GSWorld: Closed-Loop Photorealistic Simulation Suite for Robot Manipulation via Real-to-sim-to-real

Anonymous Authors.

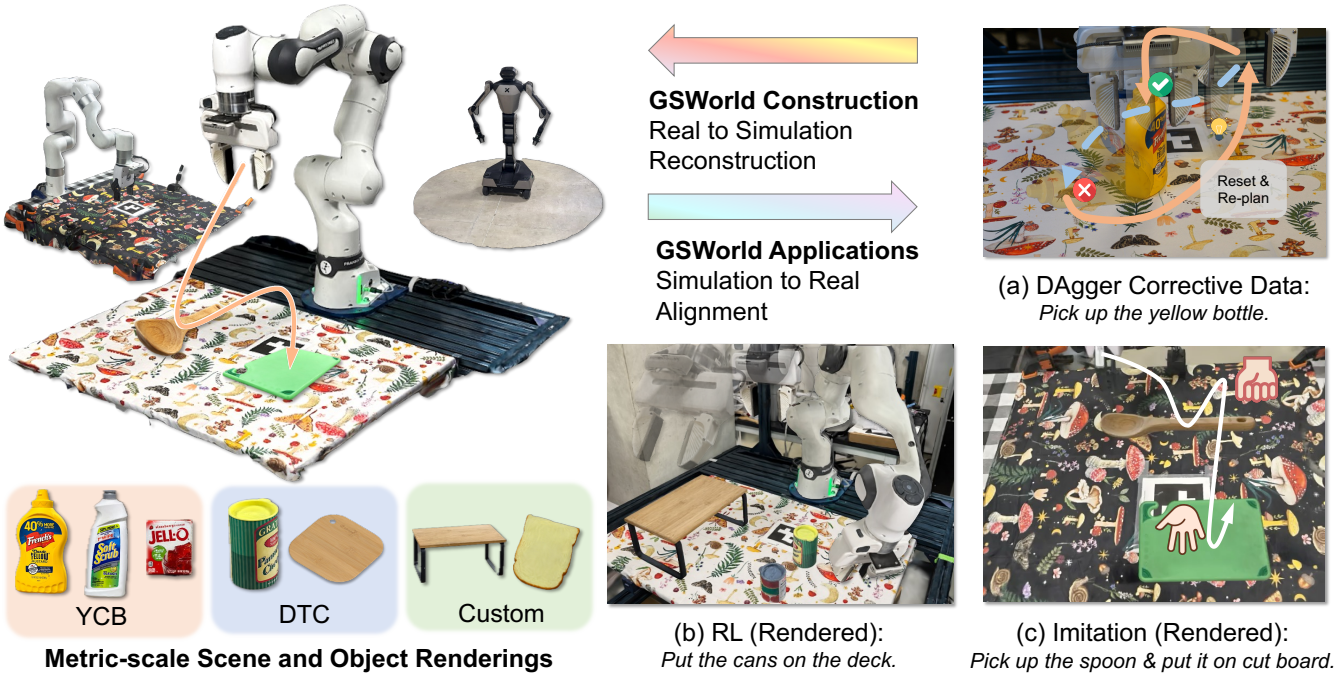

Fig. 1: GSWorld leverages 3DGS reconstruction to render photo-realistic robot scenes, supports multiple policy learning recipes in the simulation, and realizes zero-shot sim2real transfer. GSWorld also applies to policy visual benchmarking and virtual teleoperation for data collection.

***Abstract*— This paper presents GSWorld, a robust, photo-realistic simulator for robotics manipulation that combines 3D Gaussian Splatting with physics engines. Our framework advocates 'closing the loop' of developing manipulation policies with reproducible evaluation of policies learned from real-robot data and sim2real policy training without using real robots. To enable photo-realistic rendering of diverse scenes, we propose a new asset format, which we term GSDF (Gaussian Scene Description File), that infuses Gaussian-on-Mesh representation with robot URDF and other objects. With a streamlined reconstruction pipeline, we curate a database of GSDF that contains 3 robot embodiments for single-arm and bimanual manipulation, as well as more than 40 objects. Combining GSDF with physics engines, we demonstrate several immediate interesting applications: (1) learning zero-shot sim2real pixel-to-action manipulation policy with photo-realistic rendering, (2) automated high-quality DAgger data collection for adapting policies to deployment environments, (3) reproducible benchmarking of real-robot manipulation policies in simulation, (4) simulation data collection by virtual teleoperation, and (5) zero-shot sim2real visual reinforcement learning.**

I. Introduction

Training manipulation policies typically relies on three data sources—simulation, human videos, and real teleoperation—each presenting a distinct trade-off. While simulation provides a perfectly aligned action space for the robot, it often suffers from many sim-to-real gaps. Human videos offer the benefit of photo-realistic scenes and real physics, but lack temporally aligned robot actions and operate in a mismatched action space. Teleoperation successfully aligns both perception and actions, yet its high cost and difficulty to scale are significant limitations.

To resolve these trade-offs, we introduce GSWorld, a closed-loop, photo-realistic simulation suite that couples 3D Gaussian Splatting (3DGS) with physics to narrow both the visual and action-space gaps for manipulation. "Closed-loop" here means the same environment can be used to train, evaluate, diagnose failures, and relabel, enabling rapid iteration: policies perceive photo-realistic renderings while issuing controls in the robot's native space, all inside a simulator that mirrors real scenes closely enough to support zero-shot transfer and efficient adaptation. We demonstrate that GSWorld enables a range of downstream applications and, in particular, effective sim-to-real transfer for imitation learning, reinforcement learning, and DAgger-style data collection.

This closed-loop capability is powered by a bidirectional pipeline that ensures tight alignment between the physical

world and its digital twin. In the ***real-to-sim*** direction, our pipeline reconstructs a metric-accurate digital twin from short multi-view captures, sets an absolute scale with ArUco markers, and aligns the robot URDF to the scene via surface fitting (e.g., ICP). We then attach collision meshes and material properties to produce a versatile GSDF asset. The ***sim-to-real*** direction is the reverse: policies trained in GSWorld deploy on hardware without interface translation because their control and observation spaces match the robot's native APIs. Policies trained with both sim and real data can be deployed in the sim to evaluate, detect failures, and gather DAgger corrections, thus closing the iteration loop. Achieving this sim-to-real alignment requires that the scene's geometry, camera properties, and action semantics remain consistent across the real-to-sim divide. We measure the degree of this correspondence through a suite of metrics evaluating the visual, geometric, and functional similarity between the two worlds.

