

Axis Robotics: The Compounding Data Engine Accelerating Physical AI

Scaling Physical AI for the Real World, Contributed by Everyone

Axis Robotics Team · October 2026 · axisrobotics.ai

Abstract

Physical AI is not bottlenecked by model capacity, compute, or hardware; it is bottlenecked by data. The total volume of robot training data available today is orders of magnitude below what general-purpose manipulation requires, and the structural forces that keep it scarce—hardware-bound collection, the difficulty of manufacturing diversity, and the inability of demonstration-only policies to see their own failures—are not addressed by a better architecture. Axis Robotics is built around this constraint. It is a *compounding data engine*: a closed loop in which each contribution improves the next policy, and each improved policy determines what data to collect next, so data ceases to be a one-time cost and becomes an appreciating asset. This whitepaper describes the full system—the flywheel that makes data compound, the three cornerstones and three products that expose it (a browser simulation platform, a mobile ego-centric app, and the data-to-model pipeline), and the four-layer architecture beneath them, from a globally accessible data interface up to policy deployment. Two design choices are central: browser-native collection that runs on commodity devices with no hardware, and human-gated correction on the policy’s own rollouts, made scalable by executing the policy client-side. Continual pretraining of Physical Intelligence’s $\pi_{0.5}$ on the Axis dataset raises LIBERO-Plus success rate from **83.9% to 88.8%** and beats a volume-matched RoboCasa365 control by **31.3 points**, with gains scaling in the amount of Axis data and concentrated on the perturbation axes the engine is built to cover. We close with the evaluation and provenance layer that turns crowd work into trusted data with an auditable contribution record, and the roadmap from today’s deployed platform toward foundation-model-scale data. A final section sets out the \$AXIS tokenomics: the allocation and release schedule of a fixed **1,000,000,000 \$AXIS** supply, **61.08%** of which is allocated to the network, the two-season Community Contributor Airdrop, and the token’s utility.

The peer-facing paper [1] covers the data engine, benchmark, and experimental protocol in full detail; this document is the system-level overview.

1 Introduction

Large language models were built on a foundation that already existed: decades of internet text, trillions of tokens, spanning virtually every domain of human knowledge. With that

corpus in place, progress became largely a matter of scaling compute and refining architectures. Physical AI has no such foundation. The physical world has no search-engine index, and there is no “internet of robotic interactions” to train on. The total volume of available robot training data is a tiny fraction of the text corpora behind modern LLMs—not a long-tail gap, but a shortage of foundational data at every level.

This shortage is the field’s binding constraint. Model architectures and robot hardware are mature enough to learn general manipulation; robot foundation models [2–4] demonstrate that the recipe works when data is available. What is missing is the massive, diverse, every-day interaction data that would trigger a GPT-1-style step change—the moment scale unlocks emergent generalization. And the ways the field has tried to produce that data run into three structural walls, examined in Section 2: real-world collection is hardware-bound and cannot reach foundation-model scale; hand-collected datasets cannot manufacture the compounding diversity that generalization demands; and policies trained only on expert demonstrations are structurally blind to their own failure modes.

The thesis of Axis is that the answer is not a better model but a fundamentally different way to *produce* data—one that generates diversity on demand, collects it without specialized hardware, and turns each policy’s failures into the next batch of training data. We realize this as a **compounding data engine**: a closed loop where every contribution improves the next policy and every improved policy decides what to collect next. This whitepaper is a complete description of that engine.

Contributions and structure. This document presents (i) a precise account of the data bottleneck and why it is structural (Section 2); (ii) the compounding-data-engine flywheel and the cornerstones and products that expose it (Section 3); (iii) the four-layer architecture beneath the products (Section 4); (iv) the three technical subsystems that do the work—task generation and native diversity (Section 5), data collection at scale (Section 6), and scalable online post-training via human-gated correction (Section 7)—followed by the model-centric processing pipeline (Section 8); (v) empirical validation, including our headline crowdsourcing result (Section 9); and (vi) the two-tier evaluation oracle and on-chain provenance that turn crowd work into trusted data with an auditable contribution record, together with the \$AXIS tokenomics (Section 11), plus the roadmap and go-to-market (Sections 10–12).

2 The Data Bottleneck

2.1 Physical AI Has No Internet

The defining asymmetry between language models and Physical AI is the starting corpus. LLMs inherited one; robots did not. There is no passively available record of embodied interaction at anything approaching web scale, and the interaction data that does exist is fragmented across incompatible platforms, embodiments, and formats. This is why progress in Physical AI has not tracked progress in language: the missing ingredient is not a smarter learner but the foundational data a learner would need.

