Scaling Robot-Learning by Crowdsourcing Simulation Environments

Marcel Torne[1] **Arhan Jain**[*2] **Vidyaaranya Macha**[*2], **Jiayi Yuan**[*2] **Lars Lien Ankile**[1]
Anthony Simeonov[1] **Pulkit Agrawal**[1] **Abhishek Gupta**[2]
[1]Massachusets Institute of Technology [2]University of Washington
* equal contribution

Abstract: Scaling robot learning requires data collection pipelines that scale favorably with human effort to ensure a sufficient diversity and quality of expert data. In this work, we propose *Scaling, Crowdsourcing and Amortizing Real-to-Sim-to-Real* - **SCAR**, a pipeline for scaling up data collection and learning generalist policies where human effort scales sublinearly with the number of environments where data is collected. The key idea is to crowdsource digital twins of real-world scenes using 3D reconstruction techniques and collect large-scale data in these simulation scenes, rather than in the real-world. Data collection in simulation is initially driven by reinforcement learning bootstrapped with human demonstrations. However, as the training of a generalist policy progresses across environments, the generalization capabilities of the learned generalist policy can be used to replace human effort with model generated demonstrations. This results in a pipeline where environments are easily sourced from non-experts through 3D capture, while behavioral data is collected with continually reducing amounts of human effort. We analyze the zero-shot and few-shot scaling laws of **SCAR** on two real-world tasks: *placing mugs/bowls/cups into a sink* and *placing boxes on a shelf* across a diverse range of environments. We also demonstrate the capabilities of the **SCAR** pipeline to finetune trained policies in a target scenario using a novel unsupervised fine-tuning technique that can improve behavior simply using 3D environments scans at test time, without requiring additional human demonstrations.

Keywords: Real-to-Sim-to-Real, Continual Data Collection, Scaling-up

1 Introduction

Robot learning has the potential to revolutionize decision-making for robots by leveraging data to learn behaviors deployable in unstructured environments, showing generalization and robustness. Critical to the success of robot learning, beyond the algorithms and model architectures, is the data used for training. As in most fields of machine learning, getting the "right" type, quality and quantity is key to generalization. Robot learning is still grappling with the question of what the right type of data is and how to obtain it at scale. The type of data we can train on is inherently tied to the abundance of this data - good data is both high-quality and abundant. This paper proposes a system for obtaining this type of diverse, high-quality data at scale with sublinear human effort.

Unlike vision and language, data for learning is not available passively - there are relatively few robots that are already finding use in the world. This makes applying the same recipes we did in vision and language challenging, necessitating more careful consideration of how and where this data comes from. One option is to rely on teleoperation to collect this data. This approach is inherently limited by human effort, since the cost to collect data scales linearly with human involvement. Recent work [1, 2, 3] has attempted to scale the amount of teleoperation data however the quantity of data collected is still orders of magnitude smaller than the scale at which vision and language models show emergent capabilities.

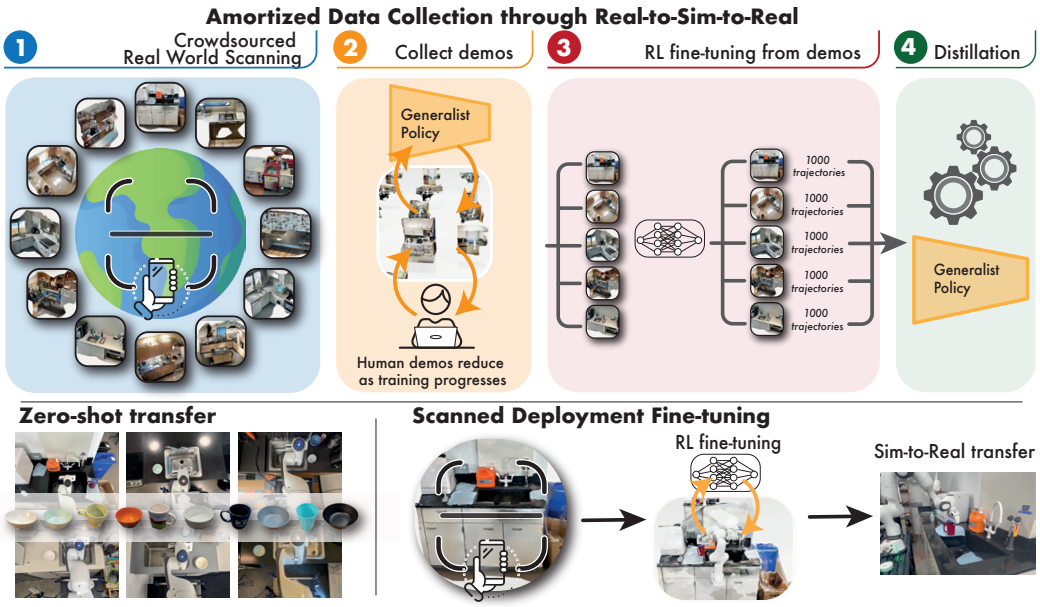

Figure 1: Overview of **SCAR**, we propose a system for training generalist policies leveraging real-to-sim simulation on crowdsourced scans. These have zero-shot transfer and scanned fine-tuning capabilities.

So where might we find data that scales sublinearly with human effort? Simulation offers a potential solution, at face value providing free data up to the limit of computing. However, this hides a significant cost - scene, task, and reward creation per domain is non-trivial, and even with scenes generated, behaviors are costly to obtain. This suggests that despite the promise, simulation data isn't quite free of cost, and requires considerable amounts of human efforts for content and behavior creation per environment. While it is possible to generate random environments procedurally, generating thousands of environments randomly is unlikely to cover the distribution of "natural environments", and generating behaviors randomly is unlikely to lead to success.

In this work, we propose a method to scale up continual data collection, ensuring human effort amortizes sublinearly with the number of environments. Our key idea is to leverage simulation for data scaling without the corresponding increase in content and behavior creation effort. For content scaling, we utilize 3D reconstruction methods, shifting the burden from designers to non-expert users. For behavior generation, we employ techniques that leverage model generalization to reduce the required human data over time. The insight is that as we go across many simulated environments, models will show some levels of generalization. This generalization can be leveraged to continually reduce the amount of human data needed as new environments are encountered. **SCAR** creates a data flywheel, where data begets more data through model generalization.

Our contributions include 1) a novel continual data collection system based on real-to-sim-to-real for training generalist policies, 2) a novel scanned deployment fine-tuning technique for improving the accuracy of a generalist policy on a target environment without additional human demonstrations, 3) a detailed analysis of the scaling laws for zero-shot performance of our generalist policies, 4) evaluation of the few-shot performance of the resulting generalist policies.

2 Related Work

Large Scale Data Collection for Robotics: Learning from real-world demonstrations has proven effective [4, 5, 6]. To facilitate this, various studies have focused on improving hardware to ease the data collection process for teleoperators [5, 7, 8]. Efforts have also scaled up the volume of data from real-world demonstrations [2, 1, 3], staying nevertheless in the low-data regime. Moreover, real-world data collection is costly, requiring expert supervision and physical robots, which limits scalability. **SCAR**, instead, trains entirely in simulation, using real-world scans obtained via standard smartphones. Additionally, while traditional teleoperation data collection scales linearly with

human effort, **SCAR** reduces the human effort needed for subsequent learning steps by leveraging the knowledge acquired during training.

