

RoboRender: Robot-Oriented Video Generation for Visual Sim-to-Real Transfer

Huang Huang^{1*}, Wensi Ai^{1*}, Ziyu Chen^{1*}, Youhui Wang^{1*},
Zijian Du², Yang Liu³, Jiaolong Yang³, Li Fei-Fei¹, Jiajun Wu¹

¹Stanford University

²Nvidia

³Microsoft Research

*Equal Contribution

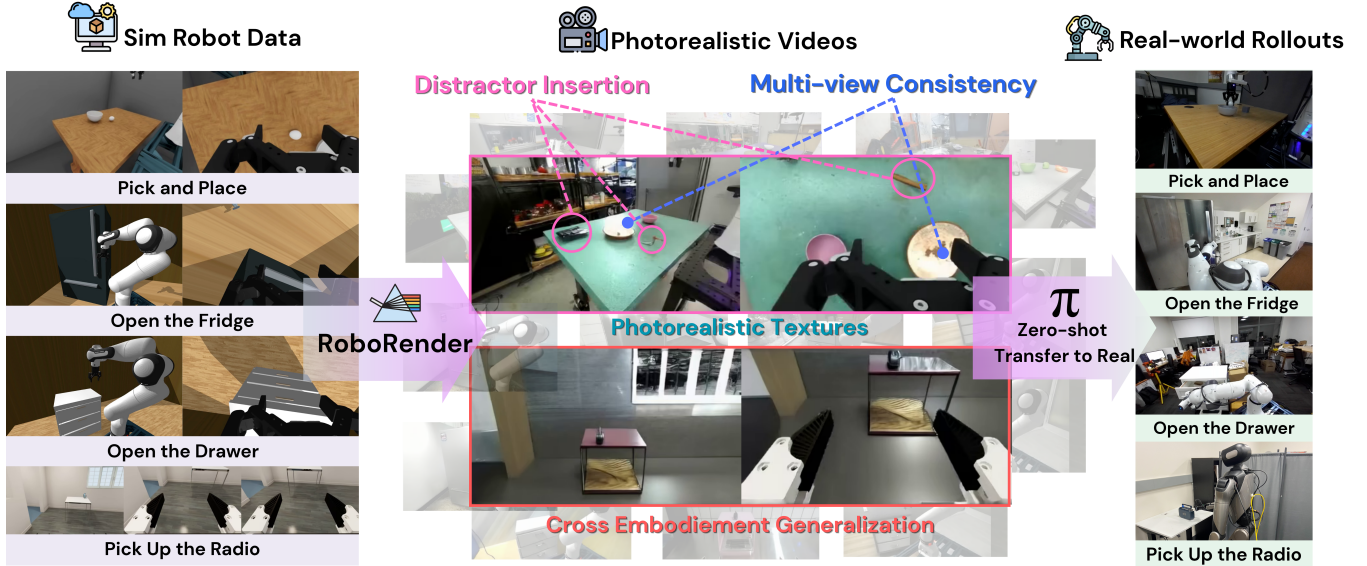


Fig. 1: **Overview.** From simulation trajectories (**Left**), RoboRender generates **diverse** and **photorealistic** videos oriented for visual sim-to-real robot policy transfer (**Middle**). These videos preserve simulator geometry and robot motion while supporting **multi-view consistency** across different robot camera views, **distractor insertion** for robustness, and **cross-embodiment generation** for unseen robot embodiments. Policies trained on RoboRender-generated videos achieve strong zero-shot sim-to-real performance across pick-and-place, articulated-object manipulation, and mobile manipulation tasks (**Right**), demonstrating that RoboRender effectively mitigates the visual sim-to-real gap.

Abstract—Simulation enables large-scale, low-cost robot data generation, but policies trained in simulation often fail to transfer to the real world due to the sim-to-real visual discrepancies. Existing approaches often rely on intermediate representations, which can discard rich semantic information or require additional perception modules at deployment. We address this visual sim-to-real gap with RoboRender, a framework that converts simulated trajectories into photorealistic RGB videos for policy learning. RoboRender trains a robot-oriented video generation model conditioned on simulated depth videos, language instructions, and robot RGB mask videos, preserving simulator geometry, robot motion, and action labels while synthesizing realistic textures, backgrounds, and distractors. The generated RGB videos are paired with simulator-provided states and actions to train policies for zero-shot real-world deployment. On robot video test sets, our video model outperforms depth-conditioned video generation baselines in generation quality. In real-world experiments across pick-and-place, articulated-object manipulation, and mobile manipulation tasks, policies trained on RoboRender-generated data achieve a 71% average success rate, outperforming raw simulation renderings and conventional visual domain randomization by approximately $7.1\times$ and $3.6\times$, respectively. We further show that policy performance improves with more generated videos per simulation trajectory, increasing

opening-task success by 65 percentage points. These results demonstrate that generative video rendering mitigates the visual sim-to-real gap for zero-shot policy transfer. Project website: <https://robo-render.github.io/>.

I. INTRODUCTION

Simulation provides privileged state information, supports large-scale parallel data generation, and enables low-cost and automatic environment reset, making it effective for robot policy learning. However, transferring simulation-trained policies to the real world remains challenging due to the sim-to-real gap, which includes both dynamics and visual discrepancies. For vision-based manipulation, the visual gap is particularly limiting, as policies must act from observations whose appearance can differ substantially between rendered and real scenes. Existing approaches reduce this gap by reconstructing realistic digital twins [1]–[4], replacing RGB with depth or object states [5]–[7], or applying visual domain randomization [8], [9], but can be scene-specific, semantically lossy, require additional perception modules, or remain visually mismatched to real observations.

Video generation models encode rich priors over real-world appearance, semantics, and temporal dynamics. In light of this, we propose **RoboRender** (Fig. 1), a framework that renders simulated trajectories into photorealistic RGB-action data for zero-shot visual sim-to-real policy transfer. At its core is a robot-oriented video generation model conditioned on simulated depth videos, language instructions, and robot RGB masks. Depth preserves scene geometry and object motion across views, enabling multi-view-consistent generation; language controls appearance, materials, lighting, and background semantics; and robot masks preserve robot motion and support generation for unseen embodiments. Together, these conditions allow RoboRender to synthesize rich, diverse and realistic textures, backgrounds, and distractors while retaining simulator geometry and action labels, making it efficient for generating visually diverse and physically grounded data. Policies are then trained on the resulting photorealistic RGB-action trajectories and deployed zero-shot in the real world.

We evaluate RoboRender on five pick-and-place tasks, two opening tasks, and one mobile manipulation task. Compared with policies trained on raw simulator RGB and with visual domain randomization, policies trained on RoboRender-generated data achieve $7.1\times$ and $3.6\times$ higher real-world success rates, respectively. We further show that generating multiple visually diverse videos from each simulated trajectory with different language prompts improves performance by up to **65** percentage points on opening tasks and **10** points on pick-and-place tasks. These results demonstrate the effectiveness of RoboRender in mitigating the visual sim-to-real gap for policy transfer.

In summary, our contributions are:

- We introduce RoboRender, a framework that converts simulated trajectories into photorealistic RGB demonstrations for zero-shot visual sim-to-real policy transfer.
- We develop a robot-oriented video generation model conditioned on depth, language, and robot masks for grounded, controllable, and embodiment-aware video synthesis.
- Real-robot experiments indicate that RoboRender enables significantly stronger zero-shot sim-to-real transfer than policies trained on raw simulation or conventional visual domain randomization, across pick-and-place, articulated-object manipulation, and mobile manipulation tasks.

II. RELATED WORK

A. Visual Sim-to-Real Transfer

Simulation enables scalable robot data generation, but policies trained in simulation often fail to transfer due to dynamics and visual discrepancies. While dynamics gaps can be mitigated through system identification, dynamics randomization, or adaptation [10], [11], vision-based manipulation remains particularly sensitive to the visual gap between simulation and real observations. Prior work addresses this gap through three main strategies. Real2sim2real methods reconstruct photorealistic digital twins using neural rendering

or 3D Gaussian Splatting [1]–[4], but require per-scene capture, calibration, or asset preparation, and may struggle with accurate interactive geometry [2], [3]. Geometry- or object-centric methods avoid sim-to-real RGB gap by using depth, point clouds, object poses, or object states [5]–[7], [12], but can discard semantic appearance cues and require additional perception modules. Visual domain randomization trains policies on randomized textures, lighting, cameras, and backgrounds [8], [9], [13], but often requires careful tuning and may remain far from real observations. In contrast, RoboRender retains the simulator as the source of geometry, dynamics, and action labels, while using a generative video prior to render simulated depth trajectories into photorealistic RGB videos. This approach combines the scalability and physical grounding of simulation with the rich semantic and visual priors of video generation models.

