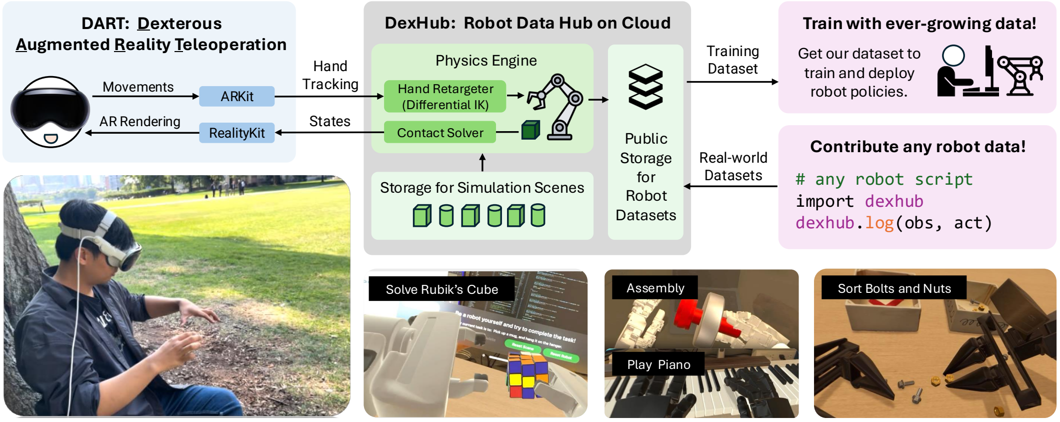
DexHub and DART:
Towards Internet Scale Robot Data Collection

Younghyo Park
MIT CSAIL
United States
younghyo@mit.edu

Jagdeep Singh Bhatia
MIT CSAIL
United States
jagdeep@mit.edu

Lars Ankile
MIT CSAIL
United States
ankile@mit.edu

Pulkit Agrawal
MIT CSAIL
United States
pulkitag@mit.edu


DART: Dexterous Augmented Reality Teleoperation
Movements
ARKit
AR Rendering
RealityKit
Hand Tracking
States
DexHub: Robot Data Hub on Cloud
Physics Engine
Hand Retargeter (Differential IK)
Contact Solver
Storage for Simulation Scenes
Public Storage for Robot Datasets
Training Dataset
Real-world Datasets
Train with ever-growing data!
Get our dataset to train and deploy robot policies.
Contribute any robot data!
Solve Rubik's Cube
Assembly
Play Piano
Sort Bolts and Nuts

```
# any robot script
import dexhub
dexhub.log(obs, act)
```

Figure 1: We present **DART**, Dexterous Augmented Reality Teleoperation system, enabling intuitive, low-latency teleoperation with cloud-hosted simulation. Through a user study, we found that DART enables first-time robot teleoperators to achieve 2.1× faster data collection throughput with significantly lower physical fatigue than existing real-world teleoperation platforms. To further support scaling up data collection efforts in the community, we also release **DexHub**, a cloud-hosted data hub for robot learning where data collected in DART is automatically stored. https://dexhub.ai/project

Abstract: The field of robotics has long grappled with a critical challenge: the scarcity of diverse, high-quality data that can be used to train a generalist robot policy. While real-world data collection efforts exist, requirements for robot hardware, physical environment setups, and frequent resets significantly impede the scalability needed for modern learning frameworks. To address these limitations, this paper introduces DART, a novel teleoperation platform that reimagines robotic data collection by leveraging cloud-based simulation and augmented reality (AR). Our user studies highlight that DART enables higher data collection throughput and lower physical fatigue compared to real-world teleoperation frameworks. In addition, our policy training experiments using DART-collected datasets demonstrate successful Sim2Real transfer with robust trained behaviors. Most importantly, all data collected through DART is automatically stored in our cloud-hosted database, DexHub, and publicly available to anyone.

