the starting configurations.

Policy performance u is measured in terms of binary task success rates: in the real world, success is determined by human evaluators, while in simulation, task-specific criteria are automatically computed from privileged simulation states. In this work, we evaluate the performance of several state-of-the-art imitation learning algorithms, as well as checkpoints from different training stages for each network. Notably, the simulator is readily extensible to other policy types, as we package the entire system into the widely adopted Gym environment API [26]. We are committed to open-sourcing our implementation to encourage community adoption and enable scalable, reproducible policy evaluation.

IV. Experiments

In this section, we test the performance of imitation learning policies in both the real world and our simulation environment to examine the correlation. We aim to address the following questions: (1) How strongly do the simulation and real-world performance correlate? (2) How critical are rendering and dynamics fidelity for improving this correlation? (3) What practical benefits can the correlation provide?

A. Experiment Setup

1) Tasks: We evaluate policies on three representative manipulation tasks involving both deformable and rigid objects:

- *Toy packing*: The robot picks up a plush sloth toy from the table and packs it into a small plastic box. A trial is considered successful only if the toy's arms, legs, and

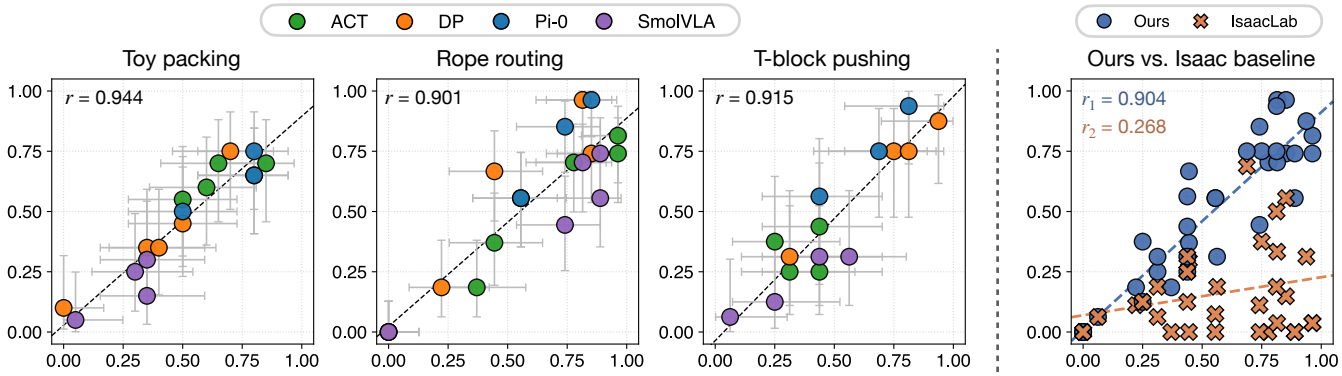


Fig. 3: **Correlation between simulation and real-world policy performance.** *Left:* Simulation success rates (y-axis) vs. real-world success rates (x-axis) for toy packing, rope routing, and T-block pushing, across multiple state-of-the-art imitation learning policies and checkpoints. The tight clustering along the diagonal indicates that, even with binary success metrics, our simulator faithfully reproduces real-world behaviors across tasks and policy robustness levels. *Right:* Compared with IsaacLab, which models rope routing and push-T tasks, our approach yields substantially stronger sim-to-real correlation, highlighting the benefit of realistic rendering and dynamics.

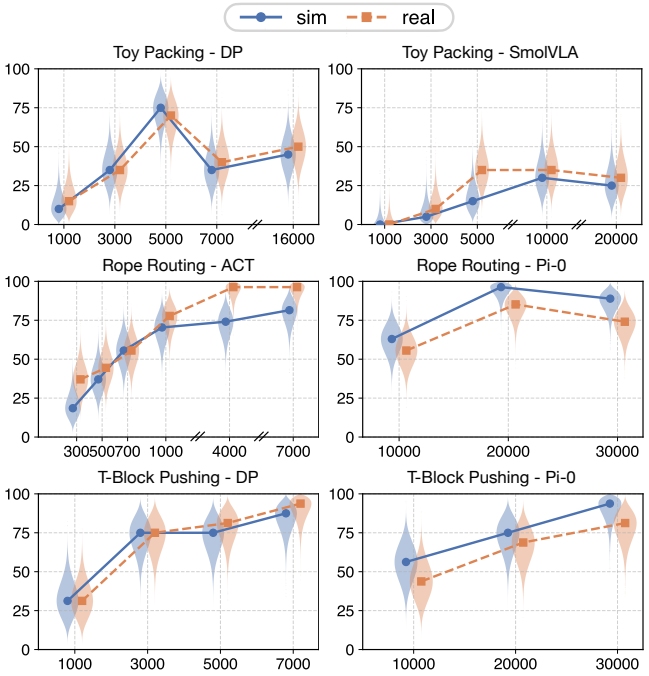


Fig. 4: **Per-policy, per-task performance across training.** x-axis: training iterations, y-axis: success rates. Simulation (blue) and real-world (orange) success rates are shown across iterations. Unlike Figure 3, which aggregates across policies, this figure shows unrolled curves for each task-policy pair. Improvements in simulation consistently correspond to improvements in the real world, establishing a positive correlation and demonstrating that our simulator can be a reliable tool for evaluating/selecting policies.

body are fully contained within the box, with no parts protruding.

- *Rope routing*: The robot grasps a cotton rope, lifts it, and routes it through a 3D-printed clip. Success is defined by the rope being fully threaded into the clip.
- *T-block pushing (push-T)*: A 3D-printed T-shaped block is placed on the table. Using a vertical cylindrical pusher, the robot must contact the block and then translate and reorient it to match a specified target pose.

Both the toy packing and rope routing tasks are challenging because the small tolerances of the box and clip require the policy to leverage visual feedback. Similarly, in push-T, the policy must infer the block's pose from images to achieve the required translation and reorientation.

2) Evaluation: To reduce variance and ensure systematic evaluation, we initialize scenes from a fixed set of configurations shared between the simulation and the real world. These initial configurations are generated in our simulator by constructing a grid over the planar position (x, y) and rotation angle θ of objects placed on the table. The grid ranges are chosen to ensure that the evaluation set provides coverage comparable to the training distribution. In the real world, objects are positioned to replicate the corresponding grid states. We use an evaluation set size of 20, 27, and 16 for toy packing, rope routing, and push-T, respectively.

We use binary success criteria for all tasks. Following [19], we quantify the alignment between simulation and real-world performance using the Mean Maximum Rank Variation (MMRV) and the Pearson correlation coefficient (r).

The number of evaluation episodes plays a critical role in the uncertainty of measured success rates [11]. To capture this variability, we report uncertainty in our results using the Clopper–Pearson confidence interval (CI). We also visualize the Bayesian posterior of policy success rates under a uniform Beta prior with violin plots.