2.2 How Far From the GPT-1 Moment?

The language-model trajectory is instructive, because each step isolates a different bottleneck. GPT-1 proved a direction—Transformer plus pre-training works—and was validation rather than scale. GPT-2 trained on roughly 10 billion tokens, about 500,000 hours of reading at a human pace, and emergent behavior appeared for the first time. GPT-3 trained on roughly 300 billion tokens, about 15 million hours; without that volume there is no ChatGPT and none

of what followed. At that stage the binding lever was not curation but volume.

Robotics has not had its GPT-1 moment. Model architectures are converging and algorithms are maturing, but the data itself is unsettled: which pre-training distribution is the right one, what the simulation / real / ego-centric ratio should be, how action resolution ought to be modeled, and how long the temporal horizon needs to be. There is no consensus on data structure to curate toward.

Carrying the comparison across sets the scale of the problem. If 15 million hours of text produced GPT-3, a genuinely general-purpose Physical AI foundation model plausibly requires *on the order of* 100 million hours of human manipulation data—an order-of-magnitude estimate, and one that remains daunting with a factor of ten in either direction. Three bars must be cleared at once, and today’s best datasets fall orders of magnitude short of each.

Table 1. The distance to general-purpose manipulation, across three dimensions. The right column reflects the current state of the field; the middle column reflects what generalization is understood to require, with the volume figure an order-of-magnitude estimate extrapolated from language-model pre-training.

Dimension	What generalization requires	Where the field stands today
Data volume	100M+ hours of manipulation data (estimate)	~2,000 hrs — the industry leader, Open X-Embodiment [5]
Scene diversity	10,000+ distinct physical environments	single digits — traditional lab collection
Atomic skills	1,000+ manipulation skills	~700 — a current benchmark, RT-1 [6]

The distance between these columns is the distance between narrow, brittle demonstrations and a general-purpose robotic policy—and the reason demand for structured manipulation data should compound over the next two to three years rather than level off.

2.3 Why the Gap Persists

Compute, hardware, and algorithms are ready. Data is what Physical AI starves for, and four forces keep it scarce.

- **Algorithms starved for data.** Architectures and hardware can learn general manipulation; what is absent is the everyday interaction data at the scale that unlocks emergent generalization.
- **Trapped in physical collection.** Real-world collection is hardware-bound: specialized rigs, safety systems, and one trained operator per robot, in the same room, in real time. This caps both volume and coverage and makes foundation-model scale structurally impossible.
- **Failing at generalizable diversity.** Real-world generalization demands compounding diversity across objects, scenes, embodiments, and viewpoints that hand-collected datasets can never reach. Simulation and ego-centric capture are the only paths that scale to it.
- **Blind to their own failures.** A policy trained only on expert trajectories sees only the states an expert visited, so the situations where it is weakest are precisely those least represented in its training data. Errors compound over a task horizon—the covariate-shift problem [7]. Closing this requires on-policy corrective data: humans supervising the policy’s own execution and correcting it exactly where it fails—which has traditionally required one human continuously supervising one robot, and therefore has not scaled.

Every one of these barriers points the same way: not toward a better model, but toward a different data-production regime. That regime is the compounding data engine.

3 The Compounding Data Engine

Most robot data pipelines are one-directional: collect data, train a model, ship it, start over. Axis is built as a closed loop in which every contribution improves the next policy, and each improved policy determines what data to collect next. Data stops being a one-time cost and becomes an asset that appreciates.

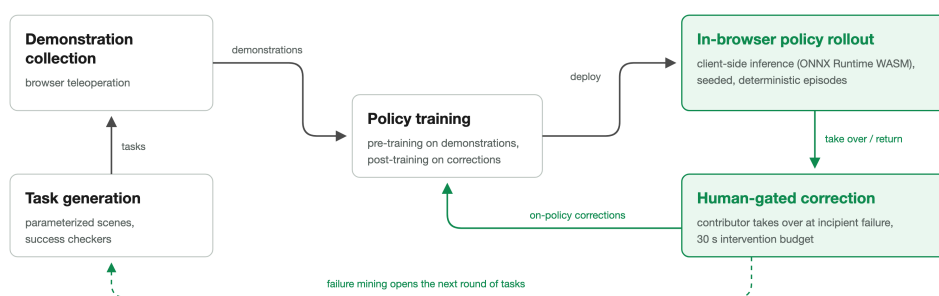


Figure 1. The compounding data engine. Contributor demonstrations train a base policy; the policy is deployed back to contributors for human-gated correction where it fails; and the policy’s measured weaknesses open the next round of tasks. Deployment feedback, not a fixed curriculum, decides what data gets collected next.