Keywords: Robot Teleoperation, Augmented Reality, Sim2Real

CoRL 2024 Workshop on Whole-Body Control and Bimanual Manipulation (CoRL 2024 WCBM).

1 Introduction

Robotics has seen impressive progress with the advent of learning-based control. However, the primary bottleneck remains the lack of large amounts of diverse and high-quality data for training robust and generalizable robot policies. It would be ideal to have an internet-scale robotics dataset that continually and rapidly grows with data coming from everywhere in the world — just like how people easily upload language and vision data on the internet. Despite recent efforts [1, 2, 3, 4], we are not there yet. In this paper, we examine and address many key bottlenecks in achieving this dream.

Consider the typical process of collecting robot data on an example task: moving dishes from sink to dishwasher. The very first point of friction is ***setting up the right environment*** for the robot to perform the task in. There are two options: physically construct a kitchen in the lab around the robot or physically move the robot to an actual kitchen. Neither is easy to scale as we will need data from many kitchens.

Once in the right environment, a common scheme for data collection is to have humans operate the robot for a task. As an operator, the second point of friction is ***observing and understanding*** what is happening in the scene. Even when in the same room with the robot, due to visual occlusions, operators may not understand how objects are moving as a result of robot's operation. Remote teleoperation adds additional challenges originating from network delays, limited field of view, and visual artifacts. Such challenges can slow down operators and in some cases prevent them from performing dynamic or precise tasks.

If the operator manages to resolve the first two challenges and move all the dishes from the sink to the dishwasher to complete the sample task, a third obstacle emerges; they must then return all the dishes to the sink to collect a new trajectory! In addition to being time-consuming, this ***resetting*** process is both physically and mentally exhausting as operators must context-switch between robot control and environment setup. Ensuring that each reset presents the robot with a diverse range of scenarios is also mentally taxing.

What makes the experience even worse for the operators is the need to ***repeat*** the process of *teleoperating* and *resetting* a large number of times. The number of required demonstrations scale with the task complexity and the extent of required generalization. Unfortunately, humans are known to lose focus when performing a repetitive job [5].

Finally, say the operator has finished collecting a few hundred demonstrations. How does the recorded data get ***processed and stored***? It is common to store collected demonstrations on a local machine or a *private* cloud, which is often not shared unless someone explicitly requests it. Additionally, different data structures and conventions for data storage make data-sharing difficult.

To address the aforementioned pain points in collecting robot data via teleoperation, we introduce **DART**, a Dexterous Augmented Reality Teleoperation system, enabling *anyone* in the world to teleoperate robots in simulation with an intuitive, game-like AR interface. Connected to a cloud-hosted simulation, DART allows users to collect demonstrations for an unlimited number of scenes in one sitting without having to physically set up environments or physically move robots to different places. DART's high-fidelity AR rendering allows users to observe the scene in great detail with minimal occlusion, enabling teleoperation of complex tasks. DART also allows users to reset the environment with a click of a button, removing the taxing process of physically resetting the scene.

As a result, our user study shows that DART achieves 2.1× **faster data collection throughput** with significantly less physical and cognitive fatigue on tasks requiring fine-grained control compared to most existing robot data collection pipelines. Our experiments also highlight the unmatched benefits of collecting demonstrations in simulation over the real world. Simulation-trained policies achieve higher robustness than real-world trained policies due to data augmentation and randomization strategies only possible in simulation.

Last but not least, all robot demonstrations collected through DART are automatically stored and logged to our public cloud-hosted database, **DexHub**, which serves as an open-sourced data hub for robot learning. In fact, DexHub is not limited to hosting DART-generated data; DexHub offers a simple Python API for *anyone* to easily log any robot data on the cloud by adding a single line of code to existing robot execution scripts, `dexhub.log(obs,act)`, offering a seamless cloud-logging solution for any real-world policy rollout or teleoperation happening around the world.

