Axis AI Platform: A Cross-Simulator, Cross-Machine Infrastructure for Scalable Teleoperation and Generalizable Robot Learning

Axis Robotics Team

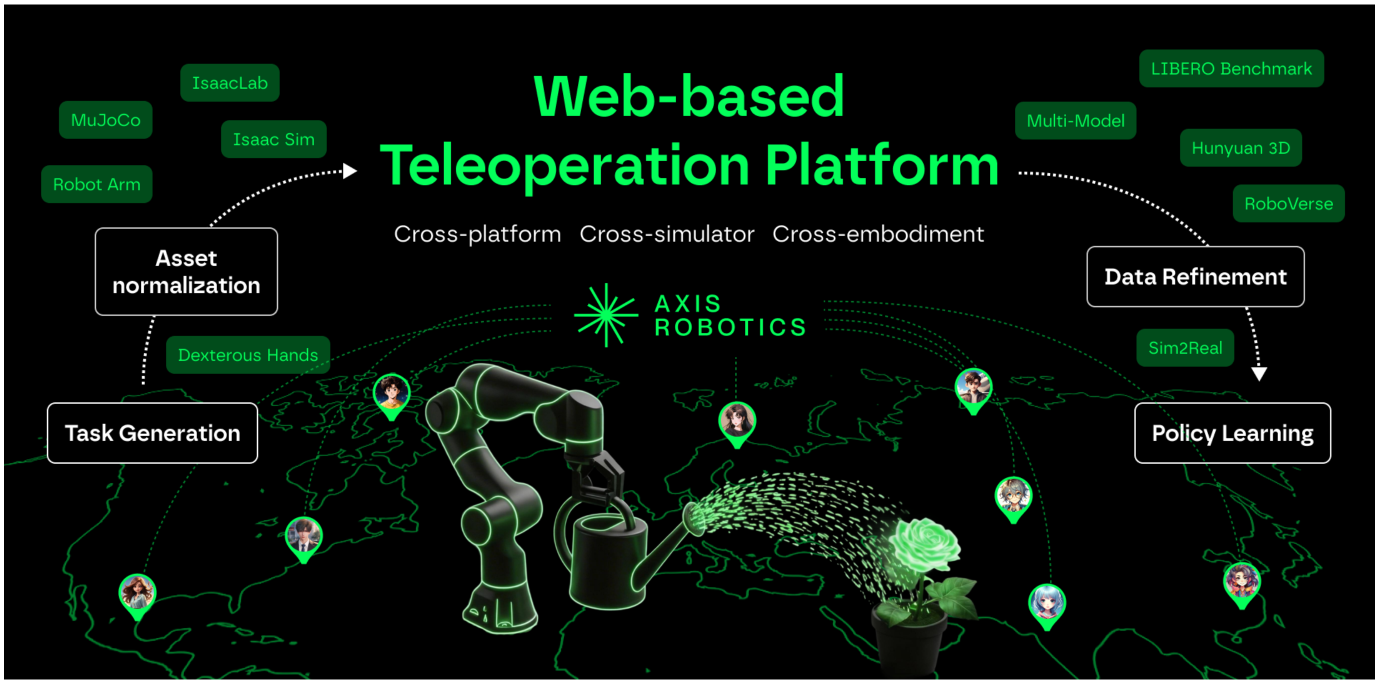


Fig. 1. Axis AI Platform: A unified infrastructure bridging web-based teleoperation, GPU-accelerated realistic augmentation, and sim-to-real deployment.

***Abstract*—Robotic learning systems increasingly rely on large-scale demonstration data and realistic simulation for training and evaluation. However, existing workflows are often fragmented across simulators, operating systems, and compute environments: lightweight simulators enable broad access for teleoperation but lack high-fidelity rendering and physics, while GPU-based simulation stacks enable realism but are expensive and restrictive to access. We present Axis AI Platform, a unified infrastructure that supports (i) multi-simulator development and execution, (ii) cross-machine orchestration spanning heterogeneous compute (web, Mac/Windows, and Linux GPU servers), and (iii) an end-to-end data pipeline from web-based teleoperation to photorealistic augmentation and downstream model training with sim-to-real deployment. In Axis AI Platform, users teleoperate robots directly in a browser via a MuJoCo WebAssembly frontend, producing demonstrations without specialized hardware or compute. Demonstrations are then uploaded in a unified trajectory format and replayed on GPU-based Linux servers to generate realistic, domain-randomized rollouts using an IsaacSim-based augmentation backend. We further provide two production pipelines: (1) data cleaning and trajectory refinement (filtering, smoothing, and segmentation), and (2) model training and sim-to-real evaluation, enabling rapid iteration from data to real-world deployment.**

***Index Terms*—robotics infrastructure, simulation, teleoperation, data collection, sim-to-real, VLA, IsaacSim, MuJoCo, WebAssembly**

I. Introduction

Learning-based robot policies increasingly benefit from large, diverse, and high-quality datasets, especially when training modern perception-action models such as vision-language-action (VLA) systems. Yet collecting robot data remains expensive: many pipelines require specialized workstations, complex simulator setups, or lab-only teleoperation rigs. Meanwhile, high-fidelity simulation for realism (photorealistic rendering, accurate physics, material properties) typically demands GPU-based Linux machines and heavyweight stacks, limiting who can contribute and how fast iteration can happen.

This paper introduces Axis AI Platform, an end-to-end infrastructure designed around a simple principle: *make data collection universally accessible, and make realism scalable on-demand.* Axis AI Platformachieves this by combining:

- **Accessible teleoperation on commodity devices:** a browser-based MuJoCo WASM interface that runs on web/Mac/Windows without requiring GPU or special software installation.
- **Unified cross-simulator abstraction:** a common API layer and data format that supports multiple simulators for complementary strengths (lightweight control iteration vs. high-fidelity augmentation).
- **Cross-machine orchestration:** seamless movement of trajectories from web clients to GPU-based Linux servers for large-scale augmentation and training.
- **Production data pipelines:** trajectory cleaning/smoothing and a training-to-deployment loop with sim-to-real validation.

