Automatic Real-to-Sim-to-Real System through Iterative Interactions for Robust Robot Manipulation Policy Learning with Unseen Objects

Minjae Kang[1], Hogun Kee[1], Hosung Lee[2], and Songhwai Oh[1]

***Abstract*— Real-to-sim-to-real systems have been studied to overcome the challenges of robot policy learning in the real world by creating a virtual environment that mimics the actual workspace. However, previous studies have limitations, requiring human assistance, such as observing the workspace with a hand-held camera or manipulating objects with a hand. To solve these limitations, we propose a novel real-to-sim-to-real framework, ARIC, that performs without human help. First, ARIC observes real objects by repeatedly changing the object poses through the pre-trained robot policy via reinforcement learning. Through iterative interactions between the robot and the environment, ARIC gradually improves the accuracy of 3D object reconstruction. Next, ARIC learns task-specific robot policies in simulation using replicated objects and applies the policies to real-world scenarios without fine-tuning. We confirm that ARIC efficiently learns robotic tasks by achieving a success rate of 83.3% on average for three real-world tasks.** [1]

I. Introduction

In recent years, there has been active research in robotics focused on developing control policies through deep learning [1]–[3], including manipulation tasks using real robots [4], [5]. However, training robots for real-world scenarios poses numerous challenges. First, learning tasks from scratch in the real world typically demands considerable time and human intervention, such as resetting the workspace and determining task success. Although robotic teleoperation systems are employed to collect real-world data [6]–[8], they require significant human effort, and the data quality can vary widely depending on the skill of the user. In addition, if the task is changed or unknown objects are used, additional human assistance is needed for further training of robotic policies.

To address these challenges, recent studies aim to learn robotic policies for real-world scenarios through real-to-sim-to-real approaches [9], [10], a pipeline that combines real-to-sim and sim-to-real frameworks. The real-to-sim phase replicates the actual objects into a virtual environment through 3D object reconstruction techniques [11]–[13], and the sim-to-real phase learns the task-specific policy in the simulated environment and transfers the learned policies back to the real world. The real-to-sim-to-real system allows a robot to learn manipulation policies by carrying out time-consuming actions efficiently in a virtual environment. However, challenges in the sim-to-real transfer occur due to the visual gap between the real and virtual environments, as well as the inaccuracies in the 3D reconstruction of objects. In addition, accurately reconstructing the 3D shape of objects requires observations from a comprehensive range of angles. However, acquiring a sufficient number of viewpoints is often challenging because the workspace of a manipulator with a fixed base is limited. Consequently, previous real-to-sim-to-real studies have limitations, requiring human assistance, such as observing the workspace with a hand-held camera [10] or manipulating objects with a hand [9].

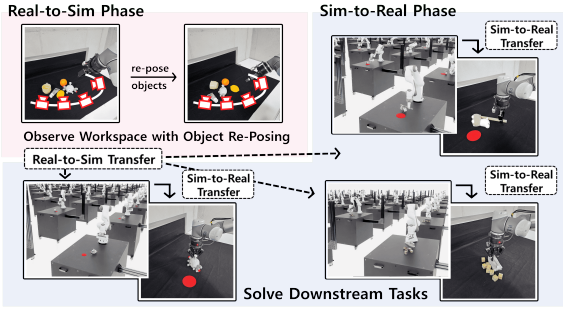

Fig. 1: ARIC consists of real-to-sim and sim-to-real phases. In the real-to-sim phase, ARIC observes the actual workspace to create 3D object meshes by repeatedly re-posing objects. In the sim-to-real phase, ARIC learns a robot behavior policy for downstream tasks in a virtual environment using generated meshes and applies the policy in a real environment.

