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

RE^{3}SIM: Generating High-Fidelity Simulation Data via 3D-Photorealistic Real-to-Sim for Robotic Manipulation

Xiaoshen Han[1,2,*] Junqiu Yu[2,*] Minghuan Liu[1] Yilun Chen[2,†] Xiaoyang Lyu[3] Yang Tian[2] Bolun Wang[2]
Weinan Zhang[1,†] and Jiangmiao Pang[2,†]
Website: https://re3sim.github.io/

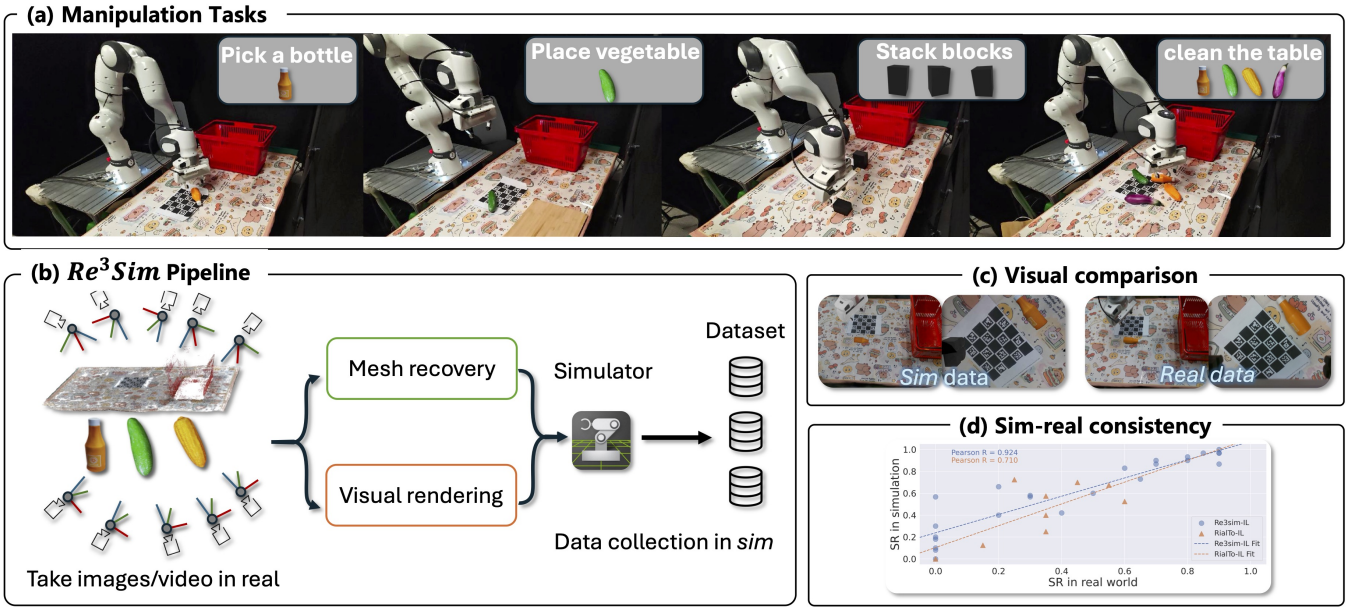


Fig. 1: **Illustration of RE^{3}SIM.** a) RE^{3}SIM allows zero-shot policy transfer on various tasks. b) The system pipeline to generate high-quality data. c) High-fidelity rendering results. d) Consistency in success rates between real and simulated environments.

***Abstract*— Real-world data collection for robotics is costly and resource-intensive, requiring skilled operators and expensive hardware. Simulations offer a scalable alternative but often fail to achieve sim-to-real generalization due to geometric and visual gaps. To address these challenges, we propose a 3D-photorealistic real-to-sim system, namely, RE^{3}SIM, addressing geometric and visual sim-to-real gaps. RE^{3}SIM employs advanced 3D reconstruction and rendering techniques to faithfully recreate real-world scenarios, enabling real-time rendering of simulated cross-view cameras within a physics-based simulator. By utilizing privileged information to collect expert demonstrations efficiently in simulation, and train robot policies with imitation learning, we validate the effectiveness of the real-to-sim-to-real system across various manipulation task scenarios. Notably, with only simulated data, we can achieve zero-shot sim-to-real transfer with an average success rate exceeding 58%. To push the limit of real-to-sim, we further generate a large-scale simulation dataset, demonstrating how a robust policy can be built from simulation data that generalizes across various objects.**

I. INTRODUCTION

We have witnessed impressive generalization capabilities of robotic models [1]–[3] in certain tabletop or kitchen scenarios, achieved through high-quality teleoperation data. However, collecting real-world expert data [4], [5] remains a time-consuming and costly process. Despite advancements in teleoperation systems [6]–[8], the labor involved is still intensive, making this a key challenge in robotics research. Recent efforts [9], [10] have explored inter-institutional collaboration to address this issue. In contrast, simulation data offers the advantage of exponential scalability with computational resources, making it an appealing and renewable alternative for training robotic policies. Research on sim-to-real transfer [11], [12] has increasingly focused on generating high-quality simulated or synthetic data to train real-world robotic policies. Unfortunately, simulation data often exhibits significant sim-to-real gaps, making it necessary to collect target domain data for effective policy fine-tuning.