We evaluate four state-of-the-art imitation learning policies: ACT [1], DP [2], SmolVLA [4], and Pi-0 [3]. The real-world setup consists of a single UFactory xArm 7 robot arm equipped with two calibrated Intel RealSense RGB-D cameras: a D405 mounted on the robot wrist and a D455 mounted on the table as a fixed external camera. All policies take as input images from both camera views, along with the current end-effector state. For push-T, the end-effector state includes only the 2D position (x, y); for the other tasks, it additionally includes the position, rotation, and gripper openness. Across all tasks, we collect 39-60 successful demonstrations via teleoperation using GELLO [79]. Training is performed using the open-source LeRobot [80] implementation, except for Pi-0, where we adopt the original implementation [3] for better performance.

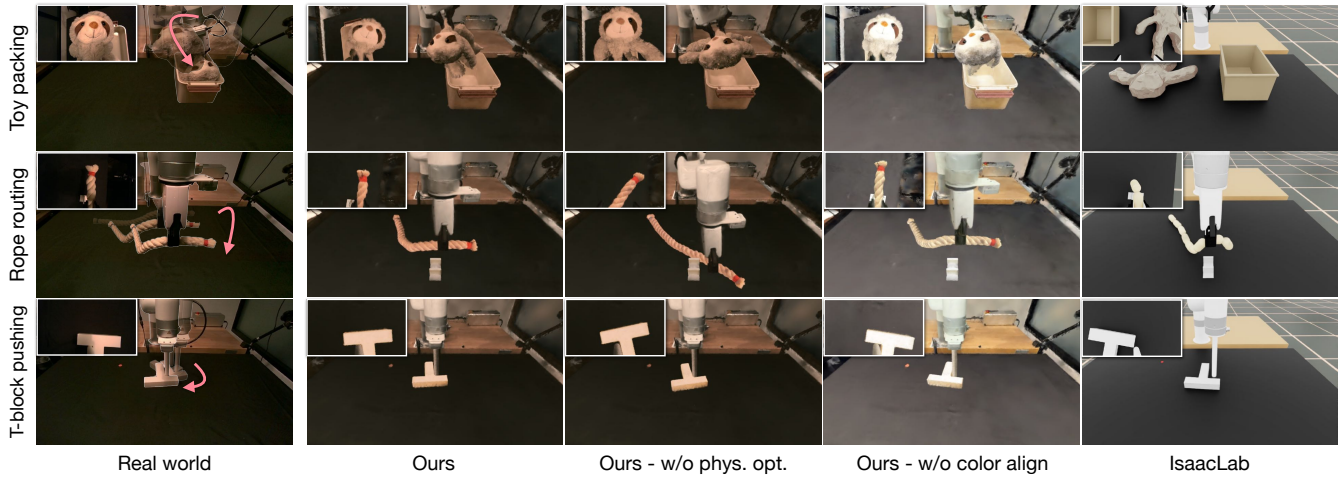


Fig. 5: **Comparison of rendering and dynamics quality.** Real-world observations (left) compared with our method, two ablations, and the IsaacLab baseline across three tasks. From right to left, visual and physical fidelity progressively improve. Without physics optimization, object dynamics deviate, causing failures such as the toy's limbs not fitting into the box or the rope slipping before routing. Without color alignment, rendered images exhibit noticeable color mismatches. The IsaacLab baseline (rightmost) shows lower realism in both rendering and dynamics compared to our approach.

B. Baseline

As a baseline, we use NVIDIA IsaacLab [13] as the simulation environment. Robot and environment assets are imported and aligned in position and color to match the real-world setup. IsaacLab provides a general-purpose robot simulation framework built on the PhysX physics engine, but its support for deformable objects remains limited. For ropes, we approximate deformable behavior using an articulated chain structure. However, for the plush toy, realistic grasping and deformation could not be stably simulated, making task completion infeasible; we therefore excluded this task from our quantitative comparisons.

C. Sim-and-Real Correlation

Figure 3 (left) shows the performance of all policy checkpoints in both simulation and the real world. We observe a strong correlation: policies that achieve higher success rates in reality also achieve higher success rates in our simulator, consistently across architectures and tasks. Figure 3 (right) further highlights that our simulator achieves stronger correlation than the IsaacLab baseline [13]. This is also confirmed by the quantitative results in Table I, with our simulator achieving a Pearson coefficient $r > 0.9$ for all policies. By contrast, the baseline yields only $r = 0.649$ on push-T, and an even lower $r = 0.237$ on rope routing as a result of the larger dynamics gap. The low MMRV value for the IsaacLab rope routing task arises from its consistently low success rates, which in turn produce fewer ranking violations.

D. Policy Performance Analysis

Figure 4 further illustrates per-policy, per-task performance curves across training iterations. We observe that simulation success rates generally follow the same progression as real-world success rates, further highlighting the correlation. For example, in the toy packing-DP case, both simulation and real success rates peak at iteration 5,000 and decline significantly by iteration 7,000. Similarly, in the rope routing-Pi-0 case, performance peaks around iteration 20,000. These results suggest that our simulator can be used as a practical tool for monitoring policy learning dynamics, selecting checkpoints for real-world testing, and setting approximate expectations for real-world performance.

In cases where simulation and real success rates do not overlap, such as toy packing-SmolVLA and rope routing-ACT, the simulator still captures the correct performance trend, even if the absolute success rates differ. We attribute these discrepancies to residual gaps in visual appearance and dynamics, as well as variance from the limited number of evaluation episodes (16–27 per checkpoint).

	Toy packing		Rope routing		T-block pushing	
	MMRV↓	r↑	MMRV↓	r↑	MMRV↓	r↑
IsaacLab [13]	-	-	**0.022**	0.237	0.031	0.649
Ours w/o color	0.200	0.805	0.156	0.714	0.031	0.529
Ours w/o phys.	0.200	0.694	0.119	0.832	0.031	0.905
Ours	**0.087**	**0.944**	0.096	**0.901**	**0.000**	**0.915**

TABLE I: **Quantitative comparison of correlation.** *Ours w/o color*: our method without color alignment. *Ours w/o phys.*: our method without physics optimization. Lower MMRV indicates fewer errors in ranking policy performance, while higher r reflects stronger statistical correlation. Best results are highlighted in bold.

E. Ablation Study

To measure the importance of the rendering and dynamics realism for our Gaussian Splatting simulator, we perform ablation studies on the correlation metrics MMRV and r. We provide two ablated variants of our simulation:

- *Ours w/o color alignment*: we skip the color alignment step in simulation construction and use the original GS colors in the iPhone camera space, creating a mismatch in the appearance.
- *Ours w/o physics optimization*: instead of using the fully-optimized spring stiffness Y, we use a global stiffness value shared across all springs. The global value is given by the gradient-free optimization stage

in PhysTwin [25]. For push-T, we keep its rigidity and change its friction coefficients with the ground and the robot to create a mismatch in dynamics.

Figure 5 presents a visual comparison between our simulator, its ablated variants, and the baseline, using the same policy model and identical initial states. Our full method achieves the best rendering and dynamics fidelity, resulting in policy rollouts that closely match real-world outcomes. In contrast, the w/o physics optimization variant produces inaccurate object dynamics, while the w/o color alignment variant shows clear color mismatches.

Empirically, both dynamics and appearance mismatches lead to deviations between simulated and real policy rollouts, though policies exhibit different sensitivities to each type of gap. For example, in the rope routing task, the rope fails to enter the clip when stiffness is mis-specified (*w/o physics optimization*). In the push-T task, color discrepancies alter the robot's perception, causing it to push the block differently (*w/o color alignment*).