To overcome this limitation, we propose a novel real-to-sim-to-real framework named ARIC (**A**utomatic **R**eal-to-sim-to-real via **I**terative **C**omplement), which learns robot manipulation policies through the automatic real-to-sim-to-real pipeline, as shown in Figure 1. For the real-to-sim phase, ARIC utilizes Gaussian surfels [14], a Gaussian splatting-based method to reconstruct an object surface accurately using only RGB observations. Next, ARIC identifies incompletely reconstructed objects using the proposed completeness score function. Then, ARIC repeatedly complements incomplete objects by obtaining additional observations from new viewpoints through repositioning objects using a robot. The reposition policy of ARIC is pre-trained through reinforcement learning (RL). Through iterative interactions between the robot and the real objects, ARIC incrementally improves the accuracy of 3D reconstruction for objects within the workspace. On the other hand, in the sim-to-real phase, reconstructed objects are used in the virtual environment, where task-specific policies are trained via RL. ARIC robustly trains robot action policies by randomly

[1] Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul, Korea (e-mail: {minjae.kang, hogun.kee}@rllab.snu.ac.kr, songhwai@snu.ac.kr), [2] Department of Electrical Engineering, Hanyang University, Seoul, Korea (e-mail: hosunglee03@hanyang.ac.kr), *(Corresponding author: Songhwai Oh.)*

This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00480, Development of Training and Inference Methods for Goal-Oriented Artificial Intelligent Agents, 50%, and No. 2019-0-01190, (SW Star Lab) Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning, 50%).

[1]The code will be made available publicly at https://github.com/rllab-snu.

altering task states, such as object positions, orientations, and goal positions within the simulation. Also, ARIC reduces the observation gap between the simulation and the real world by using precisely reconstructed objects. Finally, ARIC transfers robot manipulation policies trained in simulation to real-world tasks without fine-tuning.

In the experiments, we aim to solve the following three downstream tasks with ARIC: Push-to-Goal, Scrubbing, and Push-with-Stick, as shown in Figure 1. Note that the user of ARIC should provide reward and termination functions of the task. As a result, ARIC achieves a success rate of 83.3% on average for three tasks in the real environment, which means that the robot policies learned by ARIC can be successfully applied to real scenarios without any fine-tuning.

In summary, the main contributions of ARIC are as follows: (1) ARIC automatically performs the real-to-sim system without human assistance. (2) ARIC can gradually enhance the accuracy of 3D shapes through iterative interactions. (3) ARIC robustly solves various manipulation tasks without fine-tuning the policy in real-world environments.

II. Related Work

A. 3D Object Reconstruction

3D object reconstruction is the task of reconstructing the three dimensional shape of the target object through data collected from various sensors [11]. In particular, estimating the 3D shapes only with RGB observations is a practical but challenging task. Recent studies have introduced methods based on NeRF [13] or Gaussian splatting [14], which effectively learn scene representations to restore shapes. Typically, 3D object reconstruction methods utilizing multiple images estimate the camera poses through the structure from motion technique [15], which requires numerous highly overlapping image pairs. Conversely, since we use a camera attached to a manipulator, we can directly calculate the camera poses using the poses of the manipulator and a calibration matrix between the end effector and camera.

In ARIC, we employ Gaussian surfels [14] to reconstruct 3D object shapes. The reasons we select Gaussian surfels are as follows. First, Gaussian surfels is a Gaussian splatting-based method that generates high-quality object surfaces with RGB observations from various views. In our problem setting, it is important to obtain a better-quality shape with multiple images rather than reconstructing an object with a small number of images, i.e., one- or few-shot reconstruction [12], since we are able to gather as many observations as necessary. Second, Gaussian surfels utilizes object segmentation masks to specify the target object to be reconstructed. This approach allows us to individually reconstruct each object shape in a scene where several objects exist.

B. Real-to-Sim-to-Real Systems

The real-to-sim-to-real system in robotics is an approach that learns robot policies in a virtual environment that mimics the real world and applies them in the real environment [9], [10]. In real-to-sim-to-real systems, there are challenges, such as restoring the 3D shapes of the observed real objects and overcoming a gap between the simulated and real environments. To address these problems, existing real-to-sim-to-real studies utilize recent 3D object reconstruction methods to generate accurate object meshes. Furthermore, they perform the sim-to-real policy transfer by robustly training policies using domain randomization techniques or by reducing the visual gap using photorealistic simulations.

Ditto [9] proposes a real-to-sim system, which aims to replicate artificial objects with joints through human-object interactions. RialTo [10] utilizes an off-the-shelf scanning app to observe an actual workspace and replicates the environment using a NeRF-based algorithm. Unlike previous works, ARIC focuses on performing a real-to-sim system without human assistance. Since the assumptions of the real-to-sim system and the experimental setting are different, the above works are not used as baselines in experiments.