Our key contributions are outlined as follows:

1. In Sec. 3, we introduce DART, a novel AR-based teleoperation platform, and detail its system architecture and supported features. We also showcase the diversity of tasks we can perform with DART, unlocked by enhanced teleoperation experience.
2. In Sec. 4.1, we analyze the impact of different teleoperation interface design choices through user study. We show that DART enables higher data collection throughput and lower fatigue than alternatives.
3. In Sec. 4.2, we show that policies trained with data collected via DART can be effectively transferred to the real world and are more robust than those trained with real-world demos.
4. In Sec. 5, we provide an in-depth overview of the proposed DexHub platform that serves as a central hub for logging not only the demonstrations generated by DART, but all robot interactions happening around the world.

2 Related Works

2.1 Large-Scale Robot Data Collection Efforts

Addressing the need for large-scale datasets in robotics, there have been two primary approaches within the community. The first approach, as exemplified by projects like [1], focuses on gathering existing datasets from various robotics institutes worldwide into a single place. These initiatives involve a central team overseeing the data gathering, post-processing, and release. The second approach involves teams actively collecting large-scale datasets themselves by teleoperating robots in real-world environments. For example, [2] collected 110k trajectories for diverse tasks through real-world teleoperation with the help of volunteer participants. Similarly, [3] created a dataset of 60k trajectories using a low-cost robotic arm. Most recently, [4] have released 76k demonstrations across 564 scenes using a Franka Panda attached to a mobile platform. These efforts all unanimously highlight the value of large datasets in improving the performance of trained policies.

However, we argue that relying on disconnected, project-level efforts to create such datasets is not a scalable solution for the robotics community. The episodic, labor-intensive collection efforts seen in these examples fail to mirror the organic growth of language and vision datasets on the internet. Furthermore, these datasets are limited in scope, primarily focused on single-arm robots with parallel jaw grippers, neglecting the richness of bimanual or dexterous manipulations. Finally, these datasets are collected exclusively in real-world settings, overlooking the significant potential of simulation as a data source. Simulation allows for the refinement and augmentation of human-collected – and therefore possibly suboptimal – datasets through online reinforcement learning using massively parallelizable simulation environments [6]. Such refinement can address the potential performance saturation often observed on policies trained only with supervised learning [7, 8, 9, 10].

2.2 Collecting Robotic Dataset in Simulation

Using simulation as an alternative environment for collecting demonstrations has been explored in the community. For example, [11] utilized webcams attached to laptops to allow users to teleoperate various robot morphologies in simulation. [12] employed a VR interface where humans control simulated dexterous hands, while specialized exoskeletons capture their hand movements. More recently, with advancements in VR devices, [13, 14] have demonstrated similar technical stacks that no longer require external hand trackers, but instead utilize the built-in capabilities of modern VR/AR devices to capture hand movements. All existing VR-based systems use stereo rendering streams as a source of visual feedback. However, relying on raw visual streams of simulated renderings inevitably creates a noticeable latency in network communication, forcing designers to trade-off visual fidelity and latency to maintain real-time performance. The use of Augmented Reality (AR) for robot teleoperation, on the other hand, has not yet been extensively explored as a solution to this problem. Finally, no existing platform has fully leveraged simulation's potential by making data collection widely accessible and available to the general public – particularly to those without specialized knowledge in robotics or the ability to set up simulation servers.

3 DART: Teleoperating Robots in Sim via AR

This section details the system architecture of DART and its benefits (Sec 3.1). We then introduce the main features of the platform (Sec 3.2), which are designed to maximize the platform's capability (Sec 3.2) and enhance user experience. DART is easy to install: anyone can download and install from App Store.

3.1 System Architecture

Simulation Assets as AR Objects Enabled by Apple's RealityKit, DART presents all assets in simulation environments, including robots, as photo-realistic AR objects overlayed over each operator's real-world environment. Handling visualization locally on the AR device (a) removes unnecessary latency from transmitting large image data packets and (b) significantly improves the real-timeliness of the simulation by removing the compute-intensive rendering layer. Variation in latency critically impacts the user's data collection throughput and cognitive fatigue, as highlighted by our user study (See Sec. 4.1).