Table I details the quantitative results. Overall, our full method achieves the highest correlation values, outperforming the ablated variants. In particular, lower MMRV values reflect more accurate policy ranking, while higher Pearson correlation coefficients (r) indicate stronger and more consistent correlations without being influenced by outlier points.

V. Conclusion

In this work, we introduced a framework for evaluating robot manipulation policies in a simulator that combines Gaussian Splatting-based rendering with real-to-sim digital twins for deformable object dynamics. By addressing both appearance and dynamics, our simulator narrows the sim-to-real gap through physics-informed reconstruction, positional and color alignment, and deformation-aware rendering.

We demonstrated the framework on representative deformable and rigid body manipulation tasks, evaluating several state-of-the-art imitation learning policies. Our experiments show that policy success rates in simulation exhibit strong correlations with real-world outcomes ($r > 0.9$). Further analysis across highlights that our simulator can predict policy performance trends, enabling it to serve as a practical proxy for checkpoint selection and performance estimation. We found that both physics optimization and color alignment are critical for closing policy performance gaps.

In future work, scaling both simulation and evaluation to larger task and policy sets could provide deeper insights into the key design considerations for policy evaluation simulators. Moreover, our real-to-sim framework can be generalized to more diverse environments, supporting increasingly complex robot manipulation tasks.

Acknowledgment

This work is partially supported by the DARPA TIAMAT program (HR0011-24-9-0430), NSF Award #2409661, Toyota Research Institute (TRI), Sony Group Corporation, Samsung Research America (SRA), Google, Dalus AI, Pickle Robot, and an Amazon Research Award (Fall 2024). This article solely reflects the opinions and conclusions of its authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

We would like to thank Wenhao Yu, Chuyuan Fu, Shivansh Patel, Ethan Lipson, Philippe Wu, and all other members of the RoboPIL lab at Columbia University and SceniX Inc. for helpful discussions and assistance throughout the project.

References

[1] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, *Learning fine-grained bimanual manipulation with low-cost hardware*, 2023. arXiv: 2304.13705 [cs.RO].

[2] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," in *RSS*, 2023.

[3] K. Black et al., π_0*: A vision-language-action flow model for general robot control*, 2024. arXiv: 2410.24164 [cs.LG].

[4] M. Shukor et al., *Smolvla: A vision-language-action model for affordable and efficient robotics*, 2025. arXiv: 2506.01844 [cs.LG].

[5] C. Chi et al., "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," in *RSS*, 2024.

[6] T. Lin, K. Sachdev, L. Fan, J. Malik, and Y. Zhu, "Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids," *arXiv:2502.20396*, 2025.

[7] B. Tang et al., *Industreal: Transferring contact-rich assembly tasks from simulation to reality*, 2023. arXiv: 2305.17110 [cs.RO].

[8] A. Brohan et al., "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *arXiv preprint arXiv:2307.15818*, 2023.

[9] P. Intelligence et al., $\pi_{0.5}$*: A vision-language-action model with open-world generalization*, 2025. arXiv: 2504.16054 [cs.LG].

[10] NVIDIA et al., "GR00T N1: An open foundation model for generalist humanoid robots," in *ArXiv Preprint*, Mar. 2025. arXiv: 2503.14734.

[11] T. L. Team et al., *A careful examination of large behavior models for multitask dexterous manipulation*, 2025. arXiv: 2507.05331 [cs.RO].

[12] G. R. Team et al., *Gemini robotics: Bringing ai into the physical world*, 2025. arXiv: 2503.20020 [cs.RO].

[13] NVIDIA, *NVIDIA Isaac Sim*, 2024.

[14] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*, 2012, pp. 5026–5033.

[15] F. Xiang et al., "SAPIEN: A simulated part-based interactive environment," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[16] C. Li et al., *Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation*, 2024. arXiv: 2403.09227 [cs.RO].

[17] G. Authors, *Genesis: A generative and universal physics engine for robotics and beyond*, Dec. 2024.

[18] R. Tedrake, *Drake: Model-based design and verification for robotics*, 2019.

[19] X. Li et al., "Evaluating real-world robot manipulation policies in simulation," in *CoRL*, 2024.

[20] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023.

[21] J. Abou-Chakra et al., *Real-is-sim: Bridging the sim-to-real gap with a dynamic digital twin*, 2025. arXiv: 2504.03597 [cs.RO].

[22] M. N. Qureshi, S. Garg, F. Yandun, D. Held, G. Kantor, and A. Silwal, *Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting*, 2024. arXiv: 2409.10161 [cs.RO].

[23] X. Li et al., *Robogsim: A real2sim2real robotic gaussian splatting simulator*, 2024. arXiv: 2411.11839 [cs.RO].

[24] L. Barcellona, A. Zadaianchuk, D. Allegro, S. Papa, S. Ghidoni, and E. Gavves, *Dream to manipulate: Compositional world models empowering robot imitation learning with imagination*, 2025. arXiv: 2412.14957 [cs.RO].

[25] H. Jiang, H.-Y. Hsu, K. Zhang, H.-N. Yu, S. Wang, and Y. Li, "Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos," *ICCV*, 2025.

[26] G. Brockman et al., *Openai gym*, 2016. arXiv: 1606.01540 [cs.LG].

[27] Octo Model Team et al., "Octo: An open-source generalist robot policy," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[28] J. Wang, M. Leonard, K. Daniilidis, D. Jayaraman, and E. S. Hu, *Evaluating pi0 in the wild: Strengths, problems, and the future of generalist robot policies*, 2025.

[29] A. Padalkar et al., "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023.

[30] A. Khazatsky et al., "Droid: A large-scale in-the-wild robot manipulation dataset," 2024.

[31] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, Sep. 2015.

[32] K. Van Wyk, J. Falco, and E. Messina, "Robotic grasping and manipulation competition: Future tasks to support the development of assembly robotics," in *Robotic Grasping and Manipulation Challenge*, Springer, 2016, pp. 190–200.

[33] N. Correll et al., "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 172–188, 2018.

[34] G. Zhou et al., *Train offline, test online: A real robot learning benchmark*, 2023. arXiv: 2306.00942 [cs.RO].

[35] S. Dasari et al., *Rb2: Robotic manipulation benchmarking with a twist*, 2022. arXiv: 2203.08098 [cs.RO].

[36] S. Tao et al., "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," *RSS*, 2025.

[37] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, *Rlbench: The robot learning benchmark & learning environment*, 2019. arXiv: 1909.12271 [cs.RO].

[38] S. Srivastava et al., "Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments," in *CoRL*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. PMLR, vol. 164, Aug. 2022, pp. 477–490.

[39] X. Puig et al., *Habitat 3.0: A co-habitat for humans, avatars and robots*, 2023. arXiv: 2310.13724 [cs.HC].

[40] S. Nasiriany et al., "Robocasa: Large-scale simulation of everyday tasks for generalist robots," in *RSS*, 2024.

[41] Y. Zhu et al., *Robosuite: A modular simulation framework and benchmark for robot learning*, 2025. arXiv: 2009.12293 [cs.RO].