III. Automatic Real-to-Sim Transfer

We propose a novel real-to-sim-to-real method, ARIC, to learn robot manipulation policies for real-world scenarios. ARIC consists of a real-to-sim phase that replicates real objects into a virtual environment and a sim-to-real phase that learns the robot policy with generated replicas in the virtual environment and applies the learned policy to real-world tasks. We describe the real-to-sim phase in this section, and the sim-to-real phase is explained in the next section.

For the real-to-sim transfer, ARIC observes the real workspace using an RGB camera attached to the robot. The goal of the real-to-sim phase is to reconstruct the 3D object shapes as accurately as possible through interactions between a robot and a real environment without human assistance. To generate accurate object replications, observations of objects from various viewpoints are necessary. However, the workspace of a robot arm with a fixed base is limited, so there are viewpoints that cannot be reached depending on the location of the object. To solve this problem, ARIC collects observations from various viewpoints by repeatedly changing the pose of objects through robotic actions.

As shown in Figure 2, the real-to-sim pipeline of ARIC consists of four modules. First, the reconstruction module converts real-world objects into individual point clouds using RGB observations obtained from various viewpoints. Next, the completeness module estimates the completeness score of each generated point cloud. For each object determined to be incomplete, the manipulation module changes the pose of the object. Finally, ARIC obtains additional observations for objects from new viewpoints, and the complement module incorporate the newly collected point cloud. The above process is repeated until it is determined that the reconstructed point clouds of all objects are completed.

A. Reconstruction Module

The reconstruction module observes the workspace using an RGB camera attached to the manipulator and generates individual point clouds of detected objects. The inputs of this module are RGB images and camera poses, and the output is a set of point clouds of all objects in the workspace. First, we generate object segmentation masks for each observation using pre-trained FastSAM [16] and SAM2 [17] models. We make object masks from the first observation using FastSAM, and for subsequent images, we match the masks for the same objects using SAM2. Next, we use Gaussian surfels [14] to restore the 3D shape of the target object from RGB images.

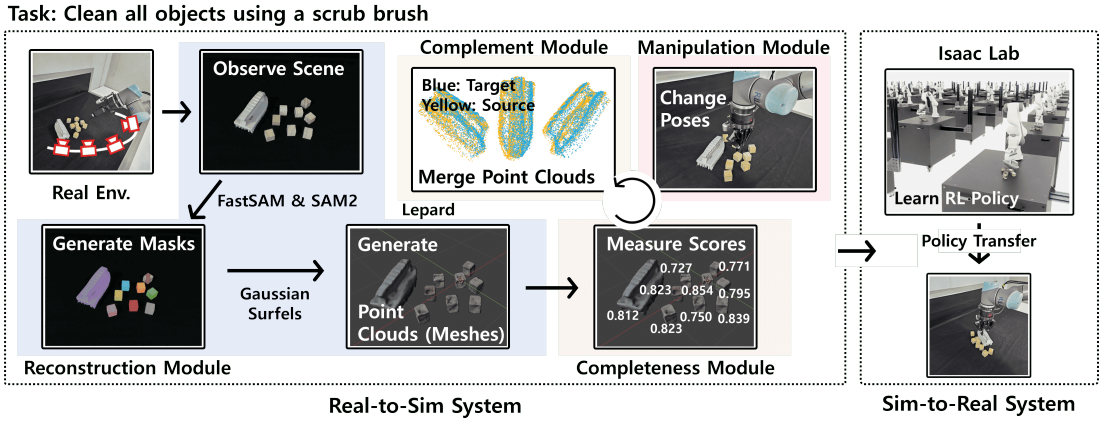

Fig. 2: Overview of ARIC. The real-to-sim phase consists of four modules. (1) The reconstruction module restores the 3D mesh of objects using the observed RGB images and camera poses. (2) The completeness module determines whether the generated meshes are completed. (3) For incomplete meshes, the manipulation module manipulates the corresponding object to obtain additional observations. (4) The complement module improves the mesh using new observations. In the sim-to-real phase, ARIC trains a task-specific policy using generated meshes and transfers the learned policy to the real world.