B. Video Generation for Robot Data Synthesis

Video generation models have emerged as synthetic data engines for robot learning. DreamGen [14] generates robot videos from language-conditioned world models, but language alone provides limited physical grounding, and actions must be inferred through inverse dynamics or latent action models. RoboCurate [15] improves generation quality through simulator-replay filtering; RoboRender instead starts from simulator rollouts with known actions, preserving grounded dynamics and action labels. Other works add grounding through robot motion or simulator-derived cues: AnchorDream [16] conditions on language and robot motion videos but lacks grounded robot-object interactions, while EmbodieDreamer [17] and CRAFT [18] translate simulated observations using object masks or Canny edges, which provide 2D structure but lack explicit 3D information and multi-view consistency. RoboRender instead conditions on multi-view depth videos, preserving 3D scene structure for physically grounded, multi-view-consistent generation. RoboTransfer [19] also generates multi-view-consistent videos using depth, surface normals, and appearance exemplars such as background and object images, but reliance on pre-collected photorealistic conditions limits flexibility. RoboRender uses language to control textures, materials, lighting, and background semantics, enabling diverse synthesis without appearance exemplars. Compared with GigaWorld-0 [20], which studies video-model-based robot data generation across appearance, view, human-to-robot, and sim-to-real transfer, RoboRender focuses on visual sim-to-real transfer from simulated depth trajectories and supports distractor insertion and unseen-embodiment generalization through robot-mask conditioning. Overall, RoboRender complements prior work with capabilities specialized for robot learning: physically grounded dynamics, simulator-provided action labels, multi-view consistency, distractor insertion, and embodiment generalization, helping mitigate the visual sim-to-real gap.

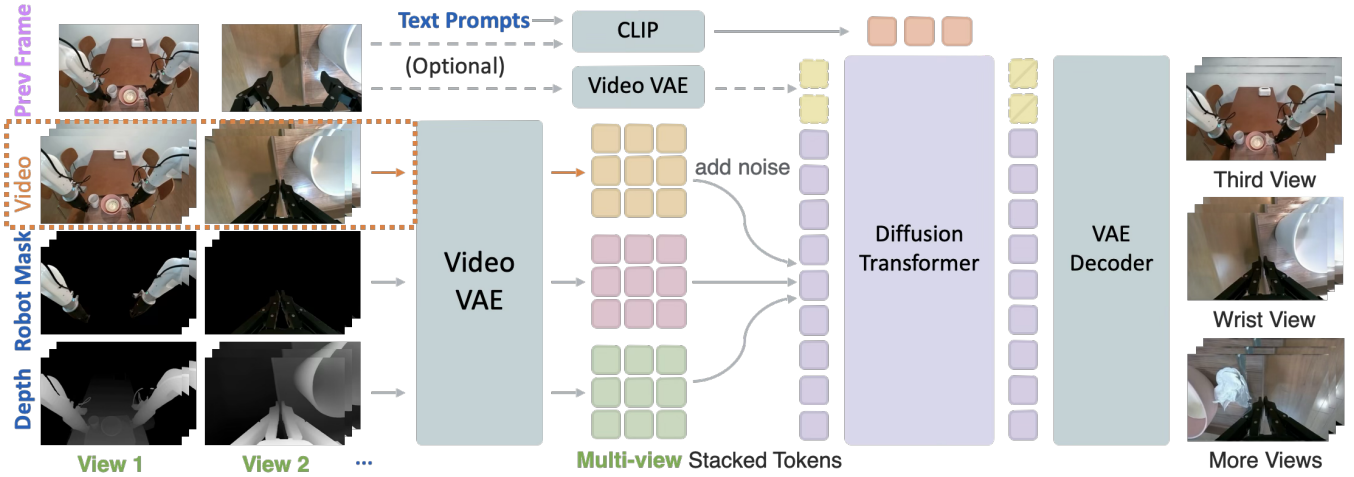


Fig. 2: **RoboRender video generation model.** During training, raw RGB videos are VAE-encoded and noised to form the diffusion input. Per-view depth videos and robot-arm RGB mask videos are encoded as spatial conditions, while CLIP-encoded language prompts and previous-frame inputs provide semantic and temporal conditioning. The per-view condition latents are concatenated with the noised latents and stacked into a unified multi-view latent, which is denoised and decoded into RGB videos. At inference, ground-truth RGB videos (orange) are omitted and the diffusion transformer samples from noise conditioned on depth, robot-arm RGB, language, and previous frames.

III. METHOD

A. Video Generation Model

Model Architectures. RGB videos for robot learning must be photorealistic, physically grounded, cross-view consistent, and action-aligned. Existing text-to-video models [21], [22] provide strong visual priors but often fail to capture accurate causal interactions [23]. Depth-conditioned variants [24], [25] improve geometric control, yet usually generate views independently and lack robot-specific manipulation priors. We therefore adapt a variant of Wan2.1-T2V-1.3B [21] into a robot-oriented video generation model for robot visual sim-to-real transfer. As shown in Fig. 2, the model is conditioned on depth videos for scene geometry and object motion, robot-mask RGB videos for embodiment and robot motion, language prompts for appearance and scene semantics, and any available previous frame for autoregressive long-horizon generation. During training, the depth and robot-mask videos are VAE-encoded and channel-wise concatenated with the noised latent of the ground-truth RGB video, then stacked across views into a unified multi-view latent. A diffusion transformer then denoises this latent to synthesize photorealistic, multi-view-consistent videos while preserving simulator-provided structure and action labels for policy learning. We fine-tune the model with LoRA at a global batch size of 16 for 12K iterations on ~ 130 K real-world data. Wan2.1-T2V-1.3B is a bidirectional diffusion model with a fixed 81-frame window, whereas robot demonstrations are variable-length and often longer. For long-horizon synthesis, we condition the model on the previous frame (purple in Fig. 2) during training [22], [26]. At inference, it generates autoregressively, conditioning each chunk on the previous chunk’s last frame. We randomly drop this condition during training, enabling first-chunk synthesis without a previous

frame and autoregressive continuation with one. During training, we additionally mask large random depth patches in each view, so the model learns to synthesize plausible, prompt-guided appearance where depth is absent rather than merely reproducing the input geometry. At inference, this enables a controllable scene edit: by dropping depth over a target region across all views and prompting for a new object, the model inserts a distractor that remains consistent across views. Used as data augmentation, these distractors make the downstream policy more robust to object clutter. We show examples of these inserted distractors in Fig. 1.

Training Data Curation. Video-model training requires large-scale paired conditioning signals and target videos. However, existing robotics datasets [27], [28] are primarily designed for policy learning, providing RGB observations and actions but not the conditions required by our model. We therefore build a scalable preprocessing pipeline that converts real-robot datasets into training-ready samples. For each clip, we extract a temporally aligned 81-frame window across all available views and generate four aligned conditions: a scene-description prompt, per-view depth videos, per-view robot-arm RGB masks, and a pre-window frame for temporal anchoring. Given each real-robot clip, we generate scene prompts with Qwen2-VL [29] from the first frame of each view, estimate depth using Video-Depth-Anything [30] and FoundationStereo [31] when stereo is available, and extract robot-arm masks with SAM3 [32]. Further preprocessing details are provided in Appendix A. Our final video-model training set contains ~ 130 k real-world robot manipulation samples: ~ 70 k from AgiBot-World-Alpha [27] and ~ 60 k from DROID [28], processed at 416×240 resolution.

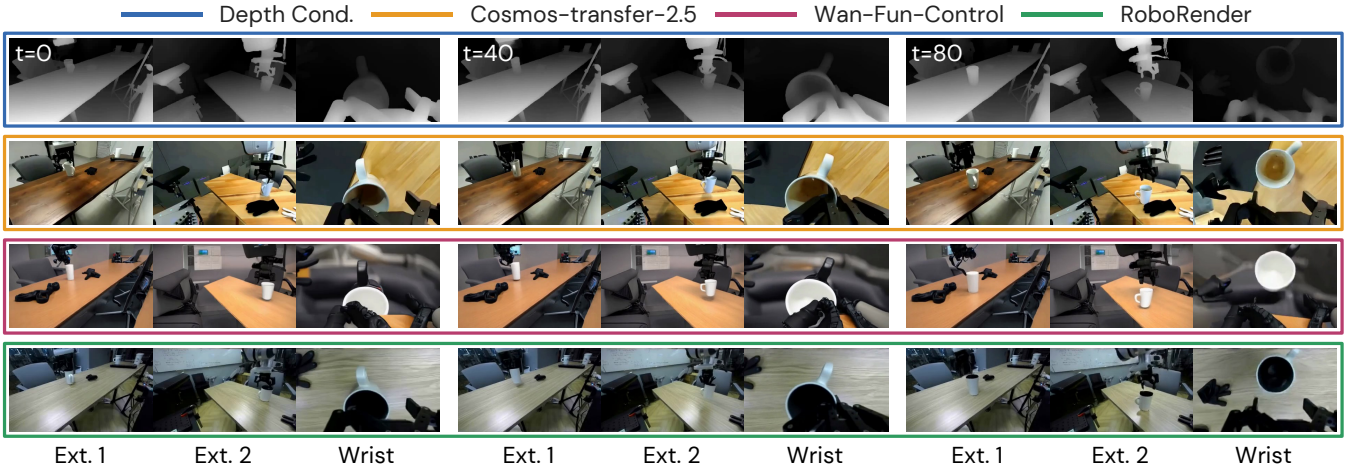


Fig. 3: **Visual comparison of video generation.** Given depth conditions, we compare RGB videos generated by RoboRender against the baselines Cosmos-Transfer2.5 [25] and Wan-Fun-Control [24] across time steps and views. RoboRender produces realistic video quality while preserving object consistency across views. See Appendix B for more visual results.