[42] A. Mandlekar et al., *Mimicgen: A data generation system for scalable robot learning using human demonstrations*, 2023. arXiv: 2310.17596 [cs.RO].

[43] X. Yang, C. Eppner, J. Tremblay, D. Fox, S. Birchfield, and F. Ramos, *Robot policy evaluation for sim-to-real transfer: A benchmarking perspective*, 2025. arXiv: 2508.11117 [cs.RO].

[44] Y. R. Wang et al., *Roboeval: Where robotic manipulation meets structured and scalable evaluation*, 2025. arXiv: 2507.00435 [cs.RO].

[45] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *ICRA*, IEEE, 2018, pp. 3803–3810.

[46] Y. Chebotar et al., "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *ICRA*, IEEE, 2019, pp. 8973–8979.

[47] OpenAI et al., *Solving rubik's cube with a robot hand*, 2019. arXiv: 1910.07113 [cs.LG].

[48] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, *Retinagan: An object-aware approach to sim-to-real transfer*, 2021. arXiv: 2011.03148 [cs.RO].

[49] S. Liu, Z. Ren, S. Gupta, and S. Wang, "Physgen: Rigid-body physics-grounded image-to-video generation," in *ECCV*, Springer, 2024, pp. 360–378.

[50] B. Chen et al., "Physgen3d: Crafting a miniature interactive world from a single image," in *CVPR*, 2025, pp. 6178–6189.

[51] Y. Jiang et al., "Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality," in *SIGGRAPH*, 2024, pp. 1–1.

[52] T. Xie et al., "Physgaussian: Physics-integrated 3d gaussians for generative dynamics," in *CVPR*, 2024, pp. 4389–4398.

[53] R.-Z. Qiu, G. Yang, W. Zeng, and X. Wang, *Feature splatting: Language-driven physics-based scene synthesis and editing*, 2024. arXiv: 2404.01223 [cs.CV].

[54] B. Bianchini, M. Zhu, M. Sun, B. Jiang, C. J. Taylor, and M. Posa, "Vysics: Object reconstruction under occlusion by fusing vision and contact-rich physics," in *RSS*, Jun. 2025.

[55] W. Yang, Z. Xie, X. Zhang, H. B. Amor, S. Lin, and W. Jin, *Twintrack: Bridging vision and contact physics for real-time tracking of unknown dynamic objects*, 2025. arXiv: 2505.22882 [cs.RO].

[56] J. Abou-Chakra, K. Rana, F. Dayoub, and N. Suenderhauf, "Physically embodied gaussian splatting: A visually learnt and physically grounded 3d representation for robotics," in *CoRL*, 2024.

[57] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing," in *IROS*, IEEE, 2016, pp. 30–37.

[58] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, *Learning mesh-based simulation with graph networks*, 2021. arXiv: 2010.03409 [cs.LG].

[59] K. Zhang, B. Li, K. Hauser, and Y. Li, "Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation," in *RSS*, 2024.

[60] K. Zhang, B. Li, K. Hauser, and Y. Li, "Particle-grid neural dynamics for learning deformable object models from rgb-d videos," in *RSS*, 2025.

[61] T. Tian, H. Li, B. Ai, X. Yuan, Z. Huang, and H. Su, "Diffusion dynamics models with generative state estimation for cloth manipulation," *arXiv preprint arXiv:2503.11999*, 2025.

[62] X. Li et al., "Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification," *arXiv preprint arXiv:2303.05512*, 2023.

[63] T. Zhang et al., "Physdreamer: Physics-based interaction with 3d objects via video generation," in *ECCV*, Springer, 2024, pp. 388–406.

[64] L. Zhong, H.-X. Yu, J. Wu, and Y. Li, "Reconstruction and simulation of elastic objects with spring-mass 3d gaussians," in *ECCV*, Springer, 2024, pp. 407–423.

[65] C. Chen et al., "Vid2sim: Generalizable, video-based reconstruction of appearance, geometry and physics for mesh-free simulation," in *CVPR*, 2025, pp. 26 545–26 555.

[66] X. Han et al., "Re^{3}sim: Generating high-fidelity simulation data via 3d-photorealistic real-to-sim for robotic manipulation," *arXiv preprint arXiv:2502.08645*, 2025.
[67] A. Escontrela et al., "Gaussgym: An open-source real-to-sim framework for learning locomotion from pixels," *arXiv preprint arXiv:2510.15352*, 2025.
[68] J. Yu et al., *Real2render2real: Scaling robot data without dynamics simulation or robot hardware*, 2025. arXiv: 2505.09601 [cs.RO].
[69] S. Yang et al., "Novel demonstration generation with gaussian splatting enables robust one-shot manipulation," *arXiv preprint arXiv:2504.13175*, 2025.
[70] G. Jiang et al., *Gsworld: Closed-loop photo-realistic simulation suite for robotic manipulation*, 2025. arXiv: 2510.20813 [cs.RO].
[71] H. Kress-Gazit et al., "Robot learning as an empirical science: Best practices for policy evaluation," *arXiv preprint arXiv:2409.09491*, 2024.
[72] Niantic, *Scaniverse*, https://scaniverse.com/.
[73] PlayCanvas and Snap Inc., *Supersplat*, https://github.com/playcanvas/supersplat, [Computer software], 2025.
[74] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.
[75] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.
[76] P. J. Green, "Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 46, no. 2, pp. 149–170, 1984.
[77] M. Macklin, *Warp: A high-performance python framework for gpu simulation and graphics*, https://github.com/nvidia/warp, NVIDIA GPU Technology Conference (GTC), Mar. 2022.
[78] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," vol. 26, no. 3, 80–es, Jul. 2007.
[79] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, "Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators," in *IROS*, 2024.
[80] R. Cadene et al., *Lerobot: State-of-the-art machine learning for real-world robotics in pytorch*, https://github.com/huggingface/lerobot, 2024.

APPENDIX

Contents

Appendix I: Additional Technical Details 10
- I-A Platform and Tasks . . . 10
  - I-A.1 Robot Setup . . . 10
  - I-A.2 Data Collection . . . 10
  - I-A.3 Task Definition . . . 10
- I-B Simulation . . . 11
  - I-B.1 Assets . . . 11
  - I-B.2 Positional Alignment . . . 11
  - I-B.3 Color Alignment . . . 11
  - I-B.4 PhysTwin Training . . . 11
  - I-B.5 Simulation Loop . . . 12
- I-C Policy Training . . . 12
  - I-C.1 Datasets . . . 12
  - I-C.2 Normalizations . . . 12
  - I-C.3 Image Augmentations . . . 12
  - I-C.4 Hyperparameters . . . 12
- I-D Evaluation . . . 12
  - I-D.1 Evaluation Protocol . . . 12
  - I-D.2 Episode Settings . . . 12
  - I-D.3 Success Criteria . . . 12