Gaussian surfels infers a point cloud of the target object using RGBs, segmentation masks of the target object, and camera poses as inputs. However, due to the limited workspace of the manipulator, not enough viewpoints are guaranteed to infer the shape of objects accurately. Therefore, we aim to generate more accurate object point clouds by supplementing the missing viewpoints through a subsequent process.

B. Completeness Module

The completeness module determines the completeness score representing the completeness of the generated object point cloud. The input of the module is an object point cloud, and the output is a real number between zero and one. The process of calculating the score is as follows. First, we obtain the center point of the given point cloud, and all object points are projected onto a unit sphere located at that center point. Next, we use the Fibonacci lattice algorithm to make N Fibonacci points evenly over the surface of the unit sphere. Then, we count the number of valid points among the Fibonacci points that have M_1 or more object points within a distance r_1. N, M_1, and r_1 are constants. Finally, the ratio of the number of valid Fibonacci points to the total number of Fibonacci points is defined as a completeness score, as shown in the second column of Figure 3. However, there are objects for which it is difficult to determine the completeness of the point cloud using the completeness score defined above. For example, for objects with geometric holes, such as cups and bowls, the projected points cannot completely cover the unit sphere, i.e., the score of the perfect object point cloud is not one. To solve this problem, we consider camera poses to compensate for the incomplete score. Since we have the camera poses that have observed the workspace, we can project all camera poses onto a unit sphere. Then, we also count the number of valid Fibonacci points that have M_2 or more projected camera poses within a distance r_2, where M_2 and r_2 are constants. By projecting the camera poses, we are able to cover the area where the hole of the object exists. For instance, we can fill the hole of the cup with projected camera poses, as shown in the third column of Figure 3. Finally, the

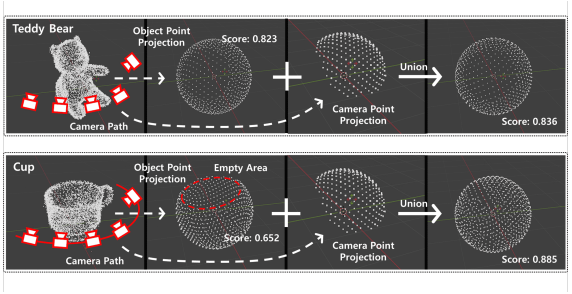

Fig. 3: (Top) The completeness of the generated object point cloud is estimated by projecting points onto a unit sphere. (Bottom) For objects with holes, such as cups, an empty area can occur. The completeness score is inferred reasonably by considering the camera poses as well.

completeness score of the object is calculated using a union of two valid Fibonacci point sets, as shown in the last column of Figure 3. If the calculated completeness score of the object exceeds a predefined threshold, it is judged to be complete. In the experiment, we set the values of N, M_1, r_1, M_2, and r_2 to 1000, 10, 0.10, 1, and 0.10, respectively.

C. Manipulation Module

The manipulation module generates robotic behaviors that change the positions and orientations of objects, which are determined to be incomplete. The input of the module is the observation of the workspace, and the output is the push action of the robot. To supplement the point cloud of an incomplete object, ARIC needs observations from new viewpoints. Through RL in the simulation, ARIC learns the robotic re-posing policy with push actions, which changes the arrangement of objects to obtain additional observations without human assistance. We describe the detailed explanation of re-posing policy learning in Section V-A.

D. Complement Module

The complement module merges the existing and newly obtained point clouds of a target object into one. The input of

the module is two point clouds for the same object, and the output is a transformation matrix between point clouds. This problem is called point cloud registration, and the goal is to find a scale $c \in \mathbb{R}$, translation vector $\mathscr{T} \in \mathbb{R}^3$, and rotation matrix $\mathscr{R} \in \mathbb{R}^{3\times3}$ applied to the source point cloud that best matches the target point cloud.