In real-to-sim-real scenarios, consistency with the real world in appearance and geometry is essential. High-quality RGB rendering ensures visual consistency between real and simulated domains, while accurate mesh reconstruction captures scene geometry for realistic interactions. To achieve this, we propose a 3D-photorealistic real-to-sim-to-real system, RE^{3}SIM, short for reconstruction-rendering-based real-to-sim. RE^{3}SIM faithfully replicates real-world scenarios by integrating photorealistic RGB rendering with precise 3D geometry reconstruction. The visual rendering is realized via Gaussian rasterization [13], while the 3D geometry is reconstructed

[1]Shanghai Jiao Tong University, [2]Shanghai AI Laboratory, [3]The University of Hong Kong. * denotes equal contribution. † denotes corresponding authors.

using multi-view stereo (MVS) techniques [14], [15]. This dual-stage pipeline enables efficient, high-fidelity simulation, where physical dynamics are handled by physics engine backends [16]–[18], and real-time rendering is achieved through a dedicated hybrid rendering engine.

Specifically, RE^{3}SIM adopts a sequential real-to-sim pipeline comprising three key steps: (a) *mesh recovery* for reconstructing scene and object geometry; (b) *hybrid visual rendering* for compositing foreground and background elements, and (c) *real-world alignment* to synchronize world coordinates between the real and simulated environments. Human involvement is minimal in the real-to-sim process and limited to: (a) Placing ArUco markers for real-sim alignment. (b) Capturing images/videos of the scene and objects, including an extra step for robot base alignment.

After importing all assets and robots into the simulator, RE^{3}SIM utilizes a privileged policy to collect high-fidelity expert data. It incorporates the following features to enhance sim-to-real transfer:

- **High-fidelity geometry and vision**: Accurately reconstructs geometry and generates photorealistic views, closely matching real-world appearances to reduce sim-to-real gaps.
- **Rapid scene reconstruction**: Reconstructs novel scenes with under *three* minutes of manual effort.
- **Efficient rendering**: Achieves 24 FPS at 480p across two independent camera views.

To validate the effectiveness of the pipeline, we design several tabletop tasks demonstrating the generality and applicability of the real-to-sim-to-real process, as shown in Fig. 1 The zero-shot sim-to-real policies achieve an average success rate of 58% with only about 10 minutes of data collection in simulation, showcasing the effectiveness of our approach. We expect these experiments to highlight the potential of the real-to-sim-to-real pipeline in reducing sim-to-real gaps, paving the way for future advancements in scalable, efficient, and generalizable robotic policy.

II. BACKGROUND

A. 3D Reconstruction

3D reconstruction recovers scene geometry from inputs like images, point clouds and etc. Camera poses and sparse point clouds can be obtained from structure-from-motion approaches (e.g., Colmap) and be used to reconstruct dense point clouds via either traditional methods [19] or network-based approaches [20]. Besides, commercial software solutions, such as ARCode and PolyCam, leverage RGB-D SLAM methods to estimate camera poses and depth information, and often include some post-processing techniques (e.g., smoothing, inpainting) for enhancing the usability and quality of the reconstructed 3D models.

B. Rendering

Rendering involves generating photorealistic images from specified camera views based on either explicit or implicit 3D representations. Implicit methods, like Neural Radiance Fields (NeRF [21]), represent the 3D space using a density field encoded in a neural network and render images through volumetric rendering techniques. Explicit representations, such as meshes or point clouds, employ texture mapping [22]. 3D Gaussian Splatting (3DGS) [13] uses Gaussian primitives and adopts Gaussian rasterization for high-fidelity rendering. Compared with other methods, 3DGS achieves more realistic rendering results and offers higher rendering efficiency. In detail, each Gaussian primitive is modeled as an ellipse with a full 3D covariance matrix Σ defined in world space, centered at point μ: $G(x) = \exp(-\frac{1}{2}x^T\Sigma^{-1}x)$, where the covariance matrix $\Sigma = RSS^TR^T$ of a Gaussian primitive is derived from its rotation R and scaling factor S. Additionally, each Gaussian is assigned a color c_i and depth-ordered using the Z-buffer. Using the alpha-blending formula, the rendered color C for a view is calculated as:

$$C = \sum_{i=1}^{N} \prod_{j<i} \alpha_j (1 - \alpha_i) c_i \ .$$

III. RE^{3}SIM

Robot simulators [16], [18], [23] offer high-accurate simulation results based on underlying physics engines [23]–[25]. However, it's hard to directly obtain an accurate 3D model of the real-world scene, and the simulators are not good at rendering high-fidelity images, leaving a large gap from real-world images. We introduce RE^{3}SIM, a 3D reconstruction-based system, to bridge the visual gap and recover realistic 3D meshes for sim-to-real transfer.

A straightforward approach is to reconstruct objects and the background together. However, using Multi-View Stereo (MVS) methods for background reconstruction often leads to suboptimal visual quality, while techniques like SuGaR [26] struggle with accurately reconstructing flat planes, causing protrusions and indentations that reduce realism in object movement. Additionally, segmenting foreground objects for manipulation is labor-intensive. To address these issues, we propose separately reconstructing background meshes for collision estimation and leveraging 3DGS to improve rendering realism, aligning the two. Since object rendering constitutes a small portion of the robot's visual observations, we opt not to use 3DGS to render objects.