With photo-realistic perception and native action-space control in one loop, GSWorld supports zero-shot sim-to-real for both visual imitation learning and visual RL, while exploiting scalable parallelism to accelerate data generation and training. Its closed-loop DAgger workflow lets practitioners reproduce on-robot failures inside the digital twin, step through them frame-by-frame, and collect targeted corrective labels with far less teleoperation overhead. Finally, GSWorld provides reproducible visual benchmarking: shared GSDF assets, fixed camera intrinsics/extrinsics, consistent lighting/materials, and standardized action semantics enable apples-to-apples comparisons across robots, scenes, and tasks—so improvements reflect algorithmic progress rather than environmental variance.

Existing GS-based simulators either target single-setup photo-realistic rendering, provide engine-tied pipelines without a portable asset standard, or limit the reproducible cross-embodiment benchmarking and deployment-oriented on-policy data collection [5, 27, 38, 45]. While GSWorld delivers an effective real-to-sim-to-real workflow that unifies photo-realistic 3DGS with contact-accurate physics, enabling scalable cross-embodiment benchmarking, zero-shot imitation and reinforcement learning, and automated high-quality DAgger data collection for continual deployment-time improvement.

In summary, our contributions are:

- A solid real-to-sim-to-real pipeline. Our robust real-to-sim-to-real pipeline accurately aligns the simulation with the real environment, enabling a wide range of subsequent applications.
- Simulation Data Collection and Visual Imitation Learning (IL). GSWorld supports multiple sim data collection methods, *e.g.* motion planning, teleoperation. IL policies trained with GSWorld data can be directly deployed to the reconstructed real-world scenes.
- Visual RL. GSWorld is designed to utilize parallel environments in the simulation to train RL policies. We provide an analysis to show that GSWorld reduces RL sim2real visual gaps.
- Closed-loop DAgger Learning with Visual Benchmarking. GSWorld shows reliable policy evaluation results in correlation with real-world deployments, contributing to using DAgger to iteratively improve real-world policies.

II. Related Work

Robotics Simulation. With developments in computer graphics for rendering [54] and object material simulation [50], the robotics community has designed various physics engines [10, 13] and simulators [47, 56, 60] to support various robotics tasks. Recently, more and more simulators have started to improve efficiency and fidelity. For example, Mujoco-Jax [60] exploits just-in-time compilers [18] to achieve impressive simulation efficiency in Python. To improve rendering fidelity and reduce **sim2real visual gaps**, recent papers have turned to sophisticated ray-tracing techniques [54] and generative AI tools [39] to reduce the visual observation gap between simulation renderings and the real world. A recent 'meta-simulator', Roboverse [15], attempts to provides a unified simulation interface to make use of the advantages in individual simulators. This paper focuses on improving the fidelity of simulation renderings by combining recent advances in 3D Gaussian Splatting and simulators.

Real2Sim (Real-to-Simulation). As an alternative to modeling basic physics and elements from the bottom-up in simulation, real2sim approaches take a different approach to build simulation assets by virtualizing real-world assets. Recent real2sim methods can be roughly divided into two categories: photo-realistic 3D reconstruction [1, 6, 17, 27, 34, 35, 44, 45, 53] and part-level (articulated) object understanding [7, 11, 36, 52].

Real2Sim (Real-to-Simulation) - Photo-realistic Rendering. Early method in Real2Sim reconstruction [44] use NeRFs [37], Diffusion [28] or Mesh [52] for photo-realistic modeling. Due to the inherent implicit representation of NeRFs, they often rely on surrogates such as deformation fields [21] to deform the visual renderings to accommodate object motion, which is unnatural and inefficient. On the other hand, mesh-based representations contain many artifacts [52] that lead to visual gap. Simpler [28] proposes to use generative modeling to advance reproducible benchmarking [22, 24] by supporting photorealistic rendering. However, Simpler requires expensive manual labor to match green screen and textures, hindering its scalability. With developments in 3DGS [26], a rasterization-based method, SplatSim [45] successfully combined 3DGS with PyBullet [10] to build a photo-realistic simulator and demonstrated zero-shot sim2real policy deployment. As 3DGS can be explicitly represented as 'Gaussian Blobs', the photo-realistic appearance can be displaced consistently with object physics. However, SplatSim relies on manual 3D segmentation of both the robot arm and objects, which is overfitted to a single scene. In parallel to the development of SplatSim, Embodied-GS [1] learns arm-object interaction without using physics engine in a single scene; Robo-GS [34] focuses on identifying physical parameters of individual objects from rendering;

Fig. 2: GSWorld provides an interface on top of existing simulators to render photorealistic assets. Our GSDF assets are compatible with existing simulators to use standard formats for rendering visuals (*e.g., depth, segmentation*) and computing physics collisions. GSWorld provides a rendering wrapper on top of simulators to make the RGB rendering photo-realistic to support various domain randomization and applications.

and ManiGaussian [35] investigates optimizing Gaussian representations from simulators with ideally synchronized multi-view information. Most recently, Re3Sim [17] extends SplatSim [45] and found that the capability to perform photo-realistic simulation leads to more robust manipulation policies with domain randomization and mixed simulation (*e.g.,* non-photorealistic simulation assets with photo-realistic scenes). We continue to advance the progress in photo-realistic simulation by combining the latest advancements in 3DGS and introducing more 3DGS assets with a unified asset format.

Real2Sim (Real-to-Simulation) - Others. Here we provide a concise review of other progress in real2sim that is orthogonal to our method. Part-level (articulated) object understanding methods [7, 11, 36, 52] apply internet-scale pre-trained visual and language models [32, 46] to create physics and articulation of simulation assets from real-world observations. While these methods focus on understanding articulation of objects, recent methods have also worked on more challenging tasks such as estimating physics for deformable objects [25] and rigid contracts [42]. PhysTwin [25] optimizes physics models for elastic objects by assuming a Spring-Mass model and estimating physics parameters from video observations. Scalable Real2Sim [42] is an advancement from previous physics estimation method [34], in which the authors built a pipeline that uses robot arm and camera setups to automate estimation of rigid physics parameters including mass, center of mass, and inertia tensor [42]. Liu et al. [31] proposes to optimize robot kinematics from differentiable rendering.