We summarize our contributions as follows:

1) **A multi-simulator, cross-machine infrastructure** that supports MuJoCo on web/Mac/Windows and IsaacLab/IsaacSim on Linux GPU servers within one unified workflow.
2) **A web-based teleoperation system** that enables low-friction demonstration collection at scale, without specialized compute.
3) **A realistic augmentation backend** that replays demonstrations in GPU-based simulation to produce photorealistic, domain-randomized training data suitable for VLA / robot policy learning.
4) **Two robust pipelines:** (i) data cleaning and trajectory refinement; and (ii) model training and sim-to-real evaluation with real-world deployment.

II. Related Work

A. Robotic Simulation Ecosystems

Robotic simulation tools span a tradeoff between accessibility and fidelity. Lightweight simulators such as MuJoCo[1], Gazebo[2], CoppeliaSim[3], Drake[4], and PyBullet[5] are widely used for rapid iteration, interactive control, and deployment on commodity hardware. While they provide user-friendly interfaces and sufficiently accurate physics for prototyping and control, they are often less suited to large-scale parallel data generation and high-fidelity rendering. In contrast, GPU-oriented platforms such as Isaac Gym[6], MuJoCo MJX[7], Brax[8], and Genesis[9] emphasize parallel simulation and training throughput, while systems such as Isaac Sim[10] and SAPIEN[11] further support photorealistic rendering and complex scene construction. However, these platforms often require specialized expertise and powerful hardware, which limits accessibility for broad human demonstration collection. In practice, lightweight simulators are commonly used for interactive teleoperation and low-cost data collection [12]–[14], whereas high-performance platforms are more often used for replay and augmentation at scale[15, 16]. Despite these complementary roles, few systems connect both within a unified workflow. Axis AI Platform addresses this gap by pairing a low threshold data collection frontend with a high-capability backend for replay, augmentation, and downstream learning, thereby enabling a scalable and practical pipeline for high-quality robot data generation.

B. Teleoperation and Demonstration Collection

Demonstration collection through teleoperation has historically relied on diverse hardware interfaces. Early methods often used kinesthetic teaching[17], which can produce high-quality data safely but is labor-intensive and difficult to scale across robots. Later work adopted leader–follower systems such as GELLO[18], ALOHA[19], and U-ARM[20], which offer more precise and effective control for dexterous manipulation but typically require dedicated hardware and careful calibration. Other approaches use more flexible external input devices, including SpaceMouse[14], joysticks[21], gamepads[22], VR controllers[23], and AR[24] interfaces. These interfaces are generally more accessible, but often sacrifice precision and responsiveness. More recently, wearable teleoperation systems such as DexCap[25] and DexUMI[26] have enabled richer human motion capture, though they also introduce additional sensing complexity and retargeting challenges.

Compared with hardware-based teleoperation, web-based interfaces further reduce onboarding cost and broaden participation. Prior web-based teleoperation systems have largely focused on remote access to physical robots[27, 28], where data collection is still constrained by robot availability, hardware maintenance, and deployment overhead. By contrast, Axis AI Platform uses the browser as a lightweight frontend for simulation-based teleoperation, enabling more convenient, lower-cost, and higher-throughput demonstration collection without direct access to physical hardware. It further strengthens this workflow by linking collected trajectories to downstream realistic simulation augmentation. In addition, the platform incorporates gamified interaction to encourage participation during data collection.

C. Sim-to-Real and Data Augmentation

To reduce the sim-to-real gap and improve the robustness of robot learning, prior work has widely adopted techniques such as domain randomization [29, 30], physics parameter variation [30, 31], and photorealistic rendering [16]. Beyond improving transfer realism, recent studies have also used high-fidelity simulators as augmentation backends for trajectory replay[15], synthetic view generation[32], and data expansion[33], providing richer supervision and more diverse training observations without requiring additional human demonstrations. The techniques above are proved to be particularly valuable in data-driven robot learning, where scalability and visual diversity are critical. Axis AI Platform builds on this direction by treating sim-to-real transfer and data augmentation as first-class pipeline stages: demonstrations collected in a lightweight MuJoCo frontend are replayed in an IsaacSim-based backend to generate realistic, domain-randomized, and multi-view training data for downstream learning and sim-to-real evaluation.

III. System Overview

As illustrated in Figure 2, Axis AI Platform provides an end-to-end data pipeline. The workflow begins with automated

