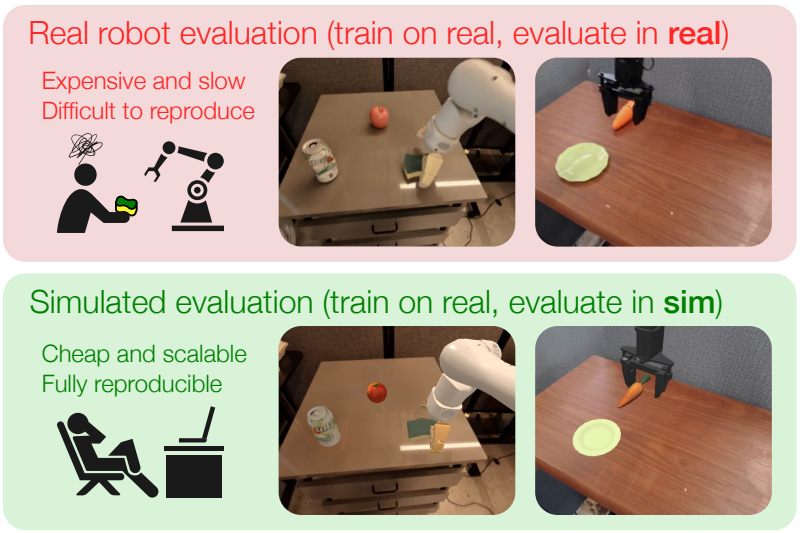
Evaluating Real-World Robot Manipulation Policies in Simulation

Xuanlin Li[*1], Kyle Hsu[*2], Jiayuan Gu[*1], Karl Pertsch[2 3 †], Oier Mees[3 †], Homer Rich Walke[3], Chuyuan Fu[4], Ishikaa Lunawat[2], Isabel Sieh[2], Sean Kirmani[4], Sergey Levine[3], Jiajun Wu[2], Chelsea Finn[2], Hao Su[‡1], Quan Vuong[‡4], Ted Xiao[‡4]

[1]UC San Diego, [2]Stanford University, [3]UC Berkeley, [4]Google Deepmind

[*]Equal contribution [†]Core contributors [‡]Co-advising

https://simpler-env.github.io



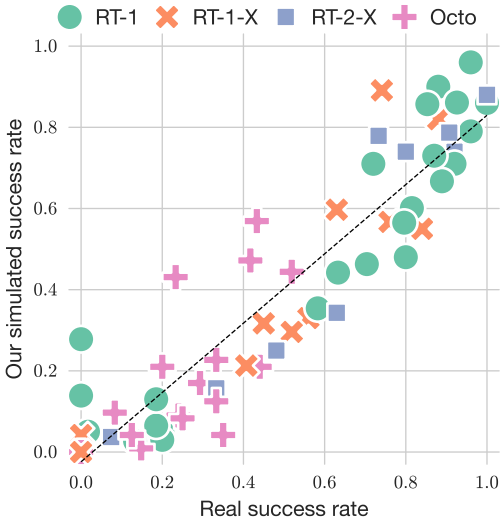


Fig. 1: Characterizing generalist robot manipulation policies typically involves evaluating them on many tasks across many scenarios, a laborious undertaking in the real world (top left). In this work, we design an evaluation procedure in which policies trained on *real* data are evaluated in purpose-built *simulated* environments (bottom left). Our approach yields a strong correlation between real-world and simulated performance (right) for various open-source robot policies [6, 11, 50] across two commonly used robot embodiments (Google Robot and WidowX) and over ∼1500 evaluation episodes. These results highlight the potential of simulation-based approaches for evaluating generalist real-world robot manipulation policies in a scalable, reproducible, and reliable way.

***Abstract*—The field of robotics has made significant advances towards generalist robot manipulation policies. However, real-world evaluation of such policies is not scalable and faces reproducibility challenges, which are likely to worsen as policies broaden the spectrum of tasks they can perform. In this work, we demonstrate that simulation-based evaluation can be a scalable, reproducible, and reliable proxy for real-world evaluation. We identify control and visual disparities between real and simulated environments as key challenges for reliable simulated evaluation and propose approaches for mitigating these gaps without needing to craft full-fidelity digital twins of real-world environments. We then employ these approaches to create SIMPLER, a collection of simulated environments for manipulation policy evaluation on common real robot setups. Through paired sim-and-real evaluations of open-source manipulation policies, we demonstrate strong correlation between policy performance in SIMPLER environments and in the real world. Additionally, we find that SIMPLER evaluations accurately reflect real-world policy behavior modes such as sensitivity to various distribution shifts. We open-source all SIMPLER environments along with our workflow for creating new environments to facilitate research on general-purpose manipulation policies and simulated evaluation frameworks.**

I. Introduction

Remarkable progress has been made in recent years towards building generalist real-world robot manipulation policies [6, 50], i.e., policies that can perform a wide range of tasks across many environments and even robot embodiments. These advances are underpinned by large-scale datasets [11, 66] and expressive models [1, 6, 29]. However, evaluating these policies in a scalable and reproducible way remains challenging as real-world evaluation is expensive and inefficient. Compared to the evaluation burden of works that study robot performance in narrower settings, the scope of evaluations required for faithful performance estimates of *generalist* policies increases with the breadth of their abilities. This underlines a growing challenge in robot manipulation research: as we scale the capabilities of robot policies, how do we correspondingly scale our ability to accurately, reproducibly, and comprehensively evaluate them?

In this work, we propose *simulated evaluation* as a possible answer, in which manipulation policies trained on real data are evaluated in purpose-built simulated environments (Fig. 1). Such real-to-sim evaluation can serve as a scal-

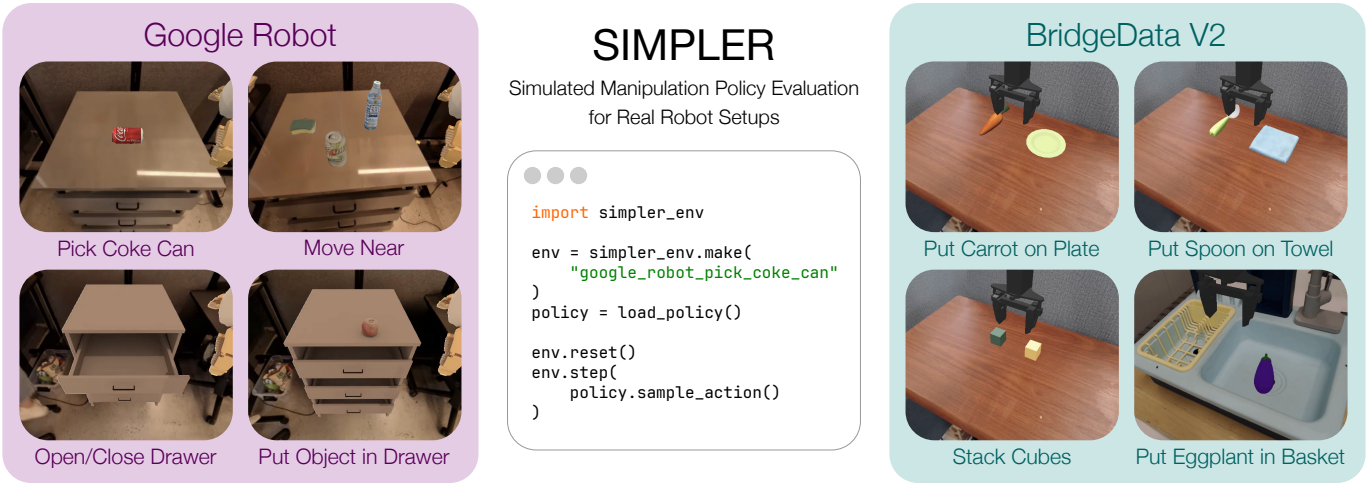


Fig. 2: We introduce SIMPLER, a suite of open-source simulated evaluation environments for common real robot manipulation setups, namely the Google Robot evaluations from the RT-series of works [6, 5, 11], and environments from the BridgeData V2 dataset [66]. All environments can be imported with a single line of code and can be interacted with through a standard Gym interface. Additionally, we open-source policy inference code for real-to-sim evaluation of common generalist robot policies (RT-1 [6], RT-1-X [11], and Octo [50]), and we provide detailed guides for evaluating new policies and creating new evaluation environments. All materials are available at https://simpler-env.github.io.

able, reproducible, and informative tool to complement gold-standard real-world evaluations. Indeed, evaluation in simulation is common practice for testing autonomous driving policies across a wide range of driving scenarios before real-world deployment [16, 44, 71]. However, performing simulated evaluations for robotic *manipulation* poses additional challenges due to the diverse interactions between agent and environment. At the same time, research on sim-to-real policy learning [55, 75] has demonstrated that considerable transfer between simulation and the real world is possible even for manipulation policies. While sim-to-real approaches typically train in simulation and evaluate in the real world, we are interested in the opposite question: how can we build systems for evaluating manipulation policies trained on *real* data in *simulated* environments?

One option to build such simulated environments is to fully replicate an existing real-world environment by creating a simulated "digital twin", an approach popular in navigation [14, 32] and autonomous driving [48]. However, for robot manipulation, reconstructing dynamic and interactive objects [70], along with realistic materials and lighting [31] in simulation remains an open research question. Furthermore, building full-fidelity digital twins demands extensive time and resources, typically requiring digital artists to manually craft object geometries and materials [14, 59]. Capturing precise physical properties of objects for manipulation simulation, such as center of mass, inertia, static and dynamic friction, further complicates scalability.

A key idea in this work is that we do not need an *exact* replica of the real-world environment. Instead, we need a simulated environment that is merely *realistic enough*, such that the performance of policies evaluated in this simulation environment correlates well with their real-world performance. This allows us to design environment creation pipelines that are more scalable than creating exact digital twins. Through extensive experiments, we examine the challenges of building effective simulated evaluation pipelines: from control disparities to visual disparities between real and simulated environments. We then propose and evaluate approaches for mitigating these differences based on offline system identification, "green-screening" simulation observations using real-world backgrounds, and object texture baking from real-world images.

Using these techniques, we create SIMPLER (Simulated Manipulation Policy Evaluation for Real Robot Setups), a suite of simulated evaluation environments for commonly used real-world robot manipulation environments, namely the RT-1 [6] and the Bridge-V2 [66] evaluation setups (Fig. 2). For both setups, we perform extensive *paired* sim-and-real evaluations for multiple open-source manipulation policies such as RT-1-X [11] and Octo [50], and we demonstrate strong correlation between policy performance as assessed by SIMPLER and the corresponding real environments (Fig. 1, right). In addition, we find that simulated policy evaluations in SIMPLER environments accurately reflect policy behavior modes in the real world, such as sensitivity to various distribution shifts. As such, SIMPLER is a first step towards using simulated evaluation as a tool for reliable, scalable, and reproducible manipulation policy evaluation. In summary, our contributions are as follows:

- We introduce SIMPLER, a suite of simulated evaluation environments for commonly-used real robot manipulation setups.
- We address the challenges inherent in simulated manipulation policy evaluation by proposing approaches to mitigate real-to-sim control and visual gaps. As a result, we demonstrate strong correlation between real and simulated policy performance and behavior modes.

- We open-source our workflow for constructing SIMPLER environments to facilitate research on general-purpose manipulation policies and simulated evaluation frameworks.

II. Related Work

Reproducible evaluation of real robot policies is a long-standing challenge in the robotics community. While benchmarks exist for specific problems such as grasping [19, 41, 42] and motion planning [8, 46], extending such benchmarks to a broader set of robotics tasks is challenging. Initiatives like YCB [7] and NIST [65] were introduced to standardize objects, yet standardizing other variables such as lighting, camera setups, and workspaces proves difficult. Many real-world robot challenges, including the Amazon Picking Challenge [12] and DARPA Robotics Challenges [35], have addressed this by using fixed physical evaluation sites. TOTO [76] provides remote users with access to shared robotic hardware and an open-source dataset for offline training, which enable the evaluation of methods on standardized tasks. RB2 [13] establishes a framework for sharing experimental data between labs, integrating local rankings from various labs to form global rankings. Such benchmarks require ongoing maintenance of real-world evaluation setups, representing a significant long-term investment. As real-world robotic datasets [6, 11, 66] and generalist policies [4, 5, 50, 58] proliferate, the demand for *reliable, scalable, and reproducible* methods of evaluating these policies grows. The need is particularly acute given the difficulty faced by the research community in conducting evaluations without standardized hardware.