Practically, since obtained point clouds are partially observed and noisy, using typical point cloud registration methods, such as RANSAC [18] and ICP [19], resulted in inaccurate outcomes. Hence, we utilize Lepard [20], a learning-based method that can infer the transformation matrix between two point clouds with less overlap. Finally, ARIC merges old and new point clouds through Lepard. The merged point cloud is determined whether or not it is completed through the completeness module.

The above process is repeated until all completeness scores of objects achieve a predetermined score threshold, or the number of performed robot actions exceeds the maximum number. The point clouds generated by Gaussian surfels include 3D positions, normal directions, and colors so that they can be converted into textured 3D meshes. Like Gaussian surfels, we generate meshes through the Poisson surface reconstruction method. The generated meshes are used for robot policy learning for downstream tasks in the sim-to-real phase, which is described in the next section.

IV. Sim-to-Real Transfer

In this section, we focus on learning object manipulation policies in simulation using replicated object meshes generated through real-to-sim transfer in Section III. We use Isaac Lab [21], the NVIDIA Isaac Sim-based robot learning framework, for the following reasons. First, Isaac Sim is a GPU-accelerated simulator that enables parallel learning for multiple agents. Time-consuming robotic behaviors make learning robot policies difficult, but in Isaac Sim, efficient RL can be possible by simultaneously demonstrating multiple robot actions. Second, with the photorealistic rendering system of Isaac Sim, we can reduce the visual gap between simulations and real-world environments. Through the delicately replicated objects of ARIC and the high rendering quality of Isaac Sim, we can transfer the manipulation policy learned in the simulation to the real world without fine-tuning.

To robustly train robot policies, we construct RL environments with variations in the simulation. Instead of transferring the objects into the simulation with the same poses as the real world, we rearrange objects on the virtual workspace with random positions and orientations. Also, for tasks in which a goal region exists, the goal position is randomly changed. It enables ARIC to obtain a robust robot policy that can successfully solve the target task even when the arrangement of objects is changed in the real world.

V. Experiment

In this section, we aim to verify whether ARIC can successfully perform the real-to-sim-to-real pipeline for real scenarios through experiments. Therefore, we focus on answering the following questions: (1) Can the re-posing task of ARIC be successfully solved through RL in the simulation? (2) Can ARIC improve the accuracy of the 3D object reconstruction through iterative interactions between the manipulator and real-world workspace? (3) Can ARIC carry out diverse downstream manipulation tasks in the real-world environment?

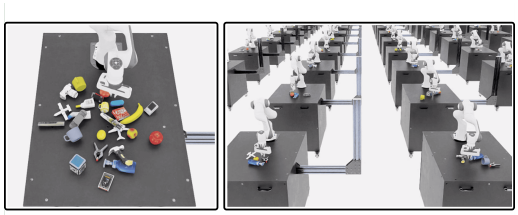

Fig. 4: (Left) 3D artificial objects used for the learning of the re-posing task. (Right) Visualization of the Isaac Sim environment for the re-posing task.

A. Target Object Re-Posing Experiment

In this experiment, we learn the robot push policy for the re-posing task. The goal of the re-posing task is to change the object arrangement, so that the target object can be observed at different viewpoints. We implement the environment for re-posing task learning in Isaac Sim, as shown in Figure 4, and learn the re-posing policy through RL.

Observation Space. The robot push policy uses a set of image observations as an input, and the image observations consist of a top-view RGB observation (three channels) and current and initial masks of the target object (one channel for each), with a total of five channels. The initial mask is used to represent the initial orientation of the target object. To effectively perform sim-to-real policy transfer, we use a background-removed RGB image as an observation.

Action Space. We define the push action as a 3D real vector (x, y, θ). x and y represent the 2D coordinates of the pushing point on the workspace, and θ indicates the pushing direction. The pushing distance and the height of the end effector of the robot are fixed.

Reward and Termination Functions. To secure the viewpoints of the target object, we aim to isolate the target from other objects. The success condition of the re-posing task is that the distances between the target and all other objects are farther than a distance threshold, and the difference between the current and initial orientations of the target is more than a rotation threshold. The change in orientation δ_{diff} is calculated as $\delta_{diff} = 1 - (q_{curr}^T \, q_{init})^2$, where q_{curr} and q_{init} are the current and initial quaternions of the target object, respectively. On the other hand, if the target leaves the workspace or the number of performed actions exceeds the maximum number of actions, the episode ends in failure. For the reward function, the agent only gets a positive reward when the task is successfully terminated. Also, to prevent meaningless push actions, if the agent fails to change the pose of any object, the agent receives a negative penalty.