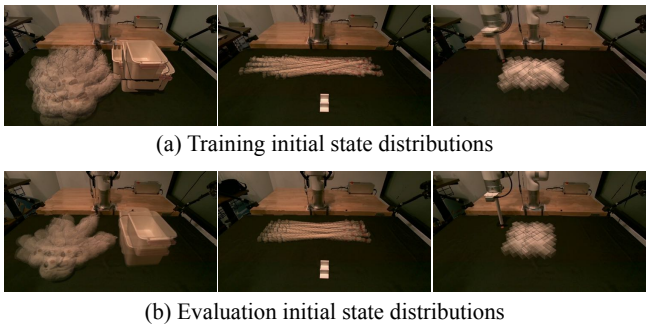
Appendix II: Additional Results 13
- II-A Scaling up Simulation Evaluation . . . 13
- II-B Replaying Real Rollouts . . . 13
- II-C Additional Qualitative Results . . . 14

APPENDIX I
ADDITIONAL TECHNICAL DETAILS

A. Platform and Tasks

1) Robot Setup: We use a UFactory xArm 7 robot mounted on a tabletop. The robot arm has 7 degrees of freedom. The robot end-effector can be interchanged between the standard xArm gripper and a custom 3D-printed pusher, depending on the task. Two Intel RealSense RGB-D cameras are connected to the robot workstation: a D455 fixed on the table overlooking the workspace, and a D405 mounted on the robot wrist via a custom 3D-printed clip. To ensure consistent appearance between real and simulated observations, we fix the white balance and exposure settings of both cameras.

2) Data Collection: We use GELLO for data collection. GELLO [79] streams high-frequency joint-angle commands to the robot, which we execute using joint-velocity control for smooth motion tracking. At each timestep, the robot computes the difference between the commanded and measured joint angles, then sets each joint's target angular velocity proportional to this delta. To prevent abrupt movements, the velocity vector is normalized such that its total ℓ_2 norm does not exceed a predefined limit. This approach enables stable and continuous trajectory following without jerky motions. During policy evaluation, we apply the same control strategy, ensuring that the policy outputs are tracked consistently in both real and simulated environments.



(a) Training initial state distributions

(b) Evaluation initial state distributions

Fig. 6: **Training and evaluation data distributions.** Top: spatial coverage of initial states in the training set. Bottom: the corresponding coverage in the evaluation set.

Name	Dynamics Type	3D Representation
xArm-gripper-tabletop	Articulated+Fixed	GS+URDF+Mesh
xArm-pusher-tabletop	Articulated+Fixed	GS+URDF+Mesh
Plush sloth	Deformable	GS+PhysTwin
Rope	Deformable	GS+PhysTwin
T-block	Rigid	GS+PhysTwin
Box	Fixed	GS+Mesh
Clip	Fixed	GS+Mesh

TABLE II: **Simulation assets.** Each row corresponds to an individual Gaussian Splatting scan, specifying its dynamics type in simulation and the 3D representation used for physical simulation and rendering. These assets are combined to instantiate all three manipulation tasks within the simulator.

3) Task Definition: To evaluate the effectiveness of our simulator, we select a set of rigid- and soft-body manipulation tasks that require the policy to leverage object dynamics while incorporating visual feedback. The formulation and setup of each task are described as follows.

a) Toy Packing: The robot grasps the plush toy by one of its limbs, lifts it above the box, and adjusts its pose such that the arm and leg on one side hang into the box. The robot then tilts the toy slightly to allow the other side's limbs to enter, before lowering it further to pack the toy snugly inside the box. Because the box is intentionally compact, the robot must adapt to the toy's pose to successfully execute the packing motion without leaving any limbs protruding over the box edges. A total of 39 human demonstration episodes are recorded for this task.

b) Rope Routing: The robot grasps one end of the rope (marked with red rubber bands), lifts it, and positions it above the cable holder before lowering it to gently place the rope into the slot. Because the rope–holder contact point is offset from the grasp location, the rope dynamics play a critical role in determining the appropriate displacement and trajectory required for successful placement. A total of 56 human demonstration episodes are collected for this task.

c) T-block Pushing: The robot begins with the pusher positioned above an orange marker on the table, while the end-effector's z-coordinate remains fixed throughout the motion. The robot must move to the T-block's location and push it toward a predefined goal region. The goal is not physically marked in the workspace but is visualized as a yellow translucent mask overlaid on the fixed-camera images.

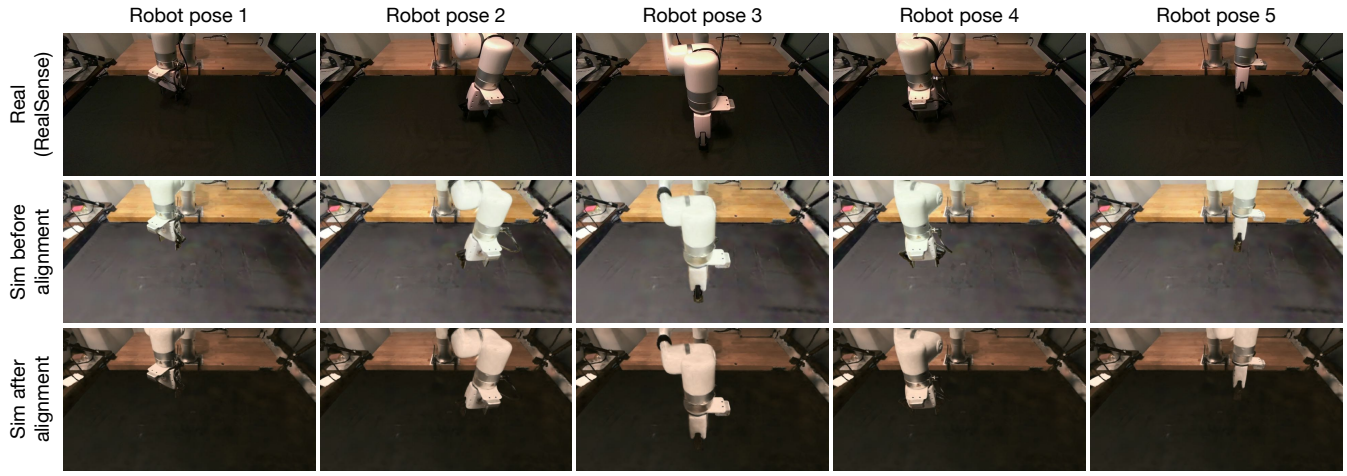
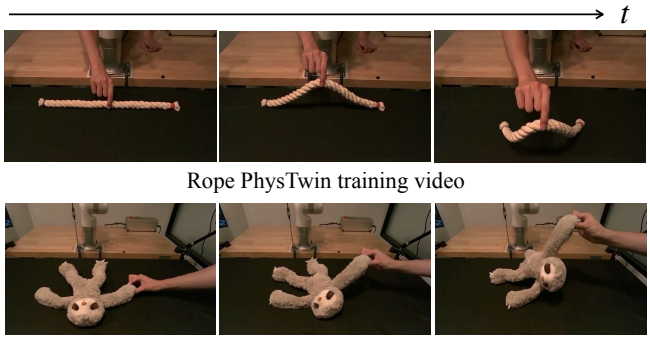


Fig. 7: **Color alignment.** Five image pairs used for the color alignment process are shown. *Top:* real images captured by the RealSense cameras. *Middle:* raw Gaussian Splatting renderings with the robot posed identically to the real images. *Bottom:* GS renderings after applying the optimized color transformation, showing improved consistency with real-world color appearance.