3.1 Why It Compounds

The engine runs as a loop, not a line:

1. **Generate** the task and world—diverse scenes, objects, and skills, produced on demand rather than hand-built.
2. **Collect** behavior—contributors teleoperate in simulation and capture real-world interactions, producing demonstration and correction data.
3. **Process** the data—verify, clean, and expand each trajectory into many photorealistic, domain-randomized training samples.
4. **Train and deploy**—policies learn from the data, run on real and simulated fleets, and expose exactly where they still fail.
5. **Feed back**—those failures decide which tasks open next, and the cycle repeats on harder, higher-value data.

Because step 5 targets step 1, the loop is self-reinforcing: the more it runs, the more valuable each new contribution becomes. A trained policy’s weaknesses are not a dead end—they are the next data request.

3.2 Three Cornerstones, Three Products

Beneath the surface sit three reusable cornerstones, and every part of the data business runs on them: the **Universal Task Generation Engine**, which continuously streams diverse scenarios and comprehensive coverage of atomic skills; the **Distributed Global Workforce**, a decentralized network that removes the physical and geographical limits of data gathering;

and the **Unified Processing Pipeline**, one standardized system that filters, augments, and transforms raw inputs into high-quality datasets. Because all three are shared, data quality is consistent no matter which source a trajectory came from.

The cornerstones reach the world as three products—the capability surfaces that contributors and customers actually touch. Together they cover the full loop.

Table 2. The three products, the flywheel stage each exposes, and current scale.

Product	Flywheel stage	Scale
Web — Simulation Data Platform	Generate → Collect (simulation)	10,000 valid trajectories per hour
Mobile — Ego-Centric Data Collection	Collect (real world)	Launching Sep 2026; target 10,000+ valid hours per month
Infrastructure — Data-to-Model Pipeline	Process → Train	—

Task generation behind both platforms. Diversity is engineered, not left to chance. From a natural-language instruction, the engine multiplies five dimensions—embodiment, object, spatial, semantic, and visual—to generate exponentially growing data at low marginal cost (Section 5).

Web — Simulation Data Platform. Data collection shifted from legacy offline, sequential methods to an online, fleet-wide, massively parallel human-in-the-loop engine in the browser—no hardware, no install. It powers both halves of the loop: expert demonstrations for pre-training, and human-gated corrections on live policy rollouts for post-training (Sections 6–7).

Mobile — Ego-Centric Data Collection. Real-world capture shifts from expensive, hardware-heavy rigs to a zero-barrier mobile app that pairs real-time hand-pose tracking with a global workforce, producing 4D hand-pose reconstruction with dense language annotation. This product is in development (Section 6).

Infrastructure — Data-to-Model Pipeline. The central processing hub, and model-centric data engineering rather than raw accumulation. A toolset turns diverse inputs into training-ready formats through cleaning, domain-randomized augmentation, and annotation; customers reach the same layer through automated evaluation toolkits and open-source datasets (Section 8).

3.3 Products Versus Architecture

The three products are *not* the same thing as the four architecture layers. Products are outward capability surfaces—flywheel stages, productized. The architecture (Section 4) is the engine underneath: four hierarchical layers from global data infrastructure up to deployment. Table 3 is the single place that maps one to the other.

3.4 Design Principles

Four principles govern the whole system and recur throughout the sections that follow:

- **Data generation must be model-conditioned.** The model’s measured weaknesses—not a fixed curriculum—determine what data gets collected next.
- **Simulation and real-world data must share a unified representation.** One embodied schema across synthetic and real sources; no fragmentation between them.

Table 3. Products (capability surfaces) mapped onto architecture layers (implementation).