III. Method

A. Problem Formulation

We consider the problem of learning robot policies from visual observations. Let $\mathcal{S}_{\text{sim}}$ denote simulated scene and $\mathcal{G}_{\text{real}}$ denote the real scene $\mathcal{S}_{\text{real}}$ reconstructed by 3DGS from multiple RGB viewpoints $\mathcal{V} = \Sigma v_i$. $\mathcal{G}_{real}$ can be used to render novel-view RGB images $I^{\text{gs}} = \mathcal{G}_{real}(p, s)$, which enables photorealistic rendering of the scene under arbitrary camera poses p and environment states s.

Our goal is to replace raw real-world RGB observations I^{real} with 3DGS-rendered images I^{gs} for downstream robot learning tasks, including IL, RL, and DAgger. Formally, at each time step t, the underlying state of the system is represented as

$$s_t = (q_t, x_t^1, \ldots, x_t^n), \tag{1}$$

where $q_t \in \mathbb{R}^m$ denotes the robot's joint position, and x_t^k represents the 6D pose of the k-th object in $\mathcal{S}_{\text{real}}$. We mainly use joint position control for the robots. The robot receives an observation

$$o_t = I_t^{\text{gs}} = \mathcal{G}_{real}(p_t, s_t), \tag{2}$$

rendered from $\mathcal{G}_{\text{real}}$ given the current camera pose p_t and environment state s_t.

In RL, the policy π_θ is trained by interacting with the environment and receiving rewards. In IL, the expert $\mathcal{E}$ provides demonstrations $\tau_{\mathcal{E}} = \{(q_1, o_1, a_1), \ldots, (q_T, o_T, a_T)\}$, which are used to supervise π_θ. In DAgger, π_θ is iteratively refined by collecting expert rollouts from previous failure cases, getting $\tau_{\mathcal{D}}$. In all cases, the policy takes as input I_t^{gs} instead of I_t^{real}, and robot proprioceptions q_t, i.e.,

$$a_t \sim \pi_\theta(I_t^{\text{gs}}, q_t). \tag{3}$$

At test time, the trained policy π_θ must generalize to real-world observations I^{real}, ensuring that GSWorld bridges sim2real visual gap.

B. Recipe for Real2Sim Reconstruction

This section describes how GSWorld creates GSDF assets to construct photo-realistic scenes. Compared with existing work [17, 27, 34, 45, 59] that focus on constructing robots in a single scene, GSWorld is designed (1) with an easy-to-use streamlined procedure to reduce manual alignment efforts and (2) to incorporate recent advancements in 3DGS such as geometric accuracy [19].

a) Collecting Training Views: To construct a scene with a robot, we use both robot sensors (*i.e.,* wrist cameras and third-person cameras) and mobile phone cameras while saving the current joint pose of robot during the scene capture.

b) Aligning Scale for Metric Representation: Existing real2sim2real methods [43, 45] relies on COLMAP [49], which introduces scale ambiguity. While such ambiguity can be dealt with manually for a single scene, it affects the scalability for multiple robot embodiments and scenes. To avoid manual scale alignment, we use a simple solution to include a printed ArUco marker [14, 23] on the tabletop during the data collection (qualitative examples in Fig. 1). The detected keypoints of ArUcO markers are projected onto the point cloud formed by 3DGS. We then scale the point cloud using the known scales of the ArUcO marker. In addition, the ArUco marker helps identify the support surface in collision and estimate gravity direction.

c) Aligning Robots and Table: Given a metric-scale $\mathcal{G}_{real}$ of a static robot $\mathcal{R}$ and a metric-scale robot URDF in $\mathcal{S}_{sim}$, we align the simulation joint position with the real world. Then, we sample and densify surface point clouds from the visual mesh of the robot URDF. We then perform an ICP to compute the rigid transform $\mathcal{T}^{gs}_{\mathcal{R},sim}$: $\mathcal{G}_{real} = \mathcal{T}^{gs}_{\mathcal{R},sim} \cdot \mathcal{S}_{sim}$. Compared to previous methods [45], our ICP has fewer degrees of freedom since the scale is fixed. Given the alignment, we use K-NN to segment robot links in $\mathcal{G}_{real}$.

d) Object Assets: Our prior reconstruction stage focuses on background and robot scans. For moveable objects $\mathcal{O}$, we consider integrating existing large-scale datasets and supporting custom objects for generalizability. Specifically, we use DTC [12] for its photo-realistic visual quality, and YCB [4]. For custom objects, we use 2DGS [19] to get the reconstruction $\mathcal{O}^{gs}$ and mesh reconstruction. Mass is estimated by weighing. The unobserved bottom regions of the object can be optionally inpainted using amodal reconstruction [2, 55] or 3D object generation [30] method. Similarly, we use ICP to get the transform $\mathcal{T}^{gs}_{k,sim}$ for the k-th object: $\mathcal{O}^{gs}_k = \mathcal{T}^{gs}_{k,sim} \cdot \mathcal{O}_k$.

C. Applications - Closing the Loop for Developing Visual Manipulation Policies

a) Closed-loop DAgger Training: In the traditional imitation learning settings, policy weights are not updated once they are trained and deployed. During deployment, policies often run into failures. DAgger [48] is a solution to this case where corrective data is used to train the model to adapt to failure cases. DAgger data has been shown to have much better data efficiency than plain data by previous works [29, 61]. However, collecting DAgger data is hard as it requires reproducing scene setups for the model to 're-experience' the failure case.

GSWorld provides an interface for automatic DAgger data collection in simulation. Given the GSDF of the target deployment environment and tasks with scripted policies, we roll out the policies with GSWorld. For failure recordings $\mathcal{D}_f = (s_1, \ldots, s_T)$, we randomly sample recovery states $s_r \sim \mathcal{D}_f$ with a uniform sampler, where the task is still achievable in s_r, and run the motion planner to obtain corrective data, as illustrated in Fig. 3.