A. System Overview

RE^{3}SIM is a real-to-sim-real system that reconstructs meshes for collision estimation, utilizes 3D Gaussian scene representations for rendering, and generates simulation data within a physics-based simulator[1]. It is worth noting that we only consider reconstructing the background and foreground objects, and robots are not involved in our system since their accurate 3D assets are usually easily accessible. Further, we take this work as an initial study of RE^{3}SIM, validated on a table-top robot arm for rigid body tasks. As shown in Fig. 2, RE^{3}SIM are composed of four components: mesh recovery for object collision computation, hybrid visual rendering combining 3DGS for the background and mesh-based rendering for the foreground, real-world alignment of

[1]Isaac Sim [16] is used by default, but other ray tracing options are also feasible.

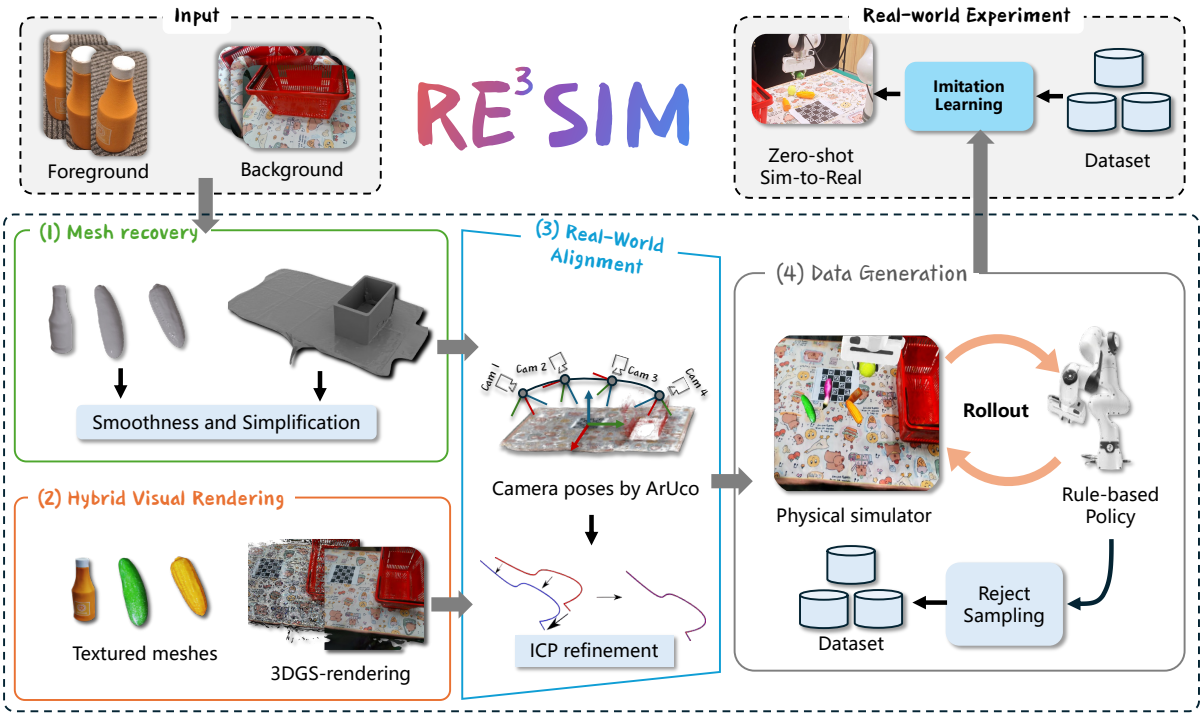


Fig. 2: **Illustration of the proposed real-to-sim-to-real system, RE^{3}SIM**. It leverages 3D reconstruction and a physics-based simulator, providing small 3D gaps that enable large-scale simulation data generation for learning manipulation skills via sim-to-real transfer.

the reconstructed scene and robot positions, and large-scale data generation for manipulation tasks.

B. Mesh Recovery

This step reconstructs objects and background geometries for accurate collision detection.

Mesh reconstruction. The meshes of backgrounds and objects are reconstructed separately for flexibility and are represented in USD for extensibility. For the background, we first employ SfM techniques (COLMAP) to estimate the pose for each image and get the sparse point cloud based on a set of images, and then utilize OpenMVS [15] for detailed mesh reconstruction. PolyCam [27] is also able to reconstruct the mesh of the background, but empirically, the geometric reconstruction results of OpenMVS exhibit fewer minor protrusions and depressions. Subsequently, we simplify this mesh to create collision bodies. This approach effectively captures intricate details while maintaining accurate planar reconstruction. For foreground objects, we employ ARCode [28], which can automatically segment the object, balancing usability and quality.

Post-processing. Reconstructed collision bodies often have voids or sharp edges due to imperfect image quality, causing erratic behavior in simulation. For example, objects may fail to rest stably on surfaces. Void filling and surface smoothing help improve stability and realism in simulation. As acquiring physical parameters like mass and friction is challenging with only initial static visual data, we use default values to simplify the process without sacrificing performance, and find it works well for many manipulation tasks.

