
DexSim: Automating Data Scaling for Sim2Real Generalizable Robot Control

Anonymous Authors¹

Abstract

A critical prerequisite for achieving generalizable robot control is the availability of a large-scale robot training dataset. Due to the expense of collecting realistic robotic data, recent studies explored simulating and recording robot skills in virtual environments. While simulated data can be generated at higher speeds, lower costs, and larger scales, the applicability of such simulated data remains questionable due to the gap of between simulated and realistic environments. To advance the Sim2Real generalization, in this study, we present DexSim, a data engine designed to perform automatic skills simulation and scaling for learning deployable robot manipulation policies. Specifically, DexSim ensures the usability of simulated skills by integrating diverse forms of realistic data into the simulated environment, preserving semantic alignment with the target tasks. For each simulated skill in the environment, DexSim facilitates effective Sim2Real data scaling by automating the process of domain randomization and adaptation. Tuned by the scaled dataset, the control policy achieves zero-shot Sim2Real generalization across diverse tasks, multiple robot embodiments, and widely studied policy model architectures, highlighting its importance in advancing Sim2Real embodied intelligence.

1. Introduction

A key milestone in advancing modern AI systems is the development of embodied intelligence, which aims to seamlessly integrate an agent's physical body, sensory perception, and environment into its learning, reasoning, and decision-making processes (Paolo et al., 2024). In recent years, robots have emerged as a widely studied form of embodiment (Duan et al., 2022). In principle, robot control skills can be highly versatile—generalizing across various do-

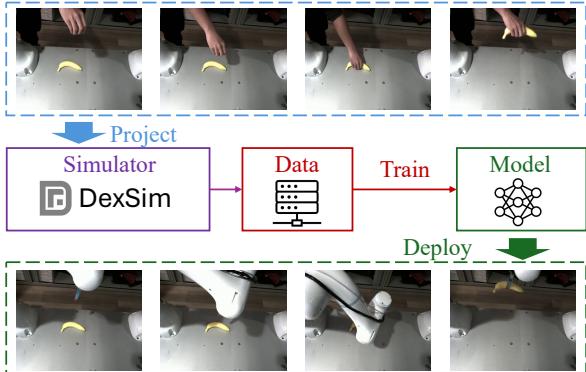


Figure 1: An example of using DexSim. Given a human video, DexSim simulates a robot skill dataset for training deployable control policy without requiring realistic data.

mains and incorporating multiple forms of sensory data. Prioritizing the availability of sensors and the impact of real-world application, we primarily focus on advancing visuomotor robotic learning for dexterous manipulation.

Many recent studies have proposed learning-based systems for manipulating various object based on image sensor signals. For example, modular approaches first predict affordances, such as grasp poses (Mousavian et al., 2019; Sundermeyer et al., 2021; Wu et al., 2020; Jiang et al., 2021; Fang et al., 2023) or keypoints (Manuelli et al., 2019; Qin et al., 2020; Mo et al., 2021; Yuan et al., 2024), for different tasks and train a low-level policies to complete these task. On the other hand, end-to-end approaches aim at learning a policy model that directly maps visual inputs to robot actions methods (Levine et al., 2016; Zhang et al., 2018; Florence et al., 2020; Chi et al., 2023). While these methods have promising performance in specific scenarios, the learned skills lack generalizability to diverse environments, making them unsuitable for deploying in the wild.

With advancements in learning algorithms, large-scale datasets, and hardware systems, large foundation models have achieved human-level performance across a wide variety of environments and downstream tasks. Their success inspired the construction of foundation models for robot control such as RDT (Liu et al., 2024b), Open-VLA (Kim et al., 2024), ACT (Zhao et al., 2023), and Octo (Team et al., 2024). Training these models depends on massive amounts of data. Apart from the text and image data readily available on the internet, learning robotic control policies requires

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

055 multi-modal sensory signals, robot self-states (i.e., proprioception), action trajectories for end-effectors, and body joints, all situated in a calibrated 3D space with real-world physics. It often requires structured pipelines or specially designed equipment to generate such datasets.

056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084 Although previous works have developed real-world data collection systems for robotic action trajectories (Zhao et al., 2023), these systems often rely on costly equipment and substantial manpower. Furthermore, in the absence of standardized data collection protocols, significant effort is required to align datasets collected from different systems. To overcome these limitations, an alternative method is data synthesis based on the simulated robot learning environment. In principle, simulated data generation can produce an infinite amount of robot skills (Wang et al., 2024), significantly addressing the "data hunger" challenge in training large-scale robotic control policy. However, skills trained on simulation data face significant challenges when deployed to real-world environments due to: 1) *Semantic Mismatch*: There is no guarantee that the semantics of the simulated environment align with the target environment. A policy trained in a misaligned simulation is unlikely to be successfully deployed in practice. 2) *Simulation Discrepancy*: Despite substantial advancements in simulators with physics-realistic and photo-realistic rendering (Todorov et al., 2012; Xiang et al., 2020; Makoviychuk et al., 2021), discrepancies between simulated and real-world environments are inevitable, continuing to hinder deployment performance.

085 In this paper, we introduce a data engine, DexSim, to address these challenges and facilitate the training of Sim2Real
086 transferable dexterous manipulation policies based on visual
087 observations. Figure 1 illustrates an example application
088 of DexSim. Given task-descriptive observations (e.g., hu-
089 man demonstration), DexSim generates a simulated dataset,
090 based on which the robot control policy can generalize to
091 realistic tasks in a zero-shot manner.

092 To ensure semantic alignment, DexSim introduces a data
093 projection mechanism that enables practitioners to incor-
094 porate heterogeneous formats of environmental and action
095 priors from the target tasks into the simulator. These priors
096 capture both static features (e.g., object types and layouts)
097 and dynamic features (e.g., human demonstrations and in-
098 teractions with objects), enabling the simulator to generate
099 desired environments for learning preferable skills in an
100 effective and reliable manner.

101 To address the Sim2Real discrepancy, DexSim employs a
102 strategic data scaling approach, incorporating an automatic
103 domain randomization process to generate diverse learning
104 configurations. This enables DexSim to actively discover
105 effective robot control trajectories across a wide range of
106 scenarios. Additionally, to remove task-irrelevant features
107 from the dataset, DexSim adapts visual sensory data from

108 both real and virtual environments into object-oriented and
109 pose-affordance representations. The resulting trajectory
110 dataset facilitates the development of imitation policies that
111 can be seamlessly deployed in real-world applications.

112 We demonstrate that the DexSim pipeline can be seamlessly
113 integrated into the training of widely studied embodied robot
114 policies, including Acting Transformer (Zhao et al., 2023),
115 Diffusion Policy (Chi et al., 2023), and Robot Diffusion
116 Transformer (Liu et al., 2024b). By leveraging DexSim,
117 these methods achieve zero-shot deployment for desired
118 tasks. Such deployments can be effectively scaled across
119 various robot configurations (e.g., single-arm and dual-arm
120 setups with different robot models) and diverse manipulation
121 tasks, including object grasping, articulated object (e.g.,
122 box) manipulation, and table rearrangement. With extensive
123 performance evaluation and case studies, we demonstrate
124 that DexSim can significantly advance the Sim2Real learning
125 of deployable policies across multiple scenarios.

2. Related works

Large-Scale Dataset for Robotic Learning. Building large-scale datasets is essential to training generalizable robotic policies. For instance, to support the development of adaptable robotic controllers, RoboNet (Dasari et al., 2019) and D4RL datasets (Fu et al., 2020) capture rich robot actions across different manipulation tasks. In recent years, to enhance the training of end-to-end foundation models, extensive data collection systems have been built across different scenarios, including object manipulation (e.g., Bridgedata (Ebert et al., 2022; Walke et al., 2023)) and language conditioned and vision-based tasks (e.g., RT-1 (Brohan et al., 2023) and RT-2 (Zitkovich et al., 2023)). To further improve generalization performance, RT-X (O'Neill et al., 2024) introduced the Open X-Embodiment (OXE) datasets, which are cross-domain and were collected from multiple types of robots across numerous institutions. A fundamental challenge in constructing real-world datasets lies in the high costs associated with equipment and manpower.

Environment Simulation for Robotic Learning. Instead of collecting data from real-world, an alternative method is generating robot skills (e.g., action trajectories) from a simulated environment based on physics engines, such as MuJoCo (Todorov et al., 2012), PyBullet (Coumans & Bai, 2016–2021), Issac Gym (Makoviychuk et al., 2021) and Sapien (Xiang et al., 2020). In this context, numerous virtual environments have been proposed for a variety of tasks, including dexterous manipulation (e.g., RL-Bench (James et al., 2020), ManiSkill (Tao et al., 2024), and RoboCasa (Nasiriany et al., 2024)), robot skill imitation (e.g., RoboMimic (Mandlekar et al., 2021)), physical interactions (e.g., ThreeDWorld (Gan et al., 2022)), rearrangement (Batra et al., 2020), high-level reasoning (e.g.,

110 Alfred (Shridhar et al., 2020) and CabiNet (Murali et al.,
 111 2023)), and generative skill learning (e.g., RoboGen (Wang
 112 et al., 2024) and GenSim (Hua et al., 2024)). To better
 113 reflect the realistic control environment, Habitat 3.0 (Puig
 114 et al., 2024) and AI2-THOR (Kolve et al., 2017) enable em-
 115 bodied agents to interact with photo-realistic environments
 116 for navigation and manipulation tasks. The continuously
 117 works ProcTHOR (Deitke et al., 2022), expand AI2-THOR
 118 with procedurally generated environments to enhance general-
 119 ization. These simulators aim to benchmark robotic skill
 120 learning by offering diverse environments and tasks. How-
 121 ever, transferring the learned control skills to real-world
 122 scenarios often requires additional fine-tuning with realistic
 123 data. The development of simulators for zero-shot Sim2Real
 124 deployment remains a challenge.

125 **Sim2Real Generalization.** In the application of robot con-
 126 trol, Sim2Real generalization techniques often involve: 1)
 127 *Domain randomization* tackles Out-of-Distribution (OoD)
 128 scenarios in practical applications augmenting the training
 129 dataset with randomized visual and physical features (Chen
 130 et al., 2022a). To determine the scale of randomization,
 131 recent studies consider updating the distribution of random-
 132 ization parameters by automatic learning (OpenAI et al.,
 133 2019), active exploration (Mehta et al., 2019), Bayesian
 134 update (Muratore et al., 2021b;a), offline inference (Ti-
 135 boni et al., 2023) and continual learning (Josifovski et al.,
 136 2024). 2) *Domain adaption* addresses the gap between
 137 simulated and real-world domains by mapping them into
 138 a feature space. Commonly studied spaces include depth
 139 images (Agarwal et al., 2022; Cheng et al., 2024) and point
 140 clouds (Lobos-Tsunekawa & Harada, 2020; Qin et al., 2022;
 141 Chen et al., 2023; Hua et al., 2024). 3) *Feature alignment*
 142 is essential when the learned policy depends on privileged
 143 information that is inaccessible in real-world applications.
 144 A commonly studied approach is knowledge distillation,
 145 where skills are transferred from the learned policy to a de-
 146 ployable policy using a teacher-student framework (Kumar
 147 et al., 2021; Qi et al., 2022; Agarwal et al., 2022; Cheng
 148 et al., 2024).