Autonomous Learning: To improve scalability of robot learning and reduce the amount of human demonstrations required, the field has explored autonomous data collection and learning methods. One approach is reinforcement learning (RL) in the real world [9, 10], but the standard RL techniques' need for resets poses scalability issues, as it requires either human supervision or substantial engineering efforts for automating resets. Reset-free reinforcement learning [11, 12, 13, 14] offers a promising alternative, but it still requires occasional human intervention and struggles with high sample complexity for learning more challenging tasks, making it hard to learn in the real-world. Autonomous learning in the real world presents significant challenges that are mitigated in simulation, where resets are manageable and data collection is more abundant. In **SCAR**, we exploit these advantages of simulation while minimizing the sim-to-real gap through real-to-sim scene transfers. Continual learning also faces challenges, such as catastrophic forgetting, as discussed in prior work [15]. We address this by decoupling the policy used to generate trajectories, which is fine-tuned with RL, from the final generalist policy, which is trained with imitation learning over the entire dataset.

Procedural and Synthetic Data Generation: Creating realistic environments for robot learning in simulation is a significant challenge. To address this, prior work has proposed using large language models (LLMs) or heuristics to generate scene plans resembling the real world [16, 17, 18], or utilizing real-world scans to replicate actual scenes [19, 20]. Despite reducing human involvement, these methods often produce scenes that are unrealistic in appearance or object distribution, such as failing to accurately simulate real-world clutter. Generating procedurally accurate training environments remains an open challenge. However, extracting digital twins from the real world mitigates this issue, as scans reflect the actual test distribution. Relevant to our work, [21] automates the creation of simulatable environments from real-world scans, which could be integrated into our pipeline to scale up environment crowdsourcing. Once the environments are available, generating valid robot trajectories that solve the task is another challenge. An option becomes procedurally generating the motions using motion planning techniques [22]. However, these techniques require some assumptions beforehand.

Real-to-Sim-to-Real Transfer for Robotics: Real-to-sim-to-real techniques have proven effective in learning robust policies for specific scenarios with minimal human supervision [23, 24]. However, these policies often fail to generalize to different scenarios, requiring significant human effort for each new environment. In this work, we address this limitation by learning generalist policies through a novel technique that amortizes the number of human demonstrations through training. Other research has tackled various challenges in real-to-sim-to-real, such as enhancing simulator accuracy with real-world interaction data [25, 26, 27], and automatically generating articulations from images [20, 28, 29]. These complementary advancements make simulators more realistic and could reduce human effort further in **SCAR**. Additionally, real-to-sim techniques have shown promise in their use for simulated evaluation of real-world policies [30].

3 Amortized Data Scaling for Learning Generalist Policies through Real-to-Sim-to-Real

This work presents **SCAR**, a pipeline for large-scale continual data collection for robotic manipulation. The primary challenge for data scaling in the realm of robotics is the absence of "passive", easy-to-collect data from naturally occurring, inadvertent sources, as is common in vision and language. While procedural generation in simulation can provide large amounts of data, the distribution and diversity of the data does not overlap with real-world environments. In this work, we argue that a multi-task, multi-environment real-to-sim-to-real pipeline can enable large-scale data generation, by leveraging model generalization to scale human-effort sublinearly as increasing numbers of environments are encountered. This is opposed to typical human teleoperated data collection that requires considerable expertise, physical infrastructure and suffers from linear scaling in human effort. This approach enables the scaling laws necessary for large scale data collection and training of robotic

foundation models, showing non-trivial zero-shot generalization performance as well as cheap and efficient fine-tuning in new environments. **SCAR** consists of three elements - 1) fast, accessible digital twin generation with 3-D reconstruction methods, 2) multi-environment model learning that amortizes the data collection process through autonomous data collection and model generalization, 3) efficient fine-tuning in new environments using 3-D scans, and minimal human demonstrations.

3.1 Real-to-Sim Scene Synthesis

Our proposed data collection pipeline adopts a real-to-sim-to-real approach, building digital twins of real-world scenes in simulation and collecting behavioral data in these simulations instead of the real world. This method offers several advantages - 1) data collection does not require a physical robot setup, and hence can occur in a broader variety of realistic environments 2) it allows for safe, decentralized, and asynchronous data collection 3) digital twins capture the complexities of real-world scenarios more accurately than procedurally generated simulations. These advantages are crucial to the democratization and scalability of data collection as it is scaled up to thousands of non-experts and real environments beyond the lab. We leverage easily accessible mobile software[31, 32] for scene reconstruction from sequences of images to easily crowdsource simulated environments [1]. These environments indicate the geometry, visuals and physics of diverse real-world scenes in simulation but do not have any demonstrations of the desired optimal behavior. We discuss how this can be obtained efficiently in the following section.

3.2 Amortized Data Collection

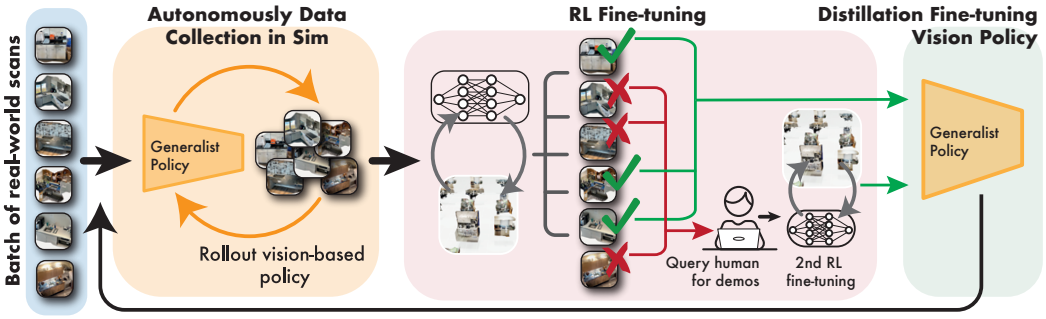

Figure 2: Overview of the proposed continual data collection system for amortizing human data collection.

Given the diversity of realistic simulation scenes available through the digital twin pipeline outlined in Section 3.1, learning generalizable decision-making policies requires a large training set of visuomotor trajectories demonstrating optimal behavior for each distinct environment. Two natural alternatives for obtaining these trajectories are: 1) human-provided demonstrations and 2) optimal policies trained via reinforcement learning [2]. While tabula-rasa reinforcement learning can provide a robust set of trajectories with extensive state coverage without expensive human intervention, it faces considerable challenges related to exploration and reward design. On the other hand, human demonstrations avoid these issues but are expensive to collect at scale.

A natural solution is to use sparse-reward reinforcement learning bootstrapped with human demonstrations [23, 33, 34][3]. This approach balances human effort for data collection and reward specification with state-space coverage. However, scaling it up to hundreds or thousands of scenes becomes tedious, as the required human effort increases linearly with the number of environments. In this work, we learn a generalist multi-environment policy to *amortize* the cost of human data collection across environments. We demonstrate that the capacity of such a multi-environment model to display non-trivial generalization allows the cost of continual human data collection to decrease as the number of training environments increases.

[1] We provide further details about the real-to-sim pipeline in Appendix 6.2, including how to stage these scenes, articulate them quickly and so on.

[2] Other techniques such as trajectory optimization or motion planning may be applicable as well

[3] We refer readers to Appendix 6.3.1 for details of demonstration bootstrapped reinforcement learning

This system, formally stated in the Appendix Algorithm 1, divides the total number of environments into batches of size K. For the first batch of K environments $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K$, we have a multi-environment visuomotor policy π_G randomnly initialized with no generalization capabilities. Thereafter, we initialize it with data from the first K environments, using reinforcement learning bootstrapped with human-provided demonstrations. Demonstration bootstrapped RL produces optimal visuomotor trajectories per environment $\mathcal{D}$, that are then distilled into a single perception-based, generalist multi-environment policy π_G with visuomotor policy distillation [35] (Appendix 6.3.2).