C. Hybrid Visual Rendering

Color images, as a key type of perception signal, often yield a large visual gap between simulation and the real world. To close such a gap, we render the foregrounds and backgrounds with hybrid real-to-sim rendering techniques. For the foreground objects, object renderings are applied from the mesh with ray tracing support. Background reconstruction contains most parts of the scenes and applies more realistic rendering using 3DGS. Z-buffer rendering is applied to mix objects according to ground-truth depth. Notably, hybrid visual rendering supports multi-view rendering according to the perspective views. The rendering time consumption of each component is shown in V.

D. Real-World Alignment

Despite the prior alignments between rendering and meshes, another key step is to align with real-world scenarios, specifically, the positions of the background/foreground relative to the robots. For foreground objects, we define a range of possible placements in the simulation. Regarding the background, we leverage the ArUco marker on the table to align the 3DGS coordinates with the marker coordinates. Then the ICP [29] post-processing is applied to optimize the coordinate alignment gap. Specifically, ICP optimizes the relative transformation between the partial point cloud obtained from the depth camera and the complete point cloud sampled from the reconstructed mesh.

E. Expert Data Collection

To verify the effectiveness of the real-to-sim-to-real system, following the designed task scenarios such as pick-and-place, we collect large-scale demonstrations using the script policy. During each rollout, we introduced domain randomization, including source/target object randomization, and robot arm's base position randomization. Define an expert policy $\pi_{\text{priv}}(a_t|o_{\text{priv},t})$ based on privileged information $o_{\text{priv},t}$, such as the exact pose of target objects. We can then make use of π_{priv} to interact with the simulator and generate observation-action pairs (o_t, a_t). Here, o_t represents observations that are accessible from the real world, including images and proprioception data. Specifically, we compute the 6D pose of the gripper at some key steps and use the motion planner based on the RRTConnect [30] algorithm to plan the path between the key steps in the joint space. During data generation, reject sampling is applied to filter out failed rollouts, improving the dataset quality and ensuring reliability. We can continue to generate such data with the amount according to our needs, obtaining the simulated datasets $\mathcal{D}$.

IV. Experiment

We mainly investigate the following three research questions based on the proposed RE^{3}SIM system:

- **Q1** Can RE^{3}SIM generate high-fidelity simulation data with small 3D sim-to-real gaps to benefit real-world manipulation problems?
- **Q2** How does large-scale simulation data help in more challenging real-world tasks?
- **Q3** Can RE^{3}SIM rapidly construct simulated scenes and synthesize data at low cost?

A. Experimental Setups

Hardware platform. In this paper, we evaluate RE^{3}SIM with a Franka Research 3 robot equipped with a parallel gripper. Two Realsense D435i depth cameras capture visual observations—one mounted on the end-effector and another positioned beside the robot for a third-person view.

Policy model and training details.

During our sim-to-real experiments, we adopt an ACT-structured policy [31] with DINOV2 [32]. The images in the dataset are compressed using JPEG format. During training, we apply visual augmentations, and for evaluation, the temporal ensemble [31] technique is used to smooth the final action for rollouts.

B. Zero-Shot Sim-to-Real Experiments

Q1 is the main question that motivates us to build RE^{3}SIM, thus we design three table-top manipulation tasks to investigate the usage of RE^{3}SIM:

1.`Pick and drop a bottle into the basket.` The robot must pick an orange bottle randomly placed within a 25cm $\times$ 35cm area on a table and place it into a basket.

2.`Place a vegetable on the board.` A cucumber model is placed on the table in an area of 35cm $\times$ 50cm, and the cutting board's position is varied within a 35cm $\times$ 80cm area. The robot must grasp the vegetable and accurately place it on the cutting board.

3.`Stack blocks.` The robot is tasked with stacking three black cubes on the table. Each cube is placed randomly within a 30cm $\times$ 10cm area separately, and the robot must grasp and stack them.

Tasks are rebuilt in simulation using RE^{3}SIM, with 100 trajectories collected to train independent policies. As a baseline, we also adopt a rule-based policy that prioritizes picking objects closer to the robot base to reduce action variability.

Visual quality evaluation. An open-loop visual comparison is conducted by selecting a trajectory from the simulation dataset, replayed in the real world to capture ground-truth images with calibrated cameras. Visual similarity is evaluated from wrist and third-person perspectives, with qualitative results in Figure 3. These confirm that RE^{3}SIM achieves high-quality visuals through accurate reconstruction and precise alignment.

We also compare with other reconstruction methods. Visual comparisons of reconstruction results from more methods are shown in Figure 4. PolyCam tends to blur details in certain areas, OpenMVS's reconstructed mesh texture shows cracks, while 3DGS produces high-quality renderings. Table I reports PSNR and SSIM for the visual sim-to-real gap, comparing RE^{3}SIM with OpenMVS and PolyCam [33]. Alignment error still causes minor pixel-level discrepancies, leading to a absolute lower PSNR and SSIM value in texture-rich scenes. 3DGS in RE^{3}SIM surpasses PolyCam in both metrics and achieves higher SSIM than OpenMVS, despite comparable PSNR. OpenMVS's PSNR is slightly higher than 3DGS's, likely because the background and crack colors are similar, making PSNR less sensitive. However, the cracks cause increased variance within patches, leading to a lower SSIM compared to 3DGS. These results validate RE^{3}SIM's rendering design for minimizing visual gaps.