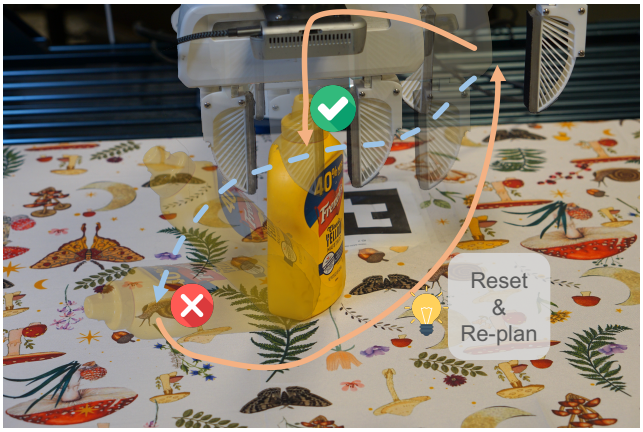

Fig. 3: **DAgger Data Collection in Simulation**. With privileged information provided by simulation, we can record and relive the failure cases during rollouts and generate corrective data for policy adaptation.

To iteratively improve zero-shot sim2real policies, we collect data, evaluate policies, record failures, and recover, resulting in an aggregated dataset $\tau_{\mathcal{S}}$:

$$\tau_{\mathcal{S}} = \Sigma_i(\mathcal{Q}_s, \mathcal{O}_s, \mathcal{A}_s)_i \tag{4}$$

, where $\mathcal{Q}_s, \mathcal{A}_s$ denote simulation robot joint positions, and action labels. $\mathcal{O}_s = I^{gs}$ is the GSWorld-rendered observations and i is the DAgger iteration.

To improve policies trained with real-world data, we first evaluate the polices and repeat the same loop, leading to dataset $\tau_{\mathcal{R}}$:

$$\tau_{\mathcal{R}} = (\mathcal{Q}_r, \mathcal{O}_r, \mathcal{A}_r) \cup \tau_{\mathcal{S}} \tag{5}$$

, where $\mathcal{Q}_r, \mathcal{O}_r, \mathcal{A}_r$ are collected in the real world by teleoperation.

b) Visual Benchmarking.: Recent VLAs [3, 33, 40] train generalist policies that work on many different robot embodiments. Learning from extensive cross-embodiment training data, these base models adapt to unseen robot hardware and language instructions in a few-shot or even zero-shot manner [3, 33]. However, due to their reliance on real robot data during training, there does not exist a standardized visual manipulation benchmark that studies the quality of the base models and their data sampling efficiency for novel embodiments.

GSWorld provides a photo-realistic rendering interface to address such an issue. With our GSDF assets, we provide photo-realistic simulation of various scenes with different robot embodiments, and a wide selection of interactable objects. We define a range of manipulation tasks using ManiSkill [51] as the simulator backend. Moreover, we use image augmentation to further reduce the visual gap between sim and real.

c) Reinforcement Learning.: RL requires massive interaction between agents and the environments, usually with parallelism [51]. To better support RL, GSWorld optimizes its implementation by only parallelizing 3DGS points that

are linked with moving parts of the scene, *i.e., robot $\mathcal{R}$ and objects $\mathcal{O}$*, and keeps other points cached with just one copy. This enables us to use a single GPU to run large parallelism to accelerate RL convergence.

IV. Experiments

This section presents an empirical evaluation of GSWorld to demonstrate its effectiveness. Our experiments are designed to address the following research questions:

- **Zero-shot Sim2real Imitation Learning.** Can GSWorld effectively bridge the sim-to-real gap to enable zero-shot policy transfer?
- **Closed-loop Policy Improvement.** Does access to a digital twin of the target deployment environment increase sampling efficiency and enable continual improvement after policy deployment via DAgger?
- **Visual Benchmarking.** Does performance in GSWorld correlate with real-world performance?
- **Virtual Teleoperation.** Does GSWorld enable simulated data collection through human teleoperation?
- **Reinforcement Learning (RL).** Can GSWorld narrow down the sim2real visual gap for visual RL?

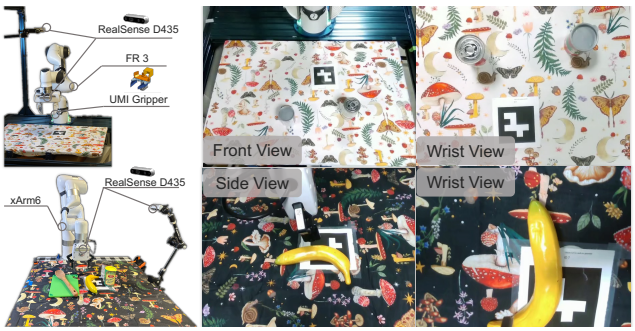

Fig. 4: **Real World Hardware Platforms.** For FR3, we set up two cameras: a third-person camera positioned in front of the robot (front view), and a wrist-mounted camera attached to the robot's end effector (wrist view). For xArm6, we set up a third-person camera besides the robot (side view) and a wrist view.

a) Hardware Platforms: Though GSWorld scales to many robot embodiments, we consider three robot platforms for evaluation: a Franka Research 3 (FR3) robot with a UMI gripper [9]; a UF xArm6 with a parallel gripper; and a bimanual Galaxea R1 robot equipped with two 6-DoF arms.

b) Experiment Protocol: In total, we evaluate GSWorld on 4 manipulation tasks on FR3 and 3 tasks on xArm. We also use R1 to demonstrate that GSWorld supports virtual simulation teleoperation to collect data. For policy implementations, we use ACT [62] and Pi0 [3] to show GSWorld to be policy-agnostic. For visual benchmarking, we intentionally train policies with varying training settings and data sizes to evaluate the performance of 'good' and 'bad' policies. Please kindly refer to our released codebase for full implementation details due to limited space.

c) Manipulation Tasks Design: We designed the following table-top manipulation tasks to evaluate performances, as shown in Fig. 5 and Fig. 6:

Fig. 5: **FR3 Task Visualizations.** We design 4 real-world robotic tasks that involve distinct manipulation skills and diverse objects on FR3.

Fig. 6: **xArm Task Visualizations.** We design 3 manipulation tasks on xArm6.

- Place Box. The bottle and the box are initialized randomly within a $45\,\text{cm} \times 45\,\text{cm}$ area. FR3 must pick up the bottle and place it onto the box.
- Pour Sauce. The mustard bottle and the bread slice are put within a $45\,\text{cm} \times 45\,\text{cm}$ area. FR3 is required to pour the sauce onto the bread slice.
- Stack Cans. Two cans are randomly placed within a $45\,\text{cm} \times 45\,\text{cm}$ area. FR3 must stack them.
- Arrange Cans. Two cans are randomly placed within a $20\,\text{cm} \times 15\,\text{cm}$ area. A rack is randomly placed beside them. FR3 is tasked with placing one object on the rack, followed by placing the second object next to it.
- Align Cans. xArm needs to grasp one can and put it next to the other can.
- Grasp Banana. xArm should grasp the banana and rotate it by 30 to 60 degrees along the z-axis.
- Tidy Table. xArm is required to pick up the kitchen spoon and place it onto the cutting board to clean the table.