Rope PhysTwin training video

Plush toy PhysTwin training video

Fig. 8: **PhysTwin training videos.** A few representative camera frames are shown for each training video, where a human subject interacts with the deformable object by hand. These videos are used by PhysTwin to reconstruct the object's geometry and estimate its physical parameters for building the digital twin models.

The initial positions and orientations of the T-block are randomized, and a total of 60 human demonstration episodes are collected for this task.

B. Simulation

1) Assets: A summary of the simulation assets used in our experiments is provided in Table II. Each asset corresponds to a single Gaussian Splatting reconstruction followed by a pose alignment process.

2) Positional Alignment: To align the robot-scene Gaussian Splatting scan with the robot's URDF model, we first perform a coarse manual alignment in SuperSplat [73] to roughly match the origins and orientations of the x, y, and z axes. Next, we manually define a bounding box to separate the robot Gaussians from the scene Gaussians. We then apply ICP registration between two point clouds: one formed by the centers of the robot Gaussians, and the other by uniformly sampled surface points from the robot URDF mesh. The resulting rigid transformation is applied to the entire GS, ensuring that both the robot and scene components are consistently aligned in the unified coordinate frame.

Algorithm 1: Simulation Loop

Data: PhysTwin particle positions and velocities x, v, PhysTwin spring-mass parameters P, robot mesh R, robot motion a, static meshes $M_{1:k}$, ground plane L, total timestep T, substep count N, Gaussians G

for $t \leftarrow 0$ **to** $T-1$ **do**
- $x^*, v^* = x_t, v_t$
- $R^*_{1:N} = \text{interpolate\_robot\_states}(R_t, a_t)$
- **for** $\tau \leftarrow 0$ **to** $N-1$ **do**
  - $v^* = \text{step\_springs}(x^*, v^*, P)$
  - $v^* = \text{self\_collision}(x^*, v^*, P)$
  - $x^*, v^* = \text{robot\_mesh\_collision}(x^*, v^*, R_\tau, a_\tau)$
  - **for** $i \leftarrow 1$ **to** k **do**
    - $x^*, v^* = \text{fixed\_mesh\_collision}(x^*, v^*, M_i)$
  - **end**
  - $x^*, v^* = \text{ground\_collision}(x^*, v^*, L)$
- **end**
- $x_{t+1}, v_{t+1} = x^*, v^*$
- $R_{t+1} = R^*_N$
- $G_{t+1} = \text{renderer\_update}(G_t, x_t, x_{t+1}, R_t, R_{t+1})$

end

3) Color Alignment: The robot–scene scan has the most significant influence on the overall color profile of the rendered images. To align its appearance with the RealSense color space, we apply Robust IRLS with Tukey bi-weight to estimate the color transformation. We use five images of resolution 848×480 for this optimization. To mitigate the imbalance between the dark tabletop and the bright robot regions, each pixel is weighted by the norm of its RGB values, giving higher weight to high-brightness pixels in the least-squares loss. The optimization is run for 50 iterations. Figure 7 visualizes the input images and the resulting color alignment.

4) PhysTwin Training: We use the original PhysTwin [25] codebase for training the rope and sloth digital twins. Phys-

Model	Visual	State	Action	Relative?
ACT	mean–std	mean–std	mean–std	False
DP	mean–std	min–max	min–max	False
SmolVLA	identity	mean–std	mean–std	True
Pi-0	mean–std	mean–std	mean–std	True

TABLE III: **Normalization schemes across models.** Columns indicate the normalization applied to each modality (visual, state, and action) and whether the model operates in a *relative action space*. *Mean–std* denotes standardization to zero mean and unit variance, while *min–max* scales values to $[-1, 1]$.

Color Transformations		Spatial Transformations	
Type	**Range**	**Type**	**Range**
Brightness	$(0.8, 1.2)$	Perspective	0.025
Contrast	$(0.8, 1.2)$	Rotation	$[-5°, 5°]$
Saturation	$(0.5, 1.5)$	Crop	$[10, 40]$ px
Hue	$(-0.05, 0.05)$		
Sharpness	$(0.5, 1.5)$		

TABLE IV: **Image augmentation configuration.** For color transformations, numeric ranges denote multiplicative or additive jitter factors applied to image intensities. For spatial transformations, ranges specify the perturbation magnitudes for projective distortion, rotation, and cropping.

Twin requires only a single multi-view RGB-D video to reconstruct object geometry and optimize physical parameters. For data capture, we record using three fixed Intel RealSense D455 cameras. The videos for the two objects are visualized in Figure 8. For the T-block pushing task, since it is a rigid object, we construct the PhysTwin object by uniformly sampling points within the mesh, connecting them with springs using a connection radius of 0.5 and a maximum of 50 neighbors, and assigning a uniform spring stiffness of 3×10^4 to all connections. This setup ensures that the object behaves like a rigid body.

5) Simulation Loop: The simulation loop, including robot action processing, PhysTwin simulation, collision handling, and renderer updates, is summarized in Algorithm 1.

C. Policy Training

1) Datasets: To better understand the data distribution used for both policy training and evaluation, we visualize the coverage of initial states in Figure 6.

2) Normalizations: Normalization plays a crucial role in ensuring stable policy learning and consistent performance across models. For input and output normalization, we follow the conventions defined in each algorithm's original implementation (summarized in Table III). Specifically, the *mean–std* scheme standardizes features to zero mean and unit variance, whereas the *min–max* scheme scales each dimension independently to $[-1, 1]$.

For the VLA (SmolVLA and Pi-0) policies, we employ *relative actions* to encourage more corrective and stable behavior, treating each action as an $\mathrm{SE}(3)$ transformation of the end-effector pose in the base frame. Inspired by [11], we compute both normalization statistics (*mean–std* or *min–max*) over a rolling window corresponding to the action chunk size across the entire dataset. Each action within a chunk is then normalized using its own statistics to maintain a consistent magnitude in the normalized space—mitigating the tendency of later actions in the chunk to exhibit larger amplitudes.

Model	Visual Res.	State Dim.	Action Dim.	$\mathbf{T_p}$	$\mathbf{T_e}$
ACT	L: 120×212; H: 240×240	8	8	50	50
DP	L: 120×212; H: 240×240	8	8	64	50
SmolVLA	L: 120×212; H: 240×240	8	8	50	50
Pi-0	L: 120×212; H: 240×240	8	8	50	50

TABLE V: **Observation and action spaces.** Low-resolution inputs are used for the rope-routing task, while high-resolution inputs are used for the other tasks. State and action vectors include end-effector position, quaternion, and gripper state, expressed in either absolute or relative coordinates. $\mathbf{T_p}$ and $\mathbf{T_e}$ denote the prediction and execution horizons, respectively.

Vision Backbone	#V-Params	#P-Params	LR	Batch Size	#Iters
ResNet-18 (ACT)	18M	34M	1×10^{-5}	512	7k
ResNet-18 (DP)	18M	245M	1×10^{-4}	512	7k
SmolVLM-2	350M	100M	1×10^{-4}	128	20k
PaliGemma (Pi-0)	260B	300M	5×10^{-5}	8	30k

TABLE VI: **Training configuration.** Model-specific hyperparameters used in policy training. #V-Params and #P-Params denote the number of parameters in the visual encoder and policy head, respectively. **LR**, **Batch Size**, and **#Iters** refer to the learning rate, batch size, and total training iterations.