Simulation-based algorithmic research offers an alternative to real-world evaluation. A wide range of simulation benchmarks [18, 22, 23, 28, 37, 43, 45, 47, 54, 62, 63, 73, 77] have been established to facilitate scalable and reproducible evaluation. Moreover, several real-world robotics challenges [21, 40] have incorporated simulation components, allowing participants to test and compete in a more affordable and flexible way. However, most prior works consider both training and evaluation in simulation, and the resulting policies might exhibit distinct behaviors when deployed on real robot hardware. In contrast, we aim to both measure and enhance simulated evaluations' ability to reflect a policy's real-world performance and behaviors.

Can simulated evaluations reliably predict real-world policy performance and behavior modes? Multiple works explore this question in the context of navigation tasks from a sim-to-real perspective. They create virtual replicas of physical rooms by either 3D scanning [32] or via artist-created assets [14], and compare the performance of various models trained in simulation in both simulated and real environments. Kadian et al. [32] highlight that the sim-to-real gap may arise from dynamic differences, noting that navigation policies trained in simulation may exploit simulator imperfections. Deitke et al. [14] demonstrate that visual discrepancies can significantly impact policies, even when the simulation assets are crafted by skilled digital artists. Zhang et al. [74] translate the real images of a navigation policy back to the synthetic domain during deployment. In contrast to these works on navigation, we focus on developing simulated evaluations for real-world *manipulation* policies. Manipulation poses distinct challenges for simulated evaluation due to the tight interaction loop between policy and environment. It often involves dynamic rather than static scenes, and complex action sequences where even slight variations can significantly impact task outcomes.

For robot manipulation, the sim-to-real setting has been extensively investigated, where one aims to train policies in simulation and deploy in the real-world. In sim-to-real, the discrepancy between simulation and reality, commonly called the "reality gap", is a key challenge. Domain randomization [52, 64], a common strategy employed to mitigate this gap, introduces variations in simulator parameters during training. This technique has proven successful in applications such as locomotion [26, 36], manipulation with complex dynamics [9, 75], and dexterous in-hand manipulation [2, 56, 72]. In addition, domain adaptation methods are extensively utilized to address visual discrepancies. For instance, Generative Adversarial Networks (GANs) [3, 24, 27, 57] are trained to modify images generated in simulations so they resemble the style of real-world images. Alternatively, Du et al. [17] aim to align the feature space of observations between simulated and real-world environments, creating a more consistent visual experience across these domains. In contrast to these works on sim-to-real learning, we focus on the opposite question: *building simulation systems that effectively and faithfully evaluate real-world robot manipulation policies*. To this end, we introduce approaches to address both real-to-sim visual and control gaps to enhance real-&-sim evaluation correlations.

III. Using Physics Simulators for Evaluation of Robot Manipulation Policies

In this work, we study the problem of using physics simulators to evaluate the performance and examine the behavior modes of real robot manipulation policies. In this section, we outline our problem definition and discuss metrics for measuring the quality of simulated evaluation pipelines.

A. Problem Formulation

Simulated evaluations have great potential as a tool for practitioners: they can reduce the need for costly real-world evaluations and provide a more comprehensive picture of a policy's performance by enabling sweeps across a wide range of controlled environment conditions at negligible added cost. Yet, we emphasize that our goal is *not* to completely replace real-world evaluations: a simulator will always be an imperfect proxy for the real world. Instead, our aim is to give practitioners a readily available signal for policy improvement to guide their research.

With this in mind, the main goal of simulated evaluations is *not* to obtain a 1:1 reproduction of a policies' real-world behavior, but instead a strong correlation in *relative* policy performance between simulation and real rollouts. That is, if one policy performs better than another in real-world

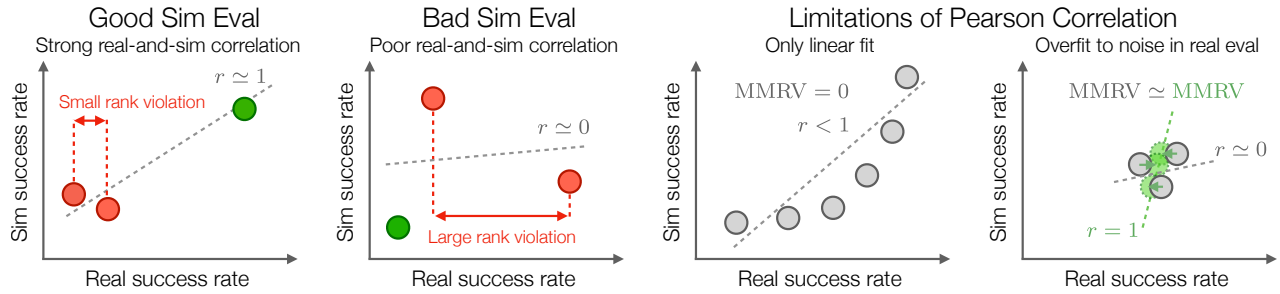


Fig. 3: Illustration of Mean Maximum Rank Violation (MMRV, range $[0, 1]$, lower is better) and Pearson correlation coefficient (Pearson r, range $[-1, 1]$, higher is better) for assessing the correlation between policy performances in real-world and simulation, as well as the overall quality of simulated evaluation pipelines. Each circle represents a policy. For the two leftmost pipelines, both metrics yield valuable insights, identifying one as poor and the other as good. The two rightmost examples highlight limitations of Pearson r: it can penalize simulation pipelines that fail to linearly recover the real results despite recovering the correct ranking, and it is overly sensitive to minor noise in evaluations when different policies perform similarly in the real world.

evaluations, we want the same comparison to hold in our simulated evaluation. The simulation would then afford a reliable proxy *improvement* signal for practitioners to iterate on design decisions. Formally, consider two policies π_a and π_b for which we have obtained real-world performance measures R_a and R_b, e.g., their average success rate across a representative set of tasks, by running real robot evaluations. Our goal is to construct a simulator $\mathcal{S}$, for which there is a strong *correlation* between the relative performances in real and the relative performances in simulation $R_{\mathcal{S},a}$ and $R_{\mathcal{S},b}$.

B. Metrics for Real-to-Sim Evaluation Pipelines

A standard approach for measuring the correlation between two variables is the **Pearson correlation coefficient (Pearson r)** [51]. It assesses the linear consistency between the two variables and has been used to measure the quality of simulated evaluation pipelines in prior work [32] by measuring the correlation between real-world and simulated evaluation performance. A high Pearson correlation of $\simeq 1$ indicates a well functioning simulated evaluation pipeline, where improvements in real-world success rate are reflected in a linear increase in simulated success rate (see dashed line in Fig. 3, far left). In contrast, a lower Pearson correlation may indicate a weaker connection between real-world and simulated evaluation performance (Fig. 3, middle left).

However, Pearson correlation has two important drawbacks when used as the sole metric for judging simulated evaluation pipelines. First, it only assesses the *linear* fit between real and simulated performance. Yet, for simulated evaluation we do not necessarily *need* to have a linear relationship, as long as the simulated pipeline reflects real-world performance improvements between different policies (Fig. 3, middle right). Second, Pearson correlation does not reflect the *range* of values it is computed over. Thus, for policy sets that lie within a narrow range of real-world performances, r may change drastically based on small real-world performance differences, which can often be attributed to the inherent noise in real-world evaluations (Fig. 3, far right grey vs. green).

To address the first point, we can additionally report a *ranking* metric that measures whether the simulated evaluation *ranks* the policies correctly based on their real-world performance, independent of a linear performance relationship. However, conventional ranking metrics like Spearman's rank correlation coefficient [61] still suffer from the second shortcoming: they operate purely on the rankings and disregard the underlying margins between real values. As a result, both simulated evaluation pipelines on the far-left and middle-left of Fig. 3 would be assigned the same rank correlation score, since both commit exactly one rank violation (red), even though the far-left evaluation pipeline provides a much stronger improvement signal and is thus clearly preferable. The key point is that we need to take the *magnitude* of the rank violation into account, measured as the difference in real-world performance between the mis-ranked policies. This provides a clear signal whether rank violations are caused by small real-world performance differences that are often a result of inherent noise in real robot evaluations, like in the far-left example of Fig. 3, or constitute clear failures of the evaluation pipeline, like in the middle-left example of Fig. 3.

Thus, we propose the **Mean Maximum Rank Violation (MMRV)** metric to better assess real-and-sim policy ranking consistency. Given N policies $\pi_{1\ldots N}$ and their respective performance measures (e.g., success rates) $R_{1\ldots N}$, $R_{\mathcal{S},1\ldots N}$ from real and simulated evaluations, MMRV is computed as follows:

$$\begin{aligned}\text{RankViolation}(i,j) = |R_i - R_j| \\ \cdot \mathbf{1}[(R_{\mathcal{S},i} < R_{\mathcal{S},j}) \neq (R_i < R_j)]\end{aligned} \tag{1}$$

$$\text{MMRV}(R, R_{\mathcal{S}}) = \frac{1}{N}\sum_{i=1}^{N} \max_{1\leq j\leq N} \text{RankViolation}(i,j). \tag{2}$$

The key underlying quantity is the *rank violation* between two policies π_i and π_j, which weighs the significance of the simulator incorrectly ranking the items by the corresponding margin in real-world performance. MMRV aggregates the N^2 rank violations by averaging the worst-case rank violation for each policy. In the remainder of this paper, we will report both the MMRV and Pearson correlation metrics.

IV. Building a Real-to-Sim Evaluation System by Addressing Control and Visual Gaps

This section introduces our approach for designing a simulation evaluation pipeline for real robot manipulation policies. When choosing aspects of the simulation problem to focus on, we take inspiration from the rich literature on sim-to-real policy learning [2, 52, 64, 75]. Commonly, there are two axes along which simulator fidelity can impact transferrability between simulation and the real world: gaps in the dynamics of the control system and gaps in visual realism. Although in this work we focus on the opposite problem, i.e., evaluating policies trained on real data in simulated environments, we demonstrate in Section VI that the same axes of simulator fidelity significantly affect the informativeness of simulated evaluations. Next, we will describe our approach for addressing the control and visual gaps between simulation and the real world.

A. Mitigating the Real-to-Sim Control Gap

The goal of mitigating the control gap between simulated and real-world environments is to ensure that policy actions executed in simulation yields comparable effects on the robot's end-effector as those observed when executed on the real robot. Practically, this means that when we execute a trajectory of actions in an open-loop manner in simulation, we want the resulting 6D end-effector pose trajectory and the joint position trajectory to closely mirror those observed in a real robot rollout of the same actions.

Concretely, let $\{(\mathbf{x}_i, R_i) : \mathbf{x}_i \in \mathbb{R}^3, R_i \in \mathbb{SO}(3)\}_{i=1}^T$ be a 6D end-effector pose trajectory recorded in the real-world when rolling out an action trajectory $\{\mathbf{a}_i\}_{i=1}^T$. Let $\{(\mathbf{x}'_i, R'_i) : \mathbf{x}'_0 = \mathbf{x}_0, R'_0 = R_0\}_{i=1}^T$ be the corresponding simulation trajectory when unrolling the same sequence of actions in the simulation in an open-loop manner using stiffness and damping parameters (i.e., PD parameters) $(\mathbf{p}, \mathbf{d})$. Then, we have the following system identification losses from translation and rotation errors:

$$\mathcal{L}_{\text{transl}}(\mathbf{p}, \mathbf{d}) = \frac{1}{T}\sum_{i=1}^{T} ||\mathbf{x}_i - \mathbf{x}'_i||_2 \tag{3}$$

$$\mathcal{L}_{\text{rot}}(\mathbf{p}, \mathbf{d}) = \frac{1}{T}\sum_{i=1}^{T} \arcsin\left(\frac{1}{2\sqrt{2}}||R_i - R'_i||_F\right) \tag{4}$$

$$\mathcal{L}_{\text{sysid}}(\mathbf{p}, \mathbf{d}) = \mathcal{L}_{\text{transl}} + \mathcal{L}_{\text{rot}} \tag{5}$$

In practice, we use a small sample of trajectories from an offline dataset $\mathcal{D}$, e.g., a demonstration dataset collected in the real environment, to retrieve action and end-effector pose trajectories and compute the system identification losses above. For all environments we consider in this work, we use trajectories from existing open-source demonstration datasets [6, 66], and thus do not need to collect any new data.