B. Robot Simulation Data Generation

We generate simulation trajectories using a pipeline built on MolmoSpaces [33] and CuRobo [34]. For each episode, we instantiate a predefined simulation scene with target objects, a robot, and synchronized multi-view cameras. A scripted task-and-motion planner executes the task while recording multi-view RGB-D observations, proprioception, actions, and robot segmentation masks. The planner uses offline grasp libraries for rigid and articulated objects, with precomputed antipodal end-effector poses for a parallel-jaw gripper. Grasps are selected by ranking [33], collision filtering, and IK feasibility checks. Once a feasible grasp is found, the planner executes task-specific primitives: pregrasp–grasp–lift–place–retreat for pick-and-place, constrained handle motion for articulated objects following an arc for hinge joints or a line for prismatic joints, and navigation–pregrasp–grasp–lift for mobile manipulation. If execution fails, the planner replans with another grasp. Episodes are retained only if object placement succeeds or the target joint displacement exceeds a task-defined threshold; otherwise, they terminate after repeated failures or a 300-step timeout. Appendix E provides additional planner and trajectory-generation details.

C. RoboRender Data Generation

We use Claude Opus 4.6 [35] to generate task-specific scene prompts. Together with the simulated depth videos and robot RGB-mask videos from Section III-B, these prompts condition our video model to synthesize multi-view photorealistic RGB videos with prompt-controlled textures, backgrounds, and scene semantics. The robot RGB masks provide robot appearance and motion cues. Since the robot appearance is relatively simple and has a small sim-to-real gap, we directly use the simulator-rendered robot RGB masks as input. Our video generation model can also insert distractor objects not present in simulation. We sample temporally consistent rectangular crops on the depth videos across views and

TABLE I: Quantitative results on generated video quality.

Model	First-chunk		AR continuation		
	FID ↓	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Wan-Fun-Control [24]	62.5	267.8	20.30	0.738	0.175
Cosmos-Transfer2.5 [25]	42.8	164.5	13.26	0.481	0.423
RoboTransfer [19]	52.8	275.4	13.75	0.509	0.418
Ours	25.7	93.9	22.28	0.784	0.142

TABLE II: 2AFC human study of the favor rate of RoboRender over baselines.

	Multi-view consist.	Gen. quality
over Wan-Fun-Control [24]	76.9%	90.6%
over Cosmos-Transfer2.5 [25]	74.9%	68.9%
over RoboTransfer [19]	85.7%	94.6%

augment the prompts with distractor descriptions, allowing the model to synthesize distractors in the cropped regions while preserving the original scene geometry and robot actions (Fig. 1; Appendix D). Finally, pairing the generated RGB videos with simulator-provided actions and states yields realistic RGB-action-state trajectories for policy training.

IV. VIDEO GENERATION EXPERIMENTS

Setup. We evaluate our video model against three baselines: Wan2.1-Fun-1.3B-Control [24] (Wan-Fun-Control), a depth-conditioned variant of our base model Wan2.1-T2V-1.3B [21], Cosmos-Transfer2.5-2B [25] (Cosmos-Transfer2.5), the latest depth-control variant in the Cosmos foundation model family [22] and RoboTransfer [19], a video generation model conditioned on depth, surface normals, and appearance reference images. We report FID [36] and FVD [37] as distribution-level video quality metrics, and additionally report PSNR, SSIM [38], and LPIPS [39] for paired evaluation against ground truth. We further conduct a two-alternative forced-choice (2AFC) human study as a complementary

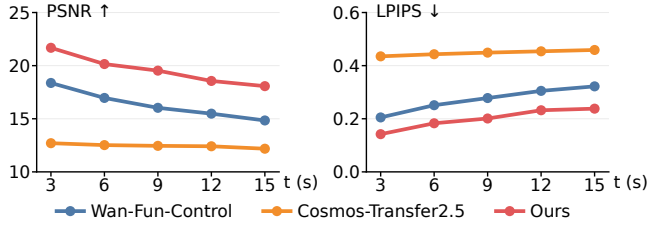


Fig. 4: Per-frame PSNR and LPIPS along 15-second autoregressive rollouts, sampled every 3 seconds.

assessment. We recruited 50 raters, each comparing 20 side-by-side pairs of RoboRender and baseline videos in randomized left-right order and judging realism and multi-view consistency.

Comparison with Baselines. We evaluate generation quality on 500 held-out clips from DROID [28] and AgiBot [27] under two inference-matched settings: *first-chunk generation*, which synthesizes the initial chunk without previous-frame conditioning, and *autoregressive continuation*, conditioned on a ground-truth previous frame. Wan-Fun-Control, RoboTransfer, and our model run at 416×240 . Since Cosmos-Transfer2.5 fails at this resolution, outside its pretraining range, we run it at 480p and downsample outputs to 240p, favoring Cosmos. For RoboTransfer, we extract depth and normal conditions following its recipe.

As shown in Table I, RoboRender outperforms all baselines across all metrics and settings, despite Cosmos-Transfer2.5 using higher-resolution 480p inputs. Qualitative first-chunk results in Fig. 3 show that Wan-Fun-Control and Cosmos-Transfer2.5 lack multi-view consistency, causing shared objects such as the table to vary across views. Wan-Fun-Control also produces unrealistic robot frames because it is a general depth-to-video model without robot-specific adaptation. By repurposing a general video model with robot-specific data, RoboRender achieves both multi-view consistency and robot realism. For sim-depth inputs, where no ground-truth RGB reference exists, we use a 2AFC human study in Table II, RoboRender is preferred over all baselines in sim-to-real generation quality.

Multi-chunk AR rollouts. To assess autoregressive generation beyond a single chunk, we roll out Wan, Cosmos and RoboRender for 15 s on 200 long real clips, initializing from a ground-truth previous frame and thereafter using its own output as the prev-frame anchor. As shown in Fig. 4, our model leads at every time step, staying well ahead of both baselines in PSNR and LPIPS throughout. This indicates that RoboRender sustains generation quality across repeated autoregressive steps, enabling the multi-chunk rollouts needed to render full simulation trajectories into realistic videos.

Multi-view consistency. We conduct a human study in which raters assess cross-view consistency across all three views. RoboRender is preferred over all baselines Table II, suggesting RoboRender can generate cross-view consistent videos. Appendix B reports additional qualitative comparisons

and results for 3-View Stacked baselines.

Accelerated Inference. To support large-scale data generation, we distill our controllable video model with DMD2 [40], reducing the 50-step bidirectional diffusion sampler to 6 steps. The distillation runs for 2,000 iterations on a single H200 node with the standard DMD recipe. The distilled video model generates one 3-view clip in 4.29s on a single H200. See Appendix C for acceleration details and TeaCache quality comparisons.

V. ROBOT PHYSICAL EXPERIMENTS

We evaluate RoboRender on five pick-and-place (PnP) tasks, two articulated-object opening tasks (opening the fridge and drawer), and one mobile manipulation task in which the robot moves to a table and picks up a radio, as shown in Fig. 6. For the four DROID PnP and opening tasks, we use a Franka robot with a Robotiq gripper, equipped with an external camera and a wrist-mounted camera following a DROID-style setup [28]. For the mobile manipulation task, we use an R1Pro robot with one egocentric camera and two wrist cameras. We additionally perform one PnP task using the YAM Arm with one egocentric camera and one wrist camera, where the robot picks up a green marker from three markers and places in the red bowl. The video model has only been trained on robot data from Droid and Agibot, as in III-A. Both R1Pro and YAM are unseen embodiment during the video model training.

We use different ways to randomize the configurations according to the type of task. For pick and place, we randomize the positions of task relevant objects and distractors, but for each trial, the positions are the same across different policies to ensure fair comparison. For the opening tasks (drawer and fridge) the randomization are done by changing the initial position and orientation of the robot. In the pick up the radio task, we randomize the physical setups by changing the mobile robot’s initial position and radio’s position on the table. See Appendix G for setup images.

A. Policy Training

Dataset Generation. For the PnP and mobile manipulation tasks, we construct scenes with a table and the task relevant objects. We generate 1K trajectories per PnP task and 2K for mobile manipulation. For opening tasks, we create empty-room scenes with a single articulated object, generating 2K drawer-opening and 1K fridge-opening trajectories. To test cross-simulator applicability, DROID PnP and mobile manipulation data are generated in OmniGibson [41], while opening-task and YAM task data are generated in MuJoCo [42]. We randomize object size, table height, robot pose, and external camera pose, yielding the simulation dataset $\mathcal{D}_{sim}$ of RGB observations, robot states, and actions. We construct a visual domain randomization baseline $\mathcal{D}_{dr}$ by replaying each simulation trajectory while randomizing object textures with DTD textures [43] and lighting conditions. With the video model from Section IV and following the procedure in Section III-C, we synthesize photorealistic RGB videos from the simulation data to form $\mathcal{D}_{gen}$. Examples of each datasets are shown in Fig. 5 and Appendix F. For DROID