Low-Latency Communication Communication between the AR device, i.e., Apple Vision Pro, and the cloud-hosted simulation is handled via gRPC, which facilitates low-latency, *asynchronous* bidirectional data transfer. The AR device sends hand-tracking data to the simulation, and asynchronously receives the simulation state. Table 1 highlights the reduced network load of our approach compared to a typical setting where real-world or simulated camera streams are transmitted over the network. Even in the most adversarial case, where robots have $n = 58$ joints and simulation scenes contain $m = 50$ objects, the data packet size is over **1,000× smaller** than that required for existing teleoperation frameworks.

		DART (Ours)	Other Immersive Teleoperation Works
Human → Robot	Data Type	Hand Tracking	Hand and Head Tracking
		25 Hand Keypoints × SE(3)	(25 Hand Keypoints + 1 Head) × SE(3)
	Packet Size	0.7kB	0.728kB
Robot → Human	Data Type	Oracle Sim States	Stereo RGB image
		n joints × `float` m objects × SE(3)	2×(480×640×3) `uint8`
	Packet Size	**1.6kB**	**1843.2kB**

Table 1. We highlight DART's **1,000× reduction** in network packet size between robot and operator's AR device compared to existing frameworks. $n = 58$, $m = 50$ assumed for DART.

Cloud-Hosted Simulation The robot simulation is powered by MuJoCo [15] and dynamically launched on AWS Elastic Container Registry (ECR) as users join. Each simulation instance runs in the cloud, enabling open access and low user setup costs. Due to compact packet sizes (Table 1), cloud-hosting does not critically impact the overall latency of our platform compared to local-hosting, as evidenced in Table 2.

	Cloud-Hosted Simulation	Local Machine and Local Network
CPU	AWS EC2 C7i	i9-13900k
Packet Travel Time	15.4 ms	10.3 ms
Simulation Step	1.8 ms	1.6 ms
Total	17.2 ms	11.9 ms

Table 2. Comparing the time profile of our system running on the cloud vs hosted on a local machine.

Hand Tracking and Mapping DART leverages Apple's ARKit to track poses of hand and wrist keypoints. We use a subset of detected keypoints, which fully determine the end-effector and finger movements, as target points for robots to track. Specifically, for robot systems with parallel-jaw grippers, we use the xyz position of 4 finger key points as tracking targets, which fully determine the SE(3) pose of the robot's end-effector (Fig. 2).

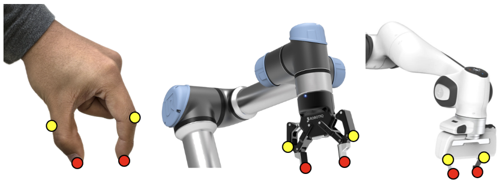


Figure 2: 4 finger keypoints used as tracking points for robots with parallel-jaw grippers.

DART uses differential inverse kinematics [16] by defining position-only tracking costs for each keypoints, $e(\mathbf{p})$. We additionally apply basic safety constraints, i.e., self-collision avoidance, expressed as $d(q)$. The resulting optimization problem is as follows,

$$\min_{v \in c} \sum_{\mathbf{p} \in \mathcal{P}} \|J_e(q)v + \alpha e(\mathbf{p})\|^2$$
$$\text{s.t. } v_{\text{min}}(q) \leq v \leq v_{\text{max}}(q),\ d(q) > 0.$$

For dexterous five-fingered hands, we use six position-only keypoints – five from the fingertips and one from the wrist.

3.2 System Features

DART supports a wide range of features to enhance the teleoperation experience while maintaining low setup costs, allowing *anyone* to participate in robotics data collection.