Next, we optimize the parameters of our controller: given initial PD parameters $(\mathbf{p}_0, \mathbf{d}_0)$ and a search range $(\mathbf{p}_{\text{low},0}, \mathbf{p}_{\text{high},0}, \mathbf{d}_{\text{low},0}, \mathbf{d}_{\text{high},0})$, we normalize the range to $[0, 1]$ and perform simulated annealing [53] to optimize $\mathcal{L}_{\text{sysid}}$.

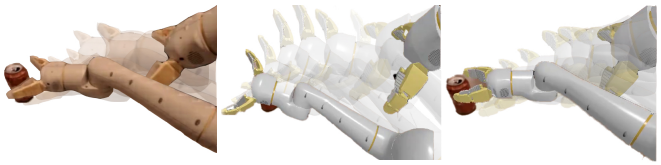


Fig. 4: We perform system identification (SysID) for closing the control gap between real and simulated environments. We visualize the open-loop execution of demonstration actions (using 6D end-effector pose control) for picking up a coke can before and after SysID (Section IV-A). Afterwards, the simulated robot arm tracks the real motion much more accurately and successfully reproduces the pick-up behavior.

We then select the PD parameters with the lowest $\mathcal{L}_{\text{sysid}}$ as $(\mathbf{p}_1, \mathbf{d}_1)$, and initialize another round of simulated annealing with a reduced parameter search range. In total, we perform 3 rounds of simulated annealing.

We qualitatively illustrate the effects of our system identification for one of our simulated environments, the Google Robot [6], in Fig. 4. We find that naively using PD parameter values from real controllers results in inaccurate tracking of the real robot's end-effector movements, which culminates in a missed grasp on the coke can. After system identification, the controller more accurately tracks the motion in simulation: the robot is able to grasp the object when replaying the demonstration's action sequence.

B. Mitigating the Real-to-Sim Visual Gap

Visual discrepancies between real-world and simulated environments can comprise a distribution shift that adversely affects a learned policy's behavior, rendering simulated evaluation unreliable [14]. While modern graphics pipelines are able to create highly realistic visuals, developing the underlying assets and determining the lighting parameters to accurately model existing environments involves significant manual labor. Our goal is to match the simulator visuals to those of the real-world environment *with only a modest amount of manual effort*. For the scene background, we propose a "green screening" approach in which we overlay an image of the real-world environment onto the background of the simulated scene (see Fig. 5). Concretely, we perform the following steps: (1) we remove the robot and foreground objects from the first frame of a real-world evaluation video I_{real} using online image inpainting tools (e.g., https://cleanup.pictures/); (2) we create a binary mask M isolating the foreground objects (robot arm and interactable objects, such as tabletop objects and articulated objects) in the simulation rendering I_{sim} by querying ground truth segmentation masks in simulation with a few lines of code; and (3) we combine the real-world background with the simulation foreground: $I' = M \odot I_{\text{sim}} + (1 - M) \odot I_{\text{real}}$, which produces the green screened image.

In practice, we find that performing this background green-screening alone can be insufficient to bridge the visual gap between simulation and real world: the tested policies are often sensitive to changes in foreground object and robot textures. At the same time, readily available simulation assets

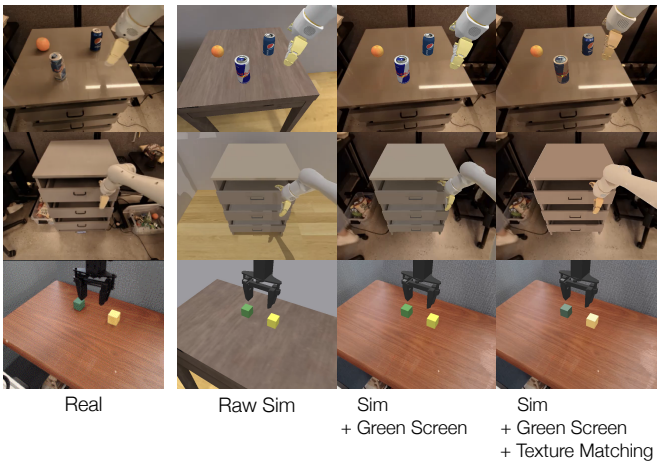


Fig. 5: Illustration of our "Visual Matching" approach for reducing the visual appearance gap between real environments and raw simulation. Visual Matching consists of (1) green screening, i.e. segmenting out interactive simulated assets and overlaying them onto real-world backgrounds; and (2) texture matching, which involves projecting real object textures onto simulation assets and tuning robot arm colors using real videos.

exhibit appearance differences from real-world objects due to a combination of texture, material, and lighting factors. Thus, for objects and robot links with the most noticeable real-to-sim visual gap, we tune simulation asset textures to more closely match their real-world counterparts. Concretely, we employ two strategies:

- For objects with substantially different textures, we project the real texture onto the simulation object by (1) segmenting the object in a real-world image; (2) aligning the simulated object pose to the real image; and (3) "unprojecting" the texture onto the object mesh in simulation. We provide step-by-step instructions and a command line script for performing this "texture matching" in Appendix Section C-B.
- For assets like robot visual meshes with texture maps already resembling their real-world counterparts, we can instead selectively copy and paste color values from real to simulated texture maps, e.g. using bucket-paint tools in the GNU Image Manipulation program.

We visualize results of this texture tuning in Fig. 5. Finally, as robot arms may change colors during movement, we found it helpful to obtain multiple tuned robot arm colors that match the real-world textures from different phases of a manipulation task. We then average their evaluation results to mitigate this confounding factor.

As an alternative to the "Visual Matching" strategy described above, we explore a mitigation strategy for real-to-sim visual gaps inspired by domain randomization: instead of minimizing the gap, we heavily randomize visual aspects of the scene to create environment "variants". We test whether we can get a more faithful estimate of a policy's performance by aggregating evaluation results across multiple such variants. See Appendix Section C-C for a detailed description and visualization of the applied randomizations. We empirically compare this randomization approach, which we denote as **Variant Aggregation**, to our main Visual Matching method in Section VI.

V. SIMPLER: Simulated Manipulation Policy Evaluation for Real Robot Setups

We apply the approaches introduced in Section IV to create SIMPLER (Simulated Manipulation Policy Evaluation for Real Robot Setups), a suite of simulated evaluation environments for commonly used real robot evaluation setups: the Google Robot from the RT series of works [6, 5, 11] and the WidowX BridgeV2 setup [66].

For each setup, we provide simulations for multiple tasks spanning a range of skills, interacted objects, object positions and orientations, backgrounds, and lighting conditions (see Fig. 2). The tasks are chosen to be representative of those in the corresponding training datasets, while also involving largely rigid body objects whose dynamics can be reasonably well-approximated by modern physics simulators.

We instantiate SIMPLER on top of the SAPIEN physics simulator [68], but show in Section VI-D that our contributions are independent of the used simulator and can be reproduced in Isaac Sim [49]. To obtain assets and scenes for SIMPLER environments, we perform the following procedure (further details are presented in Appendix, Section C):

- We obtain robot URDF assets either from public GitHub repositories or through ROS export. If robot camera intrinsics are unknown, we obtain them from real evaluation video frames using efficient interactive GUI tools such as fSpy.
- We obtain assets for common objects like cans and apples from the Objaverse 3D model repository [15], and we obtain textured meshes for less common objects through 3D scanning or via online single-view 3D reconstruction API [39] given a reference image. We then adjust their sizes in Blender to match the real dimensions, and we create texture-tuned assets for Visual Matching evaluations through the process described in Section IV-B. Finally, we use CoACD [67] to obtain convex collision shapes for all assets.
- For articulated objects such as the cabinet used in the Google Robot Drawer tasks, we manually construct its articulated model and then use the aforementioned processes to "bake" its textures. This articulated object modeling process takes the most human effort among the entire simulation environment construction process, and we highlight the acceleration of this process through approaches like multi-view [25] or interactive [30] articulated object generation as an avenue for future work.
- We set a uniform density for the object assets by querying their common material density in GPT-4 or Google search, or, for objects with non-uniform densities like empty coke can), querying their mass and dividing by their volume. We also assign the friction coefficients of objects based on their common material properties.

- Finally, we tune our robot and camera poses in simulation such that the edges of fixed objects (e.g., tables or cabinets) along with the observed robot gripper at initialization roughly align between simulation and the real-world.

Users can easily install SIMPLER environments via `pip` and interact with them via the common Gym environment API (see Fig. 2). A single environment renders at 3.5k simulation steps per second on a consumer NVIDIA 4090 GPU at 640×512 image resolution. Under a 500 Hz simulation frequency, this amounts to a $7\times$ speedup over real eval. We open-source all SIMPLER environments as well as a detailed guide for environment creation at https://simpler-env.github.io.

VI. Experimental Results

In this section, we empirically test the performance correlation between real-world robot evaluations and simulated evaluations in SIMPLER environments for a representative set of open-source generalist robot manipulation policies. Concretely, we aim to answer the following questions:

1) Do **relative performances** of different manipulation policies in simulation strongly correlate with the relative performances observed in real evaluation?
2) Can simulated evaluations not only capture the performance relationships across different policies, but also accurately reproduce real-world policy behavior modes within the same policy, like **sensitivity to various visual distribution shifts**? Additionally, can simulated evaluations predict the robustness of policies to *novel distribution shifts* in the real world?
3) To what extent do **control and visual gaps** affect the effectiveness of simulated evaluation?
4) When building simulation environments, we simplified object and robot's physical properties like center of mass and static & dynamic friction, as their precise modeling and system identification are challenging and time-consuming. Is our simulated evaluation sensitive to such **physical property gaps**?
5) Are our results applicable for a **different physics simulator**?

A. Experimental Setup

To quantitatively evaluate correlations between real and simulation policy performance, we perform paired sim-and-real experiments. We use popular open-source generalist robot policies: RT-1-X [11] and Octo [50] (Octo-Base and Octo-Small). For evaluations in the Google Robot environments, we additionally use a number of RT-1 [6] checkpoints at various stages of training: RT-1 trained to convergence (RT-1 (Converged)), RT-1 at 15% of training steps (RT-1 (15%)), and RT-1 at the beginning of training (RT-1 (Begin)). We also report results on RT-2-X [5]. Detailed evaluation protocols for each task, including the number of evaluation trials, are presented in the supplementary.

For Octo simulated evaluations, since the model involves a non-deterministic diffusion head, we average its success rates across three different random seeds to produce a lower-variance estimate of the policy's simulation performance. Additionally, for Google Robot simulated evaluations, we average results over four versions of robot arm and gripper colors to account for changes in arm texture during real robot rollouts (see Section IV-B). For the WidowX environments, given the consistent black color of the arm and gripper across videos, we skip this step.