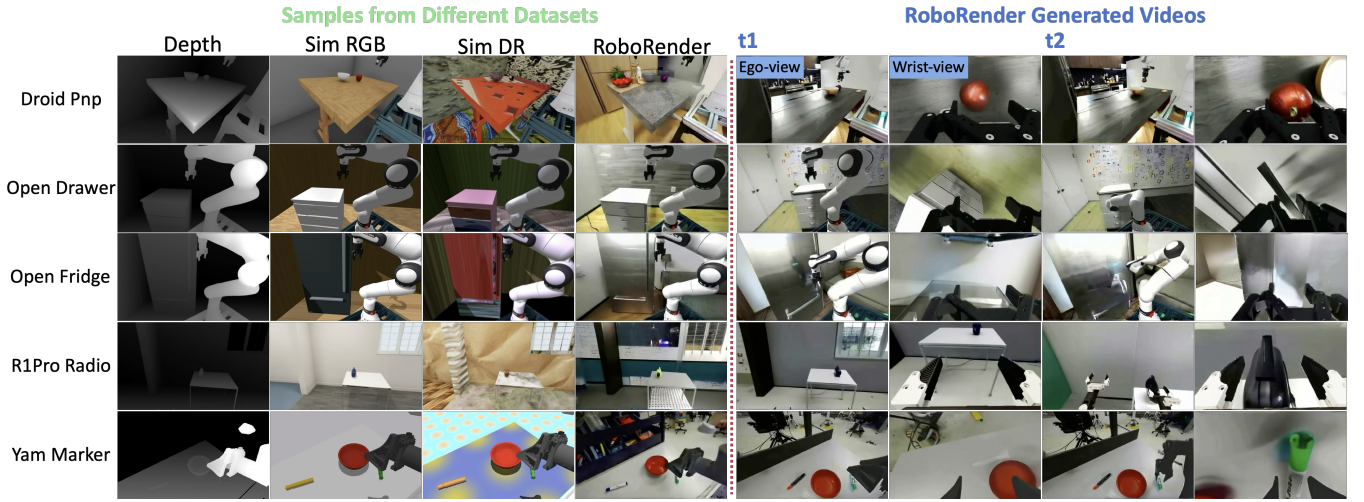


Fig. 5: Dataset samples from different methods (Left) and RoboRender generated videos examples (Right).

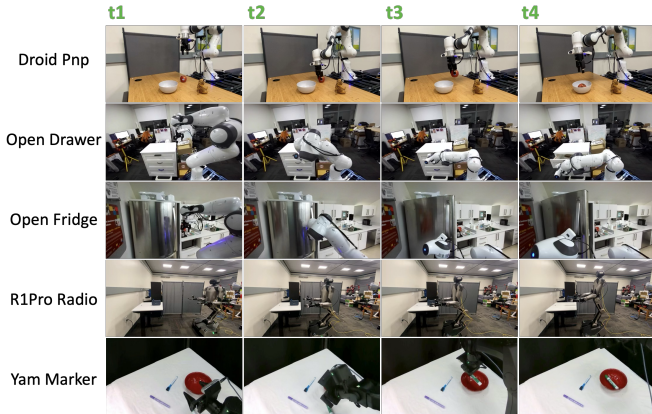


Fig. 6: Robot physical rollout examples.

PnP tasks, we additionally insert distractors using cropped depth patches: 2–5 patches in the external view, covering 20%–50% of the image, and 1–2 patches in the wrist view.

Training Setup. We use $\pi_{0.5}$ [44] as the robot policy, which maps visual observations, language instructions, and robot proprioception to actions. We finetune the pretrained $\pi_{0.5}$ base model on each dataset to predict 32-step action chunks, training one joint policy for all PnP tasks and separate policies for each opening task and mobile manipulation. All policies are trained for 40k steps with batch size 72. At inference, we execute the first 16 predicted steps.

B. Experiments and Analysis

We compare RoboRender with two baselines: **Sim**, trained on $\mathcal{D}_{sim}$, and **Sim + DR**, trained on $\mathcal{D}_{dr}$. Each task is evaluated over 10 rollouts with varied robot starts and object locations. For PnP tasks, we add random table distractors to test robustness. All real-robot evaluations are conducted in a natural office environment with object size, geometry, camera viewpoint, and background differing from simulation (right column of Fig. 6). Table III shows the results.

Does RoboRender mitigate the visual sim-to-real gap for zero-shot sim-to-real transfer? As shown in Table III,

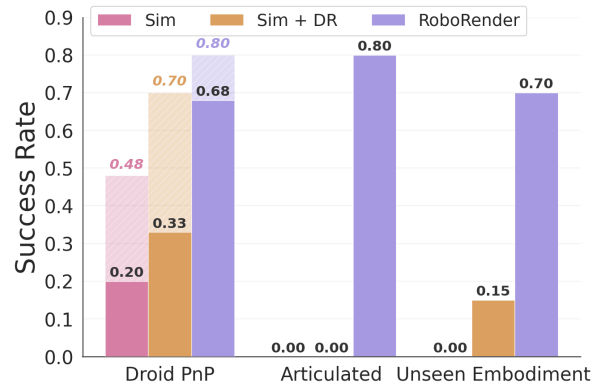


Fig. 7: Average success rate of each task category for each method. For the Droid PnP category, the lighter *hatched* extension on top of each bar shows the same method under the evaluation setup without distractor objects.

RoboRender achieves the highest average success rate across tasks, reaching 71%, about $3.6\times$ and $7.1\times$ higher than Sim + DR and Sim, respectively. The category-level results in Fig. 7 show the largest gains on opening tasks, where RoboRender is the only method with non-zero success. These tasks require recognizing fine visual details, such as drawer or fridge handles whose textures resemble the object body. Although visual domain randomization improves over raw simulation on PnP and mobile manipulation, it still fails on these visually precise tasks, suggesting that coarse appearance randomization is insufficient to close the visual sim-to-real gap. In contrast, RoboRender performs well across all task categories, showing that photorealistic RGB generation mitigates the visual sim-to-real gap for policy zero-shot sim-to-real transfer.

We further test whether distractor insertion improves robustness to clutter. In the easier no-distractor setting (hatched bars in Fig. 7), Sim, Sim + DR, and RoboRender achieve success rates of 48%, 70%, and 80%, respectively. When distractors are added during real-world evaluation, the baselines degrade substantially because their training data lacks clutter. By

TABLE III: Real-world success rate across different tasks.

Embodiment	Droid						R1Pro	Yam	
Data	Mug	Bowl	Apple	Egg	Open Fridge	Open Drawer	Pick up Radio	Place Green Marker	Avg.
Sim	40%	30%	10%	0%	0%	0%	0%	0%	10%
Sim + DR	60%	20%	20%	30%	0%	0%	30%	0%	20%
RoboRender	80%	80%	60%	50%	70%	90%	70%	70%	71%

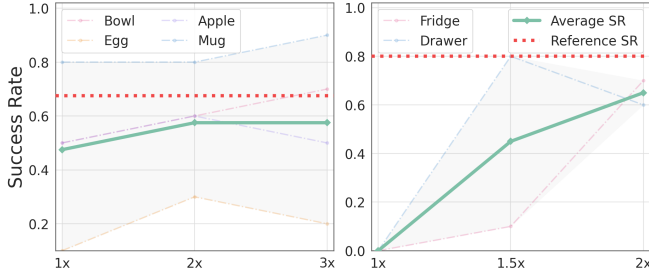


Fig. 8: **Policy success rate v.s. the number of videos generated per unique simulation trajectory**, for PnP (left) and Opening (right). *Solid green* marks the per-category average, *dash-dot* lines the individual tasks, and the *dotted red* line the reference policy (one generated video per trajectory with 2x more simulation trajectories). Performance improves with more generated videos even though the underlying set of unique simulation trajectories is unchanged.

contrast, RoboRender shows only a small gap between the distractor and no-distractor settings, as it can insert random distractors during video generation. Thus RoboRender not only reduces the visual sim-to-real gap through photorealistic generation, but also improves robustness to cluttered real-world scenes by supporting realistic distractor insertion.

Does RoboRender generalize to unseen robot embodiments? While trained only on DROID and AgiBot embodiments, the video model uses robot RGB-mask conditioning to generate videos for unseen robots. These masks are rendered in simulation, which suffices given the simple robot textures: on tasks with unseen R1Pro and YAM robots (Fig. 6), RoboRender substantially outperforms Sim and Sim + DR, suggesting that it can synthesize effective photorealistic training videos for new embodiments, reducing the visual sim-to-real gap.

Does RoboRender outperform depth-conditioned policies? Depth-conditioned policies mitigate the visual sim-to-real gap but lack semantic information available in RGB observations. To demonstrate the value of high-quality RGB training videos, we compare RoboRender with depth-only policies trained on simulated depth for three PnP and mobile PnP tasks. At inference, the depth-only policies use depth estimated from RGB observations via VideoDepthAnything. Table IV shows that policies trained with RoboRender achieve substantially higher success rates across all three setups, suggesting that RGB semantic cues are important for identifying the correct object to complete the tasks.

TABLE IV: Comparison against policies trained with depth data.

Setup	YAM	Droid PnP	R1Pro
Depth Only	30%	25%	10%
RoboRender	70%	68%	70%

TABLE V: Comparison across video generation models for training data.

Setup	Drawer	Fridge
Wan-Fun-Control	0%	0%
Cosmos-Transfer2.5	30%	0%
RoboRender	90%	70%

Does higher video generation quality lead to better policies? We compare policies trained on videos generated by RoboRender and baseline models on two opening tasks requiring high-quality RGB input. Specifically, we generated the same number of videos on the same simulation data using Cosmos and Wan. Since Cosmos and Wan don’t take robot RGB mask videos, we add texture descriptions of the robot in the prompt for the baseline models. Results in Table V show that Cosmos outperforms Wan but underperforms RoboRender. This aligns with the video quality results in Table I, suggesting that higher-quality RGB videos improve policy performance.