Experimental Setting. We set the distance threshold to 12 cm and the rotation threshold to 0.2. In the re-posing task, we use 25 objects of diverse categories selected from the YCB [22] and Shapenet [23] datasets. In each episode, one to five objects are randomly used in the workspace. The RL agent is trained through TAC [24], a maximum entropy actor-critic RL algorithm. Also, we create 256 environments to learn the re-posing policy efficiently.

Fig. 5: (Left) Everyday objects used in 3D reconstruction. (Right) 3D reconstruction with a re-posing policy.

Experiment Result. The agent achieves an average task success rate of 84.6%. This result indicates that the learned policy successfully solves re-posing tasks. Furthermore, for most failure cases, we can get new viewpoints of the target because the pose of the target object has been changed.

B. 3D Reconstruction Experiment

In this experiment, we measure the 3D object reconstruction error of ARIC using general objects placed in a real workspace, as shown in Figure 5. ARIC restores the 3D shape of unseen objects by repeatedly interacting with objects using a pre-trained re-posing policy. Through this experiment, we aim to evaluates the 3D reconstruction error according to the number of performed re-posing actions.

Metric of 3D Reconstruction Error. We define the 3D reconstruction error as the chamfer distance between the correct object point cloud and the generated point cloud through ARIC. However, since there is no ground truth point-cloud for everyday objects, we produce a correct point cloud of each object using collected depth images.

Experimental Setting. We perform 3D object reconstruction experiments using 20 everyday objects in various categories. We make five batches, each consisting of four objects and carry out the reconstruction experiment for each batch. First, ARIC reconstructs the point clouds in the initial state. Next, ARIC selects the object with the lowest completeness score as the target object and performs a re-posing action. ARIC repeats the above process four times, recording the chamfer distances and completeness scores of all objects.

Experiment Result. The experiment results for 3D reconstruction errors are shown in Table I. As a result, the number of performed re-posing actions increases, the chamfer distance decreases and the completeness score increases. It means that the learned re-posing policy helps to improve the accuracy of 3D object reconstruction through interactions with the environment.

C. Solving Downstream Tasks

In this experiment, we solve three robot manipulation tasks in the real world using ARIC, as shown in Figure 6. The purpose of the experiment is to verify that the robot action policy trained via the real-to-sim-to-real pipeline can robustly solve real-world tasks. The downstream tasks to be solved are as follows: (1) **Push-to-Goal.** The task is to push a target object into a circular goal area (the first column of Figure 6). The pose of the target and the position of the goal region change randomly in each episode. The success condition is that the distance between the target and the goal is less than 4 cm. (2) **Scrubbing.** The task is to clean the target area by scrubbing small objects using a scrub brush (the second column of Figure 6). The number and poses of small objects are randomly determined for each episode. The success condition is to remove all small objects from a target area. (3) **Push-with-Stick.** The task is to push a target object into a circular goal area using a stick (the third column of Figure 6). The pose of the target and the position of the goal region are randomly determined for each episode. The success condition is that the distance between the target and the goal is within 4 cm.

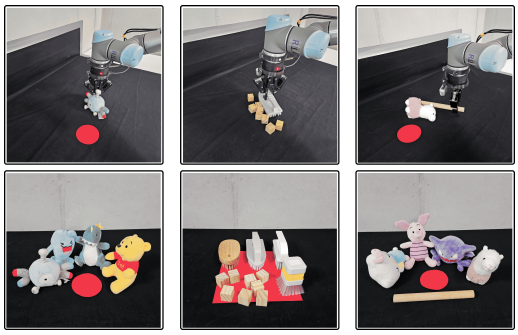

Fig. 6: ARIC solves three manipulation tasks using diverse objects: Push-to-Goal (First column), Scrubbing (Second column), and Push-with-Stick (Last column).