B. SIMPLER Environments Show Strong Performance Correlations with Real-World Evaluations

We summarize the results of our main paired real-world and simulation evaluations in Fig. 6 and Fig. 7. We observe a strong correlation between the relative performances in simulation and in the real world across most policy checkpoints

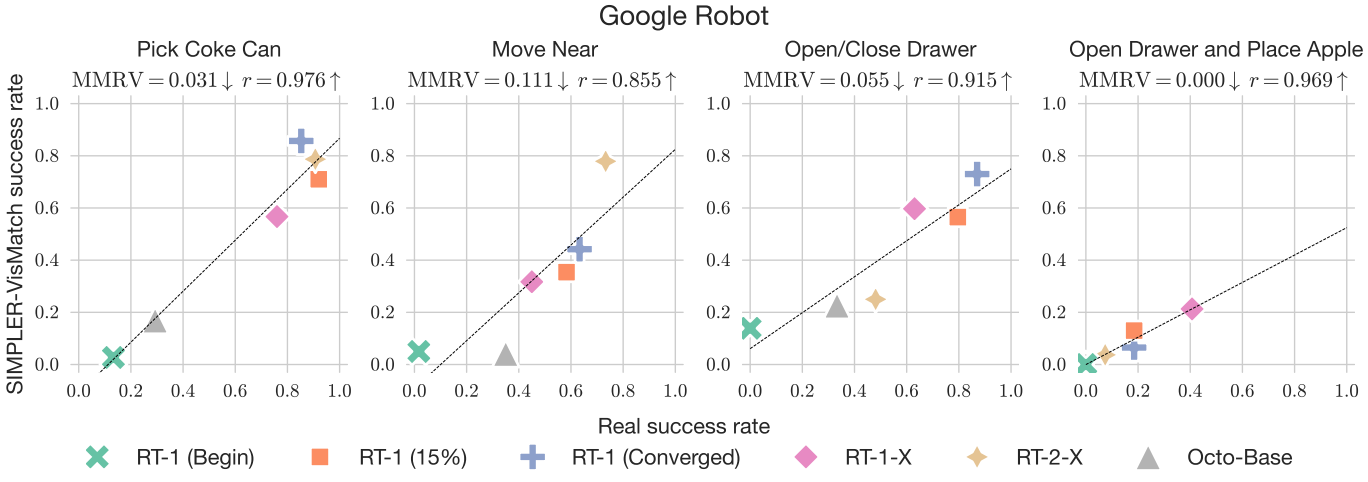


Fig. 6: Real vs. SIMPLER success rates on Google Robot tasks. SIMPLER environments with the "Visual Matching" evaluation setup show strong correlation to real policy performance. Good simulated evaluation environments have *low* MMRV, i.e., only produce rank violations for policies with similar real-world success rates, along with *high* Pearson correlation (r). See Table IV for detailed results.

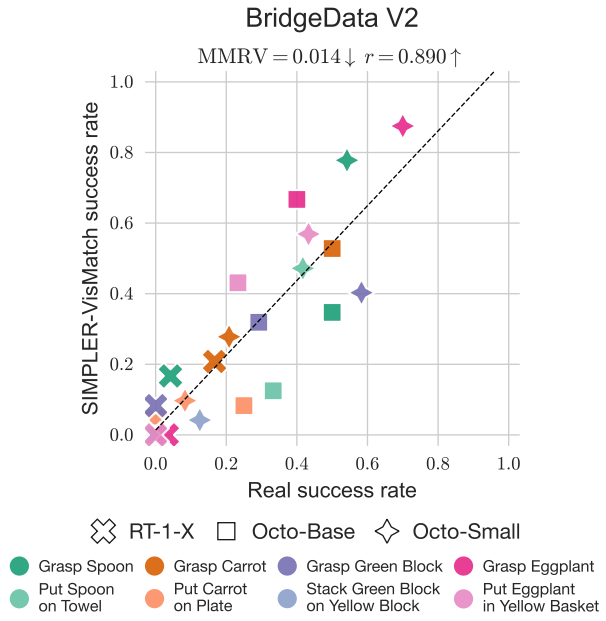


Fig. 7: Real vs. simulation success rates for BridgeData V2 tasks. SIMPLER evaluations have strong correlation with real policy performance: for all but one task (light orange), all policies are ranked correctly. MMRV and Pearson r are estimated per task and averaged. See Table V for detailed results.

Evaluation Protocol	Pick Coke Can	Move Near	Drawer	Average
		MMRV ↓		
Validation MSE	0.412	0.408	0.306	0.375
SIMPLER-VarAgg (ours)	0.084	**0.111**	0.235	0.143
SIMPLER-VisMatch (ours)	**0.031**	**0.111**	**0.027**	**0.056**
		Pearson r ↑		
Validation MSE	0.464	0.230	0.231	0.308
SIMPLER-VarAgg (ours)	0.960	**0.887**	0.486	0.778
SIMPLER-VisMatch (ours)	**0.976**	0.855	**0.942**	**0.924**

TABLE I: Comparison of manipulation policy evaluation protocols for ranking 6 common open-source policy checkpoints (3 RT-1 checkpoints, RT-1-X, RT-2-X, Octo-Base) on Google Robot tasks. Using SIMPLER results in much stronger correlation with real evaluation than using validation MSE. Furthermore, "Visual Matching" (VisMatch) outperforms "Variant Aggregation" (VarAgg). See Fig. 6 and Table IV for a detailed breakdown of results per policy.

and tasks we evaluate. This is an encouraging result for using SIMPLER as a performance measurement tool during policy development. Concretely, we find that policies with high real-world performance, such as RT-1 (Converged) and RT-2-X on Google Robot tasks and Octo-Small on BridgeData V2 tasks, also obtain high performance in our simulated evaluations. Models that obtain low real-world performance, such as RT-1 (Begin) on Google Robot tasks and RT-1-X on BridgeData V2 tasks, similarly have low performance in SIMPLER evaluations in simulation. This consistency is reflected in low values for the MMRV metric introduced in Section III and high values for Pearson r. Additionally, we find that some policies have higher sensitivity to visual differences between simulation and real-world environments. For example, RT-1 (15%) and Octo-Base exhibit the most significant success rate change between real world and simulation.

In Table I, we compare SIMPLER with using action MSE on validation episodes for policy ranking. Using validation loss for model selection is common in supervised learning. However, our analysis supports an intuition shared by many robotics practitioners: we show that for imitation learning, validation MSE is *not* a good proxy for a policy's real-world performance, leading to high MMRV and low Pearson r. On the other hand, SIMPLER evaluations more accurately reflect relative policy performances in the real-world, obtaining significantly lower MMRV and higher Pearson r. Additionally, we find that an alternative implementation of SIMPLER using "Variant Aggregation" (Section IV-B) instead of "Visual Matching" performs worse. As mentioned before, some policies are more sensitive to visual discrepancies between reality and simulation. These issues are exacerbated under Variant Aggregation, which has much larger visual distribution shifts to the real world (Fig. 11), leading to higher MMRV and lower Pearson correlation. In summary, **simulated manipulation policy evaluation with SIMPLER leads to strong correlation with real-world policy performance**, and we recommend SIMPLER-"Visual Matching" as the default approach since it directly minimizes visual discrepancies between real and simulated environments.

C. SIMPLER Evaluations Accurately Model Policy Robustness to Distribution Shifts

In the previous section, we demonstrated that policy evaluations on SIMPLER environments exhibit strong performance relationship correlations with real robot evaluations, based on average performances across evaluation trials. Beyond comparing average policy performances, it would be beneficial to let practitioners gauge more nuanced aspects of a policy's behavior, such as its robustness to distribution shifts like lighting, background, and texture changes. We ask: do SIMPLER evaluations accurately reflect a policy's real-world behavior under such distribution shifts, and can they thus be used for more fine-grained policy evaluation beyond average performance?

To test this, we use SIMPLER environments to perform controlled experiments along five distribution shift axes inspired by Xie et al. [69]: background, lighting, distractors, table texture, and robot camera pose. We adopt the base environment setup and the two variations per axis from our Variant Aggregation evaluation (see Section IV-B and Section C-C), adding two more variations for the new camera pose axis. We evaluate two RT-1 checkpoints with different robustness behaviors to distribution shifts. For simulated results and real-world results, we report the difference in success rate with and without each distribution shift:

$$\Delta\text{Success}(\text{shift}) = \frac{1}{2}\sum_{k=1}^{2}\left(\text{Success}(\text{shift}, k) - \text{Success}(\text{base})\right) \tag{6}$$

Results are summarized in Fig. 8 and Table VI. We find that SIMPLER evaluations accurately reflect the policies'

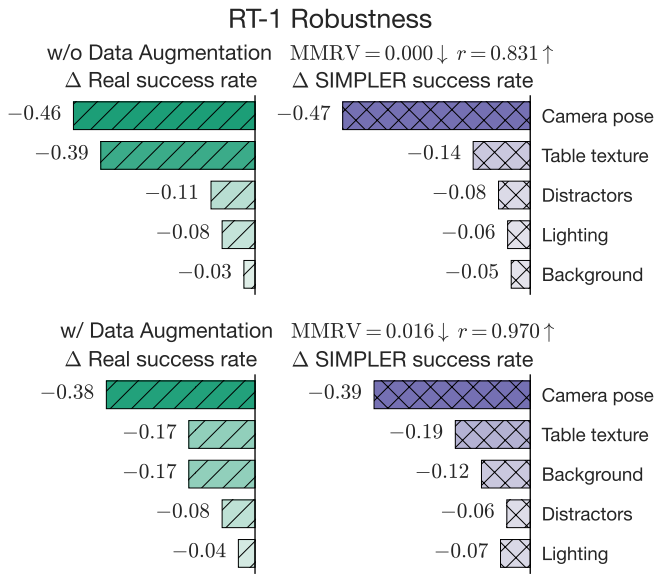


Fig. 8: Change in success rate under various distribution shifts for two RT-1 policies trained without and with data augmentation. Success rates are averaged across Google Robot "Pick Coke Can" and "Move Near" tasks. SIMPLER evaluations accurately capture each policy's sensitivity to distribution shifts as well as the effect of training with data augmentation. MMRV and Pearson r are calculated over different factors of distribution shifts within the same policy. See Table VI for detailed results.

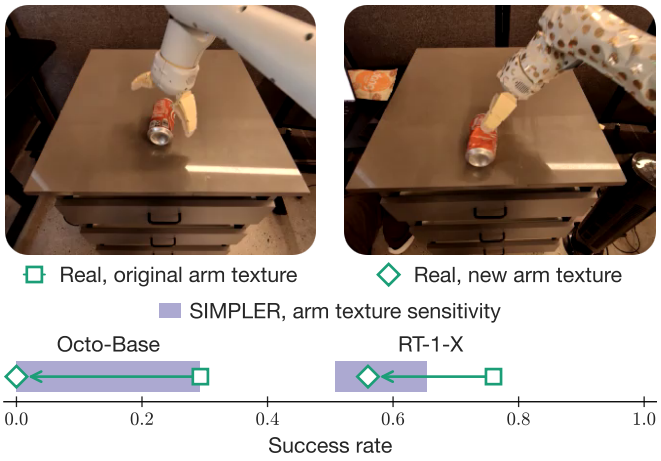


Fig. 9: SIMPLER evaluations can predict policy sensitivity to changes in arm texture for multiple policies in Google Robot "Pick Coke Can" task. See Table VIII for detailed results.

robustness to various distribution shifts in the real world. Notably, in both simulation and reality, changing robot camera poses and table textures has a significant impact on policy performance, while the impact of lighting changes and distractors are relatively minor.

Furthermore, we find that an even more fine-grained analysis beyond Fig. 8 is possible in our simulated evaluations. For example, when varying the table texture in our real-world evaluations, we find that both policies are more robust to unseen *solid* table colors than unseen *patterned* table textures (4% vs 25% performance decrease on average). This behavior is well reflected in our simulated evaluations as well: policy performance in simulation decreases by 2% on average when introducing new solid table colors and by 24% for new patterned textures.

Testing novel distribution shifts. Based on these results, we put our simulated evaluations to the test and ask: can SIMPLER evaluations be used to *predict* the robustness of policies to new distribution shifts in the real world? Throughout our simulated evaluations, we observe that Octo-Base is particularly sensitive to changes in the simulated robot arm textures. Specifically, under our "Visual Matching" evaluation setup, its success rate is 0% using the untuned robot arm, but 29.3% using one of our tuned robot arms. On the other hand, RT-1-X, also trained on the same Open-X-Embodiment dataset [11], exhibits higher robustness to different simulated robot arm textures. To test whether this trend observed in simulation holds in real-world evaluations, we design a novel real-world distribution shift evaluation, where we change the real robot arm texture by wrapping it using multiple gift wrapping papers (see Fig. 9 for an example). We report results in Fig. 9 and Appendix Table VIII. The real-world evaluations support the simulated results: Octo-Base is more sensitive to changes in arm texture than RT-1-X. This indicates that simulated evaluations in SIMPLER environments can be predictive of real-world policy behaviors under novel distribution shifts.

D. Ablation Studies

We ablate the effect of the approaches we introduced in Section IV for closing the control and visual gaps between simulation and real-world evaluations. We also investigate the sensitivity of our simulated evaluation to the real-to-sim physical property gap. Additionally, we show that our previous findings are independent of the choice of the underlying physics simulator.