Pre-Designed Robots and Scenes Out-of-the-box, DART supports many robots: Unitree Humanoid Series (H1, G1), ALOHA [17], UR5 with multiple end-effectors (Robotiq 2F-85 gripper, Allegro Hand, Shadow Hand), Franka Research 3 with Panda Hand. High-fidelity MuJoCo models of these robots were provided by [18].

Importing Custom Scenes Users can import custom simulation environments and assets to extend the platform's capabilities further. Assets can be uploaded through our online portal [`https://dexhub.ai`] and accessed via DART App on VisionOS App Store.

One-Click Reset DART includes an efficient task-resetting feature in simulation. Users can reset the environment with a single click of a button, significantly reducing operator fatigue and increasing data collection throughput.

Instant Task Switching In addition to resetting a single scene, DART enables quick switching between various tasks and simulation environments. This functionality minimizes the operator's mental fatigue that arises from repetitively performing the same task, allowing for a more engaging data collection experience.

Capability and Task Diversity DART is capable and versatile. It supports a wide range of tasks, from simple object manipulation to complex, precise, and dexterous maneuvers, as highlighted in Figure 1. These examples and those below illustrate the platform's potential to support various research and practical applications in robotics. Please find videos on our website.

- Fine motor skills: e.g., picking up small objects.
- Household chores: e.g., organizing kitchen countertop, hanging mugs on a rack.
- Dexterous Manipulation: e.g., solving a Rubik's cube.

4 Experiments

Our experiments address two key questions:

1. *How **intuitive** is DART for robotics novices to use?* We conduct a formal user study to assess the platform's accessibility to individuals without robotics expertise. (Section 4.1)
2. *Can the data collected in simulation be effectively **transferred to real-world** robots?* We demonstrate that policies trained on data collected through DART transfer zero-shot to real environments with simple Sim2Real techniques. We also highlight the generalizability of DART policies compared to those trained with real-world data. (Section 4.2)

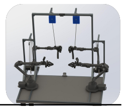
	DART	Modulation of **Command Interface**	Modulation of **Visual Feedback Design**			ALOHA [17]
		Finger Tracking ↓ Kinematic Double	Rendering as AR Objects ↓ Sim Rendering (RGB, Stereo)	Rendering as AR Objects ↓ Sim Rendering (RGB, Mono)	Active Viewpoint ↓ Fixed Viewpoint	
Data Throughput	**7.8** parts / min	6.8 parts / min	3.6 parts / min	3.0 parts / min	2.7 parts / min	**3.7** parts / min

Table 3. Quantitative comparison between different teleoperation setups for two ViperX arms with parallel-jaw gripper [17]. Users are tasked to organize ten bolts and nuts into two boxes, and DART allowed users to organize 7.77 parts per minute on average, while modulation of both command interface and visual feedback settings dropped the performance significantly. We report percent change in throughput relative to DART averaged across users.

4.1 User Study

Through a controlled user study, we analyze the impact of DART's design decisions on intuitiveness and usability. Specifically, we compare: (a) the experience of collecting data in real-world versus simulation environments (Sec 4.1.1), (b) methods of visual perception (Sec 4.1.2), and (c) control interfaces (Sec 4.1.3). A total of nine participants with no prior experience in robotics were recruited.

In varying settings, participants spent 7 minutes collecting as many robot demonstrations as possible. We asked the participants to organize 10 bolts and nuts from a table into boxes. Participants were responsible for resetting the scene both in simulation and real-world environments via reset button or manual effort, respectively. Participants teleoperated two ViperX arms with parallel-jaw grippers, and kinematically equivalent teacher devices were used as a real-world teleoperation interface [17]. Quantitative results are presented in Table 3; further analysis follows.

4.1.1 Teleoperating in Real-World vs Simulation

Our user study comparing DART and real-world teleoperation revealed two key findings. First, a significant portion of time in real-world data collection is spent physically resetting the environment and managing unexpected hardware failures (e.g., performing electrical resets after motor malfunctions) as reported in Fig. 3. By contrast, most of the time in DART is dedicated to actual data collection.