Sim-to-real results. The real-world success rates of policies trained on simulation data from RE^{3}SIM are evaluated, compared with a real-to-sim baseline, RialTo [33], and policies trained on 50 real-world trajectories. Doubling the simulation data yields performance comparable to real-world data, indicating a minimal sim-to-real gap arising from differences in scene initialization and trajectory collection. For comparison, AnyGrasp [34], a state-of-the-art grasping method, is implemented with scripted primitives for the `pick and drop a bottle into a basket` task. AnyGrasp predicts only grasp poses, limiting its applicability to other tasks with non-fixed placement locations.

The numerical results listed in Table II demonstrate that the data generated by RE^{3}SIM help the policy achieve zero-shot sim-to-real transfer, showing strong performance even compared with the existing real-to-sim and state-of-the-art grasping method. Furthermore, policies trained on our extensive synthetic dataset can achieve comparable or even slightly better performance than those only trained on real-world data, indicating the effectiveness and huge potential of high-fidelity simulation data.

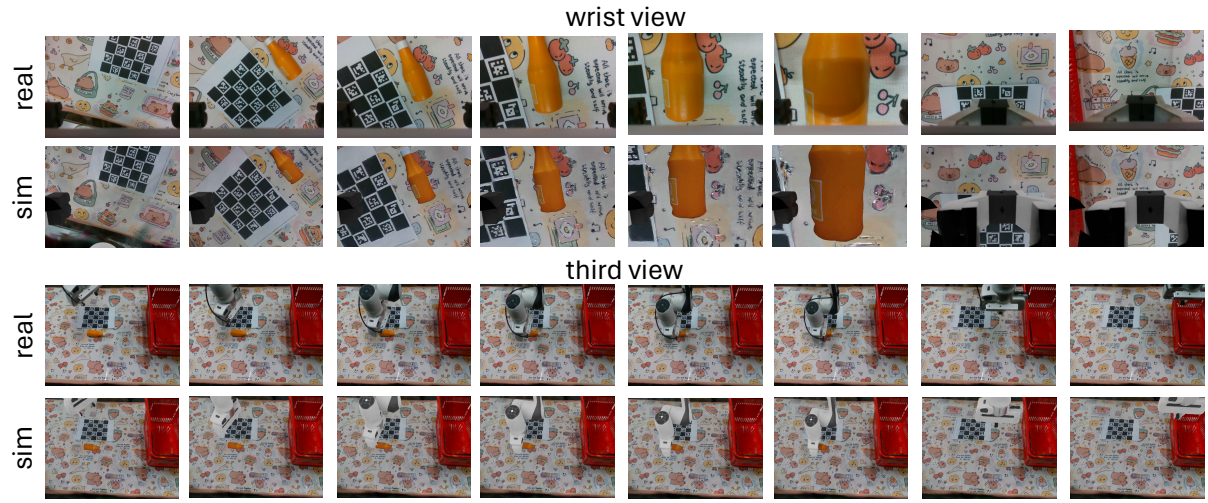


Fig. 3: **Visual comparison between real and simulation.** Rendering results from our hybrid rendering method compared with photos captured by real-world cameras, highlighting the high fidelity and realism achieved by our approach.

Fig. 4: **Visual comparison of rendering approaches.** Rendering results of reconstruction outputs from PolyCam, OpenMVS, and 3DGS, compared with real-world photos.

C. Large-Scale Sim-to-Real Experiments

To push the limit of utilizing synthetic data for real-world manipulation problems by answering **Q2**, we choose a `clear objects on the table` task: the robot has to clean up the table by picking all items that are randomly placed on the table in a 40cm $\times$ 50cm area, and put them into the basket. This is a long-horizon task requiring the policy to sequentially pick both seen and unseen objects one by one. To provide a sufficient evaluation, we design four different setups:

- *Seen*: four seen objects (the bottle, cucumber, corn, and eggplant) included in the data for training.
- *Unseen*: four objects (the green pepper, banana, red bowl, and momordica charantia) that are unseen during training.
- *Cluttered*: all eight objects from *seen* and *unseen*.
- *Darkness*: the testing brightness is significantly lower than that of the simulation data.

All models are evaluated on four *seen* objects. The results are presented in Fig. 5(a). Each setup was tested 10 times, with two additional automatic grasp attempts per object in case of failure.

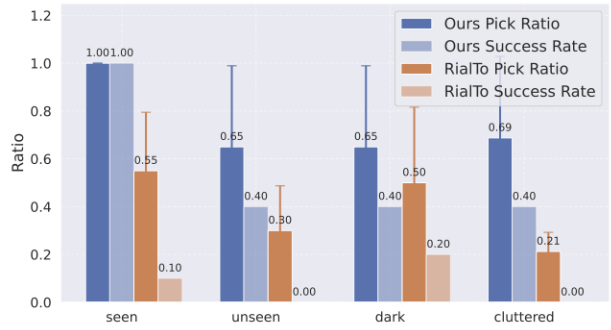
TABLE I: **Background rendering methods comparison.** Quantitative results of the reconstruction result in terms of PSNR and SSIM. Results are computed from 1029 robot arm positions in the wrist view.