A. Zero-shot Sim2real Imitation Learning

Fig. 7 shows the performances of policies trained with only simulation data. We leverage the MPlib motion planner [51], utilizing predefined motions and poses. For each task, we collect 100 trajectories per iteration and train on the sum of all generated data. We can safely conclude that GSWorld allows zero-shot sim2real policy transfer and shows promising success rates. More details are included in section IV-B.

B. Closed-loop DAgger for Continuous Policy Improvement

The data collection uses same method as the zero-shot sim2real learning. For sim2real, we collect 100 expert trajectories for the initial iteration of DAgger training. In subsequent iterations, we evaluate the current policy and identify all failed trajectories. For each failure, we reset the environment to a preceding state where the task remains solvable, and collect additional corrective data using the motion planner starting from that state. This DAgger data collection process is repeated for four additional iterations, with each iteration generating 100 expert trajectories. As shown in Figure 7, our DAgger-based approach leads to significant performance improvements across all four tasks, compared to training from scratch (TfS) using only a supervised imitation learning objective.

DAgger can also be applied to improve real-world policies. Instead of starting from a randomly initialized policy, we can do real2sim2real DAgger learning. First we train a real-world

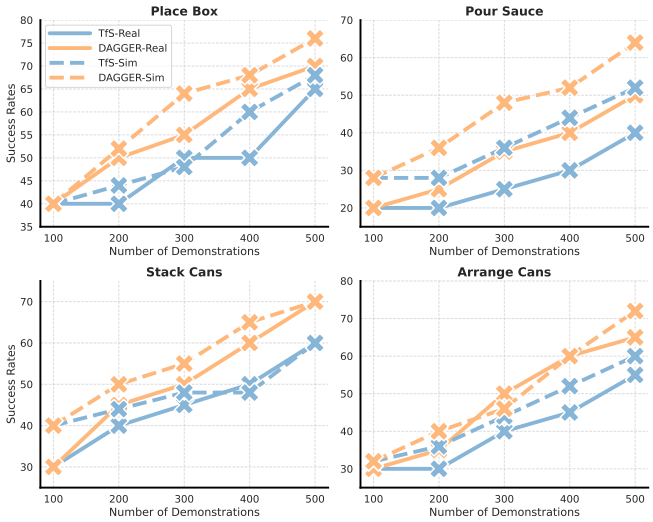

Fig. 7: **Closed-loop Sim2real DAgger Training on FR3.** Policies are trained with sim data and deployed in both sim and real environments. DAgger consistently improves policy performances and outperforms training from scratch.

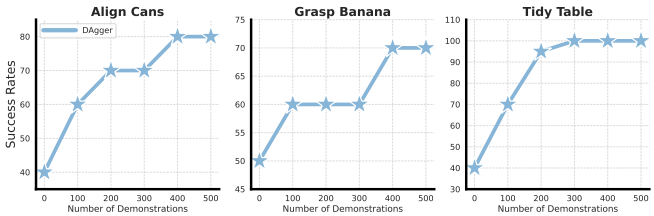

Fig. 8: **Closed-loop Real2sim2real DAgger on xArm.** DAgger can also be used to improve real-world policies.

ACT with a few demonstrations. Then we start from this checkpoint and do DAgger in GSWorld. Fig 8 shows that GSWorld can improve performance after real-world policy deployment.

These results highlight the critical role of closed-loop DAgger Training. Leveraging our photo-realistic digital twin, researchers can collect essential corrective data that would be extremely difficult to obtain in the real world, primarily due to the challenges of precisely resetting objects to their pre-failure states.

C. Benchmarking

In Fig. 9, we observe a strong correlation between simulation and real-world performance across all evaluated tasks and different policy architectures. This correlation indicates that GSWorld can reliably predict real-world outcomes without requiring physical deployment of policies in the real world scene. As shown in Table I, higher simulated performance consistently corresponds to higher success rates in real-world experiments. By leveraging the photorealistic rendering capabilities of 3DGS [26], we establish a benchmarking framework that reflects real-world behaviors.

D. Virtual Teleoperation

Simulation data can be used to scale up robot policy learning [20, 58]. We show that through mouse and keyboard,

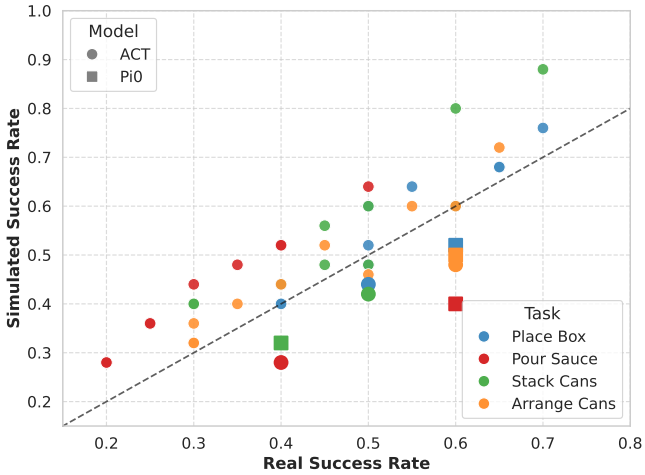

Fig. 9: **Visual Benchmarking with FR3**. We roll out various policies in both real and sim. We intentionally use different policies and data sizes to show positive correlations of sim-real performance regardless of the quality of the policies.

	ACT [62]		**Pi0** [3]	
	real	sim	real	sim
Place Box	50.0%	44.0%	60.0%	52.0%
Pour Sauce	40.0%	28.0%	60.0%	40.0%
Stack Cans	50.0%	42.0%	40.0%	32.0%
Arrange Cans	60.0%	48.0%	60.0%	50.0%
Avg.	50.0%	41.0%	55.0%	43.5%

TABLE I: **Visual Benchmarking of Real-world Policies on FR3.** We show the evaluation performance in both sim and real of policies trained with only real-world data.

we can collect teleoperation data in the simulation with real renderings, as illustrated in Fig. 10.