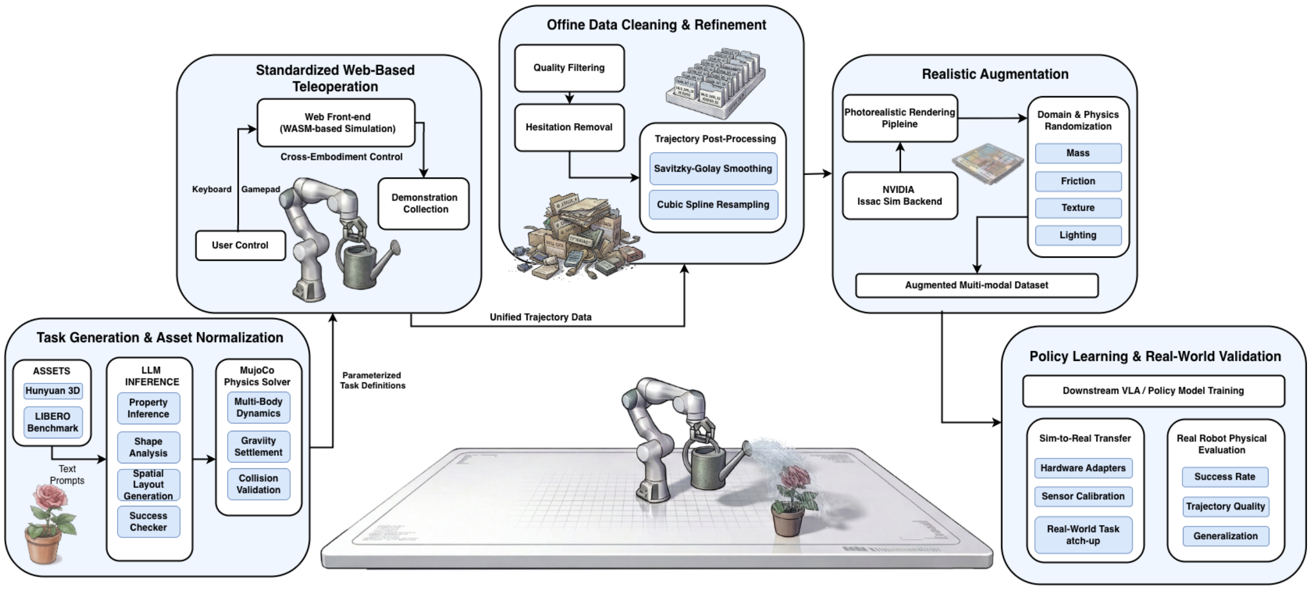


Fig. 2. **Overview of the Axis AI PlatformPipeline.** The infrastructure consists of five interconnected modules: **(1) Task Generation & Asset Normalization:** LLMs and MuJoCo physics solvers automatically generate and validate parameterized tasks. **(2) Standardized Web-Based Teleoperation:** A WASM-based MuJoCo frontend enables accessible, cross-embodiment demonstration collection via commodity inputs (e.g., keyboard/gamepad). **(3) Offline Data Cleaning:** Unified trajectories undergo quality filtering and smoothing. **(4) Realistic Augmentation:** Cleaned trajectories are routed to an NVIDIA Isaac Sim backend for photorealistic rendering and extensive domain/physics randomization. **(5) Policy Learning & Validation:** The resulting augmented multi-modal dataset is used to train downstream VLA models, followed by rigorous sim-to-real transfer and real-world physical evaluation.

task generation, where LLMs and physics solvers construct parameterized environments. These environments are then deployed to a web-based teleoperation frontend. This design enables distributed users to collect demonstrations using commodity devices. The recorded data is then automatically routed to GPU servers for cleaning, smoothing, and high-fidelity replay.

To seamlessly bridge lightweight data collection with heavy-duty simulation, Axis AI Platform is driven by five core requirements:

G1: Accessibility. Data collection should work on commodity devices (web, Mac/Windows), with minimal installation and no GPU requirement.

G2: Simulator diversity without fragmentation. Users should be able to use different simulators for different stages (teleop vs. augmentation) while maintaining consistent APIs and data semantics.

G3: Cross-machine scalability. The infrastructure should support heterogeneous compute: interactive clients and scalable GPU servers, with reliable upload, storage, and replay.

G4: Data quality and usability. Collected demonstrations must be cleaned, smoothed, and structured into training-ready datasets suitable for VLA and robot policy learning.

G5: Tight training-to-deployment loop. The system should make it easy to train on simulation data and evaluate/deploy in the real world to test sim-to-real transfer.

IV. Architecture

A. Unified Abstraction Layer

Axis AI Platformprovides a unified interface over simulators. At a high level, each environment implements:

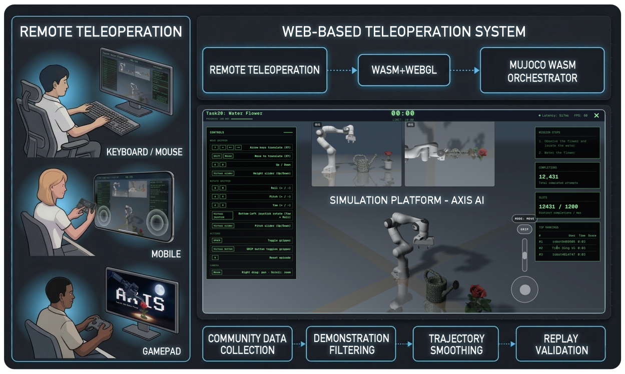


Fig. 3. Web-based Teleoperation System Overview

- **State interface:** observation retrieval, including proprioception and optional camera streams.
- **Action interface:** low-level control commands (joint targets, end-effector deltas, etc.).
- **Reset/step semantics:** deterministic stepping and standardized episode boundaries.
- **Asset specification:** robot and scene assets with consistent naming and coordinate conventions.

This abstraction ensures that a trajectory recorded in the teleop frontend can be replayed and augmented in a different simulator backend without re-implementing task logic.

B. Cross-Machine Orchestration

The system separates interactive clients (web-based MuJoCo WASM) from compute backends (Linux GPU servers). A lightweight uploader transmits demonstrations to a central storage service where they can be versioned, indexed, and retrieved for replay.

C. *Unified Trajectory Format*

We use a unified trajectory format designed for replayability and learning. Each trajectory contains:

- **Metadata:** environment ID, robot embodiment, task name, simulator version, timestamps.
- **Time-series signals:** actions, states, observations, and optional camera data.
- **Episode structure:** segmentation markers and success/failure labels when available.

V. Web-Based Teleoperation via MuJoCo WASM

A. *Motivation*

Teleoperation-based data collection is often constrained by limited access to dedicated lab hardware and by the engineering overhead of maintaining complex, specific setups. These constraints slow down iteration, reduce the scale and diversity of demonstrations, and make it difficult to involve external contributors.

In addition, traditional teleoperation pipelines typically rely on specialized simulators and input devices, which introduces deployment complexity due to installation overhead, operating-system compatibility issues, device driver dependencies, and also limits reproducibility across environments.

To address these bottlenecks, we run the simulator entirely in the browser via a WebAssembly (WASM) MuJoCo runtime. This design allows demonstrations to be collected on desktops using standard input devices (keyboard/mouse or gamepad). As a result, Axis AI Platformlowers the barrier to entry for teleoperation, enabling faster data collection at scale and improving accessibility for distributed users.