While human demonstrations are used to bootstrap the data generation and training of the first iteration of the generalist policy π_G on the first K environments, our key insight is that if π_G shows non-trivial level of generalization on visuomotor deployment in the next K simulation environments - $\mathcal{E}_{K+1}, \ldots, \mathcal{E}_{2K}$, then this policy π_G can be used to collect simulated demonstrations $\mathcal{T} = \tau_{K+1,1}, \tau_{K+1,2}, \ldots, \tau_{2K,N}$ in place of a human demonstrator. We do so by deploying the visuomotor policy $\pi_G(a_t|o_t)$ using perceptual observations o_t such as RGB point clouds, but since we are in the simulation we collect $\mathcal{T}$ with paired data of visual observations o_t, actions a_t and low-dimensional privileged Lagrangian state s_t. These privileged state-based trajectories enable the usage of efficient demonstration-bootstrapped reinforcement learning of a state-based policy π_s rather than operating from high-dimensional perceptual observations. See Eq 1 and Appendix 6.3.1 for the state-based policy update using PPO [36] with a BC loss, where $\hat{A}_t$ is the estimator of the advantage function at step t [36], and V_ϕ is the learned value function.

$$\pi_s \leftarrow \max_{\theta,\phi} \alpha \sum_{\mathcal{E}_i \in \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K\}} \sum_{(s_t, a_t, r_t) \in \mathcal{E}_i(\pi_{\theta_{\text{old}}})} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t, \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)$$
$$+\beta \sum_{\mathcal{E}_i \in \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K\}} \sum_{(s_t, V_t^{\text{targ}}) \in \mathcal{E}_i(\pi_{\theta_{\text{old}}})} (V_\phi(s_t) - V_t^{\text{targ}})^2 + \gamma \sum_{(s_i, a_i) \in \mathcal{T}} \log \pi_\theta(a_i|s_i) \tag{1}$$

$\mathcal{T}$ can be used to obtain a single robust, state-covering optimal multi-environment policy $\pi_{s1}(a_t|s_t)$ for all $\mathcal{E}_{K+1}, \ldots, \mathcal{E}_{2K}$ via demonstration-bootstrapped reinforcement learning. Nevertheless, in some environments, the policy may still perform poorly due to the occasional low-quality demonstrations from π_G. To address this, we define the set of environments where π_{s1} achieves below r success rate as $\mathcal{F} \subset \{\mathcal{E}_K, \mathcal{E}_{K+1}, \ldots, \mathcal{E}_{2K}\}$. For these environments $\mathcal{F}$, we fall back to querying the human demonstrator for high-quality demonstrations and learn a second state-based policy $\pi_{s2}(a_t|s_t)$ using demonstration-bootstrapped reinforcement learning on $\mathcal{F}$.

The two learned policies π_{s1} and π_{s2} can then be used for generating data on $\{\mathcal{E}_K, \mathcal{E}_{K+1}, \ldots, \mathcal{E}_{2K}\} \backslash \mathcal{F}$ and $\mathcal{F}$ respectively with these new trajectories being added into $\mathcal{D}$. Then, a visuomotor policy can be trained by fitting $\mathcal{D}$ on the first $2K$ environments with supervised learning (see Appendix 6.3.2 for implementation details).

$$\pi_G \leftarrow \max_\theta \mathbb{E}_{(o_i, a_i) \sim \mathcal{D}} \left[\log \pi_{G\theta}(a_i|o_i)\right] \tag{2}$$

Then the process repeats for the next K environments. As the visuomotor generalist policy π_G is trained across more environments, it demonstrates increasingly non-trivial generalization, gradually replacing the human demonstrator in more environments. This reduces the amount of human effort required for data collection as training progresses. Importantly, the generalization across environments does not need to achieve perfect success rates but should be sufficient to bootstrap a demonstration-augmented policy learning algorithm (Equation 1). This suggests an interesting scaling law - data collection becomes more human-efficient as training progresses, eventually becoming self-sustaining. For a detailed outline of the practical data collection pipeline, refer to Algorithm 1.

3.3 Fine-tuning of Generalist Policies on Deployment

The generalist policies $\pi_G(a_t|o_t)$ pretrained in Section 3.2, show non-trivial generalization across environments but may not achieve optimal performance in any one environment upon zero-shot deployment. However, these generalist policies can serve as a starting point for efficient fine-tuning at test time. In this section, we present an alternative for fine-tuning generalist policies $\pi_G(a_t|o_t)$ during deployment. We make the observation that we can follow the same procedure as model-bootstrapped autonomous data collection during training described in Section 3.2. Given a scanned digital twin $\mathcal{E}_{\text{test}}$ of the testing environment in simulation, the pre-trained multi-environment model $\pi_G(a_t|o_t)$ shows some non-trivial zero-shot generalization, but may not achieve optimal performance in $\mathcal{E}_{\text{test}}$. By executing the visuomotor policy $\pi_G(a_t|o_t)$ in $\mathcal{E}_{\text{test}}$, we collect a dataset of only successful trajectories $\mathcal{T}_{\text{test}}$ consisting of (o_t, a_t, s_t) tuples in simulation, without the need for any external human intervention. This model-generated data can then be used to train a robust, high-coverage state-based policy $\pi_s(a_t|s_t)$ using demonstration-bootstrapped reinforcement learning (Eq 1). Finally, for real-world transfer from visual observations this state-based policy $\pi_s(a_t|s_t)$ is distilled into a "fine-tuned" visuomotor policy $\pi_{Gf}(a_t|o_t)$, by collecting a set of successful rollouts $\mathcal{D}$ with $\pi_s(a_t|s_t)$ and fine-tuning the previously obtained generalist policy $\pi_G(a_t|o_t)$ as in Eq 2. This approach allows the model to retain the generalist capabilities of π_G while achieving high success in $\mathcal{E}_{\text{test}}$. Importantly, this fine-tuning step is accomplished using only a video scan of the environment, without the need for human-provided demonstrations or feedback in the physical environment. (See Algorithm 2, in Appendix 6.4.1). Finally, in the Appendix 6.4.2, we propose a second technique involving few-shot supervised fine-tuning using a limited set of human-provided demonstrations.

4 Experimental Evaluation

Our experiments are designed to answer the following questions: (a) What are the scaling laws of **SCAR**? (b) How much can we amortize the quantity of human data needed through learning without a loss in performance? (c) What are the few-shot/scanned fine-tuning capabilities of the learned generalist policies? (d) Do these scaling laws hold across different tasks? (e) Do these generalist policies extrapolate to multi-object environments when trained with single object?

To answer these questions, we design two different tasks: *placing bowls/mugs/cups in sinks* and *placing boxes in shelves*. We use a single-arm manipulator, the Franka Research 3 arm with 7 DoF and a parallel jaw gripper, see Appendix 10. We crowdsourced environment data collection, obtaining (a maximum of) 56 and 36 different scenes for the two tasks, respectively. We evaluated the policies across two institutions on 8 and 2 real-world scenes not included in the training set. Further details on the hardware setup and tasks are provided in Appendix 7 and 10.