149 3. Problem Formulation

150 **Simulation Environments for Robotic Learning.** The goal
 151 of DexSim is to learn robot control skills based on simulated
 152 environments. These environments can be modeled as an
 153 episodic Markov Decision Process (MDP), represented as
 154 $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_{\mathcal{T}}, R, \gamma, \rho_0)$, where:

- 155 • a state $s_t \in \mathcal{S}$ captures the semantic information of a
 156 scene, encompassing the configuration (e.g., layouts, ap-
 157 pearance, and physical characteristics) of various types of
 158 objects and the robots subject to control.
- 159 • an action $a_t \in \mathcal{A}$ define control signals for robot. The
 160 common representations of action include the end effec-
 161 tor pose of the robot, the angle of each joint, the torque
 162

163 applied to each joint, and the velocity of each joint.

- 164 • Transition function $\mathcal{P}_{\mathcal{T}}$ characterizes the impact of robot
 165 action a_t to the configuration of current state s_t , thereby
 166 projecting the s_t to a new scene represented by s_{t+1} .
- 167 • Reward function $R(s, a)$ captures various degrees of opti-
 168 mality of the robotic agent in completing targeted tasks
 169 after acting a in state s . For instance, $R(s, a)$ can assess
 170 "hard" optimality (i.e., sparse rewards), which determines
 171 whether a task is completed (e.g., whether the target object
 172 is successfully grasped) or "soft" optimality (i.e., dense
 173 rewards), which measures the degree of task completion
 174 (e.g., how close the gripper is to the target object).
- 175 • ρ_0 denotes the initial state distribution. Our DexSim sup-
 176 port projecting the initial state from realistic scene or
 177 automatically generating a new scene.
- 178 • $\gamma \in (0, 1]$ denotes the discounting factor which weights
 179 the importance of future rewards relative to immediate
 180 rewards in decision-making processes such that a lower γ
 181 emphasizes immediate rewards more heavily, and a higher
 182 γ gives greater significance to future rewards.

183 Note that we consider an episodic MDP with stationary
 184 policy $\pi(a|s)$, which depends only on the current state and
 185 is invariant to the decision time step t . Additionally, in
 186 many environments, robot sensors only observe partial in-
 187 formation about the underlying states. In this case, a com-
 188 mon method is to embed the historical observations into
 189 the state, such that $s_t = [o_t, o_{t-1}, \dots]$, where o_t represents
 190 the partial observation at time step t . As a data engine,
 191 DexSim supports the automatic discovery of *skills*, denot-
 192 ing the sequence of *action trajectory for finishing a task*
 193 $\tau = [s_0, a_0, \dots, s_T, a_T]$. To acquire these skills, DexSim
 194 automatically decomposes long-term tasks (e.g., cooking a
 195 meal) into atomic tasks (e.g., chopping potatoes). For tasks
 196 with a "hard" optimality (e.g., whether the target object is
 197 successfully grasped), DexSim primarily relies on motion
 198 planning techniques (e.g., antipodal point detection (Chen &
 199 Burdick, 1993) combined with inverse kinematics (D'Souza
 200 et al., 2001)). For tasks with "soft" optimality (e.g., how
 201 close the gripper is to the target object), DexSim supports the
 202 automatic design of dense reward functions (Ma et al., 2024)
 203 and employs Reinforcement Learning (RL) algorithms (Sut-
 204 ton, 2018) to acquire control skills.

205 **Sim2Real Domain gap.** The skills learned in a simulated
 206 environment often face a significant domain gap when ap-
 207 plied to the deployment environment (Xie et al., 2024). In
 208 this section, we systematically analyze the factors that cause
 209 the gap in deploying models trained with simulated data
 210 onto real robots. As it is illustrated in Table 1, we analyze
 211 by inspecting the steps in a simulation-perception-acting
 212 loop. Such analyses would give us insights on how to better
 213 design the simulated robotic learning environment that can
 214 effectively close the Sim2Real gap.

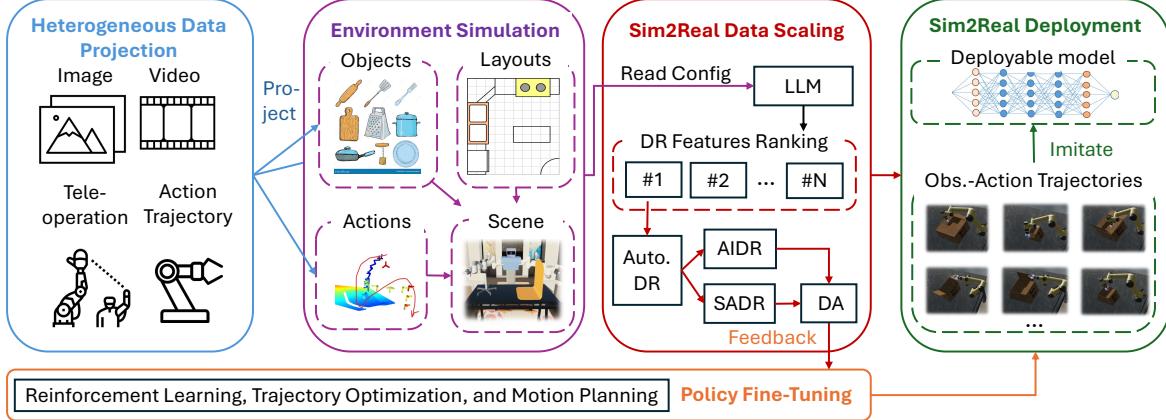


Figure 2: The pipeline of DexSim operates as a data engine, taking task-descriptive data as input and generating a skill dataset to support Sim2Real transfer. This enables the zero-shot deployment of robot policies in realistic environments.

Simulation Stage. A major factor impacting Sim2Real performance is the inaccuracy in simulating real-world scenarios. Notably, the semantic information in the simulated environment often does not align with that of the realistic environment. For instance, objects may be inaccurately scaled, either oversized or undersized, and the location and shapes may not accurately reflect those found in the real world. More importantly, there is no guarantee that the simulated environment can fully cover the realistic semantic distributions, and the disparity between simulated and real-world environments can result in Out-of-Distribution (OoD) scenarios during deployment, making the planned motions less adaptable to the deployment environment.

Perception Stage. A primary application of a simulator is to replicate or reconstruct perception from realistic sensors. A critical factor affecting Sim2Real performance during this process is the inconsistency in hardware setups between the simulated and real environments (e.g., differences in the position, orientation, and field of view of cameras).

Acting Stage. Embodied agents control various types of embodiments (e.g., arm, and hand) by executing robot actions. In this stage, mismatches between embodiments can impact deployment. For example, using different robots during simulation and deployment, or having URDF configuration errors, can lead to discrepancies. More importantly, the physical properties in the realistic environment might misalign with those in the simulated environment, inducing inconsistent outcomes of implementing the same action.

4. Data Engine for Sim-to-Real Generalization

To acquire robot control skills that effectively bridge the aforementioned Sim2Real gap, DexSim can serve as an automated data engine. As shown in Figure 2, DexSim takes descriptive data (e.g., scenes and demonstrations) from the target task as input and generates a scalable dataset that supports the efficient training of Sim2Real-deployable

Table 1: Features that might induce Domain Gap. The determination of such parameters is a recent study of the generalization gap (Xie et al., 2024).

Simulation Stage	Lighting, Table Texture, Background, Distractors, Object Locations, Object Orientations, Object Texture and Object Shape
Perception Stage	Camera Position, Camera Orientation, Camera Field of View
Acting Stage	Robot Configuration, Physics Properties

policies. We detail the design of DexSim in the following.

4.1. Heterogeneous Data Projection

The goal of heterogeneous data projection is to map descriptive observations of the target task from real-world applications to a simulated environment, ensuring the applicability of the generated skills. To achieve this, DexSim primarily supports two key types of projection: scene projection and action trajectory projection, as detailed below.

Scene Projection. To overcome the Sim2Real gap, a critical challenge to handle the semantics disparity between simulated and real-world environments. For example, in the scene constructed by generative simulation, there is no guarantee the distribution of objects and their configurations can semantically align with these in realistic scenarios. To overcome this limitation, DexSim enables projecting the *static* scene information to the simulated environment. Specifically, inspired by Digital Cousin (Dai et al., 2024), DexSim extracts relevant per-object information from the input RGB image. Using this information, DexSim matches each detected object to their "digital cousins" (i.e., similar objects) in the asset dataset. For the articulated objects (e.g., drawers and boxes), DexSim can post-process them to create a fully interactive simulated scene by matching them with CAD

models and the generated objects (Liu et al., 2024a). Besides the automatic scene project, DexSim provides user-friendly interfaces to support manual adjustment of the retrieved objects and the generated scenes. This adjustment is essential if the single-view image can capture only partial information about the scene (e.g., details on sides not visible to the camera are unknown) or if the retrieved scene can not match the realistic semantics.

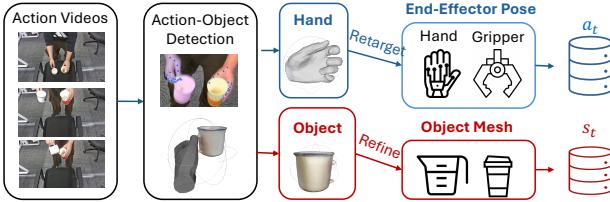


Figure 3: The flowchart of action-trajectory projection.

Action-Trajectory Projection. Given an egocentric video of humans manipulating objects (e.g., picking up a teapot and pouring tea into a paper cup), DexSim projects actions and objects from realistic human motion onto robot control signals within the simulated environment. We illustrate this process in Figure 3. This action trajectory captures *dynamic* information from human motion and can act as the seeding trajectories for the following skill scaling.

1) *Robot-Action Projection.* DexSim enables detecting movements of hand models from video data and retargeting these models to robotic end effectors, such as grippers and dexterous hands, thereby transforming human hand poses into those compatible with robotic devices. Within this process, a significant challenge is the disparity in the Degrees of Freedom (DoF) between the hand model and the robotic end effectors. For instance, a robot hand might possess 6 DoFs, whereas the human hand model features over 20 DoFs. To align the movements of the human and robotic hands effectively, it is essential to consolidate multiple DoFs from the human hand into a single DoF of the robotic hand, ensuring minimal impact on model performance. In the case of retargeting to a gripper, that has only one DoF (either open or closed), we select two fingers to act as pivotal points for picking up and placing objects, and then retargeting these fingers' movements to the jaw movements of the gripper.