Product	Flywheel stage	Runs on architecture layer(s)
Web — Simulation Data Platform	Generate → Collect (sim)	L1 assets, schema & data interface → L2 structured generation and simulation modality → L3 post-training loop
Mobile — Ego-Centric Data Collection	Collect (real)	L1 mobile interface → L2 ego-centric modality
Infrastructure — Data-to-Model Pipeline	Process	L2 model-conditioned generation → L3 model-ready data
<i>Model training & deployment</i>	Train & deploy	L3 model training → L4 deployment & rollout

- **Post-training is as critical as pre-training.** Demonstrations establish coverage; corrections establish reliability. The platform is built to produce both.
- **Deployment feedback must directly shape the data distribution.** Failure signals from policy rollouts feed straight back into task construction and distribution reweighting.

4 Architecture: A Four-Layer Closed Loop

Modern Physical AI models—embodied foundation models and world models alike—are constrained not by model capacity but by the structure of their data loops. Axis is a vertically integrated infrastructure designed around that constraint, organized into four hierarchical layers spanning foundational infrastructure to real-world deployment. Dependencies flow upward; deployment feedback closes the loop.

4.1 Layer 1 — Global Data Infrastructure

The foundational layer provides scalable, standardized data-generation capabilities and governs contributor access, quality control, and representation alignment.

- **Retail-accessible global data interface.** A low-friction, human-in-the-loop interaction layer reachable via web, desktop, and mobile, through which globally distributed contributors capture embodied interactions, execute task instructions, and provide corrective demonstrations on model rollouts. The web interface is live today; mobile and everyday ego-centric capture arrive with the ego-centric pipeline (Section 6).
- **Automated and human data validation.** All trajectories pass a multi-stage validation pipeline—automated scoring, failure and anomaly filtering, and human validation for corrective supervision—augmented by model-policy gap analysis that identifies high corrective-value samples. Dataset growth is therefore quality-aligned rather than volume-driven.
- **Model-ready embodied data schema.** All data, simulated and real, is normalized into a unified schema standardizing scene representations, action trajectories, 4D hand-pose sequences, and subtask-level temporal alignment. This eliminates fragmentation between synthetic and real sources and ensures cross-simulator interoperability.

4.2 Layer 2 — Closed-Loop Hybrid Data Engine

Built atop the infrastructure layer, the data engine performs structured generation and adaptive distribution shaping across three complementary modalities.

- **Scalable simulation data collection.** Simulation supplies controllable diversity, task parameterization, distribution shaping, and rapid policy iteration, and is used to explore scaling laws and cover rare or structured regimes.
- **Ego-centric real-world capture.** High-fidelity ego-centric data—4D hand-pose trajectories, subtask-level annotations, object-interaction dynamics—grounds simulation priors and bridges the sim-to-real representation gap. This modality is in development.
- **Model-conditioned data generation.** Generation is not static: using deployment feedback and policy-gap signals, the engine performs failure mining, weakness-targeted task construction, and adaptive distribution reweighting. Data generation becomes an optimization process conditioned on model behavior—the mechanism that decides which post-training tasks open next.

4.3 Layer 3 — 3D-aware Model

The foundation-model layer consumes structured embodied datasets and produces evolving policies through two tightly coupled phases. *Large-scale pre-training* on off-policy expert demonstrations establishes task representations, manipulation priors, and long-horizon planning structure—broad behavioral coverage across environments. *Post-training with human-gated DAgger* corrects on-policy rollouts through structured human supervision collected directly on model-generated trajectories, enabling systematic failure recovery, long-horizon stability, and reduction of compounding errors. The model evolves continuously through feedback rather than isolated retraining cycles. Section 7 formalizes this layer.

4.4 Layer 4 — Deployment & Policy Rollout

The top layer executes policies across real-world and simulation fleets, generating on-policy rollouts, failure trajectories, and distribution-drift signals. These feed directly back into the data engine, closing the loop. On the platform, this is what contributors experience as post-training tasks: every community rollout is simultaneously deployment, evaluation, and targeted data collection.

4.5 System-Level Property

Composed, the layers form a self-reinforcing flywheel:

Infrastructure → Adaptive Data Engine → Training → Deployment → Failure Signals → Targeted Generation

Embodied-intelligence training shifts from static dataset accumulation to continuous, closed-loop optimization. The concrete production pipeline that implements Layers 1–3—world generation, behavior collection, data refinement, and model training—is detailed in Sections 5–8, and every artifact it produces carries provenance (Section 10).

5 Task Generation & Native Diversity

Generalization is a function of diversity, and Axis treats diversity as a first-class design primitive rather than a byproduct of collection. The Task Generation Engine manufactures it along five independent axes, whose combinations grow multiplicatively.