Effect of system identification. To test the effect of system identification on the correlation between simulated and real-world evaluations, we repeat the simulated Google Robot evaluations from Section VI-B (using the "Visual Matching" approach), but perturb the stiffness and damping parameters of the robot's joints that were determined with the system identification approach introduced in Section IV-A. In Table II, we show that the noisy system identification parameters lead to worse MMRV, i.e., worse correlation between simulated and real-world evaluations. This underlines the importance of accurate system identification for simulated evaluation.

Effect of visual matching. We ablate the impact of the approaches we introduced in Section IV-B for matching the visual appearances between simulated and real-world evaluations. We use the RT-1 (Converged), RT-1 (Begin), and RT-1-X checkpoints on the Google robot drawer opening and closing tasks, and we compare the correlations between simulated and real-world evaluations for different combinations of background "green-screening", object texture, and robot texture settings. Results are reported in Table III. We observe the lowest MMRV and real-to-sim performance gap when combining background "green-screening" with object texture tuning for *both* drawer and robot assets. Interestingly, only

Control Parameters	Control Loss ↓	MMRV ↓
Setting 1	0.267	0.070
Setting 2	0.432	0.100
SIMPLER SysID	**0.131**	**0.031**

TABLE II: Ablation of different control parameters for the Google Robot "Pick Coke Can" task. SIMPLER's system identification approach (Section IV-A, Fig. 4) achieves the most accurate trajectory tracking (control loss) and the best real-to-sim performance correlation (MMRV).

Green Screen	Drawer Matching	Robot Matching	MMRV ↓	Real-Sim Success Gap ↓
✗	✗	✗	0.087	0.272
✗	✓	✗	0.087	0.266
✗	✗	✓	0.087	0.272
✗	✓	✓	0.087	0.328
✓	✗	✗	0.087	0.198
✓	✓	✗	0.142	0.253
✓	✓	✓	**0.050**	**0.136**

TABLE III: Ablation of methods for closing the visual gap between simulated and real environments, evaluated with three policies on the Google Robot "Open / Close Drawer" tasks (see Table IX for detailed results). Using a combination of "green-screened" background and curated foreground object and robot assets provides the best real-to-sim performance correlations.

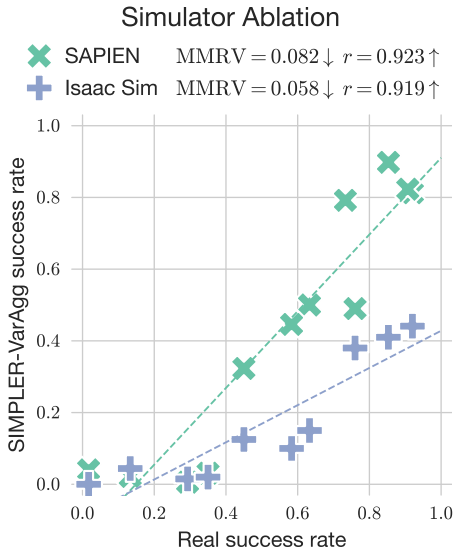


Fig. 10: Comparison of SIMPLER-"Variant Aggregation" using SAPIEN (default) vs. Isaac Sim [49] on Google Robot "Pick Coke Can" and "Move Near" tasks. Both physics simulators lead to good correlation between simulated and real-world evaluation success rates. See Table IV and Table XI for detailed results.

tuning the drawer but not the robot texture, or only using tuned textures but no background "green-screening", leads to no correlation improvement over the baseline that does not apply any approach for narrowing the real-to-sim visual appearance gap. We hypothesize that *inconsistencies* between the appearance of different parts of a scene can deteriorate simulation policy performance. Thus, the approaches we introduced in Section IV-B for narrowing the visual gap between simulated and real scene can significantly improve real-and-sim evaluation performance correlation, but *only* if applied jointly and to all parts of a scene.

Sensitivity to physical property gap. When developing SIMPLER environments, we simplified the physical properties (e.g., center of mass and friction coefficients) of objects and robots due to the complexity and time-consuming nature of precise modeling and system identification. In this section, we investigate whether our simulated evaluation is sensitive to such real-to-sim physical property gap. We conduct 2 experiments: **(1)** For the "pick coke can" task, we vary the mass of the empty coke can (by varying its density), along with the static friction of the gripper finger; (2) For the "open / close drawer" task, we vary the joint frictions of the articulated cabinet. We report the MMRV and the Pearson correlation results in Appendix Tab. X. We find that our simulated evaluation remains effective across a spectrum of plausible physical property parameters, evidenced by the low MMRV and the high Pearson correlation, even though altering these parameters has a moderate ($\leq$ 15%) impact on the success rates of different policies.

Sensitivity to choice of physics simulator. To investigate whether our results are sensitive to the underlying physics simulator, we reproduce the Google Robot evaluation in Isaac Sim [49]. The results in Fig. 10 and Appendix Table XI show that the performance of SIMPLER is reproducible with the Isaac simulator. In particular, we also observe a strong real-to-sim performance correlation across most checkpoints for SIMPLER-Isaac. This suggests that for the tasks we tested, which for the most part involve rigid body manipulations, the choice of physics simulator is not critical and our framework can be reproduced on alternative physics simulators.

VII. Conclusion

As the capabilities of generalist robot manipulation policies grow, developing scalable approaches for policy evaluation will be crucial to enable rapid iteration and improvement of algorithms, models and datasets. In this work, we investigate simulated evaluation as a tool to complement costly real-world robot evaluations. We introduce SIMPLER, a suite of simulated manipulation evaluation environments for commonly used real robot evaluation setups. In paired sim-&-real experiments across multiple open-source generalist policies, we show that SIMPLER results in strong performance relationship correlations with real evaluations. Additionally, we demonstrate that SIMPLER evaluations accurately capture fine-grained characteristics of real-world policies beyond average performance, such as their robustness to various distribution shifts. Finally, we ablate which design decisions for building simulated manipulation evaluation environments are important for strong real-to-sim correlation.

Limitations. Our current set of environments has several limitations. First, we focus our evaluations on rigid-object manipulation tasks, as their physics are most straight-forward to simulate with modern physics simulators. Much recent work in robotic manipulation has demonstrated impressive tasks that

go beyond rigid object manipulation [20, 10, 33]. Extending simulated evaluation beyond rigid object tasks, e.g., by leveraging recent work on soft-object physics simulation [38], is an exciting avenue for future work. Additionally, our current "green-screening" approach is limited to fixed cameras and does not accurately capture object shadows and other visual details. Finally, our current pipeline for generating simulated evaluation environments involves some manual effort in curating assets and assembling scenes. Enabling a fully-automatic and more scalable pipeline for creating thousands of realistic simulated environments is an ambitious goal for future work.

VIII. Acknowledgements

We sincerely thank Jeffery Bingham and Paul Wohlhart from Google DeepMind for clarifying some details of the Google Robot controller. We also thank Justice Carbajal, Samuel Wan, Jornell Quiambao, Deeksha Manjunath, Jaspiar Singh, Sarah Nguyen, Jodilyn Peralta, and Grecia Salazar for conducting real-world Google Robot experiments. Additionally, we thank Fanbo Xiang from UC San Diego for his help on matching the real and simulation visual appearances of foreground objects. Kyle Hsu was supported by a Sequoia Capital Stanford Graduate Fellowship.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*, 2022.

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[3] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018.

[4] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. Robocat: A self-improving foundation agent for robotic manipulation, 2023.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[7] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[8] Constantinos Chamzas, Carlos Quintero-Pena, Zachary

Kingston, Andreas Orthey, Daniel Rakita, Michael Gleicher, Marc Toussaint, and Lydia E Kavraki. Motion-benchmaker: A tool to generate and benchmark motion planning datasets. *IEEE Robotics and Automation Letters*, 7(2):882–889, 2021.

[9] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[10] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[11] Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[12] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.

[13] Sudeep Dasari, Jianren Wang, Joyce Hong, Shikhar Bahl, Yixin Lin, Austin S. Wang, Abitha Thankaraj, Karanbir Chahal, Berk Çalli, Saurabh Gupta, David Held, Lerrel Pinto, Deepak Pathak, Vikash Kumar, and Abhinav Gupta. RB2: robotic manipulation benchmarking with a twist. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[14] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174, 2020.

[15] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

[16] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[17] Yuqing Du, Daniel Ho, Alex Alemi, Eric Jang, and Mohi Khansari. Bayesian imitation learning for end-to-end mobile manipulation. In *International Conference on Machine Learning*, pages 5531–5546. PMLR, 2022.

[18] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Manipulathor: A framework for visual object manipulation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*,

pages 4495–4504, 2021.

[19] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11444–11453, 2020.

[20] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[21] Niklas Funk, Charles Schaff, Rishabh Madan, Takuma Yoneda, Julen Urain De Jesus, Joe Watson, Ethan K Gordon, Felix Widmaier, Stefan Bauer, Siddhartha S Srinivasa, et al. Benchmarking structured policies and policy optimization for real-world dexterous object manipulation. *IEEE Robotics and Automation Letters*, 7(1): 478–485, 2021.

[22] Ran Gong, Jiangyong Huang, Yizhou Zhao, Haoran Geng, Xiaofeng Gao, Qingyang Wu, Wensi Ai, Ziheng Zhou, Demetri Terzopoulos, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. Arnold: A benchmark for language-grounded task learning with continuous states in realistic 3d scenes. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20426–20438, 2023.

[23] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Z. Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yuan Yao, Xiao Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *The Eleventh International Conference on Learning Representations*, 2023.

[24] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10920–10926, 2020.

[25] Xiaoxia Huang, Ian Walker, and Stan Birchfield. Occlusion-aware reconstruction and manipulation of 3d articulated objects. In *2012 IEEE international conference on robotics and automation*, pages 1365–1371. IEEE, 2012.

[26] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[27] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[28] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[29] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021.

[30] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[31] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensoir: Tensorial inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[32] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, S. Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5: 6670–6677, 2019.

[33] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[34] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3992–4003, 2023.

[35] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt, and Christopher Orlowski. The darpa robotics challenge finals: Results and perspectives. *The DARPA robotics challenge finals: Humanoid robots to the rescue*, pages 1–26, 2018.

[36] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

[37] Chengshu Li, Ruohan Zhang, J. Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.

[38] Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chen-

fanfu Jiang, and Danny M. Kaufman. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.*, 39(4), aug 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392425.

[39] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. *arXiv preprint arXiv:2311.07885*, 2023.

[40] Ziyuan Liu, Wei Liu, Yuzhe Qin, Fanbo Xiang, Minghao Gou, Songyan Xin, Máximo A. Roa, Berk Çalli, Hao Su, Yu Sun, and Ping Tan. Ocrtoc: A cloud-based competition and benchmark for robotic grasping and manipulation. *IEEE Robotics and Automation Letters*, 7:486–493, 2021.

[41] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1957–1964. IEEE, 2016.

[42] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[43] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[44] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2020.

[45] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[46] Mark Moll, Ioan A Sucan, and Lydia E Kavraki. Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. *IEEE Robotics & Automation Magazine*, 22(3):96–102, 2015.

[47] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Cathera Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[48] Galen E Mullins, Paul G Stankiewicz, and Satyandra K Gupta. Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1443–1450. IEEE, 2017.

[49] NVIDIA. Nvidia isaac sim. https://developer.nvidia.com/isaac-sim, 2022.

[50] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. https://octo-models.github.io, 2023.

[51] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895. ISSN 03701662. URL http://www.jstor.org/stable/115794.

[52] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[53] Martin Pincus. A monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research*, 18(6):1225–1228, 1970. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/169420.

[54] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023.

[55] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yinsong Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, 2022.

[56] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023.

[57] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11154–11163, 2020.

[58] Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022.

[59] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets,

Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

[60] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023.

[61] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 100(3/4):441–471, 1987. ISSN 00029556. URL http://www.jstor.org/stable/1422689.

[62] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Mart'in-Mart'in, Fei Xia, Kent Vainio, Zheng Lian, Cem Gokmen, S. Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pages 477–490. PMLR, 2022.

[63] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimír Vondruš, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel X. Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266, 2021.

[64] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pages 23–30. IEEE, 2017.

[65] Karl Van Wyk, Joe Falco, and Elena Messina. Robotic grasping and manipulation competition: Future tasks to support the development of assembly robotics. In *Robotic Grasping and Manipulation: First Robotic Grasping and Manipulation Challenge, RGMC 2016, Held in Conjunction with IROS 2016, Daejeon, South Korea, October 10–12, 2016, Revised Papers 1*, pages 190–200. Springer, 2018.