B. *Frontend Design and Implementation*

Built on the Next.js framework, the teleoperation frontend is designed to support an end-to-end workflow from task selection to high-fidelity demonstration as shown in Figure 3. The web interface is organized into three modules: a landing page, a task-selection hub, and the core teleoperation dashboard. The dashboard serves as a heads-up display (HUD), integrating interactive 3D visualization, camera control, and real-time status monitoring. To maintain responsive control in the browser, we manage the MuJoCo WASM runtime using an orchestrator pattern rather than a monolithic control loop. The system provides:

1) Interactive visualization and camera controls: The interface provides full-screen MuJoCo (WASM+WebGL) rendering with a floating, semi-transparent UI overlay with background blur. This design ensures the readability of all UI elements without completely obscuring the robotic arm's movements. The workspace is organized into functional zones, including a top global status bar for real-time metrics (time, FPS) and a structured controls panel.To facilitate quick user onboarding, this panel logically groups complex key combinations into gripper movement, rotation, actions, and intuitive camera controls , which enable users to easily orbit, pan, and zoom the 3D viewport via mouse.

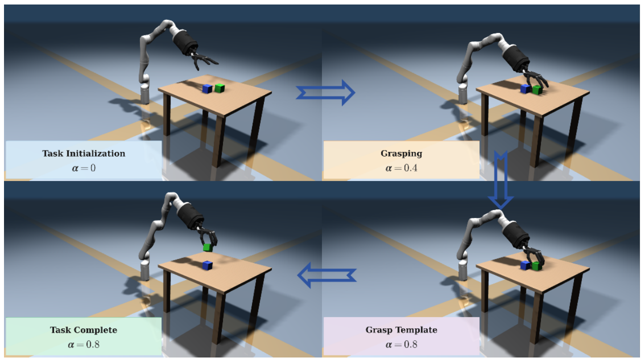


Fig. 4. Visulization of Grasping Pipeline

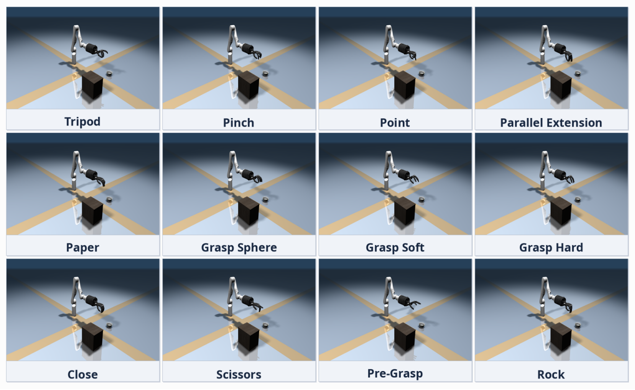


Fig. 5. Visulization of Diverse Grasping Templates

2) Versatile Input Mapping: Teleoperation controller integrates diverse input modalities, supporting both precise discrete step control via a keyboard and continuous trajectory control via a virtual joystick or gamepad. User commands are mapped into 6-DoF Cartesian targets for the end-effector and seamlessly processed by an embedded Inverse Kinematics solver to compute joint-level commands for the robot.

3) low-latency stepping and logging: To maintain a high control frequency and reliable data collection, we separate the MuJoCo physics loop and Three.js rendering from the main React UI thread. While the dedicated broadcaster sends occasional status updates to the UI, the trajectory manager runs synchronously within the high-speed physics loop. By buffering state samples directly in memor, the system avoids the overhead of frequent UI re-renders and potential lag during active teleoperation.

4) Automated trajectory packaging: Once the embedded checker runtime verifies task completion, the trajectory manager automatically compiles the buffered samples and task metadata into a serialized JSON file, ensuring data standardization.

C. *Multi-Embodiment Keyboard Control via Interpolation Mapping*

To support diverse end-effectors, ranging from simple parallel-jaw grippers to complex dexterous hands, Axis AI Platform employs a unified teleoperation interface based on interpolation mapping. This strategy, shown in Figure 4 abstracts the high-dimensional joint control into a single scalar parameter, enabling intuitive, cross-embodiment operation

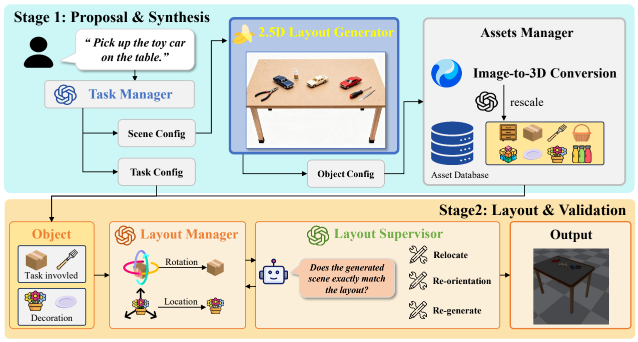


Fig. 6. Overview of task generation and assets normalization

without exposing the operator to the underlying degrees of freedom (DoFs).

1) Grasp Templates: We predefine a library of *grasp templates* for various manipulation tasks as shown in Figure 5. Each template specifies a target joint-angle vector $\mathbf{q}_{\text{tmpl}} \in \mathbb{R}^n$ corresponding to a specific grasp category, such as power, precision, or functional grasps (e.g., key-turning). Let $\mathbf{q}_{\text{rest}} \in \mathbb{R}^n$ represent the fully open rest pose of the hand, where n denotes the number of actuated joints.

2) Interpolation Control: The instantaneous configuration of the hand is computed by linearly interpolating between the rest pose and the active template:

$$\mathbf{q}(t) = \mathbf{q}_{\text{rest}} + \alpha(t)\left(\mathbf{q}_{\text{tmpl}} - \mathbf{q}_{\text{rest}}\right), \tag{1}$$

where $\alpha(t) \in [0,\, 1]$ is a scalar coefficient. The operator adjusts α via two keyboard keys that increment or decrement its value at a fixed rate. Here, $\alpha = 0$ corresponds to the fully open state, while $\alpha = 1$ reaches the exact template pose.