Does generating multiple videos per simulation trajectory improve policy learning? Because the video model is language-conditioned, it can generate diverse backgrounds, textures, and distractors from the same simulation trajectory. We study whether this visual diversity improves policy learning independent of collecting more unique trajectories. We use only **half** of the simulation trajectories and vary the number of generated videos per trajectory by resampling language prompts and depth patches. For PnP tasks, we train with 1x, 2x, and 3x generations per trajectory using 500 trajectories per task. For Opening tasks, we train with 1x, 1.5x, and 2x generations per trajectory, using 500 fridge-opening and 1k drawer-opening trajectories. Results are shown in Fig. 8, where the red dotted lines mark the performance of the policies in Table III, which train on **all** unique trajectories with one generated video each. While the gap to 1x reflects the value of physical trajectory diversity, additional generations recover much of this gap using only half the simulation data, especially on Opening tasks. On PnP tasks, average success rises from 48% at 1x to 58% at 2x and saturates at 3x. On Opening tasks, average success increases

more sharply, from 0% at $1\times$ to 45% at $1.5\times$ and 65% at $2\times$, indicating that visual diversity is particularly useful for precise grounding, such as handle identification. These results suggest that RoboRender can generate diverse photorealistic videos from a fixed simulation dataset, improving policy learning efficiency without a proportional increase in simulation data.

VI. CONCLUSION AND LIMITATIONS

We presented RoboRender, a framework that converts simulated trajectories into photorealistic RGB-action-state data with a robot-oriented video generation model. Conditioned on depth videos, language prompts, and robot RGB masks, RoboRender preserves simulator geometry and robot motion while synthesizing realistic textures, lighting, backgrounds, and distractors for policy training. Experiments show that policies trained on RoboRender-generated data achieve about $7.1\times$ and $3.6\times$ the average success rate of policies trained on raw simulation renderings and visual domain randomized data, respectively, across pick-and-place, articulated-object opening, and mobile manipulation tasks. These results demonstrate that RoboRender mitigates the visual sim-to-real gap for zero-shot transfer. Generating diverse videos from each simulated trajectory further improves policy learning efficiency without proportional increases in simulated data.

Limitations. RoboRender has several limitations. First, it addresses the visual sim-to-real gap and therefore remains bounded by the simulator’s fidelity in dynamics and contact modeling, making deformable-object tasks especially challenging. Second, although we show generalization to the unseen R1Pro embodiment, broader validation across robot morphologies, camera configurations, and real-world environments is needed. We also aim to scale RoboRender to more complex long-horizon tasks in the future.

REFERENCES

- [1] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, “3D Gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, pp. 139:1–139:14, 2023.
- [2] M. Torne *et al.*, “Robot learning with super-linear scaling,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2025.
- [3] J. Yu *et al.*, “Real2Render2Real: Scaling robot data without dynamics simulation or robot hardware,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.
- [4] P. Dan *et al.*, “X-Sim: Cross-embodiment learning via real-to-sim-to-real,” in *EgoAct: 1st Workshop on Egocentric Perception and Action for Robot Learning at RSS 2025*, 2025.
- [5] Z. Yuan *et al.*, “HERMES: Human-to-robot embodied learning from multi-source motion data for mobile dexterous manipulation,” in *Workshop on Dexterous Manipulation at CoRL 2025*, 2025.
- [6] T. G. W. Lum, O. Y. Lee, C. K. Liu, and J. Bohg, “Crossing the human-robot embodiment gap with sim-to-real RL using one human demonstration,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.
- [7] K. Kedia, T. G. W. Lum, J. Bohg, and C. K. Liu, “SimToolReal: An object-centric policy for zero-shot dexterous tool manipulation,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2026.
- [8] T. He *et al.*, “VIRAL: Visual sim-to-real at scale for humanoid loco-manipulation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
- [9] A. Deshpande *et al.*, “MolmoBOT: Large-scale simulation enables zero-shot manipulation,” arXiv:2603.16861, 2026.
- [10] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3803–3810.
- [11] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters, “Robot learning from randomized simulations: A review,” *Frontiers in Robotics and AI*, vol. 9, p. 799893, 2022.
- [12] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang, “DexPoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023, pp. 594–605.
- [13] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.
- [14] J. Jang *et al.*, “DreamGen: Unlocking generalization in robot learning through video world models,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.
- [15] S. Kim, S. Jang, B. Yoon, D. Kim, J. Won, and J. Shin, “RoboCurate: Harnessing diversity with action-verified neural trajectory for robot learning,” arXiv:2602.18742, 2026.
- [16] J. Ye *et al.*, “AnchorDream: Repurposing video diffusion for embodiment-aware robot data synthesis,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2026.
- [17] B. Wang *et al.*, “EmbodieDreamer: Advancing Real2Sim2Real transfer for policy training via embodied world modeling,” arXiv:2507.05198, 2025.
- [18] J. Chen, I.-C. A. Liu, G. Sukhatme, and D. Seita, “CRAFT: Video diffusion for bimanual robot data generation,” in *Workshop on Synthetic Data for Robot Learning at ICRA 2026*, 2026.
- [19] L. Liu *et al.*, “RoboTransfer: Controllable geometry-consistent video diffusion for manipulation policy transfer,” arXiv:2505.23171, 2026.
- [20] A. Ye *et al.*, “GigaWorld-0: World models as data engine to empower embodied AI,” arXiv:2511.19861, 2025.
- [21] A. Wang *et al.*, “Wan: Open and advanced large-scale video generative models,” arXiv:2503.20314, 2025.
- [22] N. Agarwal *et al.*, “Cosmos world foundation model platform for physical AI,” arXiv:2501.03575, 2025.
- [23] OpenAI, “Video generation models as world simulators,” <https://openai.com/index/video-generation-models-as-world-simulators/>, 2024, openAI technical report.
- [24] Alibaba PAI and aigc-apps, “Wan2.1-Fun: Controllable video generation with Wan (VideoX-Fun),” <https://github.com/aigc-apps/VideoX-Fun>, 2025, wan2.1-Fun-Control variants supporting depth, pose, canny, MLSD conditioning.
- [25] NVIDIA, “World simulation with video foundation models for physical AI,” arXiv:2511.00062, 2025, nVIDIA Cosmos-Predict2.5 / Cosmos-Transfer2.5.
- [26] G. Chen *et al.*, “SkyReels-V2: Infinite-length film generative model,” arXiv:2504.13074, 2025.
- [27] Q. Bu *et al.*, “AgiBot world colosseum: A large-scale manipulation platform for scalable and intelligent embodied systems,” arXiv:2503.06669, 2025.
- [28] A. Khazatsky *et al.*, “DROID: A large-scale in-the-wild robot manipulation dataset,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.
- [29] P. Wang *et al.*, “Qwen2-VL: Enhancing vision-language model’s perception of the world at any resolution,” arXiv:2409.12191, 2024.
- [30] S. Chen *et al.*, “Video depth anything: Consistent depth estimation for super-long videos,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 22 831–22 840.
- [31] B. Wen, M. Trepte, J. Aribido, J. Kautz, O. Gallo, and S. Birchfield, “FoundationStereo: Zero-shot stereo matching,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 5249–5260.
- [32] N. Carion *et al.*, “SAM 3: Segment anything with concepts,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.
- [33] Y. Kim *et al.*, “MolmoSpaces: A large-scale open ecosystem for robot navigation and manipulation,” arXiv:2602.11337, 2026.
- [34] B. Sundaralingam *et al.*, “Curobo: Parallelized collision-free robot motion generation,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8112–8119.

- [35] Anthropic, “Claude Opus 4.6 system card,” Anthropic, Tech. Rep., Feb. 2026, version dated February 10, 2026.
- [36] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local Nash equilibrium,” in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6626–6637.
- [37] I. Skorokhodov, S. Tulyakov, and M. Elhoseiny, “StyleGAN-V: A continuous video generator with the price, image quality and perks of StyleGAN2,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 3616–3626.
- [38] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Transactions on Image Processing (TIP)*, vol. 13, pp. 600–612, 2004.
- [39] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 586–595.
- [40] T. Yin *et al.*, “Improved distribution matching distillation for fast image synthesis,” in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2024.
- [41] C. Li *et al.*, “BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.
- [42] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 5026–5033.
- [43] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, “Describing textures in the wild,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3606–3613.
- [44] K. Black *et al.*, “ $\pi_{0.5}$: A vision-language-action model with open-world generalization,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.
- [45] F. Liu *et al.*, “Timestep embedding tells: It’s time to cache for video diffusion model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 7353–7363.

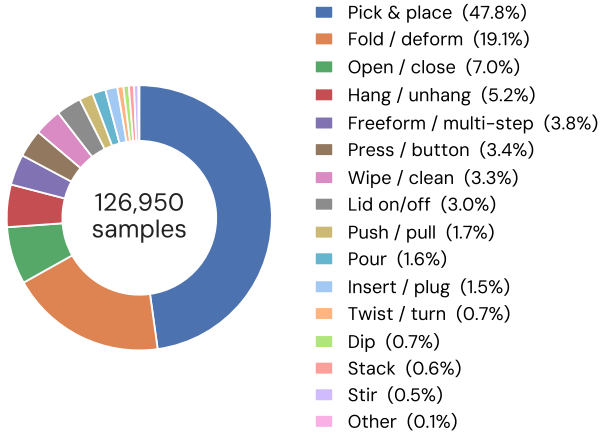
APPENDIX

A VIDEO MODEL TRAINING DATA CURATION

Training our video model requires large-scale paired (condition, video) data. However, most existing real-world robotics datasets [27], [28] are collected for policy learning rather than video generation, and lack the prompts, geometric conditions, and embodiment signals required by our model. We therefore design a scalable preprocessing pipeline that converts existing robot demonstrations into training-ready samples. Complete training data samples are shown in Fig. 10.