- **Embodiment** — one native multi-embodiment infrastructure spanning 13+ major robots, from single-arm to bimanual, wheeled, dexterous-hand, and humanoid systems, integrating new hardware at near-zero marginal cost.

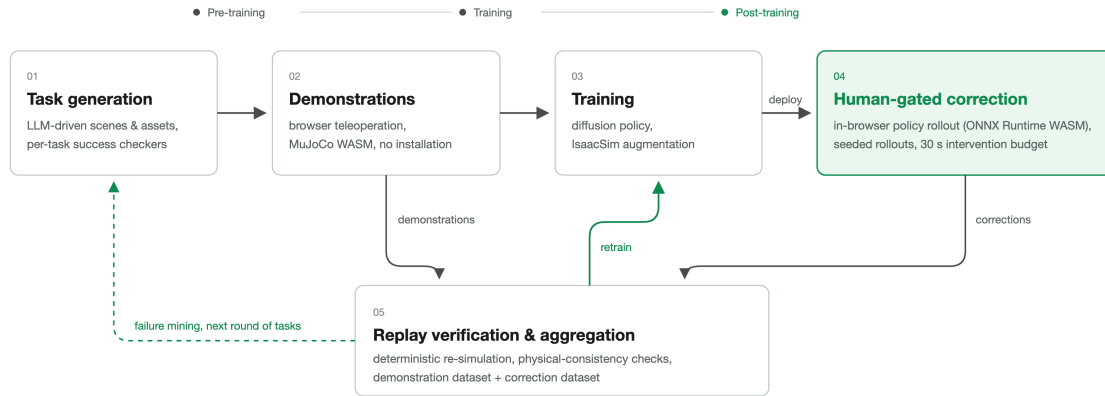


Figure 2. The production pipeline that implements Layers 1–3. A generated task (01) collects browser demonstrations (02) that train a base policy (03); the policy deploys back into contributors’ browsers for human-gated correction (04); every submission passes replay verification (05), and verified corrections plus failure mining open the next round of tasks.

- **Object** — the full spectrum of manipulation, from simple rigid items to complex articulated mechanisms for home and industrial use.
- **Spatial** — layout randomization that samples object placements across the robot’s entire reachable workspace.
- **Semantic** — asset randomization that translates LLM-level knowledge into physical variety, generating sub-types within any object class.
- **Visual** — procedural randomization of textures, lighting, and camera poses for zero-shot robustness.

Because these axes are independent, the reachable data distribution is their Cartesian product—tasks $\times$ worlds $\times$ behaviors $\times$ visual randomization—and capacity scales multiplicatively, not linearly.

5.1 From Language to a Simulator-Ready World

The engine turns a natural-language instruction into a fully specified, verifiable task through a five-stage pipeline: a *task manager* parses the instruction into structured configuration (manipulation goal, object interactions, workspace context, and a difficulty level from 1 to 5); an *assets manager* sources 3D objects from a database of 2,000+ assets, with an image-to-3D conversion path and a normalization step that rescales objects to plausible physical dimensions; a *2.5D layout generator* arranges objects, distinguishing task-involved from decorative items; a *layout supervisor* validates the 3D instantiation and iteratively corrects it; and a *success checker*, generated alongside the scene, encodes the geometric completion conditions that later gate every submission. The deployed engine has published 1,600+ tasks spanning pick-and-place, push, sort, pour, open/close, stack, articulated-object interaction, and long-horizon multi-stage sequences.

5.2 RoboVerse and MetaSim: The Cross-Simulator Foundation

Task generation and every downstream stage run on **RoboVerse** [8], a unified cross-simulator framework developed by the core team, and its scene layer **MetaSim**. No single simulator

is ideal for every job: MuJoCo (compiled to WebAssembly) is lightweight and runs in any browser, making it ideal for interactive collection; IsaacSim is GPU-powered and photorealistic, making it ideal for augmentation and replay. MetaSim decouples task definitions from simulator-specific implementations through a standardized scene representation that compiles to native formats—MJCF for MuJoCo, USD for IsaacSim, URDF for others—and normalizes coordinate frames, collision meshes, and visual properties. A trajectory collected in browser-native MuJoCo can therefore be replayed in the IsaacSim backend without re-implementing task logic, and the same asset and scene APIs feed the Task Generation Engine. Cross-simulator replay is understood as a scalable augmentation mechanism rather than exact reproduction: contact models, actuator dynamics, and solver settings differ, so replay is verified for physical consistency within tolerances (Section 8).