2) *Robot-Object Interaction.* A crucial prerequisite for action projection is reconstructing the objects and their relationships with the operators (Liu et al., 2024c). For example, as shown in Figure 3, it is essential to determine the 3D pose at which the end effectors (e.g., hands) interact with the objects (e.g., the cup), thereby ensuring the projected action trajectory can accurately reflect the interaction under real-world physics. To achieve this goal, DexSim reconstructs the 3D meshes and the poses of objects from the sequences of images in the action trajectory video data. The sequence of object poses, along with the object models, is then jointly

optimized with the poses of the target end effectors captured in the robot-action projections (as mentioned above). For each image, the goal of joint optimization is to refine the interaction between objects and end effectors, accurately capturing realistic contact dynamics such as grasping forces and stability. DexSim further refines the trajectory by ensuring smoothness and plausibility for the sequence of actions. Figure 7 shows an example of the recovered 3D mesh.

4.2. Environment Simulation for Robot Learning

DexSim enables the conditional generation of a robot learning environment based on the projected data. This environment can simulate static and dynamic features by integrating scenes and action trajectories as described below.

Scene Simulation. Dexsim supports the construction of different scenarios by leveraging scene projection, which focuses on providing static features such as shape, pose, appearance, object layout, and other background details. However, since our scene projection primarily relies on single-view or multi-view images, there is no guarantee that the scene is fully observable, nor that these visual observations capture complete information about the scene. As a result, in addition to the projected information, DexSim enables the automatic construction of a complete scene based on the available data. To achieve this, we follow (Wang et al., 2024; Hua et al., 2024) and leverage large foundation models, such as GPT-4, to generate the scene configuration based on the projected information and task description. DexSim supports retrieving objects in the scene from the Objaverse-XL dataset (Deitke et al., 2023) or generating them based on language descriptions. Furthermore, if the generated scene does not align with practical requirements, DexSim provides a user-friendly interface that allows users to manually adjust and refine the generated scenarios.

Action-Trajectory Simulation. The objective of action simulation is to generate continuous action trajectories for the robot, enabling it to execute the tasks effectively and achieve the desired results. To construct such action trajectories, DexSim primarily relies on the projected poses of end effector extracted from human video, or the robot joint positions collected from tele-operation. For the projected end-effector poses, DexSim utilizes a generalized Inverse Kinematics (IK) algorithm to compute the corresponding joint configurations, ensuring precise achievement of the desired end-effector pose. To maintain a smooth trajectory, joint-space interpolation is applied across the motion. In scenarios requiring obstacle avoidance, DexSim integrates widely used motion planning algorithms, such as RRT-Connect (Kuffner & LaValle, 2000), to efficiently generate collision-free trajectory. In applications where action projection is unavailable, DexSim supports the automatic design of reward and goal functions by consulting with large language models (Ma et al., 2024). This enables the RL and

275 trajectory optimization algorithms to effectively learn robot
276 control skills. To ensure that the simulated actions align
277 with the projected action trajectories, DexSim allows for
278 the replay and automatic refinement of these trajectories
279 via learning-based methods. Using these refined trajectories,
280 we can train an imitation policy $\pi_{\theta^0}(a|s)$, for robot
281 control. However, directly deploying $\pi_{\theta^0}(a|s)$ in real-world
282 applications is challenging due to the existing Sim2Real
283 gap between the simulated and real-world environments.
284 To address this issue, DexSim incorporates the following
285 Sim2Real scaling techniques to bridge this gap.

286 4.3. Sim2Real Data Scaling

287 In a simulated environment, various factors can contribute
288 to the Sim2Real gap (Section 3), undermining the effec-
289 tiveness of generalizing the learned robot control policy
290 to real-world applications. To address this gap, DexSim
291 primarily focuses on establishing a pipeline for automatic
292 domain randomization and adaptation, as outlined below.

293 **Automatic Domain Randomization.** Domain Randomiza-
294 tion (DR) enables Sim2Real transfer by varying simulated
295 environments (Chen et al., 2022b). We achieve DR by mod-
296 ifying the original environment \mathcal{M} with configuration ξ ,
297 creating a new environment \mathcal{M}^ξ . In robotic learning en-
298 vironments, to better assess the effects of different DRs,
299 we categorize these DRs based on their impact on agents’
300 actions in the following.

301 1) *Action-Invariant Domain Randomization (AI-DR).* Within
302 our DexSim, the goal of AI-DR is to prevent the embodied
303 agent from overfitting to the simulated features that
304 are irrelevant or ineffective in completing a task (e.g., the
305 lighting condition on manipulations). By removing the
306 effects of these features and concentrating only on essen-
307 tial features for finishing a task, the agent experiences
308 a less complex sim-to-real gap, and the learned skills
309 have stronger transferability. For these DRs (configured
310 by ξ), they have limited impact on the agents’ decisions
311 and movements. To better formulate this consistency, we
312 assume the optimal actions remain uninfluenced before
313 and after applying AI-DR to an environment \mathcal{M} , so that
314 $\pi_{\theta^\xi}^*(a|s) = \pi_{\theta^0}(a|s) \forall (s, a) \in \mathcal{S} \times \mathcal{A}$ where $\pi_{\theta^\xi}^*$ denotes
315 the optimal policy under the DR environment \mathcal{M}^ξ .

316 2) *Semantic-Aware Domain Randomization (SA-DR).* The
317 goal of SA-DR is to generalize the embodied agents’ skills
318 from the source environment \mathcal{M} to different variations of
319 DR environment $\mathcal{M}^{\xi'}$. This approach allows the agent’s
320 skills to overcome the Sim2Real gap, provided that the gap
321 falls within the range of simulated variations. To better
322 handle these DRs (configured by ξ'), the agent must adapt
323 the decisions and movements so that the optimal actions may
324 vary before and after applying SA-DR to an environment
325 \mathcal{M} . The adapted policy $\pi^{*,\xi'}$ can be represented as:

$$\pi_{\theta^{\xi'}}^* = \arg \max_{\pi_{\theta^{\xi'}}} \mathcal{J}(\mathcal{M}^{\xi'}, \pi_{\theta^{\xi'}}) - \text{Div}(\pi_{\theta^{\xi'}} \| \pi_{\theta^0}) \quad (1)$$

Where $\mathcal{J}(\mathcal{M}^{\xi'}, \cdot)$ denotes the optimality function under
the environment $\mathcal{M}^{\xi'}$ and $\text{Div}(\pi_{\theta^{\xi'}} \| \pi_{\theta^0}^*)$ indicate the di-
vergence between policies $\pi_{\theta^{\xi'}}$ and $\pi_{\theta^0}^*$. For example, in
the continual learning setting, by setting $\text{Div}(\pi_{\theta^{\xi'}} \| \pi_{\theta^0}^*) = \lambda \|\theta^{\xi'} - \theta^0\|$, the objective (1) can effectively represent the
Elastic Weight Consolidation (EWC) objective. More impor-
tantly, as opposed to the action invariance in AI-DR, we
must finetune the policy $\pi_{\theta^0}^*$ based on the updated objective
 $\mathcal{J}(\mathcal{M}^{\xi'}, \pi_{\theta^{\xi'}})$ under the new environment $\mathcal{M}^{\xi'}$ after apply-
ing SA-DR. In this setting, both RL and motion planning
algorithms can be utilized to fine-tune the original policy
and adapt it to the new environment.

Automating DR. Each DR configuration ξ corresponds to
a control policy $\pi_{\theta^\xi}^*$, which captures skill for solving the
task under the DR environment \mathcal{M}^ξ . By generating skills
 τ using $\pi_{\theta^\xi}^*$, we can construct a Sim2Real dataset \mathcal{D}_{DR} . Based on this dataset, the imitation policy can effectively
bridge the Sim2Real gap, provided that the experimented
DR configurations and their combinations accurately cap-
ture the underlying Sim2Real discrepancies. To achieve this
goal, a critical prerequisite is being able to characterize the
types, ranges, and styles of applied DRs. Within DexSim,
our DR selection primarily focuses on the key factors for
the Sim2Real gap (Section 3). For example, given an en-
vironment \mathcal{M} , we must first identify which features are
suitable for DR and determine the corresponding types of
DRs, such as AI-DR and SA-DR. Additionally, for a specific
DR parameter ξ , we need to model the distribution of DR
parameters, $p_\phi(\xi) \in \Delta^\Xi$ (Chen et al., 2022a). While tra-
ditional simulators often manually specify $p_\phi(\xi)$, DexSim
seeks to automate the selection of DR features and model-
ing $p_\phi(\xi)$ conditioning on specific tasks. As illustrated in
Figure 2, DexSim integrates a large foundation model into
the DR process by leveraging its ability to rank DR features
based on the configuration of the simulated environment.
For each DR configuration ξ , DexSim employs the ADR al-
gorithm (OpenAI et al., 2019) to calculate and update $p_\phi(\xi)$
by incorporating feedback from the fine-tuned policies and
their outcomes in the environment.

Domain Adaption (DA). To more effectively bridge the gap
between simulation and reality, domain adaptation project
the observations from the simulated and realistic environ-
ment to a into a unified target space. DexSim supports
various types of DAs, including 1) *Object-Oriented Repre-
sentations*, where DexSim excludes background information
from image observations from RGB cameras and focuses
only on each object. To better capture the geometric and spa-
tial properties of objects, DexSim can map objects into point
clouds based on their masks within the simulated environ-
ment. 2) *Pose Affordance Representations*, where DexSim

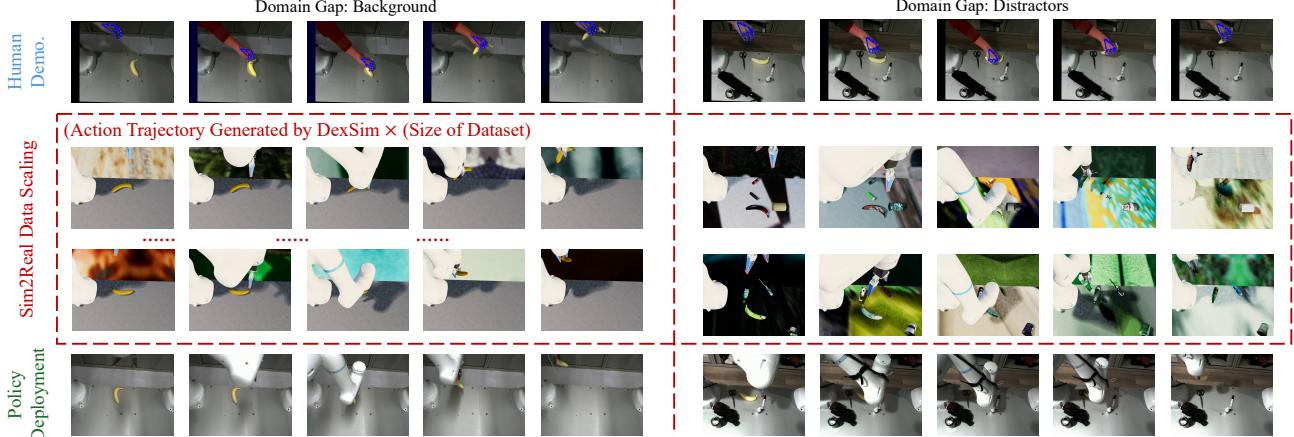


Figure 4: Visualizing the input demonstration (top row), examples of the scaled data (middle rows), and the realistic deployment of the robot policy (bottom row). The Sim2Real domain gaps are background (left) and distractors (right).

selects key affordances (e.g., pose) from an action trajectory, capturing the critical poses necessary to complete the task. For example, when grasping an object, the affordance represents the gripper’s pose when it first contacts the object, enabling the robot to complete the task by reaching these poses using inverse kinematics. These prediction targets, derived from affordance representations, occur less frequently than full action trajectories, thereby reducing accumulated errors during deployment. With these methods, DexSim can map \mathcal{D}_{DR} to the post-adaptation dataset $\mathcal{D}_{\text{DR+AR}}$.