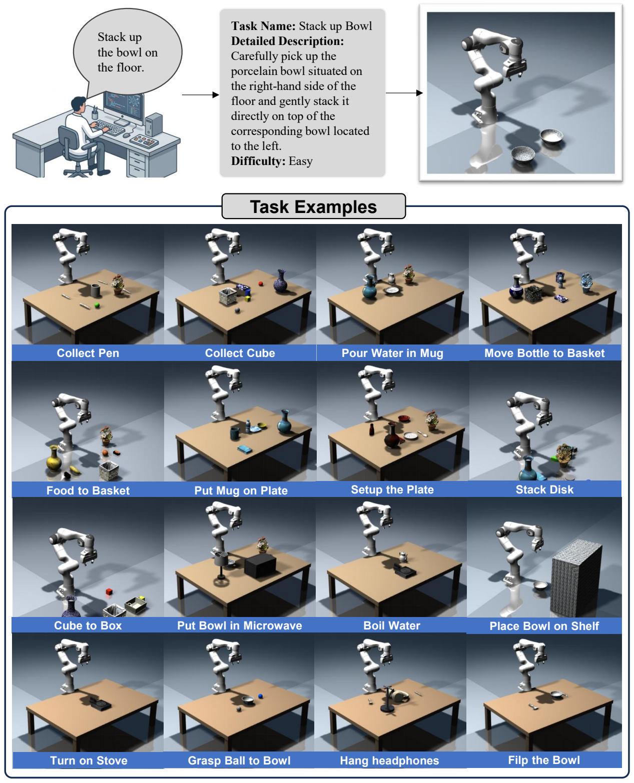
This formulation effectively collapses the high-dimensional joint space (often $n > 16$ for dexterous hands) into a single control axis. Consequently, operators can execute sophisticated grasps using only two keys. Furthermore, since the interpolation logic is morphology-agnostic, new robot embodiments can be integrated by simply defining their $\mathbf{q}_{\text{rest}}$ and a task-specific $\mathbf{q}_{\text{tmpl}}$ library, requiring no modifications to the control core.

VI. Data Pipelines

A. Pipeline 1: Task Generation and Deployment

Before collecting demonstrations, Axis AI Platform constructs task specifications and simulation-ready environments through a *TaskGen* stage. As illustrated in Fig. 6, given a high-level instruction, the pipeline automatically synthesizes a structured scene configuration, instantiates the required assets, and validates the resulting layout before deployment. This process enables scalable generation of diverse manipulation tasks while maintaining consistency between language instructions and simulated environments.

1) Task generation: Given an instruction (e.g., *"pick up the toy car on the table"*), a *Task Manager* parses the language input into structured task, scene, and object configurations. The *task configuration* defines the manipulation goal and required object interactions, while the *scene configuration* specifies the workspace context such as the supporting surface and the level



Task Name: Stack up Bowl
Detailed Description: Carefully pick up the porcelain bowl situated on the right-hand side of the floor and gently stack it directly on top of the corresponding bowl located to the left.
Difficulty: Easy

Fig. 7. Sample tasks generated by the *TaskGen* module.

of clutter. The *object configuration* determines which objects appear in the scene and their semantic roles. Each task is also associated with a *difficulty level* ranging from 1 to 5, which controls the number of objects, spatial constraints, and overall scene complexity.

To populate scenes with diverse objects, Axis AI Platform maintains an *Assets Manager* backed by an asset database. Assets can be retrieved from existing repositories or generated via an image-to-3D conversion pipeline. Because generated meshes often exhibit inconsistent scales, we apply a normalization step that rescales objects to plausible physical dimensions before adding them to the database.

Conditioned on the task specification, selected assets, and difficulty level, a *2.5D Layout Generator* produces an initial scene proposal describing which objects should appear on the workspace and their approximate spatial arrangement. Objects are categorized as *task-involved objects*, which are required for task completion, and optional *decorative objects* that increase visual diversity.

The proposed layout is instantiated into a full 3D environment by a *Layout Manager*, which assigns each object a precise pose including location and orientation. A *Layout Supervisor* then verifies whether the generated scene faithfully matches the intended layout. When inconsistencies are detected, the system iteratively corrects the scene through relocation, re-orientation, or partial regeneration until the layout satisfies the specification.

2) Scenario sampling and deployment: After a valid task layout is constructed, Axis AI Platform defines distributions

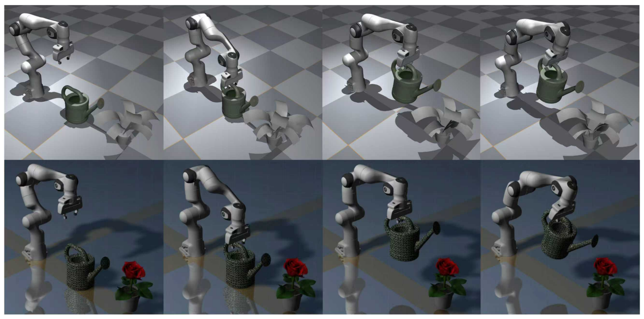


Fig. 8. Visulization of Web-teleoperated and simulation-replayed trajectory

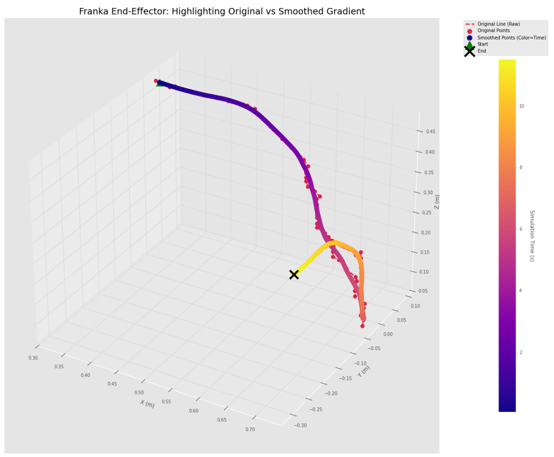


Fig. 9. Comparison betweeen Smoothed and Original End-efffector Trajectory

over initial conditions to generate diverse task instances. These distributions include variations in object poses, clutter configurations, and camera viewpoints, enabling large-scale scenario sampling while preserving the core task semantics.

Each generated task instance is augmented with a task-specific *success checker* that encodes completion conditions based on geometric constraints and object interactions. The tasks are then deployed to a browser-based teleoperation interface built on MuJoCo compiled to WebAssembly (MuJoCo-WASM), allowing contributors to access the simulation environment directly from a web browser.