4.1 Zero-Shot Scaling Laws Analysis

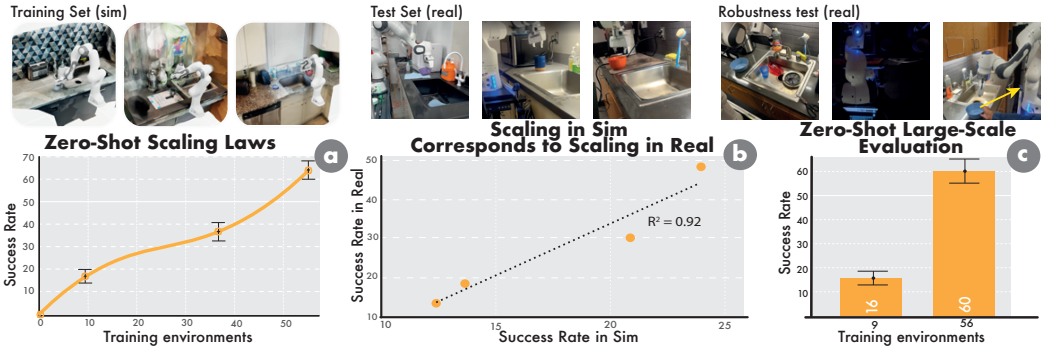

Figure 3: *a*) **SCAR**'s zero-shot scaling laws on the task of *pick and placing bowl/cup/mugs to sinks*; *b*) in the proposed real-to-sim-to-real setup there is a linear relation between performance in sim and performance in real; *c*) evaluation on a broader set of environments confirms the robustness of the zero-shot policies.

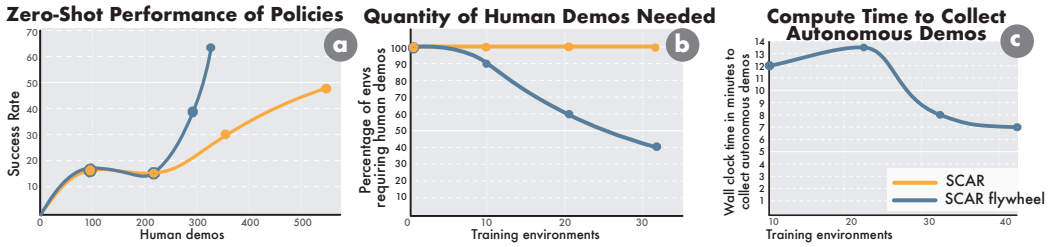

Figure 4: *a*) **SCAR** with continual data collection becomes more efficient in number of human demos and achieves higher performance than running **SCAR** uniquely from human demos. *b*) with continual data collection the number of human demos required decreases throughout training. *c*) even though **SCAR** relies on compute we observe the amount of compute needed also tends to decrease when scaling up this process.

In this section, we analyze the zero-shot performance of multiple generalist policies trained with varying amounts of training environments on the task of *put a mug/bowl/cup in a sink*. For fair comparison, we train these policies using human demonstrations in each environment. In Section 4.2, we compare this baseline to the autonomous data collection system presented in Section 3.2.

The first experiment involves a thorough real-world evaluation of these policies across two institutions, using three different kitchens and six different objects, with six rollouts each (a total of 108 rollouts per policy). As shown in Figure 3 *a*, we confirm the real-to-sim-to-real pipeline scaling law: as the number of trained environments increases, the zero-shot success rate also increases, reaching a 62% when trained on 56 environments. Furthermore, Figure 3 b shows a linear correlation between simulation and real world performance, indicating that our real-to-sim-to-real scaling approach in simulation proportionally corresponds to improved performance in the real world.

To verify the robustness of the learned policies, we ran evaluation on eight additional kitchens. The results highlight an improvement of 16% to 60% rate as the number of training environments increased from 9 to 56 (Figure 3 c). Figure 7 shows a sample of the objects and environments used for evaluation. Finally, we stress-tested against other types of robustness (Figure 3), including extreme lighting changes, clutter and physical disturbances, and observed that the policies suffer a drop in performance but keep obtaining success rates above 30% (see Appendix 9.2). On the same lines we evaluate the policy on multiple objects in the scene and observe that even though it was only trained to pick up one object, it still succeeds 10% of the times to clean a scene with 3 objects (See Figure 6 and Appendix 9.1).

4.2 Amortized Human Data Needed Through Continual Data Collection

In this section, we evaluate the amortization of number of human demonstrations needed as learning progresses across multiple environments. We compare two approaches: our proposed system using continual data collection performed in four sequential batches of 10 environments each, and another baseline providing human demonstrations for each environment individually. The evaluation is conducted in a single real-world kitchen with six different objects for the task of *put a bowl/mug/cup in a sink*, performing 6 rollouts per object. Figure 4(a) shows that the performance per number of demonstrations significantly increases as the policy starts developing generalization. Specifically, as shown in Figure 4(b), the quantity of human demonstrations needed decreases as the policy improves with each subsequent batch. Although **SCAR** shifts the burden to compute rather than human effort, Figure 4(c) indicates that the compute required decreases as well when scaling up the system, since the success rate of the generalist policy is higher, the number of trials performed to reach the same number of successful rollout decreases. Finally, we observe that the performance of the continually learned policy is higher than of the policy learned solely from human demonstrations. We hypothesize that this is due to the multimodality in behaviors from the human demonstrations. When the policy autonomously collects the data, behaviors remain closer to those already learned, whereas human-provided demonstrations may introduce more variability, making learning harder.

4.3 Fine-Tuning of Generalist Policies

Unsupervised scanned deployment fine-tuning: To evaluate the efficacy of unsupervised fine-tuning through a scan (Section 3.3), we select two scenes for the task of *placing a mug/cup/bowl in*

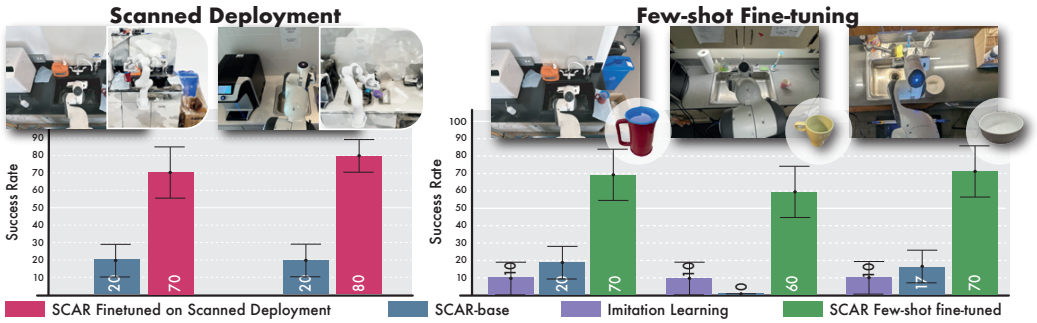

Figure 5: Fine-tuning results. *left*: **SCAR** successfully improves its performance fine-tuning autonomously on a scanned deployment environment. *right*: Few-shot fine-tuning on 10 demos we can significantly improve the performance of the generalist policy on the target scene.

a sink where the policy trained on 36 environments performs poorly ($\leq$ 20%). We then apply the scanned deployment fine-tuning algorithm as described in Section 3.3. As shown in Figure 5, this results in an average performance increase of 55% without any additional human demonstrations.

Few-shot supervised fine-tuning: We select three environments where the base policy trained on 36 environments performs poorly ($\leq$20%). We then collect 10 demonstrations for each environment and apply the few-shot fine-tuning procedure described in Section 6.4.2. This fine-tuning improves the performance of the base policy by an average of a 54% in success rate.