4.4. Sim2Real Deployment

DexSim adopts a data-driven approach to achieve Sim2Real deployment. Specifically, the deployable policy is trained by imitating the data trajectories in $\mathcal{D}_{\text{DR+AR}}$. Since $\mathcal{D}_{\text{DR+AR}}$ captures diverse and rich action trajectories across various environments, the resulting control policy remains effective under the Sim2Real gap, provided the realistic environment lies within the support of the dataset’s distribution. In the following experiment section, we demonstrate that various high-performing imitation models—including action transformer policies (Zhao et al., 2023), diffusion policies (Chi et al., 2023), and Vision-Language-Action (VLA) policies (Liu et al., 2024b)—can be trained and deployed to control real-world robots under different environments.

5. Experiments

The pipeline of DexSim is generic and agnostic to simulation platforms (Appendix A.1 shows our specifications). We evaluate the performance of Sim2Real deployment across various applications and assess the validity of Real2Sim projection in terms of action and object mapping by addressing the following questions: 1) **Generalizability:** How effectively does DexSim bridge the Sim2Real gap between simulated environments and real-world applications? 2) **Scalability:** Can the control skills learned by DexSim be scaled across different models and embodiments?

5.1. Generalizability: Bridging the Sim2Real Gap

Experiment Setting. This experiment aims to quantify DexSim by how effectively it can overcome the Sim2Real gap. We first simulate a baseline environment by projecting the real-world scene into a simulation (via our Real2Sim projection in Section 4.1) and then manually refining it to ensure alignment with the realistic setting. Using this baseline environment, we can modify its parameters and intentionally introduce the Sim2Real gaps that frequently occur in practice based on our analysis (see Table 1). We summarize such modifications in Appendix A.2. DexSim is then evaluated based on its effectiveness in overcoming these gaps and acquiring robust skills that can be seamlessly applied to real-world applications. To gain a deeper understanding of DexSim, we conducted an ablation study by removing either the strategic Domain Adaptation (DA) or Domain Randomization (DR) components from our DexSim dataset. This resulted in a skill only dataset that records only the action trajectory for the target task in the simulated environment. Additionally, To prevent overfitting, we augment the skill dataset by adding random noise to skills.

To evaluate the model’s control performance in bridging the Sim2Real gap, we trained the control model using different methods and assessed its performance through 100 trials in the simulated environment and 10 trials in the real-world environment. The success rates are presented in Table 2. We found that the Sim2Real data scaling performed by DexSim significantly enhances the control model’s ability to consistently bridge the domain gap across both simulated and real-world environments. Removing the designed DA and DR components leads to a notable decline in performance. To better understand the performance of DexSim, Figure 4 shows keyframes of scaled data and policy deployment in realistic scenarios (Appendix B.2 shows more examples). To address the domain gap caused by distracting items (e.g., umbrellas, scissors, or pens not being the target in the right

Table 2: Success rates of imitation policies learned by different datasets under various Sim2Real gaps. For the first eight domain gaps, we employ the transformer-based policy (Zhao et al., 2023) to tackle grasping tasks. For the last two domain gaps, we use the diffusion-based policy (Chi et al., 2023) to address the open-box task.

Dataset	Skills Only		Skills+DR		Skills+DA		DexSim	
	Sim.	Real.	Sim.	Real.	Sim.	Real.	Sim.	Real.
Domain Gap								
Light	64/100	0/10	75/100	4/10	74/100	1/10	73/10	4/10
Object Texture	49/100	1/10	81/100	5/10	75/100	3/10	83/100	6/10
Table Texture	55/100	2/10	81/100	4/10	73/100	2/10	82/100	4/10
Background	67/100	1/10	83/100	3/10	71/100	1/10	82/100	3/10
Distractors	46/100	0/10	65/100	2/10	73/100	2/10	72/100	4/10
Camera Position	72/100	2/10	73/100	4/10	71/100	2/10	80/100	5/10
Camera Orientation	69/100	0/10	74/100	4/10	66/100	0/10	78/100	3/10
Camera Field of View	69/100	1/10	80/100	3/10	66/100	1/10	82/100	6/10
Object Pose	15/100	1/10	33/100	2/10	42/100	3/10	69/100	4/10
Object Shape	11/100	0/10	27/100	2/10	38/100	2/10	61/100	3/10

columns of Figure 4), an interesting observation is that DexSim not only randomizes different objects but also backgrounds and textures in the scene. While textures are not directly related to distractors, they are effectively in bridging the Sim2Real gap and DexSim discovers this relation.

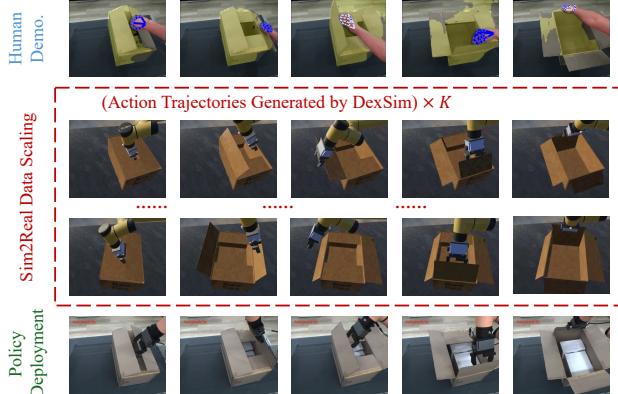


Figure 5: Visualizing the human demonstration (top row), scaled trajectories (middle rows), and realistic robot action (bottom row) for the task of opening box.

5.2. Scalability across Diverse Tasks and Embodiments
Experiment Setting. A critical prerequisite for developing a Sim2Real simulator is its ability to scale across diverse robot control tasks and environments. Unlike most prior work, which primarily demonstrates realistic performance with a single embodied robot, our approach emphasizes cross-embodiment evaluation to comprehensively assess Sim2Real performance. In this experiment, we evaluate our approach on the following tasks: 1) object grasping, 2) paper box manipulation, and 3) dual-arm table rearrangement. For these tasks, we train imitation models using various architectures, including the transformer-based policy (Zhao et al., 2023) for grasping, the diffusion policy (Chi et al., 2023) for manipulation, and Vision-Language-Action (VLA) models (Liu et al., 2024b) for table rearrangement. Appendix A.3 reports the specific training configurations.

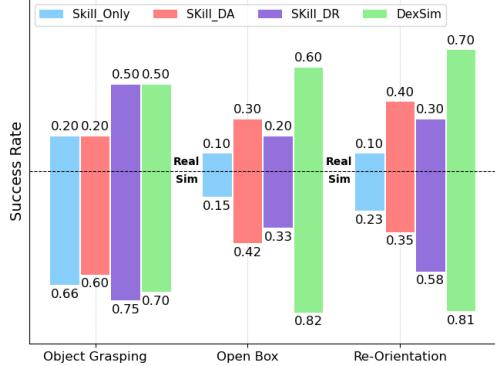


Table 3: Robot control performance for different tasks under both realistic (upper) and simulated (lower) environments.

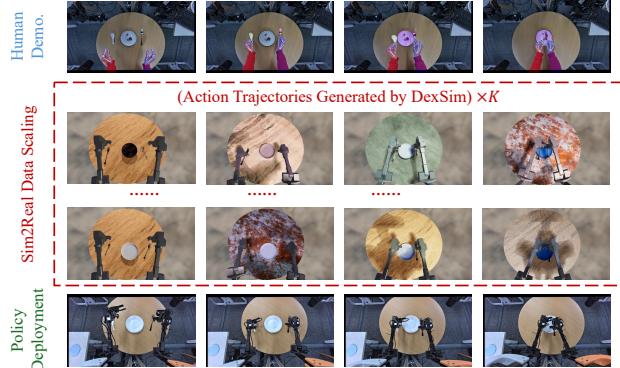


Figure 6: Visualizing the human demonstration (top row), scaled trajectories (middle rows), and realistic robot action (bottom row) for the task of table rearrangement.

Figures 4, 5, and 6 illustrate examples of action trajectories for the tasks of object grasping, box manipulation, and table rearrangement. To demonstrate the scalability of DexSim, the control policies are deployed on different robots, including two single-arm robots and a dual-arm robot equipped with wrist-mounted cameras. Figure 3 presents the end-to-end performance across various tasks, showcasing both real-to-sim (from task-descriptive data to the simulator) and sim-to-real (from simulation to real-world deployment) transitions. By leveraging the scaled data generated by DexSim, our models consistently outperform baseline methods across different tasks, model architectures, and robot platforms, highlighting the scalability and effectiveness of DexSim.

6. Conclusion

We introduce DexSim, a data engine designed to generate simulated robot action trajectories for learning deployable control policies by mapping realistic observations into simulation and utilizing automated domain randomization and adaptation for data scaling. These capabilities can be efficiently expanded to accommodate a wide range of tasks, learning models, and robotic platforms. In future work, we aim to extend DexSim to more complex robots (e.g., humanoid robots) and long-term tasks (e.g., cooking a meal).

440
441 **Impact Statement**
442

The broader impact of this work lies in its potential to significantly advance the field of robotics by addressing one of its most pressing challenges: the gap between simulated and real-world environments.

From an ethical perspective, DexSim promotes more inclusive and equitable innovation by making the development of robotic systems more accessible, even to those with limited resources. However, as with any technology that advances automation, there are potential societal consequences to consider. The widespread adoption of robots in tasks such as manufacturing, service, or even domestic work could lead to significant shifts in labor markets, raising concerns about job displacement and economic inequality.

In the long term, DexSim’s ability to scale robotic intelligence across diverse applications could have transformative societal implications. Robots trained using DexSim could be deployed in healthcare, disaster response, education, and other critical areas, amplifying their positive impact on society. However, ensuring the ethical use of such robots will require careful consideration of privacy, safety, and accountability. Future research should also explore ways to mitigate potential biases in simulated datasets to ensure fairness and reliability in real-world deployments.

467 **References**

Agarwal, A., Kumar, A., Malik, J., and Pathak, D. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205, pp. 403–415. PMLR, 2022.

Batra, D., Chang, A. X., Chernova, S., Davison, A. J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., Mottaghi, R., et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.

Brohan, A., Brown, N., Carballo, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al. RT-1: robotics transformer for real-world control at scale. In *Robotics: Science and Systems, RSS*, 2023.