For each demonstration clip, we extract a temporally consistent 81-frame window across all available camera views, and produce four condition signals aligned to this window: a scene description prompt, per-view depth videos, per-view robot-arm RGB and masks, and a pre-action frame.

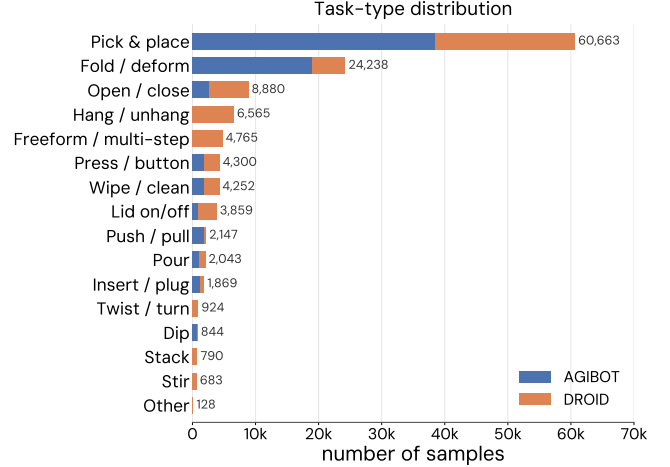
Scene Description Prompt. We use a Vision-Language



(a) Pie chart showing the distribution of 16 unique tasks over the 126,950 training samples combining DROID and AgiBot-Alpha.

Model [29] in a two-pass scheme: a first pass produces a draft caption from the first frame, and a second pass refines it by cross-verifying objects across all non-wrist views. The model is constrained to output structured metadata covering background, work surface, and foreground objects, substantially reducing hallucination compared to single-pass captioning.

Per-view Depth. We extract per-view relative depth using Video-Depth-Anything [30], and additionally run FoundationStereo [31] on samples with available stereo recordings. Both outputs are normalized via inverse-depth mapping. To avoid inductive bias from any single depth estimator, we randomly mix depth videos from different sources per view during training, improving robustness of the learned depth-to-appearance mapping.



(b) 16 task categories breakdown with specific number of samples between DROID and AgiBot-Alpha.

Fig. 9: Training data composition for the video generation model.

Robot-arm RGB and Masks. We use SAM3 [32] to segment the robot arm and gripper across all views. These signals provide explicit embodiment grounding, isolating the agent’s pose and motion from the rest of the scene.

Pre-action Frame. We include the frame immediately preceding the 81-frame clip as an additional condition, anchoring the visual appearance of the generated video and supporting autoregressive long-video generation at inference.

Our training dataset for the video model consists of $\sim 130k$ real-world robot manipulation samples drawn from two sources: $\sim 70k$ from AgiBot-World-Alpha [27] and $\sim 60k$ from DROID [28]. We process all data at 416×240 resolution. We will release the processed paired data as an additional contribution to facilitate future research on robotics video generation.

Fig. 9b, Fig. 9a, and Table VI summarize the diversity of our training dataset. Our training dataset contains 126,950 paired samples spanning 16 manipulation task types. Fig. 9a gives their overall proportions, led by pick & place (47.8%)

and fold / deform (19.1%), and the remaining 14 types, such as open / close, hang / unhang, or stack, and Fig. 9b shows per-type sample counts, with each bar split by source (AgiBot, $\sim 70k$; DROID, $\sim 60k$), and dataset-level totals are listed in Table VI. In addition to showing the total sample number and task diversity, we also analyzed the number of distinct scenes in our training dataset. We found in total 39,692 distinct scenes in our training datasets, which further demonstrates the high scene diversity in our training dataset.

B VIDEO GENERATION BASELINE COMPARISON

Additional baseline comparisons. Fig. 11 shows more qualitative comparisons against the baselines, where RoboRender produces realistic video quality while preserving object consistency across views.

Effect of 3-View Stacked Input. At inference, our model generates the three views jointly by concatenating them in the latent space. To confirm that our advantage comes from dedicated multi-view training rather than this inference

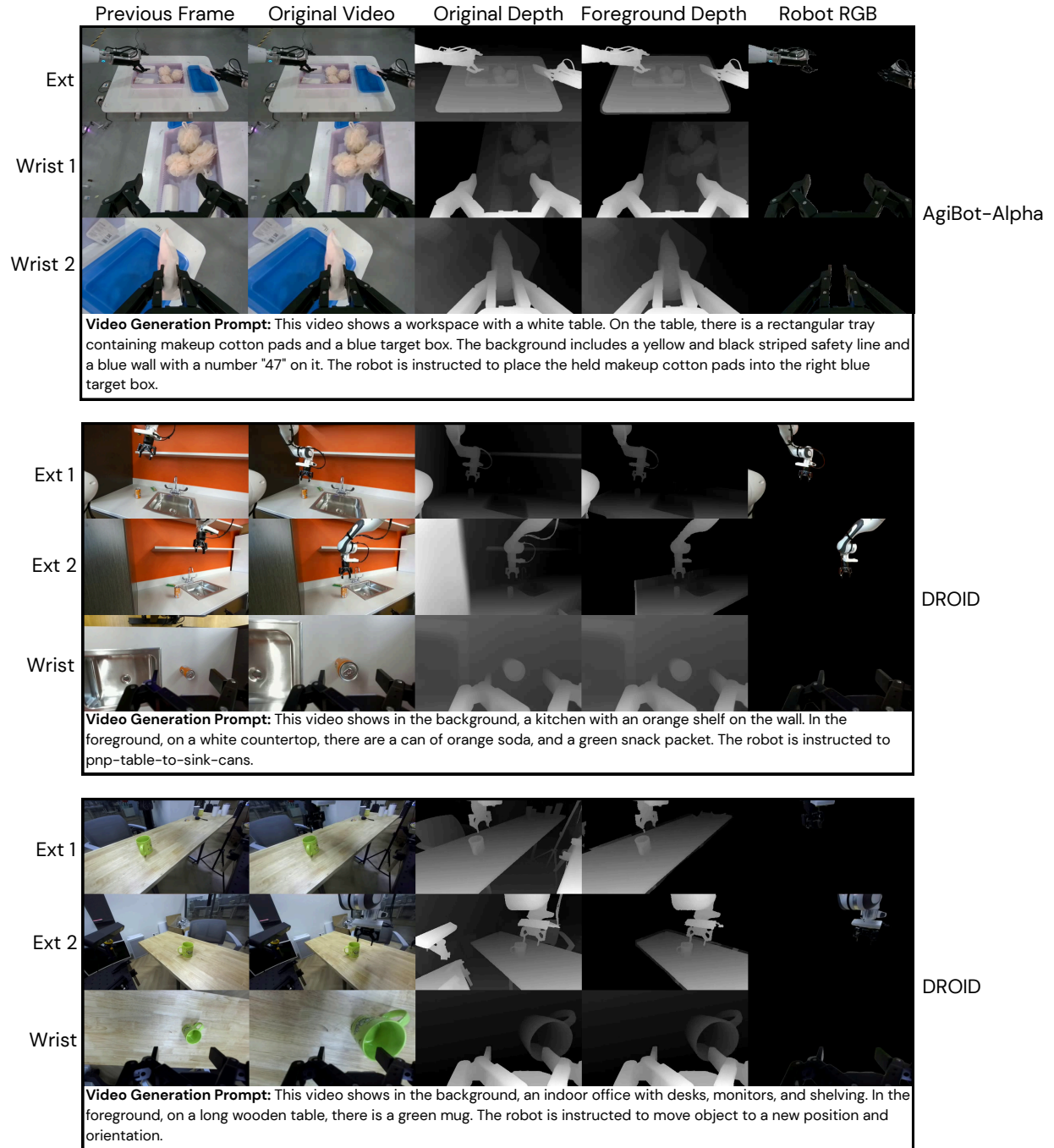


Fig. 10: Example training samples for the video generation model, shown across camera views for AgiBot-Alpha and DROID. Each sample includes a Video Generation Prompt and five aligned conditions: Previous Frame, Original Video, Original Depth, Foreground Depth, and Robot RGB mask.

scheme alone, we grant both baselines the same latent concatenation, feeding all three views as a single stacked input (the *3-View Stacked* variants). As shown in Table VII, latent concatenation does not close the gap: it only marginally improves Wan-Fun-Control [24] and even degrades Cosmos-Transfer2.5 [25], while our model remains far ahead. Fig. 12 shows the same qualitatively, where the stacked baselines still render shared objects inconsistently across views. These results show that latent concatenation at inference alone is insufficient for multi-view consistency, which instead requires

dedicated multi-view training.

C ON ACCELERATING VIDEO MODEL GENERATION

Rendering an entire simulation dataset into video requires many diffusion forward passes, so inference cost is the main bottleneck for scaling up data generation. We explore two directions to reduce it.