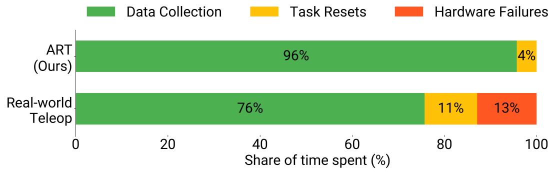


Figure 3: Time spent in DART is more productive than real-world equivalent.

Second, even after accounting for reset times and hardware malfunctions, participants in real-world teleoperation showed around **2**× lower data collection throughput. For a comparison experiment with wide range of real-world data collection systems, we used two different robot systems: dual ViperX arms [17] and RB-Y1 from Rainbow Robotics. Both data collection system has kinematic double as its teleoperation interface. Total 20 participants were asked to perform 4 bimanual tasks ranging from relatively simple object rearrangment task to precise insertion tasks. Figure 4 shows the data throughput comparison between DART and two different real-world robot systems.

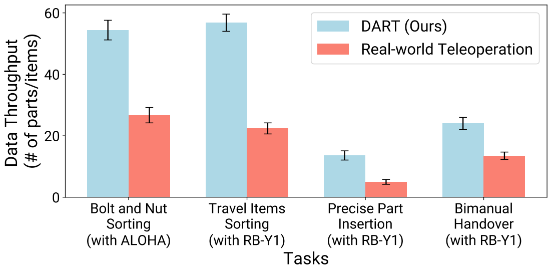


Figure 4: Data throughput comparison between DART and real-world teleoperation systems. For each robot and task, five participants were asked to teleoperate the tasks as many as possible for 7 minutes. For real-world teleoperation, kinematically equivalent teacher device was used as a teleoperation interface.

4.1.2 Effect of Visual Observation on Human Operator's Performance

Our key findings are threefold. First, transmitting images over a network inevitably introduces a tradeoff between latency and decreased visual fidelity, which can negatively impact teleoperation experience. All methods transmitting simulation renderings over the network (those with stero and mono rendering) suffered a significant drop in user's data collection throughput compared to DART which transmits only the raw simulation states. Second, we find that mono rendering, which limits the ability to properly perceive depth, suffered a performance drop over stereo rendering. Additionally, some participants reported feeling nauseous (Table 3) with stereo rendering – which uses a fixed interpupillary distance (IPD). By contrast, DART relies on VisionOS's [1] native rendering engine, which dynamically adjusts to each user's IPD [19]. Finally, we found that active perception, where users can explore their surroundings and adjust their viewpoint by moving their heads, is critical. Teleoperation without active perception reduces the data collection rate by 21.7%.

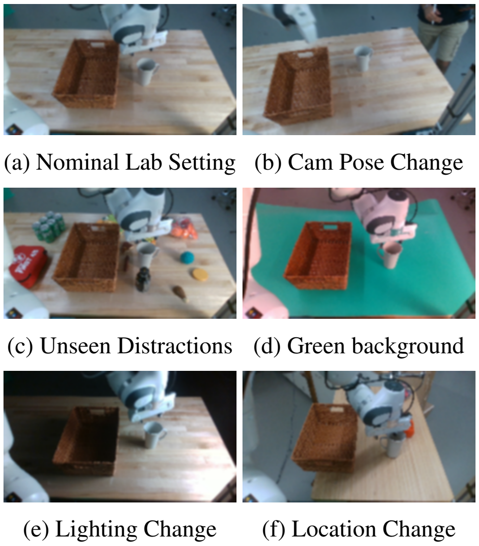
4.1.3 Control Method

We compared two methods for operating robots in simulation: a) a kinematically equivalent teacher device and b) inverse kinematics (IK) using hand tracking keypoints as targets. Our findings indicate that the kinematic double did not significantly improve task success rate over its IK equivalent. While the kinematic double provides more direct control over the robot's joints, users reported that the intuitive hand tracking offered by DART was sufficient, or even better, due to reduced weight and strain on the operator (Table 3).