Chen, I.-M. and Burdick, J. W. Finding antipodal point grasps on irregularly shaped objects. *IEEE transactions on Robotics and Automation*, 9(4):507–512, 1993.

Chen, S., Xu, H., Li, R., Liu, G., Fu, C., and Liu, S. SIRA-PCR: sim-to-real adaptation for 3d point cloud registration. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 14348–14359. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01324.

Chen, X., Hu, J., Jin, C., Li, L., and Wang, L. Understanding domain randomization for sim-to-real transfer. In *International Conference on Learning Representations, ICLR*. OpenReview.net, 2022a.

Chen, X., Hu, J., Jin, C., Li, L., and Wang, L. Understanding domain randomization for sim-to-real transfer. In *The Tenth International Conference on Learning Representations, ICLR*, 2022b.

Cheng, X., Shi, K., Agarwal, A., and Pathak, D. Extreme parkour with legged robots. In *IEEE International Conference on Robotics and Automation, ICRA*, pp. 11443–11450. IEEE, 2024.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems, RSS*, 2023.

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

Dai, T., Wong, J., Jiang, Y., Wang, C., Gokmen, C., Zhang, R., Wu, J., and Fei-Fei, L. Acdc: Automated creation of digital cousins for robust policy learning. *arXiv preprint arXiv:2410.07408*, 2024.

Dasari, S., Ebert, F., Tian, S., Nair, S., Bucher, B., Schmeckpeper, K., Singh, S., Levine, S., and Finn, C. Robonet: Large-scale multi-robot learning. In *Annual Conference on Robot Learning, CoRL*, volume 100, pp. 885–897. PMLR, 2019.

Deitke, M., VanderBilt, E., Herrasti, A., Weihs, L., Ehsani, K., Salvador, J., Han, W., Kolve, E., Kembhavi, A., and Mottaghi, R. Procthor: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., LaForte, C., Voleti, V., Gadre, S. Y., VanderBilt, E., Kembhavi, A., Vondrick, C., Gkioxari, G., Ehsani, K., Schmidt, L., and Farhadi, A. Objaverse-xl: A universe of 10m+ 3d objects. In *Advances in Neural Information Processing Systems, NeurIPS*, 2023.

D’Souza, A., Vijayakumar, S., and Schaal, S. Learning inverse kinematics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, volume 1, pp. 298–303, 2001.

Duan, J., Yu, S., Tan, H. L., Zhu, H., and Tan, C. A survey of embodied AI: from simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244, 2022.

- 495 Ebert, F., Yang, Y., Schmeckpeper, K., Bucher, B., Georgakis, G., Daniilidis, K., Finn, C., and Levine, S. Bridge
496 data: Boosting generalization of robotic skills with cross-
497 domain datasets. In *Robotics: Science and Systems, RSS*,
498 2022.
499
- 500 Fang, H., Wang, C., Fang, H., Gou, M., Liu, J., Yan, H., Liu,
501 W., Xie, Y., and Lu, C. Anygrasp: Robust and efficient
502 grasp perception in spatial and temporal domains. *IEEE
503 Transactions on Robotics*, 39(5):3929–3945, 2023.
504
- 505 Florence, P. R., Manuelli, L., and Tedrake, R. Self-
506 supervised correspondence in visuomotor policy learning.
507 *IEEE Robotics Autom. Lett.*, 5(2):492–499, 2020.
508
- 509 Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine,
510 S. D4RL: datasets for deep data-driven reinforcement
511 learning. *CoRR*, abs/2004.07219, 2020.
512
- 513 Gan, C., Zhou, S., Schwartz, J., Alter, S., Bhandwaldar, A.,
514 Gutfreund, D., Yamins, D. L. K., DiCarlo, J. J., McDermott,
515 J. H., Torralba, A., and Tenenbaum, J. B. The three-
516 world transport challenge: A visually guided task-and-
517 motion planning benchmark towards physically realistic
518 embodied AI. In *International Conference on Robotics
519 and Automation, ICRA*, pp. 8847–8854, 2022.
- 520 Hua, P., Liu, M., Macaluso, A., Lin, Y., Zhang, W., Xu, H.,
521 and Wang, L. Gensim2: Scaling robot data generation
522 with multi-modal and reasoning llms. *arXiv preprint
523 arXiv:2410.03645*, 2024.
524
- 525 James, S., Ma, Z., Arrojo, D. R., and Davison, A. J. Rlbench:
526 The robot learning benchmark & learning environment.
527 *IEEE Robotics and Automation Letters*, 5(2):3019–3026,
528 2020.
- 529 Jiang, Z., Zhu, Y., Svetlik, M., Fang, K., and Zhu, Y. Synergies
530 between affordance and geometry: 6-dof grasp
531 detection via implicit representations. In *Robotics: Sci-
532 ence and Systems, RSS*, 2021.
533
- 534 Josifovski, J., Audy, S., Malmir, M., Piater, J. H., Knoll, A.,
535 and Navarro-Guerrero, N. Continual domain randomiza-
536 tion. In *IEEE/RSJ International Conference on Intelligent
537 Robots and Systems, IROS*, pp. 4965–4972, 2024.
538
- 539 Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Bal-
540 akrishna, A., Nair, S., Rafailov, R., Foster, E. P., Lam,
541 G., Sanketi, P., Vuong, Q., Kollar, T., Burchfiel, B.,
542 Tedrake, R., Sadigh, D., Levine, S., Liang, P., and Finn, C.
543 Openvla: An open-source vision-language-action model.
544 *CoRR*, abs/2406.09246, 2024.
- 545 Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L.,
546 Herrasti, A., Deitke, M., Ehsani, K., Gordon, D., Zhu, Y.,
547 et al. Ai2-thor: An interactive 3d environment for visual
548 ai. *arXiv preprint arXiv:1712.05474*, 2017.
549
- 550 Kuffner, J. J. and LaValle, S. M. Rrt-connect: An efficient
551 approach to single-query path planning. In *Proceedings
552 2000 ICRA. Millennium Conference. IEEE International
553 Conference on Robotics and Automation. Symposia Pro-
554 ceedings (Cat. No. 00CH37065)*, volume 2, pp. 995–1001.
555 IEEE, 2000.
- 556 Kumar, A., Fu, Z., Pathak, D., and Malik, J. RMA: rapid
557 motor adaptation for legged robots. In *Robotics: Science
558 and Systems*, 2021.
- 559 Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end
560 training of deep visuomotor policies. *Journal of Machine
561 Learning Research*, 17:39:1–39:40, 2016.
- 562 Liu, J., Savva, M., and Mahdavi-Amiri, A. Survey
563 on modeling of articulated objects. *arXiv preprint
564 arXiv:2403.14937*, 2024a.
- 565 Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K.,
566 Su, H., and Zhu, J. RDT-1B: a diffusion foundation model
567 for bimanual manipulation. *CoRR*, abs/2410.07864,
568 2024b.
- 569 Liu, Y., Long, X., Yang, Z., Liu, Y., Habermann, M.,
570 Theobalt, C., Ma, Y., and Wang, W. Easyhoi: Unleashing
571 the power of large models for reconstructing hand-object
572 interactions in the wild. *arXiv preprint arXiv:2411.14280*,
573 2024c.
- 574 Lobos-Tsunekawa, K. and Harada, T. Point cloud based
575 reinforcement learning for sim-to-real and partial observ-
576 ability in visual navigation. In *IEEE/RSJ International
577 Conference on Intelligent Robots and Systems, IROS*, pp.
578 5871–5878, 2020.
- 579 Ma, Y. J., Liang, W., Wang, G., Huang, D., Bastani, O.,
580 Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A.
581 Eureka: Human-level reward design via coding large lan-
582 guage models. In *The Twelfth International Conference
583 on Learning Representations, ICLR*, 2024.
- 584 Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey,
585 K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A.,
586 Handa, A., and State, G. Isaac gym: High performance
587 GPU based physics simulation for robot learning. In
588 Vanschoren, J. and Yeung, S. (eds.), *Neural Information
589 Processing Systems Track on Datasets and Benchmarks,
590 NeurIPS Datasets and Benchmarks*, 2021.
- 591 Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C.,
592 Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and
593 Martín-Martín, R. What matters in learning from of-
594 fline human demonstrations for robot manipulation. In
595 Faust, A., Hsu, D., and Neumann, G. (eds.), *Conference
596 on Robot Learning, CoRL*, volume 164, pp. 1678–1690,
597 2021.