Accelerated Inference. We accelerate inference with Tea-Cache [45], which caches and reuses intermediate diffusion features to skip redundant computation across denoising steps.

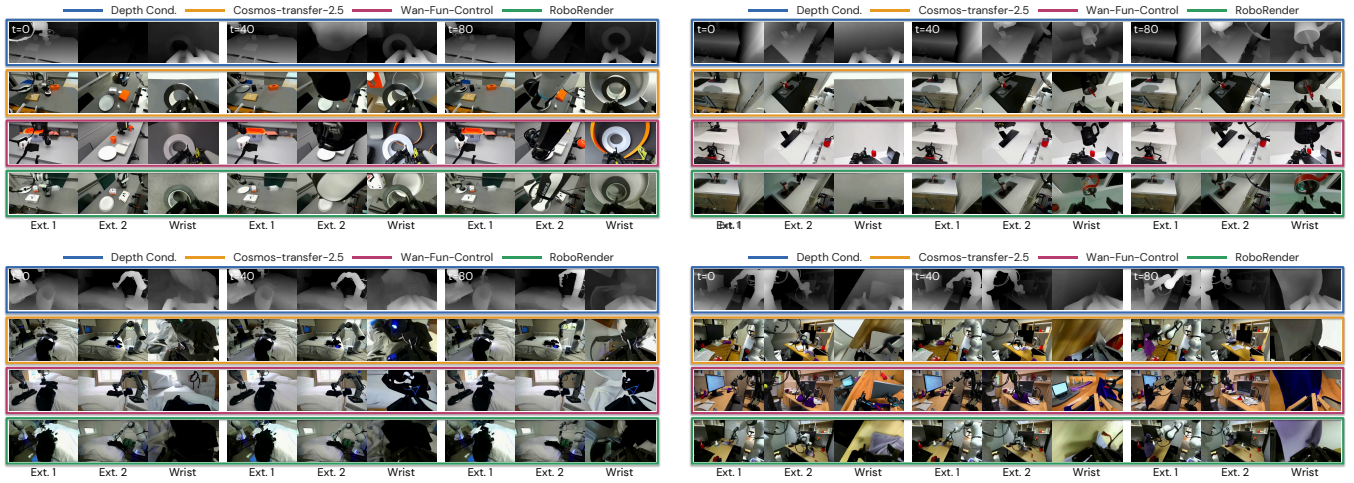


Fig. 11: **Additional baseline comparison.** Conditioned on the same depth videos (top row), RoboRender renders realistic frames that stay consistent across the two external views and the wrist view, whereas Cosmos-Transfer2.5 [25] and Wan-Fun-Control [24] generate each view independently and produce cross-view inconsistencies.

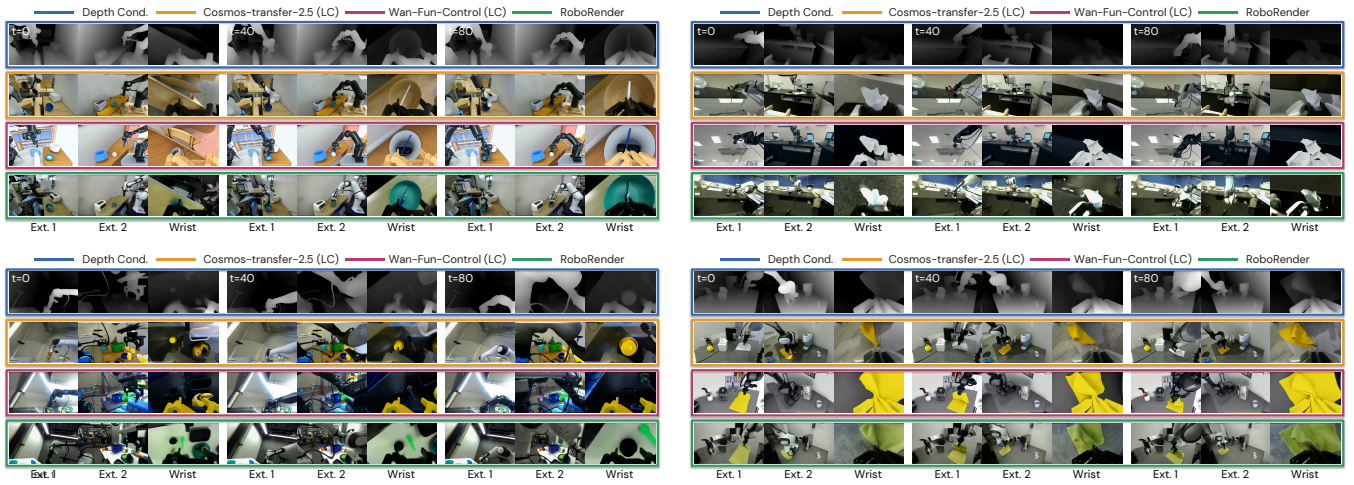


Fig. 12: **Effect of 3-View Stacked.** Even with all three views stacked into a single input, Cosmos-Transfer2.5 [25] (LC) and Wan-Fun-Control [24] (LC) render shared objects, such as the table, inconsistently across views, whereas RoboRender stays consistent.

TABLE VI: Sample counts for our processed corpus, broken down by source dataset (AgiBot-World-Alpha and DROID) and combined. The dataset in total includes 126,950 paired samples across 39,692 distinct scenes and 16 manipulation task types.

Dataset	# Samples	# Distinct scenes	# Task categories
AGIBOT	69,910	22,650	10
DROID	57,040	17,042	15
Total	126,950	39,692	16

On a single NVIDIA H200 GPU with 50 denoising steps and a CFG scale of 5.0, our default pipeline generates a 3-view 81-frame clip in 68.8 s, while the TeaCache variant (threshold $\tau = 0.10$) produces the same clip in 35.3 s, a $1.95\times$ speedup. As shown in Fig. 13, the error map is small and the two

TABLE VII: Effect of 3-View Stacked input on baselines (first-chunk synthesis, real input).

Model	FID ↓	FVD ↓
Wan-Fun-Control	62.5	267.8
Wan-Fun-Control (3-View Stacked)	61.8	255.6
Cosmos-Transfer2.5	42.8	164.5
Cosmos-Transfer2.5 (3-View Stacked)	57.1	236.0
Ours	25.7	93.9

pipelines are visually close in quality.

Step distillation. We further distill our controllable video model with DMD2 [40], reducing the 50-step bidirectional diffusion sampler to 6 steps. The distillation runs for 2,000 iterations on a single node with the standard DMD recipe.

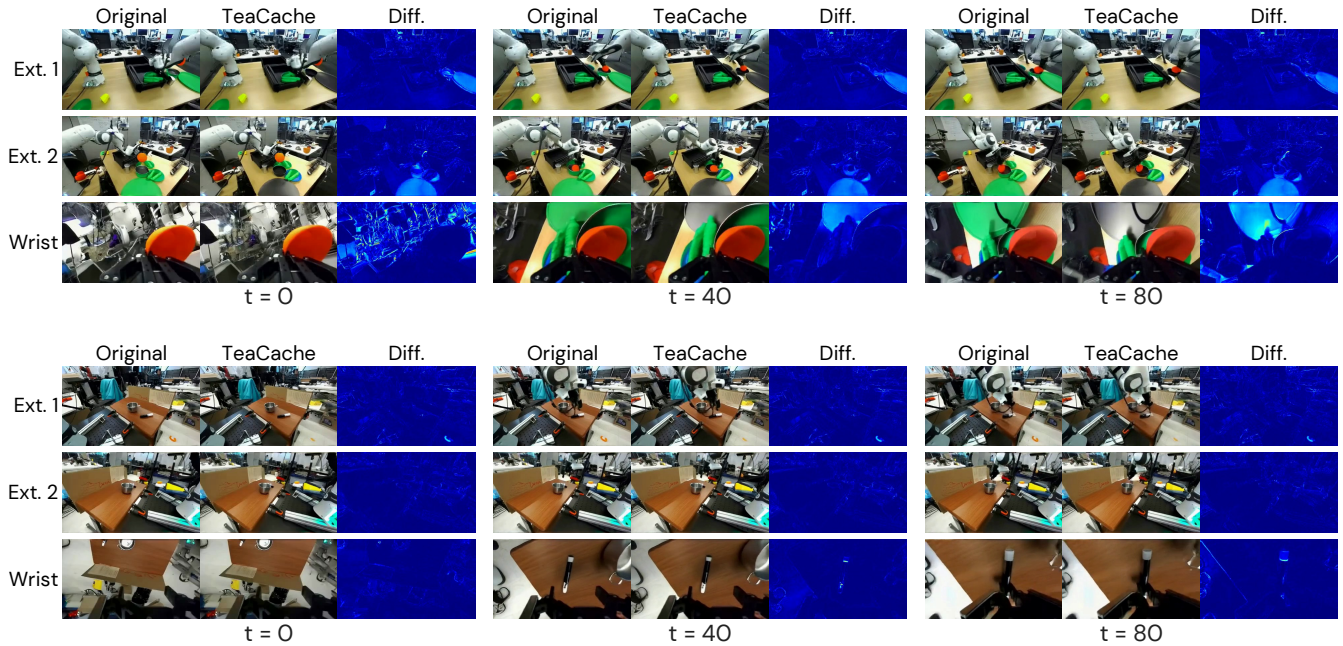


Fig. 13: **Quality preservation under TeaCache.** For each view and time step, we show the default-pipeline frame (Original), the TeaCache frame, and their per-pixel difference (Diff). The difference is small and confined to fine texture, leaving scene layout, objects, and robot pose unchanged.