To facilitate large-scale data collection and cross-platform compatibility, Axis AI Platform exposes a unified observation and action interface shared across tasks and simulators. This unified interface enables demonstrations collected through browser teleoperation to be seamlessly replayed, evaluated, and used for downstream policy learning.

Overall, *TaskGen* produces three components: (i) a parameterized task definition describing the scene configuration and object layout, (ii) a unified observation and action interface for consistent execution across environments, and (iii) a task-specific success checker for automatic evaluation. Examples of automatically generated tasks are shown in Fig. 7.

B. Pipeline 2: Data Cleaning and Trajectory Refinement

Teleoperation data can contain artifacts: jitter, stalls, unintended oscillations, or uninformative motion. Axis AI Platform provides a cleaning pipeline with two goals: filter unusable episodes and refine borderline episodes.

Algorithm 1 Trajectory cleaning (high-level summary).

```
Require: Raw trajectory $S$, static threshold $\epsilon$, target dt $\Delta t$,
    SG params $W, P$
Ensure: Smoothed and resampled trajectory $S_{final}$
    Step 1: Segmentation & Re-timing
    $S_{clean} \leftarrow \text{FilterStaticFrames}(S, \epsilon)$
    $T_{clean} \leftarrow \text{RebuildUniformTime}(S_{clean}, \text{original\_dt})$
    Step 2: Smoothing
    $Q_{smooth} \leftarrow \text{SavitzkyGolay}(S_{clean}.\text{joints}, W, P)$
    {Note: Gripper joints are excluded from SG smoothing}
    Step 3: Resampling
    $T_{new} \leftarrow \text{Arange}(T_{clean}[0], T_{clean}[-1], \Delta t)$
    $Q_{final} \leftarrow \text{CubicSpline}(T_{clean}, Q_{smooth})(T_{new})$
    Step 4: Finalization
    $S_{final} \leftarrow \text{InterpolateExtraStates}(T_{new}, Q_{final}, S_{clean}.\text{extras})$
    return $S_{final}$
```

1) Filtering: We remove trajectories that violate basic validity checks (e.g., extreme discontinuities, corrupted frames, invalid state transitions), or that fail task-specific constraints.Specifically, to address extreme discontinuities, we compute the state difference between adjacent frames and ensure that this delta does not exceed predefined, physically plausible thresholds. Furthermore, to account for corrupted frames and invalid state transitions, we filter out trajectories that lack required data keys or contain anomalous numerical values. For strict rigor, although the trajectories dumped from the source website are accompanied by metadata indicating task success, we manually re-verify them within RoboVerse to validate their success flags in the simulation environment.

2) Smoothing and segmentation: We smooth action sequences and optionally re-time trajectories to remove teleop jitter and stuck segments. Algorithm 1 outlines the procedure.We first process the raw trajectories downloaded from the Axis AI Platform by evaluating the action sequences at each frame to identify and remove static segments. Specifically, frames are classified as static and subsequently discarded if the absolute variation across all joints falls below a threshold of 5×10^{-3}. This step effectively eliminates artificial pauses and idle frames caused by human operator hesitation. Following the removal of these static frames, we apply a Savitzky-Golay filter (using a window size of 15 and a polynomial order of 3) to smooth the robot's action trajectories as shown in Figure 11. Because the original teleoperation data recorded via the web interface has a low sampling rate of only 6–8 Hz, which is insufficient for downstream high-frequency robot control and policy training, we subsequently upsample the data using cubic spline interpolation to reach the required target frequency of 20 Hz. For non-robot data, such as object states, we bypass the filtering process and exclusively apply cubic spline interpolation to temporally align them with the robot's control steps.

TABLE I

QUANTITATIVE EVALUATION OF TRAJECTORY OPTIMIZATION ACROSS 16 LIBERO TASKS. SMOOTHNESS (MAX ACCELERATION AND MAX JERK) IS COMPARED BEFORE AND AFTER THE PROPOSED PROCESSING.

LIBERO Task	**Smoothness**				**Pos. Dev. (m)**	**Removed Ratio**
	Mean Acc.		**Mean Jerk**			
	Before	After	Before	After		
Task 1: Place Black Bowl on Top of cabinet	0.2458	0.0738	2.6425	1.3363	0.0235	6.46%
Task 2: Place Rear Butter in Cabinet Top Drawer and Close It	0.3067	0.0943	3.7395	1.6208	0.0654	5.73%
Task 3: Place the black bowl on the plate	0.3559	0.1011	5.1358	2.3634	0.0201	7.17%
Task 4: Place the black bowl on top of the cabinet	0.3225	0.0859	4.6724	2.0785	0.0142	3.77%
Task 5: Place the frying pan on the stove	0.1832	0.1055	2.4771	1.1374	0.0106	1.24%
Task 6: Place the moka pot on the stove	0.1819	0.1090	2.5239	1.6784	0.0064	1.94%
Task 7: Turn on the stove	0.2689	0.1295	3.8291	2.3017	0.0142	3.81%
Task 8: Close Cabinet Bottom Drawer	0.1574	0.0741	1.9118	0.6659	0.0057	4.59%
Task 9: Place the black bowl into the cabinet’s bottom drawer	0.1551	0.0913	1.8665	1.3193	0.0161	2.03%
Task 10: Place Wine Bottle on Wine Rack	0.1841	0.1114	2.0298	1.0022	0.0625	4.86%
Task 11: Close Cabinet Top Drawer	0.1224	0.0683	1.5424	0.5822	0.0053	2.62%
Task 12: Place the black bowl into the cabinet’s top drawer	0.2109	0.1418	2.6997	2.1633	0.0961	11.4%
Task 13: Place Black Bowl on Plate	0.1682	0.0816	2.2262	1.1552	0.2511	2.14%
Task 14: Place the black bowl on top of the cabinet	0.1375	0.0736	1.8499	0.9924	0.0108	6.73%
Task 15: Place the right moka pot on the stove	0.1890	0.1130	2.5768	1.4929	0.0134	3.19%
Task 16: Turn off the stove	0.2149	0.1213	2.8268	1.5823	0.0219	8.93%
Average	**0.2128**	**0.0985**	**2.7844**	**1.4670**	**0.0398**	**4.79%**

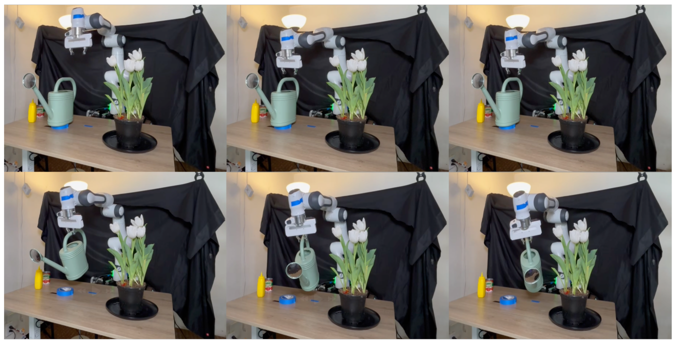


Fig. 10. Real World Experiment Setup.