[66] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[67] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

[68] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[69] Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. *arXiv preprint arXiv:2307.03659*, 2023.

[70] Kaizhi Yang, Xiaoshuai Zhang, Zhiao Huang, Xuejin Chen, Zexiang Xu, and Hao Su. Movingparts: Motion-based 3d part discovery in dynamic radiance field. In *The Twelfth International Conference on Learning Representations*, 2024.

[71] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023.

[72] Zhao-Heng Yin, Binghao Huang, Yuzhe Qin, Qifeng Chen, and Xiaolong Wang. Rotating without seeing: Towards in-hand dexterity through touch. In *Robotics: Science and Systems*, 2023.

[73] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[74] Jingwei Zhang, Lei Tai, Peng Yun, Yufeng Xiong, Ming Liu, Joschka Boedecker, and Wolfram Burgard. Vr-goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robotics and Automation Letters*, 4 (2):1148–1155, 2019.

[75] Yunchu Zhang, Liyiming Ke, Abhay Deshpande, Abhishek Gupta, and Siddhartha S. Srinivasa. Cherry-picking with reinforcement learning. In *Robotics: Science and Systems*, 2023.

[76] Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, Chelsea Finn, and Abhinav Gupta. Train offline, test online: A real robot learning benchmark. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 9197–9203. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10160594.

[77] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

APPENDIX A
CONTRIBUTIONS

Project leads: Xuanlin Li, Kyle Hsu
Main Methodology: Xuanlin Li, Jiayuan Gu, Kyle Hsu
SIMPLER Environment Building: Xuanlin Li, Jiayuan Gu, Kyle Hsu
SIMPLER-SAPIEN Implementation & Experiments: Xuanlin Li, Jiayuan Gu
SIMPLER-Isaac Sim Implementation & Experiments: Kyle Hsu
Real Robot Experiments: Homer Rich Walke, Oier Mees, Ted Xiao, Kyle Hsu, Karl Pertsch, Ishikaa Lunawat, Isabel Sieh, and people in acknowledgements
Paper Writing: Xuanlin Li, Karl Pertsch, Kyle Hsu, Quan Vuong, Ted Xiao, Jiayuan Gu, Oier Mees
SIMPLER Codebase Release: Xuanlin Li, Jiayuan Gu, Karl Pertsch, Kyle Hsu, Oier Mees
Website: Oier Mees, Xuanlin Li, Ted Xiao, Karl Pertsch
Video: Kyle Hsu, Karl Pertsch
Advising: Karl Pertsch, Oier Mees, Chuyuan Fu, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, Ted Xiao

APPENDIX B
FULL ENVIRONMENT AND EVALUATION PROTOCOL DETAILS

In this section, we provide detailed descriptions of our SIMPLER environments along with our simulation and real-world evaluation protocols.

For the Google Robot, we adopt the following language-conditioned tasks:

- **"pick coke can"**. The robot is instructed to grasp the empty coke can on the table and lift it up. In the default setting, no distractors are added to the scene. We place the coke can in 3 different orientations: horizontally laying, vertically laying, and standing. For each orientation, we place the coke can at 25 grid positions within a rectangle on the tabletop, yielding 25 trials per orientation and 75 trials in total.
- **"move {obj1} near {obj2}"**. We place a triplet of objects on the tabletop in a triangle pattern. In each trial, one object serves as the source object, one serves as the target, and the other serves as the distractor (this creates 6 trials for each triplet and each triangle pattern). We randomly choose 5 triplets of objects among a total of 8 objects (blue plastic bottle, pepsi can, orange, 7up can, apple, sponge, coke can, redbull can), and adopt 2 triangle patterns (upright and inverted). This creates a total of $5 \times 2 \times 6 = 60$ trials. The 5 triplets chosen are:
  - blue plastic bottle, pepsi can, orange
  - 7up can, apple, sponge
  - coke can, redbull can, apple
  - sponge, blue plastic bottle, 7up can
  - orange, pepsi can, redbull can
- **"(open / close) (top / middle / bottom) drawer"**. The robot is positioned in front of a cabinet that contains 3 drawers and instructed to open / close a specific drawer, testing its ability to manipulate articulated objects. We place the robot at 9 grid positions within a rectangle on the floor, yielding a total of $9 \times 3 \times 2 = 54$ trials.
- **"open top drawer; place apple into top drawer"**. The robot opens the top drawer and places the apple from the cabinet top into the top drawer, testing its ability to perform longer-horizon tasks. We place the robot at 3 different positions on the floor and the apple at 9 different positions within a grid on the cabinet top, yielding a total of $3 \times 9 = 27$ trials. Initially, the policies receive the "open top drawer" instruction. We switch to the "place apple into top drawer" instruction once the robot outputs the "terminate" token or after half of the time limit has elapsed.

For the WidowX + Bridge (with WidowX-250 6DOF robot), we adopt the following tasks:

- **"put the spoon on the towel"**. We place the spoon on a vertex of a square (with edge length 15cm) on the tabletop, and we place the towel on another vertex. The spoon's initial orientation switches between horizontal and vertical, requiring the robot to perform gripper re-orientation. This creates a total of $2 \times 12 = 24$ trials.
- **"put carrot on plate"**. We adopt a similar setup as "put the spoon on the towel", replacing the spoon with carrot and the towel with plate.
- **"stack the green block on the yellow block"**. We place a green block on a vertex of a square on the tabletop, and we position a yellow block on another vertex. The block dimensions are 3cm. We also adopt two differently-sized squares (edge length 10cm and 20cm). This creates a total of $2 \times 12 = 24$ trials.
- **"put eggplant into yellow basket"**. We place an eggplant on the right basin of a sink, and we place a yellow basket on the left basin. The eggplant is dropped into the sink at a random position and orientation, and we ensure that the eggplant is directly graspable (i.e., not too close to the edges of the sink basin). We perform a total of 24 trials.

The number of evaluation trials we present above pertain to the real-world evaluation setup. For our "Variant Aggregation" simulation evaluation setup, the number of trials is multiplied by the number of simulation environment variants. For our "Visual Matching" simulation evaluation setup, the number of trials is multiplied by the number of tuned robot arm colors for the Google Robot evaluation setup, along with the number of seeds for the Octo policies.

APPENDIX C
MORE IMPLEMENTATION DETAILS OF OUR REAL-TO-SIM EVALUATION SYSTEM

A. Robot Controllers

Google Robot Given translation, rotation, and gripper action output from a model, we adopt Algorithm 1 in simulator to execute the action commands. The simulation frequency

Algorithm 1 Google Robot Controller in Simulation

Require: (1) Current end-effector action $(\mathbf{x}_a, R_a)$, along with sensed arm joint positions and velocities $q_{\text{arm}}, v_{\text{arm}}$; (2) Current gripper action g_a, along with sensed gripper joint position and velocity $q_{\text{grip}}, v_{\text{grip}}$; (3) Simulation frequency H_{sim} (501 in our implementation), action output frequency (control frequency) H_{ctrl} (3 in our implementation following [6]); (4) Arm velocity, acceleration, and jerk limits L_{arm} (equal to 1.5, 2.0, 50.0 respectively); (5) Gripper velocity, acceleration, and jerk limits L_{grip} (equal to 1.0, 7.0, 50.0 respectively); (6) Current action timestep T within an episode; (7) A planner that takes goal and initial joint positions and velocities as input (along with velocity, acceleration, and jerk constraints), and outputs a time-parametrized trajectory.

Arm motion planning
$(\mathbf{x}, R) = \text{ForwardKinematics}(q_{\text{arm}})$
$(\mathbf{x}_{\text{goal}}, R_{\text{goal}}) = (\mathbf{x}_a + \mathbf{x}, R_a \cdot R_{\text{arm}})$
$(q_{\text{goal}}, v_{\text{goal}}) = (\text{InverseKinematics}(\mathbf{x}_{\text{goal}}, R_{\text{goal}}, q_{\text{arm}}), 0.0)$
ArmPlan = Planner($q_{\text{goal}}, v_{\text{goal}}, q_{\text{arm}}, v_{\text{arm}}, L_{\text{arm}}$)
# Gripper motion planning
if $T = 0$ **then** ▷ At the beginning of episode
  $q_{\text{lastplan,grip}}, v_{\text{lastplan,grip}} = q_{\text{grip}}, 0.0$
  $q_{\text{lastgoal,grip}} = q_{\text{grip}}$
end if
if $|g_a| < 0.01$ **then** ▷ Small action filtering
  $q_{\text{goal,grip}} = q_{\text{lastgoal,grip}}$
else
  $q_{\text{goal,grip}} = q_{\text{lastplan,grip}} + g_a$
end if
$v_{\text{goal,grip}} = 0.0$
GripPlan = Planner($q_{\text{goal,grip}}, v_{\text{goal,grip}}, q_{\text{lastplan,grip}}, v_{\text{lastplan,grip}}, L_{\text{grip}}$)
# Execute arm and gripper plans at each simulation step
for each $i = 1 \cdots \frac{H_{\text{sim}}}{H_{\text{ctrl}}}$ **do**
  $t = \frac{i}{H_{\text{sim}}}$
  $q_{\text{lastplan}}, \_ = \text{ArmPlan}(t)$
  SetArmJointPosTarget(q_{lastplan})
  $q_{\text{lastplan,grip}}, v_{\text{lastplan,grip}} = \text{GripPlan}(t)$
  SetGripperJointPosTarget($q_{\text{lastplan,grip}}$)
  SetGripperJointVelTarget($v_{\text{lastplan,grip}}$)
end for each
$q_{\text{lastgoal,grip}} = q_{\text{goal,grip}}$
$T = T + 1$

in the algorithm refers to the number of simulation steps per second, while the control frequency refers to the number of control commands (policy action outputs) per second. We use the open-source library Ruckig[1] for time-optimal joint motion planning with velocity, acceleration, and jerk constraints. Note that the duration of planned trajectories may exceed the interval between two control commands.

WidowX We present our WidowX controller implementation in Algorithm 2.

B. Robot and Object Assets

Robots For Google Robot, we convert the publically-released MuJoCo `.mjcf` robot description to URDF robot description. We also refine the collision mesh of the robot base link from the original assets to prevent erroneous mesh penetrations. For WidowX, we directly export the URDF robot

[1]https://github.com/pantor/ruckig

Algorithm 2 WidowX Controller in Simulation

Require: (1) Current end-effector action $(\mathbf{x}_a, R_a)$, along with sensed arm joint positions q_{arm}; (2) Current gripper action g_a, along with sensed gripper joint position q_{grip}; (3) Simulation frequency H_{sim} (500 in our implementation), action output frequency (control frequency) H_{ctrl} (5 in our implementation following); (4) Current action timestep T within an episode; (5) A function S that maps a $\mathbb{R}^3$ position vector and a 3x3 $\mathbb{SO}(3)$ rotation matrix to a 4x4 $\mathbb{SE}(3)$ matrix.

if $T = 0$ **then** ▷ At the beginning of episode
  $q_{\text{lastgoal}} = q_{\text{arm}}$
end if
$(\mathbf{x}, R) = \text{ForwardKinematics}(q_{\text{lastgoal}})$
$(\mathbf{x}_{\text{goal}}, R_{\text{goal}}) = S^{-1}(S(\mathbf{x}, I) \cdot S(\mathbf{x}_a, R_a) \cdot S(-\mathbf{x}, I) \cdot S(\mathbf{x}, R_{\text{arm}}))$
$q_{\text{goal}} = \text{InverseKinematics}(\mathbf{x}_{\text{goal}}, R_{\text{goal}}, q_{\text{arm}})$
$q_{\text{goal,grip}} = g_a$
SetArmJointPosTarget(q_{goal})
SetGripperJointPosTarget($q_{\text{goal, grip}}$)
$q_{\text{lastgoal}} = q_{\text{goal}}$
$T = T + 1$

descriptions from the official Interbotix repository using ROS. To simulate the Google Robot, we find that the Projected Gauss-Seidel solver in PhysX causes mesh penetration behaviors during the process of object grasping. Thus, we enable the Temporal Gauss-Seidel solver in both SAPIEN and Isaac Sim's simulation backends to produce correct grasping behaviors.