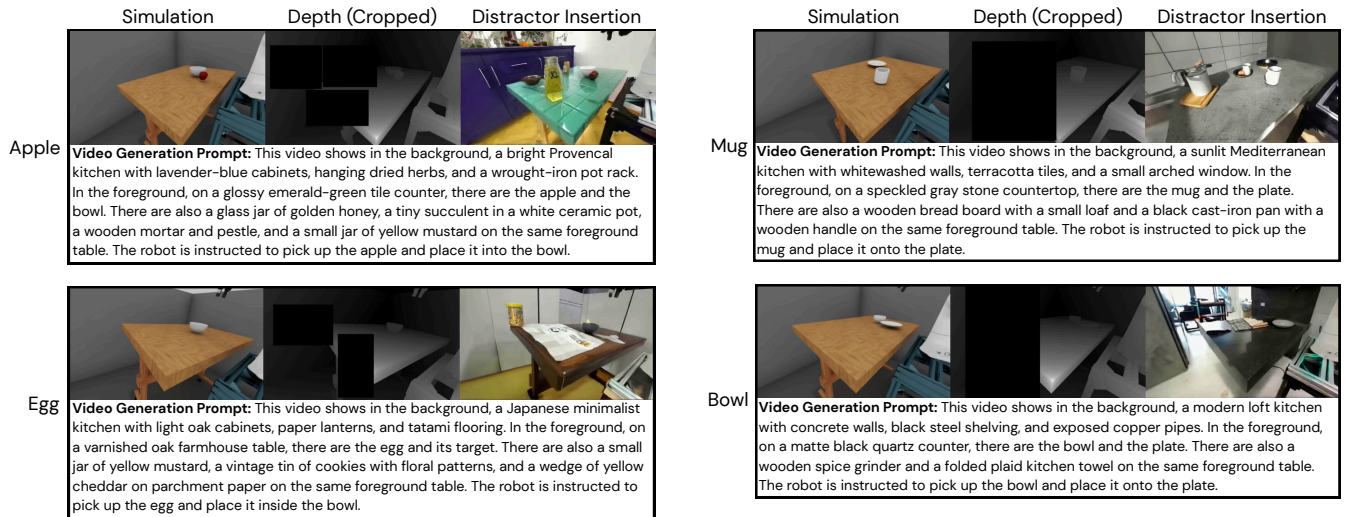


Fig. 14: Distractor insertion via depth cropping. We crop regions of the simulation depth (middle, black), eliminating those areas from geometric constraint so that RoboRender (right) synthesizes photorealistic distractor objects in these places while preserving the task-relevant geometry.

The distilled model generates one clip in 4.29s on a single H200 GPU.

D DISTRACTOR INSERTION

Fig. 14 illustrates how RoboRender enables controllable distractor insertion. Starting from the simulation depth (middle), we randomly crop out regions of the depth map on the table (shown in black), removing the geometric constraint in those areas in both views. When RoboRender renders the photorealistic video (right), these unconstrained regions are no longer pinned to the simulated geometry, allowing

the model to synthesize plausible distractor objects and background clutter in these places. The inserted distractors remain consistent over time, augmenting the visual diversity of the training data without introducing geometric artifacts in the task-relevant regions.

E ROBOT SIMULATION DATA GENERATION

We generate task trajectories with a pipeline built on MolmoSpaces [33] and CuRobo [34]. For each episode, we instantiate a scene with a target object or articulated object-joint pair, a robot at a feasible base pose, and synchronized

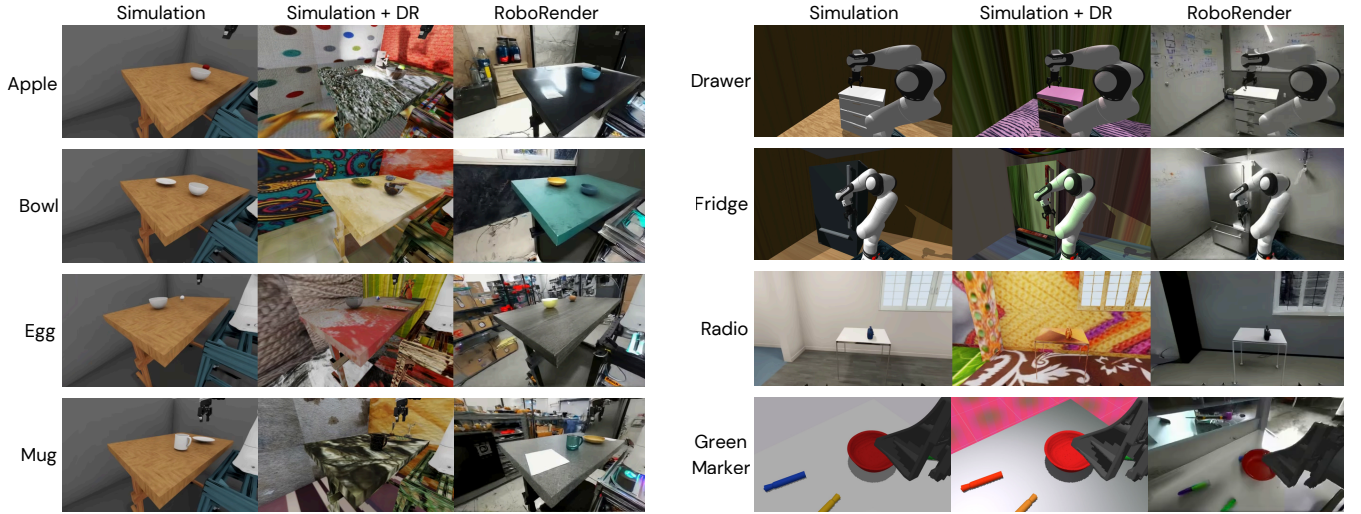


Fig. 15: Visualization of policy training data across eight evaluation tasks between raw Simulation, Simulation + Domain Randomization, and **RoboRender** (ours). Comparing to the other two methods, RoboRender yields photorealistic textures and lighting while preserving geometry, minimizing the appearance gap to the real world.



Fig. 16: Physical setups and the initial poses of the robot in completion for each task

multi-view cameras. A scripted task-and-motion planner executes the task while recording RGBD observations of each view, robot proprioception, and robot actions. We also record the robot segmentation mask videos from the simulation.

Actions are produced by a scripted planner with offline grasp libraries for both rigid-object and articulated-object manipulation. Each manipulable target is associated with precomputed antipodal end-effector poses for a parallel-jaw gripper; for articulated tasks, grasp libraries are defined per object-joint pair, such as a door or drawer handle. At episode start, candidate grasps are transformed into the world frame and augmented with 180-degree rotations to account for gripper symmetry.

The planner selects a grasp through ranking, collision filtering, and IK feasibility checking. Candidates are ranked by reachability, orientation alignment, approach stability, and proximity to the task-relevant manipulation region, such as the object center of mass for pick-and-place or the handle midpoint for articulated objects [33]. The top candidates are collision-checked against the object, support surface, clutter, and robot links, then passed through batched IK checks. The first feasible grasp is selected.

After grasp selection, the planner executes task-specific mo-

tion primitives. Pick-and-place tasks follow a pregrasp-grasp-lift-place-retreat sequence, while articulated tasks grasp the handle and move along the target joint constraint, using an arc for hinge joints and a linear path for prismatic joints. For mobile manipulation, the planner follow navigation-pregrasp-grasp-lift sequence. If execution fails due to IK loss, motion stalling, downstream infeasibility, or gripper slip, the planner replans with a different grasp. Success is measured by object placement or object lifting or by joint displacement crossing a task-defined threshold. The episode fails after the maximum number of retries is exceeded or a timeout of 300 simulator steps. Successful trajectories are retained for policy training.

F POLICY TRAINING DATA VISUALIZATION

Fig. 15 shows exemplar data we used to train each robot evaluation policy for completing the 8 tasks. Raw simulation (left) provides clean geometry but exhibits a large appearance gap from the real world, specifically synthetic textures, uniform lighting, and an unrealistic background. Standard domain randomization on simulation data (middle) perturbs textures and colors to widen the training distribution, but the resulting frames are visually incoherent and still fail to match real-world material and lighting statistics. Our method

RoboRender (right) generates photorealistic observations that preserve the underlying scene geometry while producing realistic surface textures, reflections, and lighting, which closes the appearance gap without sacrificing the geometric grounding required for the task. This higher visual realism allows policies trained on RoboRender data to transfer zero-shot to the real robot across different manipulation tasks, including opening tasks and mobile robot tasks that are often considered to be more difficult.

G PHYSICAL SETUP

Fig. 16 shows the physical setups and initial poses of our robots in completion for each task. Note that we used different ways to randomize the configurations according to the type of task. For pick and place, we randomized the positions of task relevant objects and distractors, but for each trial, the positions are the same across different policies to ensure fair comparison. For the opening tasks (drawer and fridge) the randomization was done by changing the initial position and orientation of the robot. In the pick up the radio task, we randomize the physical setups by changing the mobile robot’s initial position and radio’s position on the table.