VII. MODEL TRAINING AND SYSTEM EVALUATION

A. Experimental Setup

We evaluate Axis AI Platform under a unified experimental setup designed to test both systems-level benefits (accessibility and throughput) and downstream learning benefits (policy quality and sim-to-real transfer).

1) Robots and tasks: Our physical experiments are deployed on a Franka Research 3 (FR3) robotic arm equipped with a standard parallel-jaw gripper. We focus on a diverse set of representative tabletop manipulation tasks generated via our pipeline, including *pick-and-place*, *drawer opening/closing*, *pouring*, and *button pressing*. These tasks systematically evaluate policy robustness, spanning both rigid and articulated object interactions that require varying degrees of precision and contact control. Furthermore, to specifically benchmark our offline data cleaning module, we incorporate a suite of LIBERO-style tasks (e.g., spatial object placement and stove-switch manipulation).

2) Data collection protocol: Demonstrations are initially collected via the browser-based MuJoCo-WASM teleoperation frontend and stored in a unified trajectory format encompassing robot states, actions, and task metadata. This standardized representation ensures that the raw data can be seamlessly processed by our proposed cleaning pipeline and replayed across different simulators. For policy fine-tuning, we construct task-specific datasets consisting of approximately 40 to 60 demonstrations.

3) Policy training and System validation: To evaluate the effectiveness of our end-to-end system, we utilize the OpenVLA architecture to validate the complete sim-to-real pipeline. The training dataset consists of data collected via our web-based teleoperation frontend and augmented with a small set of real-world human demonstrations to bridge the physical domain gap. The ratio of web-collected simulation data to real-world demonstrations is maintained at approximately three to one. We fine-tune the pre-trained OpenVLA model on this mixed dataset and deploy the resulting policy directly on the physical FR3 robotic arm. The successful deployment, as shown in Fig. 10, of the fine-tuned policy on the real robot directly validates the complete architecture of the Axis AI Platform. This physical evaluation proves that the infrastructure effectively unifies automated task generation, accessible web teleoperation, offline trajectory refinement, and downstream policy learning. Consequently, the experiment confirms that the platform serves as a highly practical and scalable solution for end-to-end sim-to-real transfer.

B. Accessibility and Throughput

Since Axis AI Platform offloads realistic augmentation to Linux GPU backends while keeping the teleoperation frontend in a lightweight, browser-based MuJoCo environment, it substantially lowers onboarding costs. Consequently, Axis AI Platform achieves faster setup times, a higher yield of valid demonstrations per hour, and significantly fewer validation

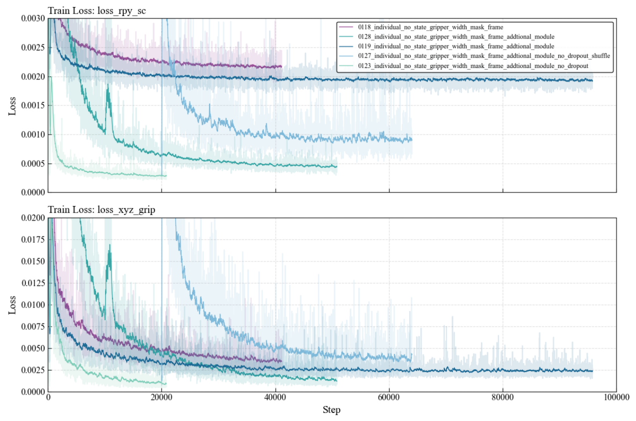


Fig. 11. Training loss curves during the fine-tuning of the OpenVLA policy.

failures compared to both baselines. This streamlined workflow proves especially critical for scaling data collection across distributed contributors using heterogeneous hardware.

C. Data Quality Metrics

We evaluate data quality along four dimensions: trajectory validity, temporal smoothness, coverage of task conditions, and visual diversity introduced by realistic replay. These metrics quantify whether Axis AI Platform produces demonstrations that are more suitable for downstream learning than raw teleoperation data alone.

1) Trajectory validity and filtering: We begin by quantifying the impact of our automated cleaning pipeline, which strictly filters out trajectories exhibiting corrupted data, extreme state discontinuities, implausible transitions, or task-specific failures. Across the evaluated task suite, this rigorous validation process discards the unqualified raw demonstrations, preventing a non-negligible portion of flawed teleoperation data from corrupting downstream policy learning.

2) Temporal smoothness: For the successfully retained trajectories, we compute mean acceleration and jerk to capture high-frequency human artifacts, such as abrupt corrections and jitter. As detailed in Table I, our kinematic refinement improves trajectory smoothness across the entire 16-task LIBERO suite. Specifically, the smoothing process halves the average mean acceleration (from 0.2128 to 0.0985) and reduces mean jerk (from 2.7844 to 1.4670). These concrete improvements confirm that the refined demonstrations are more stable, making them far better suited for downstream policy training and sim-to-real deployment.

3) Coverage and diversity: In addition to smoothness, we evaluate dataset diversity summarized by initial object pose coverage, camera configuration, task instance, and end-effector start-state variation, because a clean but overly narrow dataset limits generalization. To expand the support of the training distribution, our replay pipeline resamples task instances and rendering conditions from the same underlying demonstrations.