Background Rendering	PSNR	SSIM
Polycam	11.52 $\pm$1.40	0.34 $\pm$0.04
OpenMVS	**13.40$\pm$0.96**	0.27 $\pm$0.03
3DGS	**13.29 $\pm$1.11**	**0.37 $\pm$0.04**

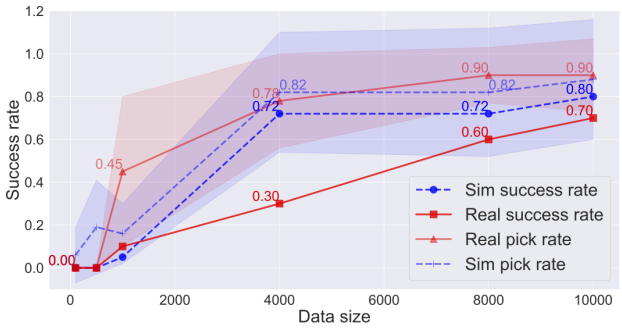
Result analysis. Despite training on only five objects, the policy generalizes effectively to grasping novel objects of similar size, regardless of shape or color variations. We hypothesize that the robot exploits the static scene, detecting objects via background color contrasts. Additionally, objects with varying shapes often have similar grasping positions, enabling the robot to execute successful grasps. Larger datasets with multiple objects enhance the policy's success rate, whereas smaller datasets limit its performance. Compared with RialTo-IL [33], policies trained on data from RE^{3}SIM exhibit superior performance across all settings. Moreover, the policy shows

TABLE II: **Quantitative results of real-to-sim-to-real evaluation.**Real-world manipulation tasks are tested 20 times each, without retries on failure. Policies are trained with 100 simulation episodes (RialTo+IL, RE^{3}SIM +IL) or 50 real-world episodes (Real+IL). AnyGrasp with scripted primitives (AnyGrasp+Prim) is evaluated in the scenario with fixed placement locations.

Real-world Tasks	RialTo+IL	AnyGrasp+Prim.	Real+IL	RE^{3}SIM +IL
Pick and drop a bottle into the basket	0.4	0.9	**0.8**	0.75
Stack cubes	0	-	0.15	**0.25**
Place a vegetable on board	0.45	-	0.6	**0.75**



(a) Real-world test for large-scale sim-to-real.



(b) Data scaling effects.

Fig. 5: (a) *Success rate* denotes the proportion of trials where all objects were grasped, while *pick rate* represents the proportion of objects grasped relative to the total on the table. (b) The success rate and pick rate were tested on seen objects in the real world. See qualitative results in the website.

TABLE III: **Human effort in reconstruction.** Estimated scene reconstruction times at the table level, alongside human effort for object reconstruction using ARCode.

Input Types	Video	Images	ARCode
Human Efforts (s)	51.5	84.5	60.5

robust performance under varying lighting conditions.

We further explored how the success rate in simulation and real-world settings changes with varying amounts of data. As shown in Fig. 5(b), increasing the size of the synthetic dataset generated by RE^{3}SIM leads to a significant improvement in imitation learning performance. When the amount of simulation data is limited, the real-world performance is almost negligible. Doubling the data size often results in a significant improvement in success rate until convergence at a high performance.

TABLE IV: **Time cost for simulation data collection.** Time to collect 100 simulation episodes per task using eight RTX 4090 GPUs.

Tasks	Time Cost (minutes)
Pick and drop a bottle into the basket	12.35
Place a vegetable on the board	13.78
Stack blocks	6.45

TABLE V: **Average time consumption per step.** Breakdown of time consumed by each process.

Process	Time (ms)
Physics Simulating	26.64
Gaussian Splatting Rendering	12.93
Motion Planning	0.36
Others	1.53
Total Time	41.46

D. Sim-Real Consistency

We evaluated the consistency of policy performance between simulated and real-world environments using policy ACT. In this experiment, we evaluate policies trained at different stages (from 0 to 120000 training steps) and with 3 different seeds on the `pick and drop a bottle into a basket` task.

As illustrated in Fig. 1 (d), the Pearson correlation coefficient of RE^{3}SIM is 0.924, while that of RialTo is 0.710, indicating the correlation between performance in simulation and the real world of RE^{3}SIM is better. Specifically, models that achieve higher success rates in simulation tend to exhibit similar success rates in real-world testing. When the success rate in simulation is low, it is often difficult for the model to perform well in real-world scenarios. This could be due to alignment errors, leading to a small sim-to-real gap, or the fact that the number of real-world test instances is much lower than in simulation, potentially introducing bias. This consistency suggests that policies trained with RE^{3}SIM are well-suited for real-world deployment, reflecting a minimal sim-to-real gap.

E. System Efficiency

To validate **Q3**, we record and compute the time of RE^{3}SIM to reconstruct both the background and objects, along with the time of generating demonstrations.

Human effort during reconstruction. In RE^{3}SIM, background reconstruction only requires a set of images or a short video, plus a single depth-aligned image. While the process is straightforward, it still involves some human effort. We report average reconstruction times for both foreground

TABLE VI: **Ablation on foreground rendering.** Task success rates for 3DGS vs. ARCode rendering.