3) Image Augmentations: To improve visual robustness and generalization, we apply a combination of color and spatial augmentations to each input image during training. For every image in a training batch, three augmentation operations are randomly sampled and composed. Table IV summarizes the augmentation types and their corresponding parameter ranges.

4) Hyperparameters: A complete overview of the observation and action spaces, as well as the training configurations for each model, is presented in Tables V and VI. For VLA-based policies, we finetune only the action head (keeping the pretrained vision-language encoder frozen) on our datasets.

D. Evaluation

1) Evaluation Protocol: During evaluation, we sample a fixed set of initial states, and rollout the policies from both sim and real. To ensure that sim and real align with each other, we first sample object initial states in simulation and render them from the same camera viewpoint as the real-world physical setup. Then, we save the set of initial frame renderings, and a real-time visualizer overlays these simulated states onto the live camera stream, enabling a human operator to manually adjust the objects to match the simulated configuration.

2) Episode Settings: In all evaluation experiments in the main paper, the number of episodes for each task and the grid-based initial configuration randomization ranges are set as in Table VII.

3) Success Criteria: Real robot experiments typically rely on human operators to record success and failure counts, which is tedious and introduces human bias. For simulated

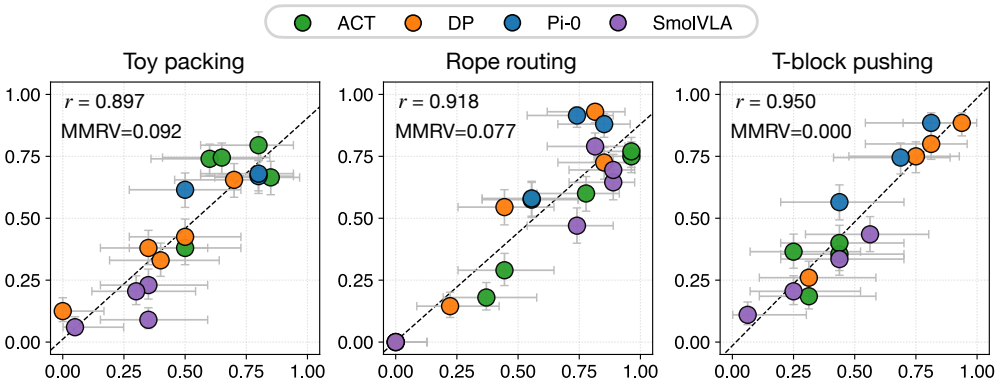


Fig. 9: **Sim-and-real correlations from scaled-up simulation evaluations.** Each point represents a policy evaluated on both domains, and the shaded region indicates the 95% confidence interval. Increasing the number of simulated episodes reduces statistical uncertainty and yields stable correlation estimates with real-world success rates, with the minimum observed correlation coefficient of 0.897. Compared to the main-paper experiments, the relative ordering of policy checkpoints remains consistent, demonstrating the robustness of the evaluation across larger-scale simulations.

Task	**Episodes**	x **(cm)**	y **(cm)**	θ **(deg)**
Toy packing (toy)	20	$[-5,5]$	$[-5,3]$	$[-5,5]$
Toy packing (box)	20	$[-5,5]$	$[0,5]$	$[-5,5]$
Rope routing (rope)	27	$[-5,5]$	$[-5,5]$	$[-10,10]$
T-block pushing (T-block)	16	$[-5,5]$	$[-5,5]$	$\{\pm 45, \pm 135\}$

TABLE VII: **Task randomization ranges used for evaluation.** For each task, the initial object configurations are randomized: the plush toy and box in toy packing, the rope in rope routing, and the T-block in T-block pushing.

experiments to scale up, automated success criteria are necessary. For all three tasks, we design metrics based on simulation states as follows:

a) Toy Packing: For each frame, we calculate the number of PhysTwin mass particles that fall within an oriented bounding box of the box's mesh. Within the final 100 frames (3.3 seconds) of a 15-second episode, if the number exceeds a certain threshold for over 30 frames, the episode is considered successful. Empirically, the total number of PhysTwin points is 3095, and we use a threshold number of 3050.

b) Rope Rouing: For each frame, we calculate the number of PhysTwin spring segments that pass through the openings of the channel of the clip. Within the final 100 frames (3.3 seconds) of a 30-second episode, if for both openings and more than 30 frames, the number of the spring segments that cross the opening is over 100, that indicates a sufficient routing through the clip and the episode is considered successful.

c) T-block Pushing: For each frame, we calculate the mean squared Euclidean distance between the current PhysTwin particles and the target-state PhysTwin particles. Within the final 100 frames (3.3 seconds) of a 60-second episode, if the mean squared distance is less than 0.002, the episode is considered successful.

APPENDIX II
ADDITIONAL RESULTS

A. Scaling up Simulation Evaluation

In the main paper, we evaluate each policy in simulation using an identical set of initial states as in the real-world experiments. This design controls for randomness but limits the number of available trials and thus results in high statistical uncertainty, as reflected by the wide Clopper-Pearson confidence intervals.

To account for the distributional differences introduced by uniformly sampling within the randomization range, we adopt slightly modified randomization settings compared to the grid-range experiments in the main paper. In the toy packing task, we use the same randomization range as described previously. For the rope routing task, we enlarge the x, y, θ randomization ranges to $[-7.5, 7.5]$ cm and $[-15, 15]$ degrees, respectively. For the T-block pushing task, we enlarge the x and y range to $[-7.5, 7.5]$ cm.

To better estimate the asymptotic correlation between simulation and real-world performance, we further scale up the number of simulation evaluations by sampling 200 randomized initial states from the task distribution. Figure 9 reports the resulting correlations between the scaled-up simulation metrics and real-world success rates.

We observe that the confidence intervals are significantly narrowed down, and the correlation estimates stabilize as the number of simulation episodes increases, suggesting that simulation fidelity becomes a reliable predictor of real-world outcomes when averaged across diverse task instances.

B. Replaying Real Rollouts

To further assess correspondence between our simulation and the real world, we perform replay-based evaluations, where real-world rollouts during policy inference are re-executed in the simulator using the same control commands. This allows us to disentangle dynamic discrepancies from appearance gaps, i.e., the difference in policy behaviors introduced by differences in perceived images is eliminated.

In total, we replay the real-world rollouts of 16 checkpoints each with 20 episodes for toy packing, 15 checkpoints each with 27 episodes for rope routing, and 12 checkpoints each with 16 episodes for T-block pushing. The object states in simulation are initialized to be identical to the corresponding real episodes.