4.2 Sim2Real and Generalizability

Both DART and real-world data collection offer distinct advantages for real-world policy training. With DART, roboticists benefit from significantly higher data throughput with reduced physical and cognitive demands, as demonstrated by our user study (Sec. 4.1). One minor downside of using DART is the need to import scenes into the simulation environment. Fortunately, with modern advances in computer vision [20, 21], scanning 3D objects from the real world has become incredibly efficient. The bigger challenge, however, lies in bridging the potentially large Sim2Real gap. Given these trade-offs, how does one weigh the benefits of faster data collection against the challenge of real-world deployment?

Our experimental results suggest that collecting data in simulation offers **more advantages than drawbacks** when paired with a proper Sim2Real pipeline. In particular, we demonstrate the unique robustness of Sim2Real-transferred policies, enabled by diverse data augmentation techniques only available in simulation environments.


(a) Nominal Lab Setting (b) Cam Pose Change

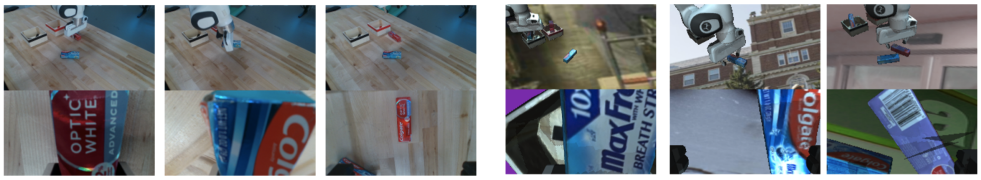
(c) Unseen Distractions (d) Green background

(e) Lighting Change (f) Location Change

Figure 5: Six different settings to evaluate the robustness of our vision-based policy.

Specifically, we compare two types of RGB vision policies: (a) a policy trained on real-world data, and (b) a policy trained on simulation data collected through DART. Both policies are trained on two tasks with 50 minutes of operator effort. Both policies also use a standard ACT [17] implementation at 20Hz. Real-world datasets are augmented with Gaussian blur and color-jitter. DART datasets were additionally augmented by randomizing the camera extrinsic and intrinsics, replacing the background with random textures and images from [22, 23, 24], and randomizing the lighting setting in simulation (Figure 6).

[1] Apple's Operating System for AR devices



(a) Real-world Images

(b) Simulation Renderings with Augmentations

Figure 6: Visual comparison between training images for Real and DART policies.

Inspired by [25], we evaluated policies in six diverse environments in the real world illustrated in Fig. 5. We found that our DART policies not only demonstrate zero-shot Sim2Real in the nominal setting but also significantly outperform the Real policy in many of the modified settings (Table 4). Our results highlight the benefit of scaling up simulation data versus real-world data: a single demo in simulation, which can be aggressively augmented, is more valuable for learning than that collected in real world.

Task	Pick Mug in Basket		Sorting Small Items	
Trained on data from	Real-world	DART	Real-world	DART
Lab Space	65%	**80%**	45%	**60%**
with:				
Lighting Changes	45%	**45%**	25%	**65%**
Background Changes	10%	**60%**	5%	**40%**
Cam. Pose Changes	5%	**35%**	0%	**40%**
Unseen Distractions	0%	**70%**	0%	**45%**
Communal Kitchen	0%	**50%**	0%	**35%**

Table 4. Success rates for policies trained with 50 minutes of data collection effort in the real-world v/s DART. The results highlight the robustness of policies trained with simulation data, enabled by diverse data augmentation strategies.

5 DexHub: Central Data Hub for Robot Learning on the Cloud

5.1 Purpose and Vision

To serve as a central data hub for logging *every* demonstration collected through DART, we developed **DexHub**, a cloud-hosted data repository where anyone can sign in and retrieve datasets collected by themselves and others.