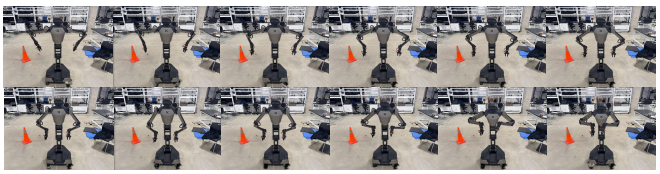

Fig. 10: **Galaxea R1 Virtual Teleoperation**. We use a keyboard to teleop R1 and render photo-realistic videos.

E. Visual Reinforcement Learning

GSWorld is designed to support parallel environments so that we can use this to learn visual RL policies, which has great potential in robot learning [20, 41]. We trained asymmetric SAC [16] with GSWorld, where the critic sees simulation-privileged information and the actor only uses robot joint position. As we aim to show that GSWorld can reduce the visual gap for RL instead of acquiring good sim2real VRL policies, we train with no domain randomization except for color jittering for efficiency. We only use the third-person view since wrist camera shows significant gaps during RL exploration. Our real-world success rates for *Grasp Banana* and *Tidy Table* are 30% and 20%, while baseline ManiSkill

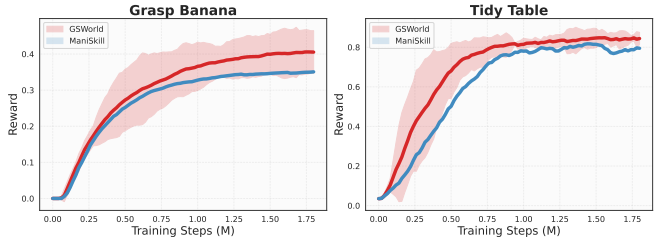

Fig. 11: **SAC**. Results are aggregated over 3 runs. Directly training from ManiSkill is also plotted for comparison.

reaches 0% and 5%, respectively. Training results are shown in Fig. 11.

V. Conclusion

In this work, we present a pipeline for constructing a photorealistic digital twin that delivers highly correlated performance metrics between simulation and real-world deployments across different policy architectures and embodiments. Furthermore, we demonstrate that our environment can efficiently collect corrective data at scale, enabling more effective policy training.

References

[1] Jad Abou-Chakra et al. "Physically Embodied Gaussian Splatting: A Visually Learnt and Physically Grounded 3D Representation for Robotics". In: *CoRL*. 2024.

[2] William Agnew et al. "Amodal 3d reconstruction for robotic manipulation via stability and connectivity". In: *CoRL*. 2021.

[3] Kevin Black et al. "π_0: A Vision-Language-Action Flow Model for General Robot Control". In: *arXiv preprint arXiv:2410.24164* (2024).

[4] Berk Calli et al. "Yale-CMU-Berkeley dataset for robotic manipulation research". In: *The International Journal of Robotics Research* 36.3 (2017), pp. 261–268.

[5] Tianxing Chen et al. "RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation". In: *arXiv preprint arXiv:2506.18088* (2025).

[6] Ziyu Chen et al. "Omnire: Omni urban scene reconstruction". In: *arXiv preprint arXiv:2408.16760* (2024).

[7] Zoey Chen et al. "Urdformer: A pipeline for constructing articulated simulation environments from real-world images". In: *RSS*. 2024.

[8] Xuxin Cheng et al. "Open-television: Teleoperation with immersive active visual feedback". In: *arXiv preprint arXiv:2407.01512* (2024).

[9] Cheng Chi et al. "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots". In: *arXiv preprint arXiv:2402.10329* (2024).

[10] Erwin Coumans and Yunfei Bai. *Pybullet, a python module for physics simulation for games, robotics and machine learning*. 2016.

[11] Tianyuan Dai et al. "Automated creation of digital cousins for robust policy learning". In: *CoRL*. 2024.

[12] Zhao Dong et al. "Digital Twin Catalog: A Large-Scale Photorealistic 3D Object Digital Twin Dataset". In: *arXiv preprint arXiv:2504.08541* (2025).

[13] Tom Erez, Yuval Tassa, and Emanuel Todorov. "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx". In: *ICRA*. 2015.

[14] Sergio Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion". In: *Pattern Recognition* (2014).

[15] Haoran Geng et al. "RoboVerse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning". In: *arXiv preprint arXiv:2504.18904* (2025).

[16] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. Pmlr. 2018, pp. 1861–1870.

[17] Xiaoshen Han et al. "Re^{3}Sim: Generating High-Fidelity Simulation Data via 3D-Photorealistic Real-to-Sim for Robotic Manipulation". In: *arXiv preprint arXiv:2502.08645* (2025).

[18] Yuanming Hu et al. "Taichi: a language for high-performance computation on spatially sparse data structures". In: *ACM Transactions on Graphics (TOG)* (2019).

[19] Binbin Huang et al. "2d gaussian splatting for geometrically accurate radiance fields". In: *ACM SIGGRAPH*. 2024.

[20] Tao Huang et al. "Diffusion reward: Learning rewards via conditional video diffusion". In: *European Conference on Computer Vision*. Springer. 2024, pp. 478–495.

[21] Clément Jambon et al. "NeRFshop: Interactive Editing of Neural Radiance Fields". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* (2023).

[22] Stephen James et al. "Rlbench: The robot learning benchmark & learning environment". In: *IEEE Robotics and Automation Letters* (2020).

[23] Mazeyu Ji et al. "Graspsplats: Efficient manipulation with 3d feature splatting". In: *CoRL*. 2024.

[24] Guangqi Jiang et al. "Robots Pre-Train Robots: Manipulation-Centric Robotic Representation from Large-Scale Robot Datasets". In: *arXiv preprint arXiv:2410.22325* (2024).

[25] Hanxiao Jiang et al. "Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos". In: *arXiv preprint arXiv:2503.17973* (2025).

[26] Bernhard Kerbl et al. "3d gaussian splatting for real-time radiance field rendering." In: *ACM Trans. Graph.* 42.4 (2023), pp. 139–1.

[27] Xinhai Li et al. "RoboGSim: A Real2Sim2Real Robotic Gaussian Splatting Simulator". In: *arXiv preprint arXiv:2411.11839* (2024).