Overall, Axis AI Platform improves data quality at both the trajectory and observation levels. Automated filtering removes invalid samples, smoothing reduces motion artifacts, and IsaacSim-based augmentation expands visual diversity, providing a stronger foundation for downstream policy learning and sim-to-real transfer.

VIII. Discussion and Limitations

While Axis AI Platform unifies trajectories through a common abstraction layer, cross-simulator replay is not perfectly lossless because of discrepancies in contacts, actuator models, solver settings, and coordinate conventions. Although we mitigate these deviations by aligning frames and standardizing conventions, replay remains a scalable augmentation mechanism instead of an exact reproduction. Furthermore, teleoperation data often lacks dense supervision such as rewards or contact labels. While our use of task-specific success checkers provides scalable weak supervision, it may overlook partial successes or intermediate events. Finally, although the platform provides infrastructure-level compatibility for multiple robots, it does not inherently solve cross-embodiment transfer due to substantial differences in kinematics, dexterity, and control properties. Future work could address these gaps through stronger tracking controllers, learned event detectors, and embodiment-aware retargeting or canonical action spaces.

IX. Conclusion

We present Axis AI Platform, a cross-simulator, cross-machine infrastructure that makes demonstration collection broadly accessible through web-based MuJoCo teleoperation, and scales realism through GPU-based IsaacSim augmentation. By unifying trajectory formats and providing production pipelines for data cleaning and model training with sim-to-real evaluation, Axis AI Platformaccelerates iteration from data collection to real-world deployment. Future work includes expanding task libraries, improving cross-simulator replay robustness, and supporting richer labels for VLA-scale training.

References

[1] 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*. IEEE, 2012, pp. 5026–5033.

[2] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep. 2004, pp. 2149–2154.

[3] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, available: www.coppeliarobotics.com.

[4] Robot Locomotion Group, "Drake: Model-based design and verification for robotics," https://drake.mit.edu/, 2026, accessed: 2026-03-06.

[5] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016, accessed: 2026-03-06.

[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," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021, neurIPS 2021 Datasets and Benchmarks Track.

[7] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel, "Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer," *arXiv preprint arXiv:2502.08844*, 2025.

[8] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, “Brax: A differentiable physics engine for large scale rigid body simulation,” in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021, neurIPS 2021 Datasets and Benchmarks Track.

[9] Genesis Authors, “Genesis: A universal and generative physics engine for robotics and beyond,” https://github.com/Genesis-Embodied-AI/Genesis, 2024, software project, accessed 2026-03-06.

[10] NVIDIA, “Isaac sim,” https://developer.nvidia.com/isaac/sim, 2025, robotics simulator, accessed 2026-03-06.

[11] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, “Sapien: A simulated part-based interactive environment,” *arXiv preprint arXiv:2003.08515*, 2020.

[12] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu, “robosuite: A modular simulation framework and benchmark for robot learning,” *arXiv preprint arXiv:2009.12293*, 2020.

[13] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei, “Roboturk: A crowdsourcing platform for robotic skill learning through imitation,” in *Proceedings of the 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87, 2018, pp. 879–893.

[14] 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*, 2024.

[15] Y. Fang, Y. Yang, X. Zhu, K. Zheng, G. Bertasius, D. Szafir, and M. Ding, “Rebot: Scaling robot learning with real-to-sim-to-real robotic video synthesis,” in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[16] J. Fan, Z. Zhao, Y. Zhang, C. Chen, P. Wang, H. Zhang, and Z. Cheng, “Robopaint: From human demonstration to any robot and any view,” *arXiv preprint arXiv:2602.05325*, 2026.

[17] P. Kormushev, S. Calinon, and D. G. Caldwell, “Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input,” *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[18] 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.

[19] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” in *Robotics: Science and Systems*, 2023.

[20] Y. Zou, Z. Zhou, C. Shi, Z. Ye, J. Huang, Y. Ding, and B. Zhao, “U-arm: Ultra low-cost general teleoperation interface for robot manipulation,” *arXiv preprint arXiv:2509.02437*, 2025.

[21] D. Honerkamp, H. Mahesheka, J. O. von Hartz, T. Welschehold, and A. Valada, “Zero-cost whole-body teleoperation for mobile manipulation,” *IEEE Robotics and Automation Letters*, 2025.

[22] C. Zhou, C. Peers, Y. Wan, R. Richardson, and D. Kanoulas, “Teleman: Teleoperation for legged robot loco-manipulation using wearable imu-based motion capture,” *arXiv preprint arXiv:2209.10314*, 2022.

[23] A. George, A. Bartsch, and A. B. Farimani, “Openvr: Teleoperation for manipulation,” *SoftwareX*, vol. 29, p. 102054, 2025.

[24] J. Wang, C.-C. Chang, J. Duan, D. Fox, and R. Krishna, “Eve: Enabling anyone to train robot using augmented reality,” *arXiv preprint arXiv:2404.06089*, 2024.

[25] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu, “Dexcap: Scalable and portable mocap data collection system for dexterous manipulation,” *arXiv preprint arXiv:2403.07788*, 2024.

[26] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso, and S. Song, “Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation,” *arXiv preprint arXiv:2505.21864*, 2025.

[27] S. Mirchandani, M. Tang, J. Duan, J. I. Hamid, M. Cho, and D. Sadigh, “Robocade: Gamifying robot data collection,” *arXiv preprint arXiv:2512.21235*, 2025.

[28] C.-L. Fok, F. Sun, M. Mangum, A. K. Mok, B. He, and L. Sentis, “Web-based teleoperation of a humanoid robot,” *arXiv preprint arXiv:1607.05402*, 2016.

[29] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” *arXiv preprint arXiv:1703.06907*, 2017.

[30] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, “Learning dexterous in-hand manipulation,” *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[31] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[32] R. Singh, J. Liu, K. V. Wyk, Y.-W. Chao, J.-F. Lafleche, F. Shkurti, N. D. Ratliff, and A. Handa, “Synthetica: Large scale synthetic data generation for robot perception,” in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 7810–7817.

[33] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, “Mimicgen: A data generation system for scalable robot learning using human demonstrations,” in *7th Annual Conference on Robot Learning*, 2023.