Real-world Tasks	Pick and drop a bottle into the basket	Place a vegetable on board
3DGS	0.70	**0.77**
ARCode	**0.75**	0.75

and background, based on five users with basic knowledge of modern phones or cameras. The workflow is simple: (1) capture photos or video using a phone or camera, and (2) follow ARCode's guided interface. Users typically need just 1–3 practice runs to become proficient. As shown in Table III, video input saves time but may slightly reduce quality due to motion blur, while images yield better results with slightly more effort.

Data collection cost. Table IV shows the time required to collect 100 simulation episodes for each task in Section IV-B, using 8 RTX 4090 GPUs. The time required for data collection is much lower than the time needed for teleoperation in real-world scenarios, especially considering RE^{3}SIM only involves machine runtime, and the latter mostly requires the time of data collection experts. This demonstrates our capability to efficiently generate large-scale simulation data at minimal cost. The breakdown of the time consumption of each step is detailed in Tab. V.

Time consumption breakdown. We measured the average time spent on each phase during the data collection of the *pick and drop a bottle* task in the simulator over 60 steps, as shown in Tab. V. The reconstructed mesh, with numerous vertices and faces, demands more time for physics simulation. Rendering 480p images from two camera views using 3DGS adds 12.93 milliseconds, representing one-third of the total time per step. RE^{3}SIM operates at approximately 24 frames per second (FPS) with two cameras, ensuring real-time performance and visual fidelity.

F. Impact of Mesh-Based vs. 3DGS Rendering Methods

We explore using 3DGS for foreground rendering as an alternative to ARCode-reconstructed meshes. Leveraging the GSDF [35] framework, we generate aligned mesh–3DGS pairs for integration into our hybrid system. As reported in Table VI, substituting ARCode with 3DGS offers no appreciable benefit in task success. In light of this, we continue to use ARCode-based meshes as the default in our main experiments.

V. RELATED WORK

Sim-to-real. Sim-to-real transfer requires techniques to enable policies to successfully adapt from simulation to the real world. The most direct approach is improving simulators [16], [18], which reduces the sim-to-real gap. Other methods, such as domain randomization [36], [37] and system identification [38], [39], also aim to bridge this gap.

Real-to-sim-to-real. Many recent works leverage real-world data to enhance simulation models. Reconstruction methods integrated with grasping techniques, such as Evo-NeRF [40] and LERF-TOGO [41], enable the grasping of objects using only RGB images. Meanwhile, GaussianGrasper [42] and GraspSplats [43], use 3D Gaussian splatting [13] for fast rendering and explicit representation in robotics tasks. Additionally, methods like URDFormer [44] and Articulate Anything [45] use a single image to reconstruct the environment directly, allowing for collecting large amounts of data for imitation learning or reinforcement learning. These approaches enhance data by varying articulations and leveraging various articulations from simulation datasets to train models deployable in the real world.

With the help of 3DGS works like RoboStudio [46], SplatSim [47] and RoboGSim [48] use multi-view images or video for world reconstruction, improving multi-view rendering quality with minimal cost. These methods are especially effective in manipulation tasks involving multiple objects or occlusions. 1) RoboStudio focuses on reconstructing the URDF of a robot, offering a photorealistic rendering result and an accurate collision mesh. 2) SplatSim utilizes pre-obtained 3D models of objects and backgrounds to collect trajectories in the physics simulator and then re-render them with 3DGS to reduce the visual gap between simulated and real-world environments, but acquiring the 3D models is hard for many tasks. 3) RoboGSim compares rendering quality, validates high-quality rendering results for novel pose synthesis, and shows the potential for evaluating various manipulation algorithms. However, its sim-to-real validation remains limited. Different from them, RE^{3}SIM reconstructs both geometric and visual aspects with small gaps, and validates robot policies trained on simulated data through extensive experiments in the real world.

VI. CONCLUSION

We introduced RE^{3}SIM, a novel Real-to-Sim-to-Real system that integrates Gaussian splatting with NVIDIA Isaac Sim's PhysX engine, improving scene reconstruction and sim-to-real transfer for robotic manipulation tasks. Extensive experiments demonstrate its efficacy and scalability, showing that RE^{3}SIM significantly reduces the sim-to-real gap and enables reliable real-world task performance. Models trained on large simulated datasets achieved competitive success rates compared with those trained on real-world data, highlighting the potential RE^{3}SIM to generate diverse, high-quality data for pre-training large-scale robot models.

Limitations. Currently, RE^{3}SIM does not support the reconstruction of deformable objects or liquids. The system also relies on manually defined physics parameters rather than automated system identification. Furthermore, rule-based policies for data collection become increasingly impractical as task complexity grows. Addressing these limitations is left to future work.

ACKNOWLEDGMENT

The SJTU team is supported by National Natural Science Foundation of China (62322603).

REFERENCES

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[2] C.-L. Cheang, G. Chen, Y. Jing, T. Kong, H. Li, Y. Li, Y. Liu, H. Wu, J. Xu, Y. Yang, H. Zhang, and M. Zhu, "Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation," *arXiv preprint arXiv:2410.06158*, 2024.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[6] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, "Open-television: Teleoperation with immersive active visual feedback," *arXiv preprint arXiv:2407.01512*, 2024.

[7] J. Aldaco, T. Armstrong, R. Baruch, J. Bingham, S. Chan, K. Draper, D. Dwibedi, C. Finn, P. Florence, S. Goodrich *et al.*, "Aloha 2: An enhanced low-cost hardware for bimanual teleoperation," *arXiv preprint arXiv:2405.02292*, 2024.