- 550 Manuelli, L., Gao, W., Florence, P. R., and Tedrake,
 551 R. KPAM: keypoint affordances for category-level
 552 robotic manipulation. In *The International Symposium
 553 of Robotics Research*, volume 20, pp. 132–157. Springer,
 554 2019.
- 555 Mehta, B., Diaz, M., Golemo, F., Pal, C. J., and Paull, L.
 556 Active domain randomization. In *Conference on Robot
 557 Learning, CoRL*, volume 100, pp. 1162–1176, 2019.
- 559 Mo, K., Guibas, L. J., Mukadam, M., Gupta, A., and Tulsiani,
 560 S. Where2act: From pixels to actions for articulated
 561 3d objects. In *IEEE/CVF International Conference on
 562 Computer Vision, ICCV*, pp. 6793–6803. IEEE, 2021.
- 564 Mousavian, A., Eppner, C., and Fox, D. 6-dof grapsnet:
 565 Variational grasp generation for object manipulation. In
 566 *IEEE/CVF International Conference on Computer Vision,
 567 ICCV*, pp. 2901–2910. IEEE, 2019.
- 569 Murali, A., Mousavian, A., Eppner, C., Fishman, A., and
 570 Fox, D. Cabinet: Scaling neural collision detection for
 571 object rearrangement with procedural scene generation.
 572 In *IEEE International Conference on Robotics and Au-
 573 tomation, ICRA*, pp. 1866–1874. IEEE, 2023.
- 574 Muratore, F., Eilers, C., Gienger, M., and Peters, J. Data-
 575 efficient domain randomization with bayesian optimiza-
 576 tion. *IEEE Robotics and Automation Letters*, 6(2):911–
 577 918, 2021a.
- 579 Muratore, F., Gruner, T., Wiese, F., Belousov, B., Gienger,
 580 M., and Peters, J. Neural posterior domain randomization.
 581 In *Conference on Robot Learning, CoRL*, volume 164, pp.
 582 1532–1542, 2021b.
- 583 Nasiriany, S., Maddukuri, A., Zhang, L., Parikh, A., Lo, A.,
 584 Joshi, A., Mandlekar, A., and Zhu, Y. Robocasa: Large-
 585 scale simulation of everyday tasks for generalist robots.
 586 *arXiv preprint arXiv:2406.02523*, 2024.
- 588 O’Neill, A., Rehman, A., Maddukuri, A., Gupta, A.,
 589 Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar,
 590 A., Jain, A., et al. Open x-embodiment: Robotic learning
 591 datasets and RT-X models : Open x-embodiment collabora-
 592 tion. In *IEEE International Conference on Robotics
 593 and Automation, ICRA*, pp. 6892–6903. IEEE, 2024.
- 595 OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M.,
 596 Litwin, M., McGrew, B., Petron, A., Paino, A., Plap-
 597 pert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N.,
 598 Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba,
 599 W., and Zhang, L. Solving rubik’s cube with a robot hand.
 600 *CoRR*, abs/1910.07113, 2019.
- 601 Paolo, G., Gonzalez-Billandon, J., and Kégl, B. Position:
 602 A call for embodied AI. In *International Conference on
 603 Machine Learning, ICML*, 2024.
- 550 Puig, X., Undersander, E., Szot, A., Cote, M. D., Yang,
 551 T., Partsey, R., Desai, R., Clegg, A., Hlavac, M., Min,
 552 S. Y., Vondrus, V., Gervet, T., Berges, V., Turner, J. M.,
 553 MakSYMets, O., Kira, Z., Kalakrishnan, M., Malik, J.,
 554 Chaplot, D. S., Jain, U., Batra, D., Rai, A., and Mot-
 555 taghi, R. Habitat 3.0: A co-habitat for humans, avatars,
 556 and robots. In *International Conference on Learning
 557 Representations (ICLR)*, 2024.
- 559 Qi, H., Kumar, A., Calandra, R., Ma, Y., and Malik, J.
 560 In-hand object rotation via rapid motor adaptation. In
 561 *Conference on Robot Learning, CoRL*, volume 205, pp.
 562 1722–1732, 2022.
- 564 Qin, Y., Huang, B., Yin, Z., Su, H., and Wang, X. Dexpoint:
 565 Generalizable point cloud reinforcement learning for sim-
 566 to-real dexterous manipulation. In *Conference on Robot
 567 Learning, CoRL 2022, 14–18 December 2022, Auckland,
 568 New Zealand*, volume 205, pp. 594–605. PMLR, 2022.
- 572 Qin, Z., Fang, K., Zhu, Y., Fei-Fei, L., and Savarese, S.
 573 KETO: learning keypoint representations for tool manip-
 574 ulation. In *IEEE International Conference on Robotics
 575 and Automation, ICRA*, pp. 7278–7285. IEEE, 2020.
- 578 Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T.,
 579 Khedr, H., Rädl, R., Rolland, C., Gustafson, L., et al.
 580 Sam 2: Segment anything in images and videos. *arXiv
 581 preprint arXiv:2408.00714*, 2024.
- 585 Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W.,
 586 Mottaghi, R., Zettlemoyer, L., and Fox, D. ALFRED:
 587 A benchmark for interpreting grounded instructions for
 588 everyday tasks. In *2020 IEEE/CVF Conference on Com-
 589 puter Vision and Pattern Recognition, CVPR*, pp. 10737–
 590 10746, 2020.
- 595 Sundermeyer, M., Mousavian, A., Triebel, R., and Fox,
 596 D. Contact-graspnet: Efficient 6-dof grasp generation
 597 in cluttered scenes. In *IEEE International Conference
 598 on Robotics and Automation, ICRA*, pp. 13438–13444.
 599 IEEE, 2021.
- 604 Sutton, R. S. Reinforcement learning: An introduction. *A
 605 Bradford Book*, 2018.
- 609 Tao, S., Xiang, F., Shukla, A., Qin, Y., Hinrichsen, X.,
 610 Yuan, X., Bao, C., Lin, X., Liu, Y., Chan, T.-k., et al.
 611 Maniskill3: Gpu parallelized robotics simulation and
 612 rendering for generalizable embodied ai. *arXiv preprint
 613 arXiv:2410.00425*, 2024.
- 618 Team, O. M., Ghosh, D., Walke, H., Pertsch, K., Black, K.,
 619 Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., Luo,
 620 J., Tan, Y. L., Chen, L. Y., Sanketi, P., Vuong, Q., Xiao,
 621 T., Sadigh, D., Finn, C., and Levine, S. Octo: An open-
 622 source generalist robot policy. *CoRR*, abs/2405.12213,
 623 2024.

- 605 Tiboni, G., Arndt, K., and Kyrki, V. DROPO: sim-to-real
 606 transfer with offline domain randomization. *Robotics and*
 607 *Autonomous Systems*, 166:104432, 2023.
- 608
- 609 Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics
 610 engine for model-based control. In *IEEE/RSJ International*
 611 *Conference on Intelligent Robots and Systems, IROS*, pp.
 612 5026–5033. IEEE, 2012.
- 613
- 614 Walke, H. R., Black, K., Zhao, T. Z., Vuong, Q., Zheng, C.,
 615 Hansen-Estruch, P., He, A. W., Myers, V., Kim, M. J., Du,
 616 M., Lee, A., Fang, K., Finn, C., and Levine, S. Bridgedata
 617 V2: A dataset for robot learning at scale. In *Conference*
 618 *on Robot Learning, CoRL*, volume 229, pp. 1723–1736,
 619 2023.
- 620
- 621 Wang, Y., Xian, Z., Chen, F., Wang, T., Wang, Y., Fragkiadaki,
 622 K., Erickson, Z., Held, D., and Gan, C. Robogen:
 623 Towards unleashing infinite data for automated robot
 624 learning via generative simulation. In *International Conference*
 625 *on Machine Learning, ICML*, 2024.
- 626
- 627 Wu, C., Chen, J., Cao, Q., Zhang, J., Tai, Y., Sun, L., and Jia,
 628 K. Grasp proposal networks: An end-to-end solution for
 629 visual learning of robotic grasps. In *Advances in Neural*
 630 *Information Processing Systems, NeurIPS*, 2020.
- 631
- 632 Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu,
 633 M., Jiang, H., Yuan, Y., Wang, H., Yi, L., Chang, A. X.,
 634 Guibas, L. J., and Su, H. SAPIEN: A simulated part-
 635 based interactive environment. In *IEEE/CVF Conference*
 636 *on Computer Vision and Pattern Recognition, CVPR*, pp.
 637 11094–11104, 2020.
- 638
- 639 Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B.,
 640 Chen, D., Tong, X., and Yang, J. Structured 3d latents
 641 for scalable and versatile 3d generation. *arXiv preprint*
 642 *arXiv:2412.01506*, 2024.
- 643
- 644 Xiao, B., Wu, H., Xu, W., Dai, X., Hu, H., Lu, Y., Zeng,
 645 M., Liu, C., and Yuan, L. Florence-2: Advancing a
 646 unified representation for a variety of vision tasks. In
 647 *Proceedings of the IEEE/CVF Conference on Computer*
 648 *Vision and Pattern Recognition*, pp. 4818–4829, 2024.
- 649
- 650 Xie, A., Lee, L., Xiao, T., and Finn, C. Decomposing
 651 the generalization gap in imitation learning for visual
 652 robotic manipulation. In *IEEE International Conference*
 653 *on Robotics and Automation, ICRA*, pp. 3153–3160.
 654 IEEE, 2024.
- 655
- 656 Xu, G., Wang, X., Ding, X., and Yang, X. Iterative geometry
 657 encoding volume for stereo matching. In *IEEE/CVF*
 658 *Conference on Computer Vision and Pattern Recognition,*
 659 *CVPR*, pp. 21919–21928. IEEE, 2023.
- Yang, J., Cao, Z.-a., Deng, C., Antonova, R., Song, S., and
 Bohg, J. Equibot: Sim(3)-equivariant diffusion policy for
 generalizable and data efficient learning. In *8th Annual*
Conference on Robot Learning, 2024.
- Yuan, W., Duan, J., Blukis, V., Pumacay, W., Krishna, R.,
 Murali, A., Mousavian, A., and Fox, D. Robopoint: A
 vision-language model for spatial affordance prediction
 for robotics. *CoRR*, abs/2406.10721, 2024.
- Zhang, T., McCarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg,
 K., and Abbeel, P. Deep imitation learning for
 complex manipulation tasks from virtual reality teleoper-
 ation. In *IEEE International Conference on Robotics and*
Automation, ICRA, pp. 1–8. IEEE, 2018.
- Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning
 fine-grained bimanual manipulation with low-cost
 hardware. In *Robotics: Science and Systems, RSS*, 2023.
- Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F.,
 Wu, J., Wohlhart, P., Welker, S., Wahid, A., et al. RT-2:
 vision-language-action models transfer web knowledge
 to robotic control. In Tan, J., Toussaint, M., and Darvish,
 K. (eds.), *Conference on Robot Learning, CoRL*, volume
 229, pp. 2165–2183, 2023.

660 A. Implementation Details

661 A.1. Specification of DexSim

663 DexSim is a scalable, automated data engine designed to bridge the Sim2Real gap in robot skill learning by combining
664 physically based rendering (PBR), physics simulation and high performance multi-threading system. Its architecture
665 is optimized for generating large-scale, diverse, and realistic robot learning environments and datasets, while ensuring
666 computational efficiency. Below, we detail its core specifications:

668 **System Architecture.** DexSim is built on a modular framework with three interconnected subsystems:

- 670 • *Physically-Based Rendering Engine*: Combines ray tracing and rasterization techniques to synthesize photorealistic visuals
671 with dynamic lighting effects and physically accurate material properties, such as surface roughness, metallic reflectance,
672 and albedo maps.
- 674 • *Physics Simulation Engine*: Employs a rigid-body dynamics solver with support for frictional contacts and articulation
675 modeling (e.g., grippers, robotic arms)
- 677 • *Scene Composition*: Compose complex scenes with customizable objects, lighting, and backgrounds. Users can define
678 object dynamics properties, such as mass, friction, and elasticity, or material properties to match real-world conditions.

679 DexSim’s modular architecture is engine-agnostic, enabling seamless integration with arbitrary rendering or physics engines
680 as backends. This design ensures adaptability to diverse use cases while decoupling core functionality from third-party
681 dependencies.

683 **Robotic Simulation.** Our robot simulation module provides a unified framework for modeling, controlling, and training
684 robotic systems through three core components:

- 687 • *Robot Abstract Models*: Predefines the parameterized templates for common robotic platforms, such as Manipulators
688 ((e.g., 6-DoF industrial arm), Dexterous Hands and Humanoids with unified URDF description and configurable kinematic
689 chains, inertial properties, Joint limits, etc. It also supports the customization of robot morphologies through a modular
690 assembly of actuators and sensors.
- 691 • *Control System*: Provides forward/inverse kinematics solvers with singularity handling and motion generation capacities
692 with trajectory interpolation and collision-aware planners. It has user-friendly interface to access robot proprioception and
693 execute the control signal in both joint space and task space (eg, euclidean space)
- 695 • *Robot Learning Environment*: Adopts OpenAI gym-compatible API with standardized method (e.g., **reset**, **step**), and
696 support domain randomization for both dynamics and visuals properties of the objects and scene. It includes task suites
697 (e.g., manipulation, locomotion) equipped with configurable callback and reward functions, enabling the implementation
698 of customized features tailored for both imitation learning and reinforcement learning.