[28] Xuanlin Li et al. "Evaluating Real-World Robot Manipulation Policies in Simulation". In: *arXiv preprint arXiv:2405.05941* (2024).

[29] Huihan Liu et al. "Robot learning on the job: Human-in-the-loop autonomy and learning during deployment". In: *The International Journal of Robotics Research* (2022).

[30] Ruoshi Liu et al. "Zero-1-to-3: Zero-shot one image to 3d object". In: *ICCV*. 2023.

[31] Ruoshi Liu et al. "Differentiable robot rendering". In: *arXiv preprint arXiv:2410.13851* (2024).

[32] Shilong Liu et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection". In: *ECCV*. 2024.

[33] Songming Liu et al. "RDT-1B: a Diffusion Foundation Model for Bimanual Manipulation". In: *arXiv preprint arXiv:2410.07864* (2024).

[34] Haozhe Lou et al. *Robo-GS: A Physics Consistent Spatial-Temporal Model for Robotic Arm with Hybrid Representation*. 2024.

[35] Guanxing Lu et al. "Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation". In: *ECCV*. 2024.

[36] Zhao Mandi et al. "Real2code: Reconstruct articulated objects via code generation". In: *arXiv preprint arXiv:2406.08474* (2024).

[37] Ben Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[38] Yao Mu et al. "RoboTwin: Dual-Arm Robot Benchmark with Generative Digital Twins". In: *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2025, pp. 27649–27660.

[39] Soroush Nasiriany et al. "Robocasa: Large-scale simulation of everyday tasks for generalist robots". In: *arXiv preprint arXiv:2406.02523* (2024).

[40] Octo Model Team et al. "Octo: An Open-Source Generalist Robot Policy". In: *Proceedings of Robotics: Science and Systems*. 2024.

[41] Guoping Pan et al. "RoboDuet: Learning a Cooperative Policy for Whole-body Legged Loco-Manipulation". In: *IEEE Robotics and Automation Letters* (2025).

[42] Nicholas Pfaff et al. "Scalable real2sim: Physics-aware asset generation via robotic pick-and-place setups". In: *arXiv preprint arXiv:2503.00370* (2025).

[43] Ri-Zhao Qiu et al. "Language-driven physics-based scene synthesis and editing via feature splatting". In: *ECCV*. 2024.

[44] Delin Qu et al. "Livescene: Language embedding interactive radiance fields for physical scene rendering and control". In: *arXiv preprint arXiv:2406.16038* (2024).

[45] M Nomaan Qureshi et al. "Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting". In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2025, pp. 6502–6509.

[46] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *ICML*. 2021.

[47] Eric Rohmer, Surya PN Singh, and Marc Freese. "V-REP: A versatile and scalable robot simulation framework". In: *IROS*. 2013.

[48] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011.

[49] Johannes L Schonberger and Jan-Michael Frahm. "Structure-from-motion revisited". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4104–4113.

[50] Deborah Sulsky, Zhen Chen, and Howard L Schreyer. "A particle method for history-dependent materials". In: *Computer methods in applied mechanics and engineering* (1994).

[51] Stone Tao et al. "ManiSkill3: GPU Parallelized Robotics Simulation and Rendering for Generalizable Embodied AI". In: *arXiv preprint arXiv:2410.00425* (2024).

[52] Marcel Torne et al. "Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation". In: *arXiv preprint arXiv:2403.03949* (2024).

[53] Yijia Weng et al. "Neural Implicit Representation for Building Digital Twins of Unknown Articulated Objects". In: *CVPR*. 2024.

[54] Turner Whitted. "An improved illumination model for shaded display". In: *Proceedings of the 6th annual conference on Computer graphics and interactive techniques*. 1979, p. 14.

[55] Tianhao Wu et al. "Amodal3R: Amodal 3D Reconstruction from Occluded 2D Images". In: *arXiv preprint arXiv:2503.13439* (2025).

[56] Fanbo Xiang et al. "Sapien: A simulated part-based interactive environment". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11097–11107.

[57] Rui Yan et al. "ACE-F: A Cross Embodiment Foldable System with Force Feedback for Dexterous Teleoperation". In: *1st Workshop on Robot Hardware-Aware Intelligence*. 2025.

[58] Ruihan Yang et al. *EgoVLA: Learning Vision-Language-Action Models from Egocentric Human Videos*. 2025.

[59] Sizhe Yang et al. "Novel Demonstration Generation with Gaussian Splatting Enables Robust One-Shot

Manipulation". In: *arXiv preprint arXiv:2504.13175* (2025).

[60] Kevin Zakka et al. "MuJoCo Playground". In: *arXiv preprint arXiv:2502.08844* (2025).

[61] Xiaoyu Zhang et al. "Diffusion meets dagger: Supercharging eye-in-hand imitation learning". In: *RSS*. 2024.

[62] Tony Z Zhao et al. "Learning fine-grained bimanual manipulation with low-cost hardware". In: *arXiv preprint arXiv:2304.13705* (2023).

APPENDIX

A. Neural Rendering Background

Point and surface splatting methods represent a scene explicitly via a mixture of 2D or 3D Gaussian ellipsoid. In the case of Gaussian Splatting, the geometry is represented as a collection of 3D Gaussian, each being the tuple $\{\mathcal{X}, \mathbf{\Sigma}\}$ where $\mathcal{X} \in \mathbb{R}^3$ is the centroid of the Gaussian and Σ is its covariance matrix in the world frame. This gives the probability density function

$$G(\mathcal{X}, \Sigma) = \exp -\frac{1}{2}\mathcal{X}^\top \Sigma^{-1} \mathcal{X}\,. \tag{6}$$

Gaussian splatting decomposes it into a scaling matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$ via $\Sigma = \mathbf{RSS}^\top\mathbf{R}^\top$. The color information in the texture is encoded with a spherical harmonics map $\mathbf{c}_i = \mathrm{SH}_\phi(\mathbf{d}_i)$, which is conditioned on the viewing direction ϕ.