Details for Reinforcement Learning. In sim-to-real transfer, ARIC learns the robot behavior policy through RL. To perform RL, we should define observation and action spaces as well as reward and termination functions. First, for all tasks, the observation space and the action space are the same. The observation is a background-removed tabletop-view RGB image. On the other hand, the action is a 3D robot push behavior, which is the same as the re-posing task.

Next, the user should provide task-specific reward and termination functions. For all tasks, the agent receives a positive reward if the success condition is satisfied. In the scrubbing task, we establish auxiliary rewards, which are determined by the number of objects removed, to encourage the cleaning of the target area. For scrubbing and push-with-stick tasks, which use tools, the agent initially starts holding the tool, and the task fails when the agent drops the tool.

Experimental Setting. Details of the experimental settings are as follows. (1) **Push-to-Goal**. We perform the push-to-goal task with four target objects. The agent should position the target object in a circular goal area with a radius of 4 cm through up to 10 pushing actions. For each object, we measure the success rate over 10 episodes by changing the position of the target object and goal region. (2) **Scrubbing**. We use four scrub brushes and eight wooden blocks for the scrubbing task. The target area is a square of 14 cm on each side. The agent should remove all wooden blocks from the square region with up to 10 pushing actions. For each brush, we record the success rate over 10 episodes by changing the pose of the wooden blocks. (3) **Push-with-Stick**. We use four target objects for the push-with-stick task. Using a wooden stick of 20 cm in length, the agent should position the target object by pushing it into a circular goal region with a radius of 4 cm, with up to 10 pushing actions. For each object, we record the success rate over 10 episodes by changing the position of the target object and goal region.

Experiment Result. The experimental results of ARIC for downstream tasks are shown in Table 2. ARIC shows

Batch Index	Initial Scene	Chamfer Distance (↓) (Completeness Score (↑)) 1 Re-Posing Task	2 Re-Posing Tasks	3 Re-Posing Tasks	4 Re-Posing Tasks
1	1.62 ± 1.01 (0.73 ± 0.12)	1.53 ± 1.32 (0.83 ± 0.13)	1.14 ± 0.85 (0.87 ± 0.09)	1.20 ± 0.58 (0.88 ± 0.10)	1.00 ± 0.77 (0.89 ± 0.09)
2	2.92 ± 1.14 (0.66 ± 0.13)	2.02 ± 0.49 (0.79 ± 0.09)	1.72 ± 0.69 (0.81 ± 0.10)	1.88 ± 0.90 (0.83 ± 0.11)	1.67 ± 0.86 (0.85 ± 0.12)
3	1.78 ± 0.25 (0.72 ± 0.17)	1.41 ± 0.28 (0.80 ± 0.14)	1.38 ± 0.24 (0.85 ± 0.11)	1.16 ± 0.37 (0.88 ± 0.10)	1.21 ± 0.35 (0.91 ± 0.09)
4	2.11 ± 0.56 (0.63 ± 0.10)	1.85 ± 1.01 (0.73 ± 0.09)	1.55 ± 0.81 (0.79 ± 0.12)	1.49 ± 0.69 (0.85 ± 0.11)	1.33 ± 0.89 (0.88 ± 0.09)
5	2.26 ± 0.85 (0.70 ± 0.13)	2.12 ± 1.07 (0.78 ± 0.14)	1.94 ± 0.54 (0.86 ± 0.09)	1.94 ± 0.68 (0.90 ± 0.08)	1.82 ± 0.50 (0.92 ± 0.06)
Average	2.14 ± 0.87 (0.69 ± 0.12)	1.79 ± 0.86 (0.79 ± 0.11)	1.55 ± 0.65 (0.84 ± 0.10)	1.53 ± 0.68 (0.87 ± 0.09)	$\mathbf{1.41 \pm 0.70}$ ($\mathbf{0.89 \pm 0.08}$)

TABLE I: The average of 3D reconstruction errors of real objects according to the number of performed re-posing actions. We record the average chamfer distance (mm) between the object point clouds generated by ARIC and the point clouds obtained from depth images. The numbers in parentheses represent the average of completeness scores.