The Google Robot uses a customized egocentric camera mounted on the robot head, while the WidowX + Bridge V2 setup uses a Logitech C920 third-view camera.

Objects We adopt the following procedure to obtain object assets. Except creating precise models for articulated objects like cabinets, the process does not involve heavy manual effort.

- Obtain raw 3D object models from public repositories (e.g., Objaverse [15]), from 3D scanning of objects purchased from Amazon, from single-view 3D generation (e.g., One-2-3-45++ [39]), or from manual modeling based on precise measurements of real-world counterparts (we only used the last technique for articulated objects like cabinets).
- Process 3D object models in Blender such that the dimensions of objects are similar to those used in the real world, and that the object meshes do not contain too many vertices (to limit the sizes of object meshes).
- Optionally, use our Visual Matching approach (see below) to improve the texture of 3D object models.
- Export visual mesh and collision mesh of objects. For collision mesh, further perform CoACD [67] to obtain watertight and locally convex collision meshes. Optionally, simplify the resulting collision mesh and perform minor modifications using Blender (e.g., make the bottom of cans or bottles flat).
- Set the object to have a simple uniform density by querying their common material density in GPT-4 or google search, or (for objects with non-uniform densities like empty coke can), querying their mass and dividing by their visual mesh volume.

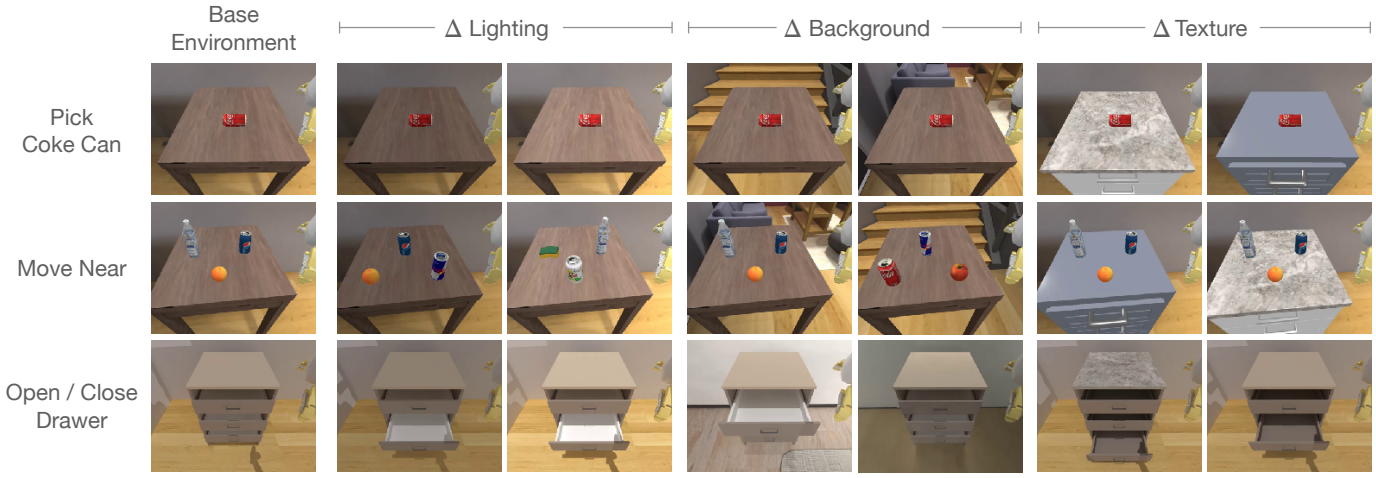


Fig. 11: Subset of environment variations under our "Variant Aggregation" evaluation setup, visualized in SAPIEN from Google Robot's egocentric view. The variations cover different lightings, backgrounds and table textures and are modified from ReplicaCAD [59] scenes.

To perform visual matching of object textures, we adopt the following steps: (1) Crop the target object in a real image using SAM [34]; (2) Perform a *coarse* estimation of object pose by importing it into the simulation and adjusting its position such that its simulation segment mask overlaps with the real one; (3) Employ differential rendering (using Nvdiffrast) to optimize the simulation asset's pose such that it *precisely aligns* with the real object's segmentation mask; (4) "Unproject" the real object's RGB texture values onto the simulation object mesh; (5) Optionally, generate the remaining views of the object through a diffusion model (Zero123++ [60]), and refine the poses of novel views using a rendering loss with the existing object view. Finally, unproject the novel view textures onto the simulation object mesh. This whole process is semi-automatic, and can thus be completed efficiently. We release command-line python scripts for this process at https://github.com/Jiayuan-Gu/GeTex.

C. SIMPLER-Variant Aggregation

A common approach for addressing visual gaps in sim-to-real policy training is domain randomization. By performing training across a range of randomized parameters, such as textures and lighting, prior works aim to obtain policies that are robust to visual distribution shifts in the real-world [52, 64]. Similarly, in real-to-sim evaluation, we can aggregate evaluation results across a range of visual simulator characteristics to obtain a more faithful signal for the policy's performance. In practice, we implement this SIMPLER-"Variant Aggregation" approach as an alternative to SIMPLER-Visual Matching, described in Section IV-B. Concretely, we create a "base" version of our simulation environment and then creating "variants" of this environment along four axes of visual variation: background, lighting, distractors, and table texture. For each axis, we construct 2 variations of the base setup similar to [69], covering backgrounds from different rooms, lighter and darker lighting, fewer and more distractors, and solid color and complex table textures. We visualize an example of such simulator variations for various table-top tasks in Fig. 11. We average policy performance in simulation across all variants of an environment to obtain our final performance estimate.

APPENDIX D
FULL RESULTS FOR REAL-AND-SIM RELATIVE POLICY PERFORMANCE CORRELATION EXPERIMENTS

In Table IV and Table V, we present full evaluation results for our experiments in Sec. VI-B, which demonstrate that SIMPLER environments show strong performance relationship correlations with real-world evaluations.

APPENDIX E
FULL RESULTS FOR REAL-AND-SIM POLICY BEHAVIOR CORRELATION EXPERIMENTS UNDER ENVIRONMENT DISTRIBUTION SHIFTS

In Table VI, Table VII, and Table VIII, we present full evaluation results for our experiments in Sec. VI-C, which demonstrate that SIMPLER environments show strong policy behavior correlations with real-world evaluations under different environment distribution shifts.

APPENDIX F
FULL RESULTS FOR MAIN PAPER ABLATION EXPERIMENTS

We present detailed results for our main paper's ablations in Table IX, Table X, and Table XI.

APPENDIX G
MORE EXPERIMENT RESULTS

A. More Ablations

Simulated vs. simulation-free evaluation approaches: To evaluate and select real-world robot manipulation policies, a

[2]After running 2 real evaluation trials, robot operators decided that since this policy would potentially damage the robot on the Drawer tasks, the real evaluation was terminated.

[3]As real evaluation was terminated due to risk of damaging the robot, we expect the MMRV to be less than this number if real evaluation were to continue.

Google Robot Evaluation Setup	Policy	Pick Coke Can				Move Near	Open / Close Drawer			Open Top Drawer and Place Apple
		Horizontal Laying	Vertical Laying	Standing	Average	Average	Open	Close	Average	Average
Real Eval	RT-1 (Converged)	0.960	0.880	0.720	0.853	0.633	0.815	0.926	0.870	0.185
	RT-1 (15%)	1.000	0.960	0.800	0.920	0.583	0.704	0.889	0.796	0.185
	RT-1-X	0.880	0.560	0.840	0.760	0.450	0.519	0.741	0.630	0.407
	RT-2-X	0.920	0.800	1.000	0.907	0.733	0.333	0.630	0.481	0.074
	Octo-Base	0.440	0.200	0.240	0.293	0.350	0.148	0.519	0.333	0.000
	RT-1 (Begin)	0.200	0.000	0.200	0.133	0.017	0.000	0.000	0.000[2]	0.000
SIMPLER Eval (Variant Aggregation)	RT-1 (Converged)	0.969	0.760	0.964	0.898	0.500	0.270	0.376	0.323	0.026
	RT-1 (15%)	0.920	0.704	0.813	0.813	0.446	0.212	0.323	0.267	0.021
	RT-1-X	0.569	0.204	0.698	0.490	0.323	0.069	0.519	0.294	0.101
	RT-2-X	0.822	0.754	0.893	0.823	0.792	0.333	0.372	0.353	0.206
	Octo-Base	0.005	0.000	0.013	0.006	0.031	0.000	0.021	0.011	0.000
	RT-1 (Begin)	0.022	0.013	0.031	0.022	0.040	0.005	0.132	0.069	0.000
	MMRV↓	0.093	0.133	0.140	0.084	0.111	0.303	0.321[3]	0.321	0.148
	Pearson r↑	0.947	0.937	0.933	0.960	0.887	0.629	0.613	0.737	0.235
SIMPLER Eval (Visual Matching)	RT-1 (Converged)	0.960	0.900	0.710	0.857	0.442	0.601	0.861	0.730	0.065
	RT-1 (15%)	0.860	0.790	0.480	0.710	0.354	0.463	0.667	0.565	0.130
	RT-1-X	0.820	0.330	0.550	0.567	0.317	0.296	0.891	0.597	0.213
	RT-2-X	0.740	0.740	0.880	0.787	0.779	0.157	0.343	0.250	0.037
	Octo-Base	0.210	0.210	0.090	0.170	0.042	0.009	0.444	0.227	0.000
	RT-1 (Begin)	0.050	0.000	0.030	0.027	0.050	0.000	0.278	0.139	0.000
	MMRV↓	0.027	0.027	0.053	0.031	0.111	0.000	0.123	0.055	0.000
	Pearson r↑	0.981	0.964	0.942	0.976	0.855	0.983	0.768	0.915	0.969

TABLE IV: Real-world and SAPIEN evaluation results across different policies on Google Robot tasks. We present success rates for the "Variant Aggregation" and "Visual Matching" approaches in Sec. IV-B. We calculate the Mean Maximum Rank Violation ("MMRV", lower is better) and the Pearson correlation coefficient ("Pearson r", higher is better) to assess the alignment between simulation and real-world relative performances across different policies.

WidowX+Bridge Evaluation Setup	Policy	Put Spoon on Towel		Put Carrot on Plate		Stack Green Block on Yellow Block		Put Eggplant in Yellow Basket	
		Grasp Spoon	Success	Grasp Carrot	Success	Grasp Green Block	Success	Grasp Eggplant	Success
Real Eval	RT-1-X	0.042	0.000	0.167	0.000	0.000	0.000	0.033	0.000
	Octo-Base	0.500	0.333	0.500	0.250	0.292	0.000	0.400	0.233
	Octo-Small	0.542	0.417	0.208	0.083	0.583	0.125	0.700	0.433
SIMPLER Eval (Visual Matching)	RT-1-X	0.167	0.000	0.208	0.042	0.083	0.000	0.000	0.000
	Octo-Base	0.347	0.125	0.528	0.083	0.319	0.000	0.667	0.431
	Octo-Small	0.778	0.472	0.278	0.097	0.403	0.042	0.875	0.569
	MMRV↓	0.000	0.000	0.000	0.111	0.000	0.000	0.000	0.000
	Pearson r↑	0.778	0.827	0.995	0.575	0.964	1.000	0.995	0.990

TABLE V: Real-world and SAPIEN simulation evaluation results for the WidowX + Bridge setup. We report both the final success rate ("Success") along with partial success (e.g., "Grasp Spoon").

Policy	Distribution Shift	Pick Coke Can			Move Near			Avg.			Real TableTop [69]
		|Δ Success|	MMRV↓	r ↑	|Δ Success|	MMRV↓	r ↑	|Δ Success|	MMRV↓	r ↑	|Δ Success|
RT-1 w/o Aug	Background	0.013	0.000	0.779	0.083	0.055	0.939	0.048	0.000	0.831	0.028
	Lighting	0.040			0.075			0.057			0.083
	Distractors	0.027			0.133			0.080			0.111
	Table Texture	0.113			0.175			0.144			0.389
	Camera Pose	0.753			0.192			0.473			0.458
RT-1 +Aug	Background	0.153	0.041	0.984	0.092	0.125	0.721	0.123	0.041	0.970	0.167
	Lighting	0.033			0.117			0.075			0.042
	Distractors	0.033			0.084			0.059			0.083
	Table Texture	0.220			0.159			0.189			0.167
	Camera Pose	0.613			0.175			0.394			0.375

TABLE VI: Impact of various distribution shifts on the tabletop manipulation performance of RT-1 policies trained with and without image augmentation. SIMPLER evaluations accurately track the policies' robustness to distribution shifts, exhibiting low Mean Maximum Rank Violation ("MMRV") and high Pearson correlation coefficient ("r") with the real world evaluations [69].