4.4 Further Scaling Laws Analysis

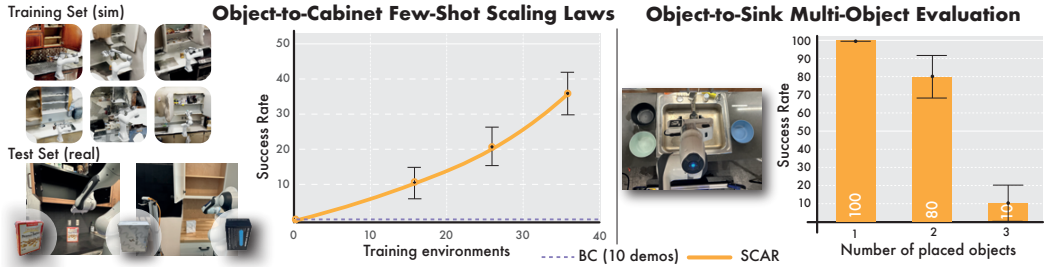

Figure 6: *left*: scaling laws for few-shot fine-tuning on the task of *pick and place a box on a shelf right*: multi-object evaluation results on the task of *pick and place mug/bowl/cups in the sink*

We attempt to solve another task, *putting a box on a cabinet*. Given the difficulty of this task, we focus our analysis on scaling laws after few-shot fine-tuning as described in Section 3.3. We crowdsourced 36 environments, collected 10 demonstrations for each of three scenes and reported the performance after fine-tuning with 10 demos. In Figure 6, we show the performance the performance increases with the number of training environments, without reaching a saturation point. Fine-tuning the policy trained on 36 environments resulted in a significant performance improvement of 36% compared to the imitation learning baseline, which had a 0% success rate.

5 Conclusion

Limitations: While with this work we demonstrate sublinear scaling of human demonstrations, the burden shifts to compute. And even though we have shown a reduction in compute time with scaling, it still exceeds the time required for collecting real-world demonstrations. Additionally, training in simulation poses challenges, as not all real-world objects can be accurately simulated yet, such as liquids and deformable objects. However, contrary to the human teleoperation efforts, with advancements in compute resources and simulator research, systems like **SCAR** will benefit from these and further improve scalability. *Conclusion*: This work presents **SCAR**, a real-to-sim-to-real system that trains generalist policies with sublinear human effort. This research paves the way for building robotic foundation models in simulation with larger datasets and enhanced robustness.

Acknowledgments

The authors would like to thank the Improbable AI Lab and the WEIRD Lab members for their valuable feedback and support in developing this project. Furthermore, we would like to thank all of the people who contributed by providing scans of scenes for training our policies. This work was partly supported by the Sony Research Award, the US Government, and Hyundai Motor Company.

References

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

[2] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

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

[4] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

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

[6] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[7] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[8] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023.

[9] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine. Serl: A software suite for sample-efficient robotic reinforcement learning, 2024.

[10] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[11] M. Balsells, M. Torne, Z. Wang, S. Desai, P. Agrawal, and A. Gupta. Autonomous robotic reinforcement learning with asynchronous human feedback. *arXiv preprint arXiv:2310.20608*, 2023.

[12] J. Yang, M. S. Mark, B. Vu, A. Sharma, J. Bohg, and C. Finn. Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning. *arXiv preprint arXiv:2310.15145*, 2023.

[13] A. Sharma, A. M. Ahmed, R. Ahmad, and C. Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. *arXiv preprint arXiv:2303.01488*, 2023.

[14] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671. IEEE, 2021.

[15] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58:52–68, 2020.

[16] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang. Gensim: Generating robotic simulation tasks via large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[17] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi. procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022.

[18] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[19] M. Deitke, R. Hendrix, A. Farhadi, K. Ehsani, and A. Kembhavi. Phone2proc: Bringing robust robots into our chaotic world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9665–9675, 2023.

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

[21] H. Xia, Z.-H. Lin, W.-C. Ma, and S. Wang. Video2game: Real-time, interactive, realistic and browser-compatible environment from a single video. *arXiv preprint arXiv:2404.09833*, 2024.

[22] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.

[23] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*, 2024.

[24] L. Wang, R. Guo, Q. Vuong, Y. Qin, H. Su, and H. Christensen. A real2sim2real method for robust object grasping with neural surface reconstruction. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–8. IEEE, 2023.

[25] M. Memmel, A. Wagenmaker, C. Zhu, P. Yin, D. Fox, and A. Gupta. Asid: Active exploration for system identification in robotic manipulation. *arXiv preprint arXiv:2404.12308*, 2024.

[26] F. Ramos, R. C. Possas, and D. Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019.

[27] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. 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.

[28] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022.

[29] N. Nie, S. Y. Gadre, K. Ehsani, and S. Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *arxiv*, 2022.

[30] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

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

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

[33] H. Hu, S. Mirchandani, and D. Sadigh. Imitation bootstrapped reinforcement learning, 2023.

[34] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[35] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[37] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[38] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020. URL https://arxiv.org/abs/2003.08934.

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

[40] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, et al. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 2023.

[41] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[42] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020.

In the Appendix, we are going to cover the following details of our work.

- **Task Specifications** Appendix 7: Specifications of tasks used for evaluating **SCAR**.
- **Implementation Specification** Appendix 8: Specification of hyper-parameters used in the network architectures, point cloud processing, and dataset used in **SCAR**.
- **Evaluation Details** Appendix 9: Details of evaluation, including robustness experiments, adding disturbance and distractors.
- **Hardware Setup** Appendix 10: Specification for hardware setup used for training and evaluating **SCAR**.
- Appendix **Crowdsourcing** 11: Specification for crowdsourcing real-world 3D scans.
- **Compute Resources** Appendix 12: Specficiations for compute resources used for data collection, training and evaluating **SCAR**.

6 Method Details

6.1 Amortized Data Collection

Algorithm 1 SCAR: Amortized Data Collection for Generalist Policies

Input: Human demonstrator $\mathcal{H}$, crowdsource humans $\mathcal{C}$
Initialize vision-based generalist policy π_G
while True **do**
  Sample set of K digital twins from crowdsourced humans $\{\mathcal{E}_K, \mathcal{E}_{K+1}, \ldots, \mathcal{E}_{2K}\} \sim \mathcal{C}$
  $\mathcal{T} \leftarrow \{\}$
  for $\mathcal{E}_i$ in $\mathcal{E}_K, \mathcal{E}_{K+1}, \ldots, \mathcal{E}_{2K}$ **do**
    $\mathcal{T}_e \leftarrow \text{RolloutPolicy}(\mathcal{E}_i, \pi_G)$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \text{FilterSuccessfulRollouts}(\mathcal{T}_e)$
  $\pi_s \leftarrow \text{RLFinetuning}(\mathcal{T}, \{\mathcal{E}_K, \mathcal{E}_{K+1}, \ldots, \mathcal{E}_{2K}\})$
  $\mathcal{T}_h \leftarrow \{\}$
  $\mathcal{F} \leftarrow \text{FailedEnvironments}(\{\mathcal{E}_K, \mathcal{E}_{K+1}, \ldots, \mathcal{E}_{2K}\}, \pi_s)$
  for $\mathcal{E}_i$ in $\mathcal{F}$ **do**
    $\mathcal{T}_h \leftarrow \mathcal{T}_h \cup \text{CollectDemos}(\mathcal{E}_i, \mathcal{H})$
  $\pi_h \leftarrow \text{PPORLFinetuning}(\mathcal{F}, \pi_h)$
  $\pi_G \leftarrow \text{TeacherStudentDistillation}(\mathcal{E}, \pi_G, \pi_s, \pi_h)$

6.2 Real-to-Sim Transfer of Scenes

Unlike prior work [23, 20], our goal is not to accurately master a single environment, but rather to train a generalist agent capable of generalizing to new, unseen environments. To obtain a wide distribution of scenes with a variety of layouts, colors, lighting conditions, We develop our general purpose, easy to use, real-to-sim-to-real pipeline that support crowdsourcing contribution of 3D scans. Digital twins are obtained directly from real-world videos or image sequences using photogrammetry methods such as Gaussian splatting [37] and neural radiance fields [38]. High fidelity 3D meshes can be scanned in under five minutes using off-the-shelf mobile software such as Polycam [32] and ARCode [31]. This easy-to-use software running on standard, commercial mobile phones enables crowdsourcing of real-world scans from non-experts worldwide with minimal instruction. The crowdsourced scenes demonstrate a natural distribution of clutter, scene layouts, colors, lighting conditions, and positional variations.