In fact, to further enhance its role as an organically growing data hub, DexHub also provides an API that enables users to log all robot interaction with ease, regardless of whether they use DART or other setups. Leveraging a cloud database, user authentication system, and secure data logging, the API allows seamless integration for individuals and institutions alike to contribute and access data. The user authentication system ensures that every data contribution is properly attributed to the individual who made it, offering potential for future reward mechanisms based on contributions.

5.2 API for End-Users

DexHub's token-protected API supports multiple key functionalities ranging from downstream (downloading from the cloud) and upstream (uploading to the cloud) operations.

Downstream API Users can retrieve the data they have personally collected through DART by simply hitting `/get-my-data` with an API key retrieved from our website. This endpoint returns a list of downloadable links for every log file that users have uploaded to the cloud. The API also allows users to access the *global* dataset which includes robot data collected and contributed by other users. Access to the global dataset is available to all DexHub contributors.

Upstream API We provide an easy-to-use upstream API allowing users to contribute to DexHub without an AR device. A simple addition of `dexhub.log(obs, act)` to any Python-based robot execution script will automatically log and upload robot interactions to DexHub. All upstream contributions will be logged in the system and properly attributed to the individual who contributed, unlocking access to the global dataset via the downstream API. To retrieve the API keys and learn more about the detailed usage instructions, visit [https://dexhub.ai/].

References

[1] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[2] H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 653–660. IEEE, 2024.

[3] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[4] 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.

[5] J. A. Häusser, S. Schulz-Hardt, T. Schultze, A. Tomaschek, and A. Mojzisch. Experimental evidence for the effects of task repetitiveness on mental strain and objective work performance. *Journal of Organizational Behavior*, 35(5):705–721, 2014.

[6] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[7] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `https://proceedings.mlr.press/v9/ross10a.html`.

[8] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[9] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, S. K. S. Ghasemipour, C. Finn, and A. Wahid. Aloha unleashed: A simple recipe for robot dexterity. In *8th Annual Conference on Robot Learning*.

[10] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement–residual rl for precise visual assembly. *arXiv preprint arXiv:2407.16677*, 2024.

[11] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.

[12] M. Mosbach, K. Moraw, and S. Behnke. Accelerating interactive human-like manipulation learning with gpu-based simulation and high-quality demonstrations. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 435–441. IEEE, 2022.

[13] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation. *arXiv preprint arXiv:2403.07870*, 2024.

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

[15] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi:10.1109/IROS.2012.6386109.

[16] S. Caron, Y. De Mont-Marin, R. Budhiraja, S. H. Bang, I. Domrachev, and S. Nedelchev. Pink: Python inverse kinematics based on Pinocchio, 2024. URL `https://github.com/stephane-caron/pink`.

[17] 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.

[18] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL `http://github.com/google-deepmind/mujoco_menagerie`.

[19] URL `https://support.apple.com/en-us/118507#:~:text=Apple%20Vision%20Pro%20features%20an,feel%20contact%20on%20your%20nose.`

[20] M. Daneshmand, A. Helmi, E. Avots, F. Noroozi, F. Alisinanoglu, H. S. Arslan, J. Gorbova, R. E. Haamer, C. Ozcinar, and G. Anbarjafari. 3d scanning: A comprehensive survey. *arXiv preprint arXiv:1801.08863*, 2018.

[21] S. Hampali, T. Hodan, L. Tran, L. Ma, C. Keskin, and V. Lepetit. In-hand 3d object scanning from an rgb sequence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17079–17088, 2023.

[22] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[23] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, 128(7):1956–1981, 2020.

[24] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *2009 IEEE conference on computer vision and pattern recognition*, pages 413–420. IEEE, 2009.

[25] A. Xie, L. Lee, T. Xiao, and C. Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3153–3160. IEEE, 2024.