To optimize for features, existing methods tend to append an additional vector $\mathbf{f}_i \in \mathbb{R}^d$ to each Gaussian, which is rendered in a view-independent manner because the semantics of an object shall remain the same regardless of view directions. The rasterization procedure starts with culling the mixture by removing points that lay outside the camera frustum. The remaining Gaussians are projected to the image plane according to the projection matrix $\mathbf{W}$ of the camera, which is then sorted from low to high using the distance from the virtual camera origin. This projection also induces the following transformation on the covariance matrix Σ:

$$\Sigma^{'} = \mathbf{J}\,\mathbf{W}\,\Sigma\,\mathbf{W}^\top\mathbf{J}^\top\,, \tag{7}$$

where $\mathbf{J}$ is the Jacobian of the projection matrix $\mathbf{W}$. We can then render both the color and the visual features with the splatting algorithm:

$$\{\hat{\mathbf{F}}, \hat{\mathbf{C}}\} = \sum_{i \in N} \{\mathbf{f}_i, \mathbf{c}_i\} \cdot \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)\,, \tag{8}$$

where α_i is the opacity of the Gaussian conditioned on $\Sigma^{'}$ and the indices $i \in N$ are in the ascending order determined by their distance to the camera origin.

Following the convention [43], GSWorld assumes that per-gaussian feature vector $\mathbf{f}_i$ is isotropic. The rendered depth, images, and features are then supervised using L2 loss.

B. More Experimental Details

a) Simulation Evaluation: We evaluated each task across 25 random seeds in simulation. At the initiation of each episode, objects were randomly configured. A partial success criterion was established, awarding half credit for completing intermediate steps (e.g., successfully picking up an object or placing one can on the rack). Episodes for the *placing box*, *pouring sauce*, and *stacking cans* tasks were limited to 500 time steps, while *arranging cans* was allocated 800 time steps due to its extended task horizon. The control framework employed joint position control. The policy architecture accepted dual visual image inputs and the robot's proprioceptive data, subsequently generating target joint positions as action outputs.

b) Real-world Evaluation: Real-world validation consisted of 10 experimental trials per policy, each with randomized initial conditions. Consistent with the simulation protocol, partial success metrics were implemented. The observational framework maintained parity with the simulation environment, utilizing visual inputs from two RealSense D435i cameras.

c) Policy Learning Implementation: In our paper we use two policy architectures, i.e., ACT [62] and PI0 [3]. For ACT, we adopted the original PyTorch implementation but replaced the visual backbone from ResNet-18 to DinoV2 ViT-S, similar to OpenTV [8]. We used 42 batch size, an SGD optimizer with a constant 0.0001 learning rate for 50 000 iterations to train the policy. The rest of the training hyperparameters are consistent with the original ACT paper. For the PI0 [3] model, we adopted the implementation on LeRobot. We used the publicly available PI0-base model weights for initialization. For the training, we freeze the visual and language backbone and train only the action expert. The optimization was done with an SGD optimizer with a constant 0.0001 learning rate for 50 000 iterations with a batch size of 10.

d) DAGGER Detailed Performance: In this section we provide the numerical results in Figure 7, which is shown in Table II.

C. Gaussian Splatting Training

a) Implementation: Our pipeline is compatible with any kind of gaussian splitting implementation. For simplicity, we use the official one [26] and their default code implementation without modifying anything. We take around 100 and 300 pictures for object and robot reconstruction, respectively. The model is trained with points initialized with Colmap [49], and saved at iteration 7000 and 30000.

D. GSWorld Data Collection

a) Simulation: We primarily rely on motion planning for data collection due to its fully automated nature. The task is decomposed into several subtasks, each associated with a specific motion. The robot executes these motions using MPlib.

Our framework also supports data collection via teleoperation in simulation. We employ a built-in click-and-drag teleoperation system provided by ManiSkill [51], which is operated using a keyboard and mouse. This system allows users to define keyframes and utilize a motion planner to achieve the desired robot pose.

b) Real World: For real-world data collection, we utilize a VR-based teleoperation system. The interface includes an HTC Vive controller and base station, which track 6-DOF hand movements that are mapped to the robot arm's end-effector pose. Integration with the HTC Vive is achieved using the triad-openvr package in conjunction with SteamVR. A position-velocity controller is implemented to ensure accurate tracking of the controller's transmitted poses. This setup enables efficient collection of demonstration trajectories, including both state and observation data. For the xArm6 platform, we use ACE-F [57].

		Place Box					Pour Sauce				
		Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5
sim	train from scratch	40%	44%	48%	60%	68%	28%	28%	36%	44%	52%
	dagger	40%	52%	64%	68%	76%	28%	36%	48%	52%	64%
real	train from scratch	40%	40%	50%	50%	65%	20%	20%	25%	30%	40%
	dagger	40%	50%	55%	65%	70%	20%	25%	35%	40%	50%

		Stack Cans					Arrange Cans				
		Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5
sim	train from scratch	40%	44%	48%	48%	60%	32%	36%	44%	52%	60%
	dagger	40%	56%	60%	80%	88%	32%	40%	46%	60%	72%
real	train from scratch	30%	40%	45%	50%	60%	30%	30%	40%	45%	55%
	dagger	30%	45%	50%	60%	70%	30%	35%	50%	60%	65%

		Avg.				
		Iter 1	Iter 2	Iter 3	Iter 4	Iter 5
sim	train from scratch	35%	38%	44%	51%	60%
	dagger	35%	46%	54.5%	65%	75%
real	train from scratch	30%	32.5%	40%	43.75%	55%
	dagger	30%	38.75%	47.5%	56.25%	63.75%

TABLE II: Performance comparison between ACT and Pi0 methods with different training approaches (train from scratch vs. dagger) across various manipulation tasks and iterations (success rates in %).

E. GSWorld Usage example

Our pipeline is highly compatible with the gym interface. GSWorld can be enabled by as few as one line of code, which add a wrapper layer on any existing environment. Below we show a code example to use GSWorld.

```
env = gym.make(
   env_id,
   robot_uids=args.robot_uid,
   obs_mode=args.obs_mode,
   control_mode=args.control_mode,
   render_mode=args.render_mode,
   reward_mode="dense",
   human_render_camera_configs=dict(
       ↪ shader_pack=args.shader),
   viewer_camera_configs=dict(shader_pack=
       ↪ args.shader),
   max_episode_steps=args.ep_len,
)

env = GSWorldWrapper(
   env=env,
   gs_cfg=args.gs_cfg,
   device="cuda",
)

# Then treat the env as the gym env.
# Pixel observations are rendered with 3
    ↪ DGS.
obs, _ = env.reset()
obs, reward, terminated, truncated, info =
    ↪  env.step(action)
```