Policy	Robustness Factor	Pick Coke Can	Move Near
RT-1 w/o Aug	Base Setup	0.920	0.467
	Background	0.933/0.907	0.533/0.567
	Lighting	0.960/0.960	0.483/0.600
	Distractors	0.880/0.901	0.600[a]
	Table Texture	0.867/0.747	0.550/0.200
	Camera Pose	0.053/0.280	0.117/0.433
RT-1 +Aug	Base Setup	0.800	0.383
	Background	0.747/0.547	0.483/0.467
	Lighting	0.760/0.773	0.517/0.483
	Distractors	0.813/0.747	0.467
	Table Texture	0.667/0.493	0.450/0.133
	Camera Pose	0.267/0.107	0.200/0.217

[a]The base setup environment already contains distractors, so we construct environment variants without distractors.

TABLE VII: Success rates of different out-of-distribution generalization factors on the tabletop manipulation performance of RT-1 policies in the SAPIEN simulator. "a/b" denote results on different environment variants (lighting: darker / brighter; table texture: solid color / contrastively patterned; camera pose: oriented lower / higher).

Policy	Sim Success Range	Real Success	
		Orig Arm Texture	OOD Arm Texture
RT-1-X	[0.507, 0.653]	0.760	0.520
Octo-Base	[0.000, 0.293]	0.293	0.000

TABLE VIII: Impact of arm textures on the success rates of "Pick Coke Can" task in the SAPIEN simulator (Visual Matching evaluation setup) and in the real-world. The ranges of simulation success rates across multiple (tuned and untuned) robot arm colors can predict policy sensitivity to real-world OOD arm textures.

widely-adopted approach involves calculating the MSE loss between predicted and ground-truth actions on a set of held-out validation demonstration trajectories. We are thus interested in the following question: *Does simulated evaluation produce significantly better real-to-sim relative performance correlation than simulation-free approaches?* We conduct an experiment where we calculate the action-prediction MSE loss on the Google Robot dataset and the Bridge dataset. For the Bridge dataset, we randomly select 25 trajectories from the validation demonstration split. For the Google Robot dataset, as a validation split is not publicly available, we randomly select 25 trajectories from the training demonstrations.

We report the results in Table XII. We find that SIMPLER evaluation produces significantly better correlations between real-and-sim performances across different policies, as highlighted by a substantially-lower MMRV and a substantially-higher Pearson correlation coefficient. Furthermore, as demonstrated in Sec. VI-C of the main paper, SIMPLER evaluation reveals finegrained policy behavior modes, such as robustness to visual distribution shifts, offering insights beyond policy performance comparisons, unlike simulation-free evaluations.

Is simulated evaluation still effective on single-task policies? Previously in the main paper, we focused our simulated evaluation on policies trained on multi-task datasets, such as the Google Robot RT-1 dataset and the Open-X-Embodiment dataset, which contain $\geq$80k demonstrations. In this section, we further ask the question: *Is SIMPLER evaluation still effective on policies trained on smaller-scale data, which are potentially more sensitive to real-to-sim visual and control gaps?* To this end, we conduct an experiment where we train RT-1 only with the "pick coke can" demonstrations and evaluate its real and simulation performance. We also compare the MMRV and the Pearson correlation before and after incorporating this single-task policy into the Google Robot experiments. Results are shown in Table XIII. We find that our simulated evaluation effectively reflects the performance rankings of the newly-added single-task policy, with the MMRV remaining low and the Pearson Coefficient remaining high. This demonstrates SIMPLER evaluation's versatility across policies trained on diverse data scales.

B. Other Metrics: Kruskal Wallis

In our previous analysis, we primarily focused on metrics that measure real-to-sim **relative performance** alignment between policies. As we match real-to-sim visual input appearance in our Visual Matching evaluation approach, it also becomes meaningful to measure the simulation distribution shift of **absolute performance** from real-world evaluations, even though we do not expect the real-to-sim absolute performances to exactly match. Let the real-world evaluation results of N policies be $\mathbf{r} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$, where $\mathbf{r}_i = (r_{ij})_{j=1}^{N_{\text{trial}}}$ is the indicator of each trial's success in the real-world. Let the corresponding simulation evaluation results be $\mathbf{s} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N\}$, where $\mathbf{s}_i = (s_{ij})_{j=1}^{N_{\text{trial}}}$. We perform *Kruskal-Wallis test* for each individual policy (i.e., between each $\mathbf{r}_i$ and $\mathbf{s}_i$) to measure whether simulation evaluations exhibit significant distribution shift from real evaluations. We then report the number of policies with significant distribution shift (which we denote as "Kruskal-#Policy $p<0.05$").

We present the Kruskal-Wallis results in Tab. XIV. We find that with the Visual Matching evaluation approach, the simulation trial success distribution is not significantly different from the real results ($p \geq 0.05$) across many tasks and policies, demonstrating the effectiveness of our simulation evaluation tool. We also note that our MMRV and the Kruskal metrics complement each other's limitations, with the former providing a real-to-sim relative performance alignment perspective, and the latter providing an absolute performance alignment perspective.

Components			Open Drawer				Close Drawer			
Background	**Drawer**	**Robot**	RT-1 (Converged)	RT-1 (15%)	RT-1-X	MMRV↓	RT-1 (Converged)	RT-1 (15%)	RT-1-X	MMRV↓
Real	Real	Real	0.815	0.704	0.519	N/A	0.926	0.889	0.741	N/A
GreenScreen	Curated	Curated	**0.703**	**0.556**	**0.333**	**0.000**	**0.889**	0.667	0.851	**0.099**
GreenScreen	Curated	Original	0.444	0.444	0.259	0.111	0.741	0.630	0.926	0.173
GreenScreen	Original	Original	0.593	0.519	0.148	**0.000**	0.852	**0.778**	0.963	0.173
ReplicaCAD	Curated	Curated	0.407	0.259	0.111	**0.000**	0.667	0.481	0.778	0.173
ReplicaCAD	Curated	Original	0.630	0.407	0.074	**0.000**	0.630	0.593	0.667	0.173
ReplicaCAD	Original	Curated	0.556	0.296	0.074	**0.000**	0.667	0.704	0.815	0.173
ReplicaCAD	Original	Original	0.556	0.333	0.074	**0.000**	0.704	0.556	**0.741**	0.173

TABLE IX: Impact of real-to-sim visual gaps on real-and-sim performance correlations. We report the success rates of 3 different policies on 2 tasks: *Open Drawer* and *Close Drawer*. The settings with the smallest MMRV and the smallest absolute performance gap with the real performance are highlighted. Using a combination of "green-screened" background and curated foreground object and robot assets provides the best correlation.

	Gripper Friction Coefficient			
Coke Can Mass	0.25	0.50	1.0	2.0
10 g	0.957	0.967	0.971	0.978
20 g	0.969	0.975	0.978	0.977
40 g	0.963	0.976	0.976	0.976
80 g	0.962	0.962	0.975	0.990

(a) Pearson r between real and SIMPLER evaluations on the Pick Coke Can task under different settings of can mass and gripper friction coefficient. The MMRV is 0.031 in all cases. The use of empty coke cans follows the setup from the Google Robot demonstration dataset and the RT-1 paper [6].

Cabinet Joint Friction	0.0125	0.025	0.05	0.10	0.15	0.20
MMRV↓	0.055	0.055	0.055	0.055	0.105	0.055
Pearson r↑	0.930	0.941	0.915	0.923	0.903	0.928

(b) MMRV and Pearson r between real and SIMPLER evaluations on the Open/Close Drawer tasks under different settings of cabinet joint friction.

TABLE X: SIMPLER is robust to imprecisely estimated physical simulation parameters such as object mass and friction coefficients, as indicated by the low MMRV and high Pearson r in both ablation studies. We use the 6 policies from our Google Robot experiments in these ablations.

Google Robot Evaluation Setup	Policy	Pick Coke Can				Move Near
		Horizontal Laying	Vertical Laying	Standing	Avg. Success	Avg. Success
Real Eval	RT-1 (Converged)	0.960	0.880	0.720	0.853	0.633
	RT-1 (15%)	1.000	0.960	0.800	0.920	0.583
	RT-1-X	0.880	0.560	0.840	0.760	0.450
	Octo-Base	0.440	0.200	0.240	0.293	0.350
	RT-1 (Begin)	0.200	0.000	0.200	0.133	0.017
SIMPLER Eval (Isaac, Variant Aggre.)	RT-1 (Converged)	0.418	0.377	0.436	0.410	0.150
	RT-1 (15%)	0.428	0.306	0.590	0.441	0.100
	RT-1-X	0.340	0.182	0.618	0.380	0.125
	Octo-Base	0.015	0.020	0.010	0.015	0.020
	RT-1 (Begin)	0.036	0.040	0.054	0.044	0.000
	MMRV↓	0.096	0.112	0.016	0.064	0.053
	Pearson r↑	0.961	0.949	0.989	0.973	0.865

TABLE XI: Real-world and Isaac Sim evaluation results for the Google Robot setup. The findings on Isaac Sim are consistent with the findings on the SAPIEN simulator.

Task	**Validation Action MSE**	**Sim Eval (Visual Matching)**
Pick Coke Can	0.412 / 0.464	0.031 / 0.976
Move Near	0.408 / 0.230	0.111 / 0.855
Open / Close Drawer	0.346 / 0.264	0.055 / 0.915
Open Drawer & Place Apple	0.265 / 0.198	0.000 / 0.969
Put Spoon on Towel	0.389 / -0.951	0.000 / 0.827
Put Carrot on Plate	0.194 / -0.342	0.111 / 0.575
Stack Block	0.125 / -0.857	0.000 / 1.000
Put Eggplant in Basket	0.366 / -1.000	0.000 / 0.990

TABLE XII: MMRV / Pearson correlation comparison between our Visual Matching simulation evaluation approach and the simulation-free approach that assesses the MSE between predicted and ground-truth actions on validation trajectories. For the latter approach, we calculate the MMRV / Pearson correlation between the negative MSE and the real policy performance. Our approach yields significantly better real-and-sim policy performance correlations.

Policy	**Avg. Real Success**	**Avg. Sim Success (Visual Matching)**
RT-1 (Converged)	0.853	0.857
RT-1 (15%)	0.920	0.710
RT-1 (Single Task Policy)	0.680	0.403
RT-1-X	0.760	0.567
RT-2-X	0.907	0.787
Octo-Base	0.293	0.170
RT-1 (Begin)	0.133	0.027
MMRV↓		0.027
Pearson r↑		0.959

TABLE XIII: Real-world and simulated evaluation results on the Pick Coke Can task, after adding an RT-1 policy trained solely on the Pick Coke Can demonstrations. Our simulated evaluation remains effective, exhibiting low MMRV and high Pearson correlation coefficient with real evaluations.

Google Robot Evaluation Setup	Metric	Pick Coke Can				Move Near	Open / Close Drawer			Open Top Drawer and Place Apple
		Horizontal Laying	Vertical Laying	Standing	Avg. Success	Avg. Success	Open	Close	Avg. Success	Avg. Success
SIMPLER - Visual Matching	Kruskal-#Policy $p<0.05$	0	0	2	3	3	1	2	2	0

(a)

WidowX+Bridge Evaluation Setup	Metric	Put Spoon on Towel		Put Carrot on Plate		Stack Green Block on Yellow Block		Put Eggplant in Yellow Basket	
		Grasp Spoon	Success	Grasp Carrot	Success	Grasp Green Block	Success	Grasp Eggplant	Success
SIMPLER - Visual Matching	Kruskal-#Policy $p<0.05$	0	0	0	0	0	0	1	0

(b)

TABLE XIV: For our Visual Matching evaluation approach, we conduct Kruskal-Wallis test to assess whether simulation and real-world policy evaluations exhibit significant distribution shift, even though we do not expect to obtain an exact reproduction of real-world performance.