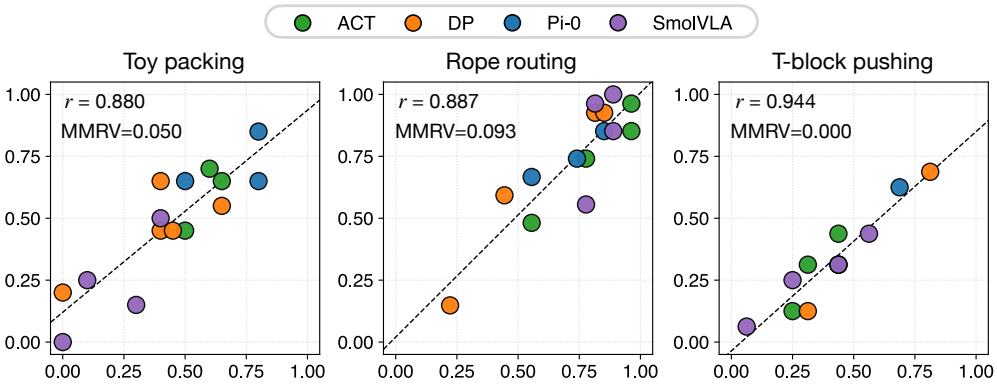


Fig. 10: **Sim-and-real correlations from replaying real-world rollouts.** Each point corresponds to a replay of a real-world policy checkpoint's evaluation results using identical control commands and camera trajectories within the simulator. The success rates are averaged over all episodes for each checkpoint. The resulting alignment highlights the degree to which our simulator reproduces the observed real-world outcomes.

Toy packing			**Rope routing**			**T-block pushing**		
	GT +	GT –		GT +	GT –		GT +	GT –
Replay +	106	37	Replay +	276	28	Replay +	63	1
Replay –	25	132	Replay –	24	77	Replay –	17	111

TABLE VIII: **Per-episode replay result.** We calculate the per-episode correlation between the replayed result and the real-world ground truth. Each subtable shows a 2×2 confusion matrix for each task (**TP**, **FP**, **FN**, **TN**), where rows indicate replay outcomes and columns indicate ground truth. Each entry records the total number of episodes, summed across all policy checkpoints. The strong diagonal dominance reflects high sim–real agreement in replayed trajectories.

Figure 10 shows the resulting correlations, and Table VIII reports the per-episode replay statistics. Across all three tasks, the confusion matrices exhibit strong diagonal dominance, indicating high agreement between replayed and real outcomes.

Notably, for toy packing, false positives (replayed success but real failure) are more frequent than false negatives, reflecting that the simulator tends to slightly overestimate success, likely due to simplified contact or friction models. For T-block pushing, false negatives are more frequent than false positives, indicating that some real success trajectories cannot be reproduced in the simulation, potentially due to a slight mismatch in friction coefficient and initial states.

Overall, the high diagonal values highlight that the simulator can reproduce real rollout outcomes most of the time, even with pure open-loop trajectory replay.

C. Additional Qualitative Results

We include further visualizations in Figure 11, which compares synchronized simulation and real-world trajectories across representative timesteps. For each task, we display both front and wrist camera views.

From the figure, we observe that the simulated trajectories closely reproduce the real-world sequences in both front-view and wrist-view observations. Object poses, contact transitions, and end-effector motions remain consistent across corresponding timesteps, indicating that the simulator effectively captures the underlying task dynamics as well as visual appearance.

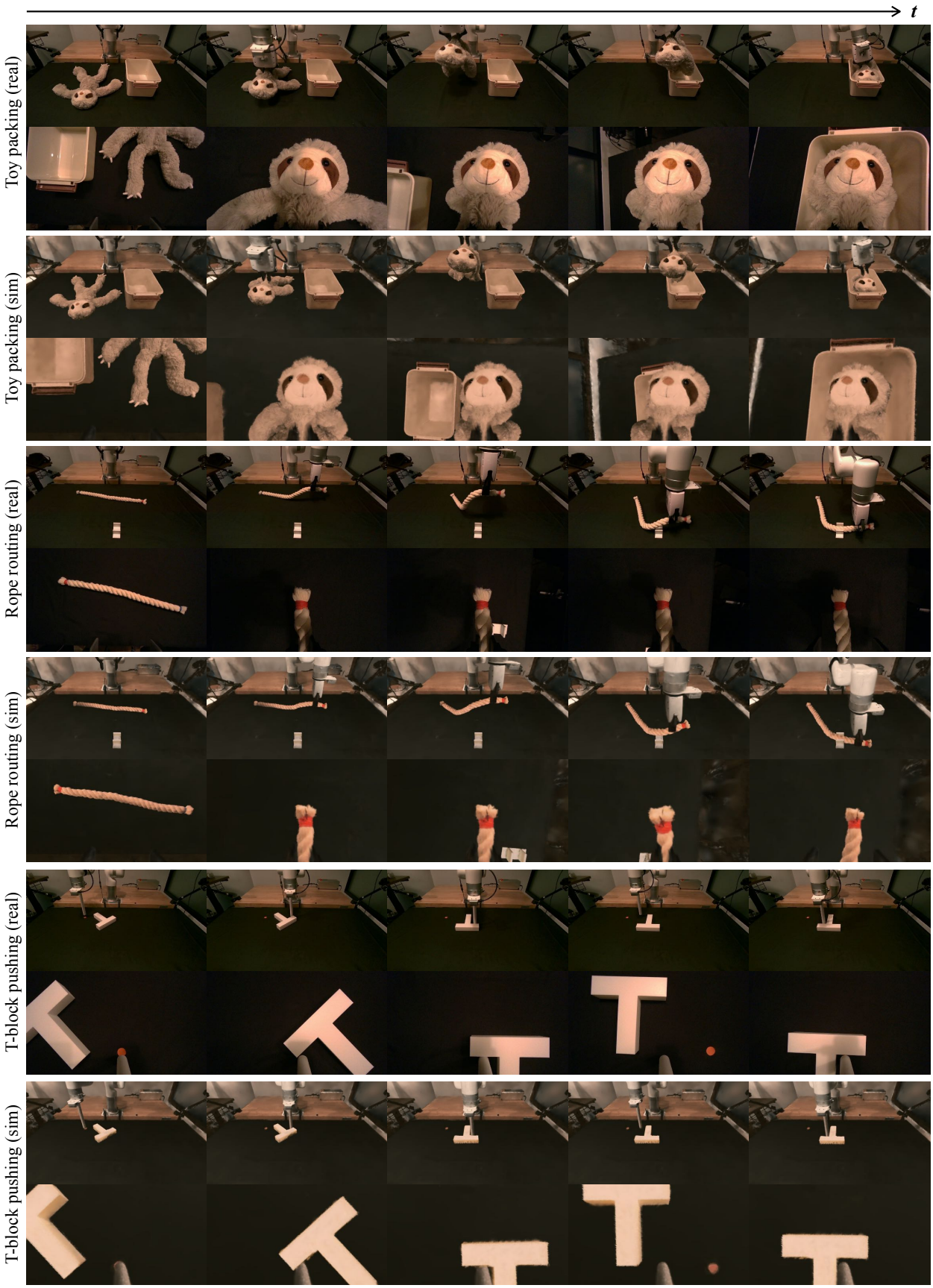


Fig. 11: **Sim and real rollout trajectories.** Columns correspond to synchronized timesteps along each rollout, with identical timestamps selected for simulation and real-world policy rollouts to illustrate correspondence. Each panel (e.g., toy packing (real)) shows front-view (top) and wrist-view (bottom) observations, with panels alternating between real and simulated trajectories.