Env.	Push-to-Goal	Scrubbing	Push-with-Stick
Simulation	0.963 ± 0.021	0.912 ± 0.045	0.869 ± 0.032
Real Env.	0.925 ± 0.050	0.875 ± 0.050	0.850 ± 0.058

TABLE II: Experimental results on downstream tasks of ARIC. In each task, four different objects are used, and for each object, we record average success rates for 200 and 10 episodes in the simulation and real workspace, respectively.

success rates of 92.5%, 87.5%, and 85.0% for Push-to-Goal, Scrubbing, and Push-with-Stick, respectively, in real-world environments. These results imply that ARIC can efficiently and effectively learn various real-world robot tasks with diverse objects through the real-to-sim-to-real system. Meanwhile, in all tasks, the success rate in the real environment is lower than the success rate in the virtual environment. Failures in the actual environment are mainly caused by failing to move the object to the desired place due to the friction between the object and the floor.

VI. Conclusion

In this paper, we propose a novel real-to-sim-to-real system, ARIC. Unlike existing methods, ARIC improves the accuracy of 3D object reconstruction through iterative interactions between a robot and an environment without human assistance. The generated objects are used for robot policy learning in the simulation, and the trained policies are applied to the real world. Through experiments, we show that ARIC can effectively learn several downstream manipulation tasks dealing with diverse objects.

References

[1] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, "RVT: Robotic view transformer for 3D object manipulation," in *Proc. of the Conference on Robot Learning (CoRL)*, Nov. 2023.

[2] S. Chen, R. Garcia, C. Schmid, and I. Laptev, "PolarNet: 3D point clouds for language-guided robotic manipulation," Nov. 2023.

[3] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3D diffuser actor: Policy diffusion with 3D scene representations," in *Proc. of the Conference on Robot Learning (CoRL)*, Nov. 2023.

[4] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "BC-Z: Zero-shot task generalization with robotic imitation learning," in *Proc. of the Conference on Robot Learning (CoRL)*, Dec. 2022.

[5] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3M: A universal visual representation for robot manipulation," in *Proc. of the Conference on Robot Learning (CoRL)*, Dec. 2022.

[6] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," in *Proc. of the Conference on Robot Learning (CoRL)*, Nov. 2024.

[7] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du *et al.*, "BridgeData v2: A dataset for robot learning at scale," in *Proc. of the Conference on Robot Learning (CoRL)*, Nov. 2023.

[8] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," *arXiv preprint arXiv:2403.12945*, 2024.

[9] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building digital twins of articulated objects from interaction," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023.

[10] 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," in *Proc. of the Robotics: Science and Systems (RSS)*, Jul. 2024.

[11] Z. Yang, Z. Ren, M. A. Bautista, Z. Zhang, Q. Shan, and Q. Huang, "FVOR: Robust joint shape and pose optimization for few-view object reconstruction," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023.

[12] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng, "Dreamgaussian: Generative gaussian splatting for efficient 3D content creation," in *Proc. of the International Conference on Learning Representations (ICLR)*, May. 2024.

[13] Y. Wang, X. He, S. Peng, H. Lin, H. Bao, and X. Zhou, "Autorecon: Automated 3D object discovery and reconstruction," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023.

[14] P. Dai, J. Xu, W. Xie, X. Liu, H. Wang, and W. Xu, "High-quality surface reconstruction using Gaussian surfels," in *Proc. of the ACM SIGGRAPH*, Jul. 2024.

[15] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[16] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," *arXiv preprint arXiv:2306.12156*, 2023.

[17] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "SAM 2: Segment anything in images and videos," Apr. 2025.

[18] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[19] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proc. of the international conference on 3D digital imaging and modeling*, 2001, pp. 145–152.

[20] Y. Li and T. Harada, "Lepard: Learning partial point cloud matching in rigid and deformable scenes." *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022.

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

[22] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols," *arXiv preprint arXiv:1502.03143*, 2015.

[23] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Tech. Rep., 2015.

[24] K. Lee, S. Kim, S. Lim, S. Choi, M. Hong, J. I. Kim, Y.-L. Park, and S. Oh, "Generalized tsallis entropy reinforcement learning and its application to soft mobile robots." in *Proc. of the Robotics: Science and Systems (RSS)*, Jul. 2024.