These real-world scans are then easily transferred into a photo-realistic physics simulator, Issac Sim [39], using an easy-to-use GUI for scene articulation and curation [23]. This flexible interface accommodates various scene complexities, from static to highly articulated environments. Using the GUI, we also add objects of interests (bowl/mug/cup for putting the object into the sink, box for

putting the box in the cabinet) into the scene, and additional site to mark the position of sink and cabinet.

With the scene and the object rendered inside the simulation, we use teleoperation with carb keyboard to collect 10 demonstrations for each articulated environment. There are 14 different discretized actions to choose from, corresponding to two directions in all spatial axis and rotational axis, and open and close the gripper. See Appendix 7 for details.

6.3 Autonomous Data Collection

6.3.1 Multi-task bootstrapped RL fine-tuning:

Given a set $\mathcal{T}$ of 10 demonstrations on each one of the digital twin scenes in the batch, and an easily defined sparse reward across all tasks, we leverage the current capabilities of fast multi-environment training on GPUs and accurate simulators to do RL fine-tuning using PPO Schulman et al. [36] to obtain a policy that is more robust to object poses, corrections, and disturbances than if we simply learned from the demos. In addition, this multi-scene policy is being trained from privileged state space, since this removes the need for rendering, making the amount of simulated parallel environments higher, and training becomes faster since we can use bigger batch sizes. Finally, we observe that although equivalent in theory, training across multiple scenes instead of a single environment at a time per GPU in practice brings a big speedup in training. With the available resources our experiments are run with 10 different scenes in parallel spread across 2048 environments, and even though the total training time is the same as for a single scene, the policy now works across 10 scenes which corresponds effectively to a 10x speedup.

6.3.2 Teacher-student distillation

In the previous RL fine-tuning step, we obtain a state-based policy that works across the whole batch of environments. However, in the real-world we do not have access to this privileged state of the environment such as object poses. For this reason we need our policy to take as input a state representation available in the real world. We decide to use colored point clouds as the sensor modality. Thereafter, we use teacher-student distillation techniques to distill the obtained policies into π_G. This consists on for each one of the scenes we use the working state-based policy to collect a set of 1000 trajectories. Out of the 1000 trajectories, 500 of them are rendered from two cameras in simulation and 500 are synthetically generated by sampling from the meshes, making the point cloud fully observable. In practice, the synthetically generated point clouds make learning with point clouds as input much smoother even for the camera-rendered point clouds. Due to the significant amount of data available, our experiments go up to 56 environments with 1000 trajectories each. We avoid catastrophic forgetting by retaining the data from previous batches during distillation.

6.4 Fine-tuning

6.4.1 Scanned Deployment Fine-tuning

Algorithm 2 Scanned-deployment fine-tuning

Input: a generalist policy π_G, a digital twin of an environment $\mathcal{E}$
$\mathcal{T} \leftarrow \{\}$
while $|\mathcal{T}| \leq 10$ **do**
  $\mathcal{T}_e \leftarrow \mathrm{RolloutPolicy}(\mathcal{E}, \pi_G)$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathrm{FilterSuccessfulRollouts}(\mathcal{T}_e)$
  $\pi_s \leftarrow \mathrm{RLFinetuning}(\mathcal{T}, \mathcal{E})$
  $\pi_G \leftarrow \mathrm{TeacherStudentDistillation}(\mathcal{E}, \mathrm{FreezeEncoder}(\pi_G), \pi_s)$

6.4.2 Few-shot Supervised Fine-tuning:

The second proposed fine-tuning technique involves using small amounts of human-provided real-world demonstrations for few-shot supervised fine-tuning. We fine-tune the generalist policy $\pi_G(a_t|o_t)$ using supervised learning on a dataset of human collected visuomotor demonstrations $\mathcal{D}_h$ via standard maximum likelihood as shown in Eq 2.

Architecturally, this involves freezing the preliminary "visual processing" layers and fine-tuning only the final fully-connected layers of the pretrained generalist network π_G. As we show in Section 4.3, this straightforward fine-tuning procedure can yield significant performance improvements with a small numbers of real-world demonstrations.

7 Task details

Figure 7: Overview of a selected number scenes and objects used for the real world evaluation of the task of placing bowls/mugs/cups in the sink.

In this section of the appendix, we describe the specification of each tasks for training and evaluating **SCAR**. For each task, the state space consists of the concantation of the following state information: object positions, object orientations, DOF positions of the tool normalized to its max and min ranges, end-effector orientation, end-effector position. The action space consists of one of the 14 discretized actions, corresponding to the end effector delta pose. In specific, the 14 actions include the following: 6 actions in position change, corresponding to ± 0.03m change in each axis; 6 actions in orientation change, corresponding to ± 0.02 radian change in each axis; 2 actions correpsonding to gripper open and close.

We define a sparse reward function for each task in PPO training:

- **Put object into sink**: $success =$ $||\text{sink\_site} - \text{object\_site}||_2 < 0.25$ && condition(object_upright) && condition(gripper_open)
- **Put object into cabinet**: $success =$ cabinet_z_axis $<$ object_z_axis && condition(object_upright) && condition(gripper_open)

7.1 Simulation details

We used the latest physical-based virtual environment simulation platform, Isaac Sim [39] for our simulation task training. Our reinforment learning codebase is inspired by the Orbit codebase [40], a unified and modular framework for robot learning built upon Isaac Sim.

For the simulation parameters of the environment, we use the default value set by the GUI in the most case, except changing the collision mesh of background from convex decomposition to SDF mesh with 256 resolution to reflect high-fidelity collision mesh. For all the other objects, we use the default value, which is convex decomposition with 64 hull vertices and 32 convex hulls as the collision mesh for all objects. We keep all the physical parameters of the environment as default in the GUI. The default value of physical parameters for all objects are as follow: dynamic and static frictions of all objects as 0.5, the joint frictions as 0.1, and the mass as 0.41kg. See Table 6 for

Task	Episode length	Randomized Object Ids	Position Min (x,y,z)	Position Max (x,y,z)	Orientation Min (z-axis)	Orientation Max (z-axis)
obj2sink	135	[Background, Object]	[[-0.1,-0.1,-0.1],[-0.1,-0.1,0]]	[[0.1,0.1,0.1], [0.1,0.1,0]]	[-0.3, -0.3]	[0.3, 0.3]
obj2cabinet	150	[Background, Object]	[[-0.1,-0.1,-0.05], [-0.1,-0.1,0]]	[[0.1,0.1,0.05], [0.1,0.1,0]]	[-0.1, -0.15]	[0.1, 0.15]

Table 1: Specific simulation parameters for each tasks.