700 **Data Generation.** Our data generation module enables large-scale synthesis of labeled simulation datasets through a
701 pipeline combining assets generation, geometric processing, and automated annotation. The architecture comprises four
702 core subsystems:

- 704 • *Scene Construction*: Combines procedural generation, leveraging parameterized templates for randomized object placement,
705 lighting configurations, and camera viewpoints, with rule-based scene assembly (e.g., clutter or ordered arrangements)
706 and collision-free guarantees. It integrates 3D AIGC for asset synthesis via state-of-the-art models like
707 TRELLIS (Xiang et al., 2024), while also supporting retrieval of 3D assets from a large-scale, multi-label database.
- 709 • *Mesh Processing*: Provides an automated pipeline to ensure 3D geometry assets are simulation-ready. This pipeline
710 includes UV mapping, geometric processing (e.g., remeshing, hole filling, and simplification), and convex decomposition
711 for efficient collision detection.
- 712 • *Domain Randomization*: Provides interface to adjust or change the parameters of the domain randomization factors described
713 in section 4.3.

- 715 • *Annotation Computation*: Supports diverse annotation types, including instance/semantic segmentation masks, 6D object
 716 poses, keypoints, and physics-based ground truth such as contact vectors at interaction points. It also provides interfaces
 717 to export datasets in standard formats like COCO and HDF5.

718

A.2. The Details of Sim2Real Gap

- 719
- *Light*. We adjust the lighting conditions by varying the distance between the light source and the scene of the given tasks.
 720 Specifically, a 3-meter difference is introduced between the simulated and real-world environments, resulting in a darker
 721 lighting condition in the real-world setting.
 - *Object Texture*. We primarily focus on the material properties and appearance, including the color, of each object.
 722 Specifically, in the real-world environment, the objects are covered with tissues, while in the simulated environment, the
 723 color and texture are tailored to match the object type (e.g., a banana has a smooth yellow surface).
 - *Table Texture*. We primarily focus on the differences in table surface patterns between the simulated and real-world
 724 environments. Specifically, in the real-world environment, the table is covered with dark paper, while in the simulated
 725 environment, a table surface in a light grey table surface is used.
 - *Background Texture*. Since the camera’s view primarily captures the floor where the table is placed, we focus on the
 726 differences in floor surface patterns between the simulated and real-world environments. Specifically, in the real-world
 727 environment, the floor is covered with a light grey surface, while in the simulated environment, a wooden surface is used.
 - *Distractors*. The distractors consist of objects that are not intended to be grasped by the gripper. In the real-world
 728 environment, these objects include a paper cup, a folding umbrella, a pull-tab can, scissors, and a spray bottle. However,
 729 in the simulated environment, these objects are intentionally removed.
 - *Camera Position*. We adjust the camera position in the simulated environment. Specifically, the camera is placed 5 cm
 730 higher and 5 cm to the right compared to the position of the camera in the real-world environment.
 - *Camera Orientation*. We adjust the camera orientation in the simulated environment. The camera is rotated 5 degrees
 731 counterclockwise around both the X and Z axes in the simulated environment.
 - *Camera Field of View*. We adjust the camera orientation by modifying its parameters. Specifically, the focal length is set
 732 to 1.15 times the actual focal length to achieve the desired adjustments.
 - *Object Pose*. We adjust the object pose in the simulated environment by applying a rotation of up to 15 degrees and
 733 shifting their x-y positions by up to 5 centimeters.
 - *Object Shape*. We rescale the width, length, and height of the object in the simulated environment, with the scaling ranging
 734 from 0.9 to 1.1 times the object’s size in the realistic environment.

735

A.3. Model Training Configurations

736

A.3.1. OBJECT GRASPING

737 **Model Structure.** We adopt the HumanPlus HIT model configuration and outline the following details:

- 738
- *Model Architecture*: The model employs 2 ResNet18 backbones to process visual inputs and output visual latents. Action
 739 data is projected into action latents using an MLP. The visual and action latents are then fed into 6 BERT-style transformer
 740 decoders for further processing.
 - *Number of Parameters*: The total number of parameters in the model is 30.4 M.
 - *Activation Functions*: GELU is used as the activation function throughout the network.

741 **Dataset Details.** The dataset consists of 2000 trajectories for each feature and each setting (gap, domain randomization,
 742 domain adaptation, domain randomization and adaptation), with each trajectory containing 50 steps. Each trajectory includes
 743 21-dimensional action data (comprising 6-dim joint angles, 7-dim end-effector pose under the robot base frame, 1-dim
 744 gripper open state, and 7-dim end-effector pose under the camera frame) and two 512×640 rectified images. The dataset is
 745

770 split into 99% training and 1% validation. Data preprocessing involves normalization for action data, ImageNet-statistics
 771 normalization for images, and image rectification. During training, color-jitter data augmentation is applied to the images.
 772

773 **Training Hyperparameters.** The model is trained with a batch size of 12 for 200,000 epochs using the AdamW optimizer
 774 with a learning rate of 1×10^{-5} . The training objective is guided by the mean squared error (MSE) loss for actions and a
 775 1-dimensional error for image latents.

776 **Training Environment.** Training was conducted on 4 NVIDIA A800 GPUs, with each policy trained for 36 hours. PyTorch
 777 version 2.0.1 was used as the deep learning framework.
 778

779 A.3.2. OPEN BOX

780 **Model Structure.** We follow the network architecture of EquiBot (Yang et al., 2024), with several modifications to enhance
 781 sim-to-real transfer. Our model is based on the SIM(3)-Equivariant Diffusion Policy (Yang et al., 2024), with the following
 782 details:
 783

- 784 • *Network Architecture:* The original EquiBot processes entire scene point clouds as the observed input environment, with a
 785 sampling point number of 1024 or 2048 depending on the task type. In our modified version, we use the cropped point
 786 clouds of manipulated objects as input and always set the sampling point number to 1024.
 787
- 788 • *Mask Detection:* The 2D mask of manipulated objects is obtained automatically in simulation using the DexSim simulator.
 789 During real-world inference, the mask is detected using vision foundation models, such as Florence2 (Xiao et al., 2024)
 790 and SAM2 (Ravi et al., 2024).
 791
- 792 • *Prediction and Observation Horizon:* The prediction horizon is set to 72, and the observation horizon is 1. Unlike the
 793 high-frequency dynamic style used in the original EquiBot (prediction-conduction-observation), we observe once and
 794 execute all predicted actions at once.
 795
- 796 • *Stereo Vision:* Our model uses a binocular camera to capture left and right images simultaneously. The left image branch
 797 lifts 2D pixels into 3D using a stereo matching algorithm (Xu et al., 2023). This approach significantly reduces domain
 798 gaps caused by hardware differences between real and simulated environments.
 799
- 800 • *Number of Parameters:* The total parameter size is 111.43 MB.
 801

802 **Dataset Details.** The dataset consists of 50 demonstrations in the training set, with each demonstration containing 75
 803 timesteps. The dataset is split into 50 demonstrations for training and 10 demonstrations for testing. Additionally, 100
 804 evaluation trials are conducted in the simulator to assess performance.
 805

806 **Training Hyperparameters.** The training process uses a batch size of 16 and runs for 150,000 epochs. The model is
 807 optimized using the Adam optimizer with a cosine learning rate scheduler, starting with a learning rate of 5×10^{-4} . The
 808 training objective is guided by the Chamfer Distance loss function for point clouds and the Mean Squared Error (MSE) loss
 809 for action predictions.
 810

811 **Training Environment.** Training was performed on a single NVIDIA A100 GPU for a total of 48 hours. PyTorch version
 812 2.0.1 was used as the deep learning framework.
 813

814 A.3.3. TABLEWARE REARRANGEMENT

815 **Model Structure.** We follow the original Robotics Diffusion Transfomer (RDT) model configuration (Liu et al., 2024b)
 816 and make the following modifications:
 817

- 818 • *Vision Encoder:* Replace SigLip with DINoV2 (base) with registers as the vision encoder.
 819
- 820 • *Image Conditions:* Use one stereo camera along with two wrist-mounted RGB cameras, resulting in the following image
 821 conditions: 2 RGB images, 1 disparity image, and 2 additional RGB images.
 822
- 823 • *Mask Predictor Module:* Add a small mask predictor module consisting of 3 convolutional layers to predict the mask for
 824 each image. The updated image condition is then computed using the following equation:
 825

$$\text{img_cond_new} = \text{img_cond} + \text{mask} \cdot \text{mask_embed}$$

- 825 • *Transformer Block Settings*: The transformer block is configured with a hidden size of 512, a depth of 8 layers, and 16
 826 attention heads.
 827
 828 • *Model Size*: The total number of parameters is 116.66 MB.

829
 830 **Dataset Details.** The dataset used in this work consists of multiple trajectories designed to capture diverse scenarios for
 831 training and evaluation. Specifically: 1) *Number of Trajectories*: The dataset includes 200 trajectories, each consisting of
 832 200 steps. 2) *Data Preprocessing*: The data is preprocessed using the same pipeline as described in the original RDT setup.

833
 834 **Training Hyperparameters.** The training process utilizes a batch size of 8 and runs for 40,000 iterations. The model is
 835 optimized using the Adam optimizer with a cosine learning rate scheduler, starting with a learning rate of 1×10^{-4} . The
 836 training objective is guided by the Mean Squared Error (MSE) loss function.

837
 838 **Training Environment.** The training was conducted on a single NVIDIA A100 GPU using PyTorch version 2.0.1 as the
 839 deep learning framework.

840 A.3.4. HARDWARE DETAILS FOR DIFFERENT TASKS

841
 842 Table 4: Robot Specifications for Different Tasks

Task	Robot Name	DOF	Maximum Reach (mm)	Maximum Payload (kg)
Object Grasping	Rokae SR3	6	705	3
Open Box	AUBO I5	6	886.5	5
Tableware Rearrangement	WidowX 250 S	7	650	0.25

843 B. More Experiment Results

844 B.1. Human-Object Interaction Example

845 Figure 7 visualize the input image alongside the output hand-object interactions as a 3D mesh, which can be directly
 846 projected into a simulated environment. To enhance the visualization of the 3D mesh details, we present observations from
 847 six different angles.