[8] S. Yang, M. Liu, Y. Qin, D. Runyu, L. Jialong, X. Cheng, R. Yang, S. Yi, and X. Wang, "Ace: A cross-platfrom visual-exoskeletons for low-cost dexterous teleoperation," *arXiv preprint arXiv:240*, 2024.

[9] A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023.

[10] A. Khazatsky *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," 2024. [Online]. Available: https://arxiv.org/abs/2403.12945

[11] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *CoRL*, 2023.

[12] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. Fan, and Y. Zhu, "Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning," 2024.

[13] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[14] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *CVPR*, 2016.

[15] D. Cernea, "OpenMVS: Multi-view stereo reconstruction library," 2020. [Online]. Available: https://cdcseacave.github.io/openMVS

[16] NVIDIA, "Nvidia isaac sim," 2021. [Online]. Available: https://developer.nvidia.com/isaac-sim

[17] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A simulated part-based interactive environment," in *CVPR*, June 2020.

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

[19] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *ECCV*, 2016.

[20] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," *ECCV*, 2018.

[21] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[22] J. F. Blinn and M. E. Newell, "Texture and reflection in computer generated images," *Commun. ACM*, vol. 19, no. 10, p. 542–547, Oct. 1976. [Online]. Available: https://doi.org/10.1145/360349.360353

[23] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[24] NVIDIA, "Nvidia physx," 2021. [Online]. Available: https://nvidia-omniverse.github.io/PhysX/

[25] R. L. Smith., "Open dynamics engine (ode)," https://ode.org, 2001.

[26] A. Guédon and V. Lepetit, "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," in *CVPR*, 2024, pp. 5354–5363.

[27] Polycam, "Polycam," *https://poly.cam*, 2020.

[28] A. Code, "Ar code," *https://ar-code.com/*, 2022.

[29] A. Censi, "An icp variant using a point-to-line metric," in *ICRA*. Ieee, 2008, pp. 19–25.

[30] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000, pp. 995–1001 vol.2.

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

[32] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[33] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal, "Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation," *Arxiv*, 2024.

[34] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Transactions on Robotics*, 2023.

[35] M. Yu, T. Lu, L. Xu, L. Jiang, Y. Xiangli, and B. Dai, "Gsdf: 3dgs meets sdf for improved rendering and reconstruction," *arXiv preprint arXiv:2403.16964*, 2024.

[36] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *CoRL*. PMLR, 2020, pp. 1162–1176.

[37] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang, "Understanding domain randomization for sim-to-real transfer," *arXiv preprint arXiv:2110.03239*, 2021.

[38] R. Song, S. Gao, and Y. Li, "Sim-to-real in unmanned surface vehicle control: A system identification-based approach for enhanced training environments," in *ICETIS*, 2024, pp. 563–570.

[39] A. Allevato, E. S. Short, M. Pryor, and A. Thomaz, "Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer," in *CoRL*, ser. PMLR, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 445–455. [Online]. Available: https://proceedings.mlr.press/v100/allevato20a.html

[40] J. Kerr, L. Fu, H. Huang, Y. Avigal, M. Tancik, J. Ichnowski, A. Kanazawa, and K. Goldberg, "Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects," in *CoRL*. PMLR, 2023, pp. 353–367.

[41] A. Rashid, S. Sharma, C. M. Kim, J. Kerr, L. Y. Chen, A. Kanazawa, and K. Goldberg, "Language embedded radiance fields for zero-shot task-oriented grasping," in *CoRL*, 2023. [Online]. Available: https://openreview.net/forum?id=k-Fg8JDQmc

[42] Y. Zheng, X. Chen, Y. Zheng, S. Gu, R. Yang, B. Jin, P. Li, C. Zhong, Z. Wang, L. Liu *et al.*, "Gaussiangrasper: 3d language gaussian splatting for open-vocabulary robotic grasping," *arXiv preprint arXiv:2403.09637*, 2024.

[43] M. Ji, R.-Z. Qiu, X. Zou, and X. Wang, "Graspsplats: Efficient manipulation with 3d feature splatting," *arXiv preprint arXiv:2409.02084*, 2024.

[44] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta, "Urdformer: A pipeline for constructing articulated simulation environments from real-world images," *arXiv preprint arXiv:2405.11656*, 2024.

[45] L. Le, J. Xie, W. Liang, H.-J. Wang, Y. Yang, Y. J. Ma, K. Vedder, A. Krishna, D. Jayaraman, and E. Eaton, "Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model," *arXiv preprint arXiv:2410.13882*, 2024.

[46] H. Lou, Y. Liu, Y. Pan, Y. Geng, J. Chen, W. Ma, C. Li, L. Wang, H. Feng, L. Shi *et al.*, "Robo-gs: A physics consistent spatial-temporal model for robotic arm with hybrid representation," *arXiv preprint arXiv:2408.14873*, 2024.

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

[48] X. Li, J. Li, Z. Zhang, R. Zhang, F. Jia, T. Wang, H. Fan, K.-K. Tseng, and R. Wang, "Robogsim: A real2sim2real robotic gaussian splatting simulator," *arXiv preprint arXiv:2411.11839*, 2024.