Figure 8: Examples of simulated environment used for RL fine-tuning. The top ones corresponds to environments for obj2sink task and the bottom ones corresponds to environments for obj2cabinet task.

task-sepcific randomization parameters, Table 2 for task-specific camera parameters and Figure 8 for examples of simulated scenes.

8 Implementation details

8.1 Architecture Details

8.1.1 State-based policy

As described in Section 3.2, we trained a series of state-based policies with privileged information in simulation. The policy model is a simple Multi-Layer Perceptron (MLP) network, with input as the privileged state in simulation as specified in 7 and outputs a probability distribution of 14 classes, corresponding to the probabilities for each discrete end-effector action. To implment PPO with BC loss algorithm, we built upon the Stable Baselines 3 repository [41]. The size of MLP network is a mix of two sizes: two layers of size 256 and 256, and three layers of size 256, 512 ans 256. See Table 3 for more details.

Task Parameters	Position (x,y,z) Camera	Rotation (quat) Camera	Crop Min Camera	Crop Max Camera	Size Image
obj2sink	[[-0.01,-0.50,0.69], [-0.01,-0.50,0.69]]	[[0.84,0.33, -0.15, -0.41], [-0.42,-0.22,-0.39,0.79]]	[-0.8,-0.8,-0.3]	[0.8,0.8, 1.0]	(640,480)
obj2cabinet	[[-0.01,-0.50,0.69], [-0.01,-0.50,0.69]]	[[0.84,0.33, -0.15, -0.41], [-0.42,-0.22,-0.39,0.79]]	[-0.2,-0.5,-0.5]	[1.5,0.5, 1.5]	(640,480)

Table 2: Camera parameters for each task.

MLP layers	PPO n_steps	PPO batch size	PPO BC batch size	PPO BC weight	Gradient Clipping
256,256 or 256, 512, 256	Episode length	31257	32	0.1	5

Table 3: State-based policy training parameters. The rest of the parameters are the default as described in Stable Baselines 3[41].

8.1.2 Point cloud policy

As mentioned in Section 3.3, when distillating the state-based teacher policy to a fine-tuned visumotor policy, we will train a point cloud policy as the student. We train an MLP network of size 256,256, that takes the embedding of the point cloud observation, that has 128 dimensions, together with the state of the robot (end-effector scaled pose, position and orientation), that has 9 dimensions, as the input, and a probability distribution of 14 actions as output. To encode the point cloud observation, we use the volumetric 3D point cloud encoder proposed in Convolutional Occupancy Networks [42], which consists of a local point net that converts the point cloud to 3D features, followed by a 3D U-Net that output a dense voxel of features. The output features are then pooled by a max pooling layer and an average pooling layers separately, with the pooling output concatenated into the resulting encoding of dimension 128.

9 Detailed Evaluation Results

We conducted experiments involving disturbance and distractors for putting the object into the sink task to study the robustness of the generalist policies. The experiments include multi-object scenario, dim lighting scenario, messy kitchen scenario and disturbance scenario.

9.1 Evaluation on Multi-Object Scenes

In this section we study the extrapolation and robustness capabilities of the learned generalist policies by evaluating them on tasks involving multi-object scenes. Specifically, the robot needs to pick and place multiple objects into the sink sequentially, even though it was trained on single objects. We evaluated this by allowing the robot six trials to place the three objects in the kitchen into the sink. As shown in Figure 6, despite not being trained for multi-object, the policy succeeds 80% of the time in placing two objects and 10% in placing all three objects sequentially.

# of Env distilled	Kitchen Ids	Bowl right of the sink	Bowl left of the sink	Mug right of the sink	Mug left of the sink	Overall
9	Kitchen 1	66.7%	0%	22.2%	0%	22.2%
9	Kitchen 2	66.7%	0%	11.1%	0%	19.4%
9	Kitchen 3	41.7%	0%	16.7%	0%	14.6%
36	Kitchen 1	55.6%	55.6%	22.2%	22.2%	38.9%
36	Kitchen 2	44.4%	33.3%	22.2%	22.2%	30.6%
36	Kitchen 3	13.3%	46.7%	20.0%	60.0%	35.0%
56	Kitchen 1	55.6%	55.6%	44.4%	66.7%	64.8%
56	Kitchen 2	77.8%	33.3%	55.6%	22.2%	47.2%
56	Kitchen 3	58.3%	83.3%	66.7%	91.7%	75.0%

Table 4: Zero-shot success rate for putting object to sink task. We tested on different types of objects such as bowls, mugs and cups. We evaluated the policy by placing the object on either sides of the sink.

Num. Env distilled	Num. demos	Kitchen Ids	Object type	Success rate
0 (IL)	10	Kitchen 1	Bowl	10.0%
36	10	Kitchen 1	Bowl	70.0%
0 (IL)	10	Kitchen 2	Mug	10.0%
36	10	Kitchen 2	Mug	70.0%
0 (IL)	10	Kitchen 3	Mug	10.0%
36	10	Kitchen 3	Mug	60.0%
0 (IL)	10	Kitchen 3	Bowl	10.0%
36	10	Kitchen 3	Bowl	60.0%

Table 5: Imitation learning baseline and few-shot supervised fine-tuning success rate for putting object to sink task.

Num. Env distilled	Num. demos	Kitchen Id	Success rate for grasping	Success rate for placing
0 (IL)	10	Kitchen 1	0.0%	0.0%
16	10	Kitchen 1	10.0%	0.0%
26	10	Kitchen 1	60.0%	20.0%
36	10	Kitchen 1	80.0%	30.0%
0 (IL)	10	Kitchen 2	0.0%	0.0%
16	10	Kitchen 2	10.0%	0.0%
26	10	Kitchen 2	20.0%	20.0%
36	10	Kitchen 2	30.0%	30.0%

Table 6: Imitation learning baseline and few-shot supervised fine-tuning success rate for putting object to cabinet task. We recorded both the success rate for grasping the object and placing the object to the cabinet.

Num. objects successfully placed in the sink	Generalist policy success rate	Imitation Learning baseline success rate	Average num. of episodes
1	100.0%	0.0%	1.6
2	80.0%	0.0%	3.5
3	10.0%	0.0%	4

Table 7: Multi-object scenario evaluation. As shown in Fig 8, three objects are placed in the scene. The policy is rolled out for 6 episodes in total. The table show the success rate for average number of episodes it takes to placing certain number of objects into the sink.