848 B.2. Examples of using DexSim

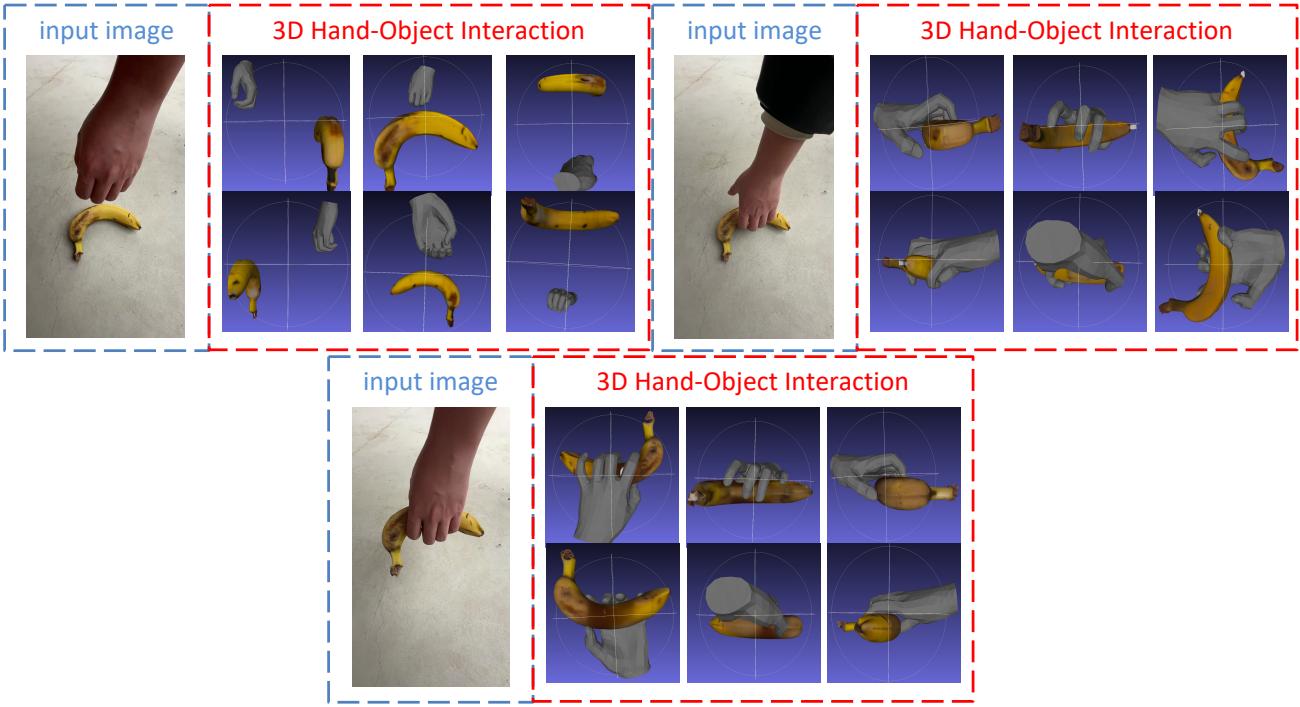


Figure 7: Visualizing the input image and the output hand-object interactions in 3D mesh.

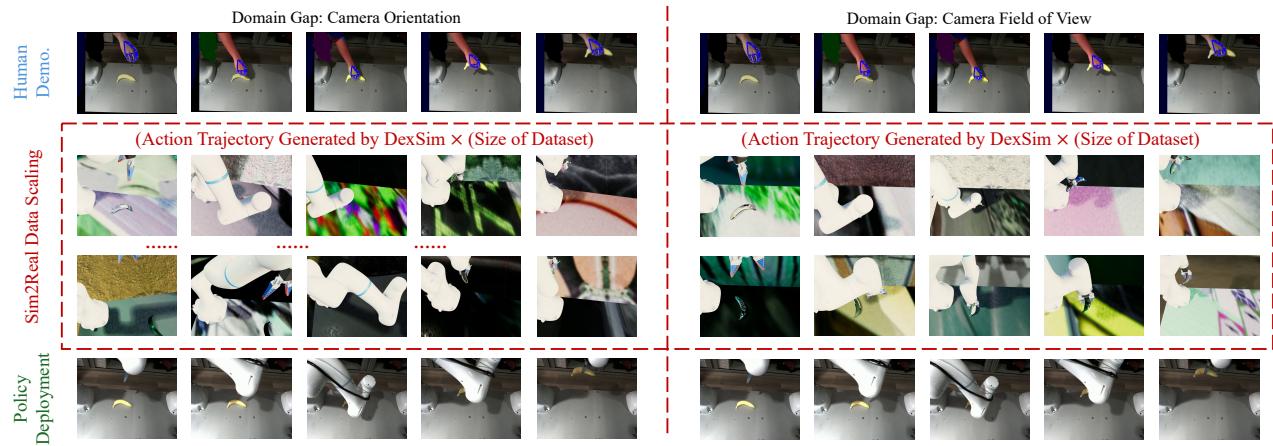


Figure 8: Visualizing the input demonstration (top row), examples of the scaled data (middle rows), and the realistic deployment of the robot policy (bottom row). The Sim2Real domain gaps are camera orientation (left) and field of views (right).

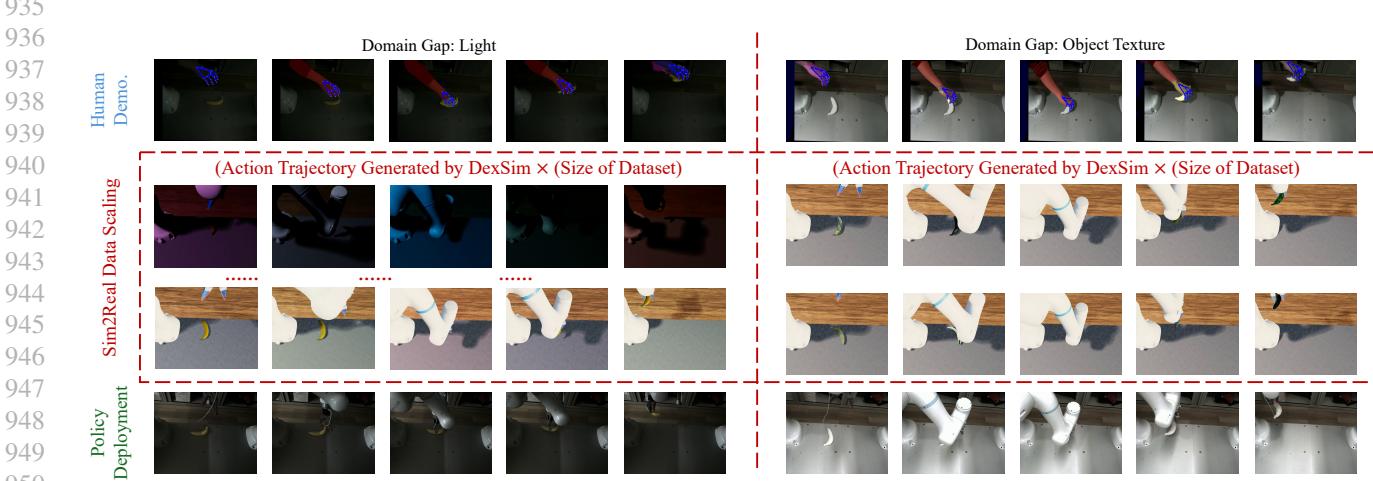


Figure 9: Visualizing the input demonstration (top row), examples of the scaled data (middle rows), and the realistic deployment of the robot policy (bottom row). The Sim2Real domain gaps are lighting (left) and object texture (right).

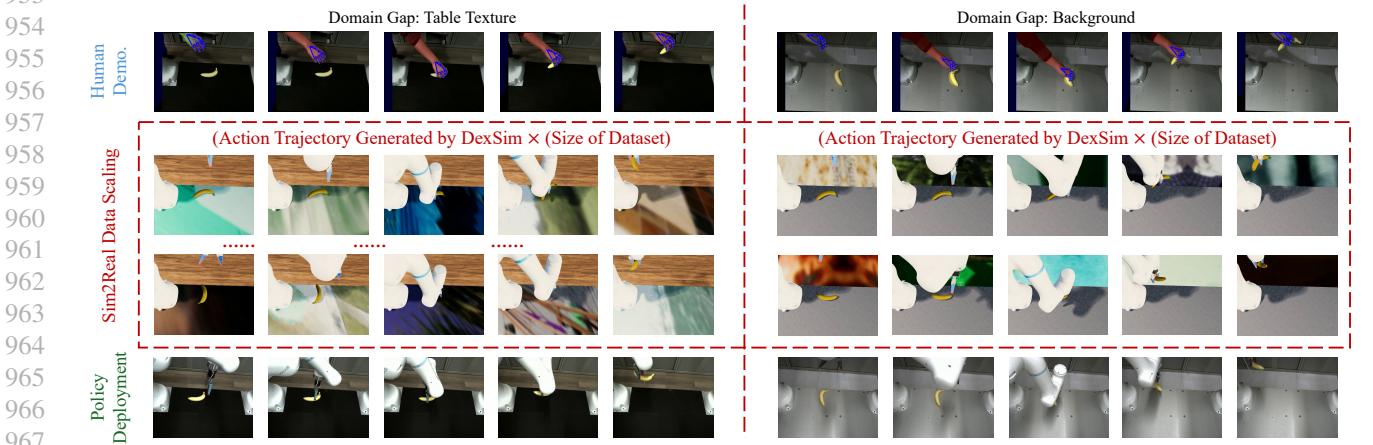


Figure 10: Visualizing the input demonstration (top row), examples of the scaled data (middle rows), and the realistic deployment of the robot policy (bottom row). The Sim2Real domain gaps are table texture (left) and background (right).

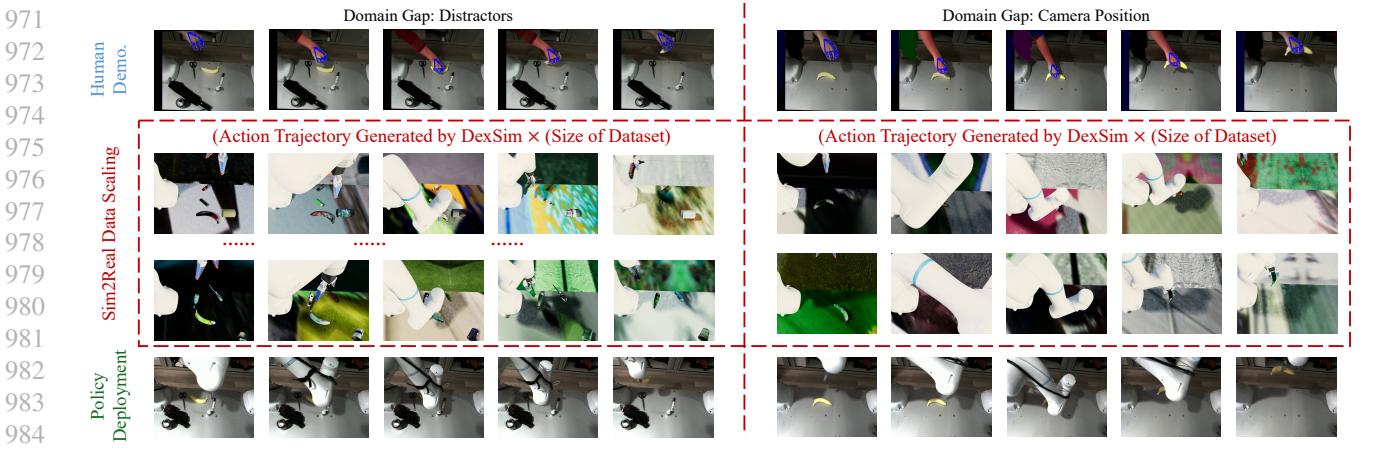


Figure 11: Visualizing the input demonstration (top row), examples of the scaled data (middle rows), and the realistic deployment of the robot policy (bottom row). The Sim2Real domain gaps are distractors (left) and camera position (right).