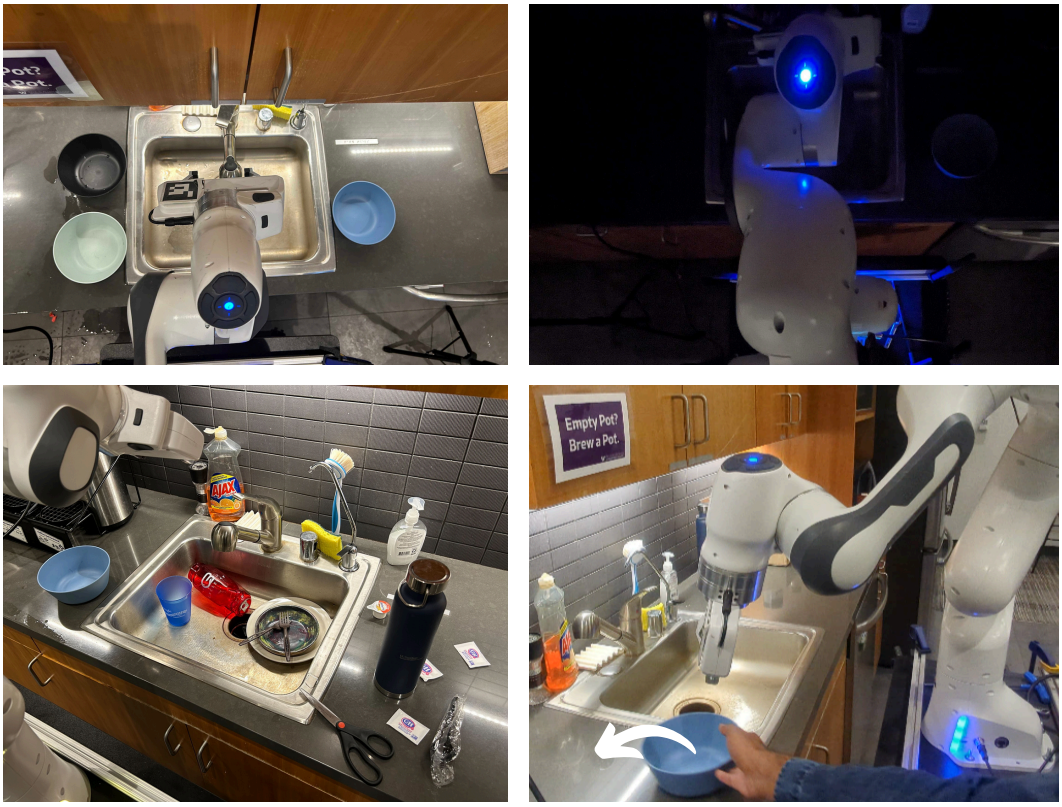

Figure 9: Overview of the experiment setup for evaluating the robustness capacity of the generalist policy. upper left: multi-object scenario. upper right: dim lightning scenario. bottom left: messy kitchen scenario. bottom right: human disturbance scenario. See Table 8 for success rate.

9.2 Evaluation on Scenes Involving Disturbance and Distractors

In the dim lighting scenario, there is minimal lightning in the scene, while the robot is only trained in the environment with sufficient lightning. The robot was able to complete the task successfully into the sink for 30% of all trials. See Figure 9 for experimental setup.

In the messy kitchen scenario, there are dirty dishes and tableware sitting in the sink, closely mimicking the realistic setting of a household kitchen sink. The robot is only trained in the environment with clean sink. The robot was able to complete the task successfully into the sink for 30% of all trials. See Figure 9 for experimental setup.

In the human disturbance scenario, the experimenter would push the object the change its position during the evaluation process. The robot was able to complete the task successfully into the sink for 50% of all trials. See Figure 9 for experimental setup.

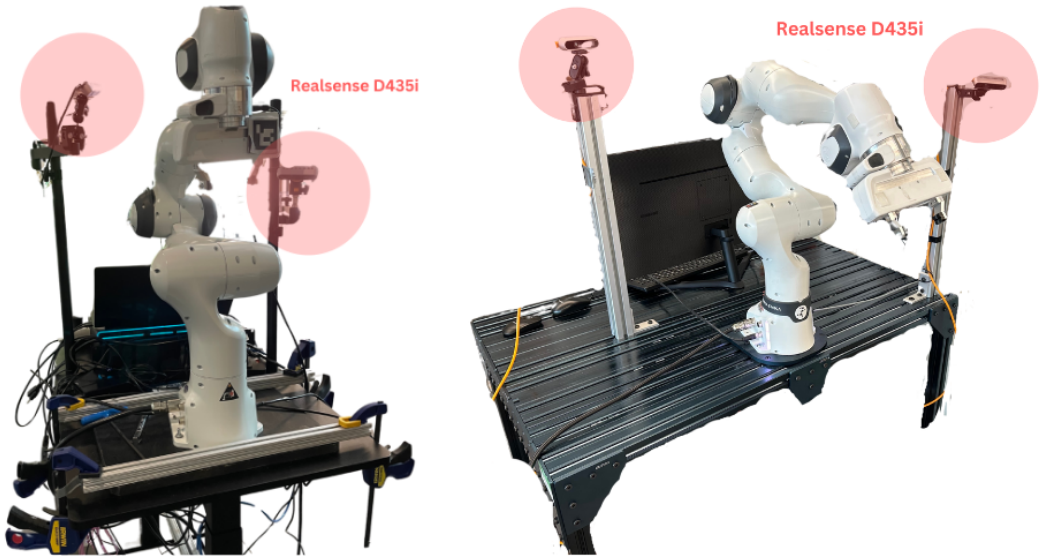

Figure 10: Overview of the hardware used to evaluting **SCAR**. left: used to evaluate in Kitchen 2 in both tasks. right: used to evaluate in Kitchen 1 and 3 in putting object to sink and Kitchen 1 in putting box to cabinet.

Scenario name	Generalist policy success rate	Imitation Learning success rate baseline
Dim lighting scenario	30.0%	0.0%
Messy kitchen scenario	30.0%	10.0%
Human disturbance scenario	50.0%	0.0%

Table 8: Success rate for various disturbance and distractor scenarios.

10 Hardware Setup

Real world experiemnts are run on two different Panda Franka arms. Both of the Panda Franka arms are mounted on mobile tables, and running the same experiments, but they are located in two different institutions and therefore having access to different real-world kitchen settings.

We mount two calibrated cameras per setup to obtain the depth perception in order to create an aligned point cloud map for vision-based policies. In particular, we use the two Intel depth Realsense cameras D435i for both setups. See Figure 10 for mode details on the robot setup.

11 Crowdsourcing

We source the kitchen scans from both expert and non-expert users. For placing the object to sink task, we collect policies on 29 sink scenes, of which 22 were collect through crowdsoucing. For putting the object to cabinet task, we collect policies on 26 cabinet scenes, of which 18 are collected through crowdsourcing. Figure 11 shows the poster we use for crowdsourcing and Figure 12 shows the geographical distribution of the crowdsourcing contributors.

Figure 11: Poster used for calling crowdsourcing contribution.

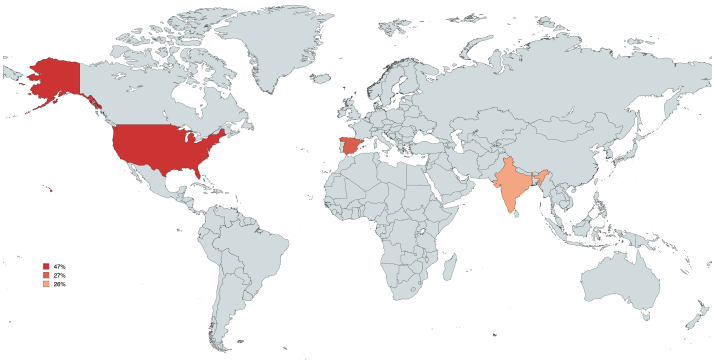

Figure 12: Geographical distribution of crowdsourcing contributors.

12 Compute Resources

We run all the experiment on an NVIDIA GeForce RTX 3090, an NVIDIA GeForce RTX 3080, an NVIDIA RTX A6000. The first step of RL fine-tuning is to use the GUI to create a task environment from a crowdsourced kitchen scan, and collecting a set of 10 demonstrations in simulation using teleopertation, which takes 1 hour per environment on average. We leverage a distributed research computing cluster to run the RL fine-tuning, where we request an NVIDIA Quadro RTX 6000, and it takes on average 20 hours to converge. Finally, during the teacher-student distillation step, it takes 4 hours on average to collect the simulation trajectories and 2 hours to collect the synthetic trajectories, and 5 days to distilling into the vision policy.