6 Data Collection at Scale

The collection stage powers both halves of the loop—demonstrations for pre-training and, on the same platform, corrections for post-training (Section 7). Two modalities feed it: browser-based simulation collection, live today, and mobile ego-centric capture, in development.

6.1 Browser-Native Simulation Collection

The defining accessibility choice is that the full physics simulation runs in the contributor’s browser. MuJoCo, compiled to WebAssembly, handles all dynamics, collision, and control integration on the contributor’s local CPU; rendering is a Three.js / WebGL layer with a floating, semi-transparent interface. There is no robot, no GPU requirement, and no installation—the platform inherits the accessibility, simulator-diversity, and cross-machine-scalability goals of the underlying infrastructure [1], and rollout capacity grows with the size of the community rather than with capital expenditure.

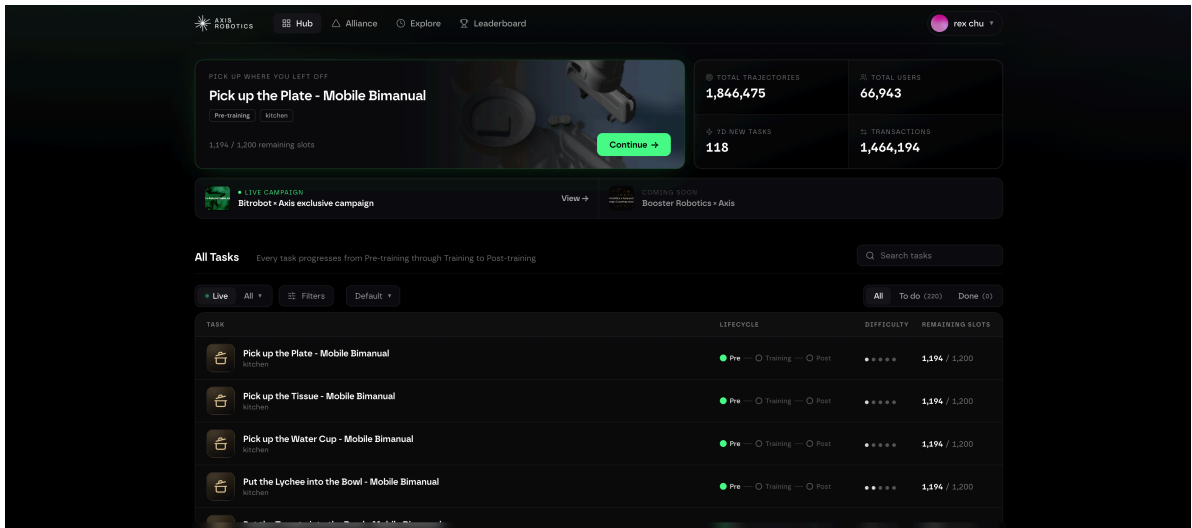


Figure 3. The deployed task catalog. Each task carries a Pre → Training → Post lifecycle indicator alongside its scenario, difficulty, and remaining collection slots; contributors filter the catalog by stage.

Control operates on an adaptive tick—a base rate that rises during active manipulation—and a contributor’s task-space inputs (end-effector position, rotation, gripper state) resolve to joint commands through an embedded inverse-kinematics solver. The control layer resolves IK *targets* rather than driving every frame, with smooth online interpolation in joint space, and adds direct manipulation (dragging the end-effector along the view plane, or double-clicking an object for an automatic pre-grasp) and simulator-state checkpointing for long-horizon tasks. A

single scalar collapses a high-DoF end-effector to one control axis via grasp-template interpolation,

$$q(t) = q_{\text{rest}} + \alpha(t) (q_{\text{tmpl}} - q_{\text{rest}}),$$

where q_{tmpl} is a target joint-angle vector from a library of grasp templates (power, precision, tripod, pinch), letting a contributor operate hands with $n > 16$ joints through two keys. New embodiments are integrated by defining a rest pose and a template library.

6.2 Mobile Ego-Centric Capture

In development. Real-world capture shifts from expensive, hardware-heavy rigs to a zero-barrier mobile ego-centric app. Pairing real-time hand-pose tracking with a global workforce, it translates human vision into robotic motion across everyday routines and domain-specific workflows, with modular setups ranging from a single iPhone to multi-camera headbands, wrist cameras, and UMI grippers. The output is 4D hand-pose reconstruction paired with dense language annotation, normalized into the same embodied schema as simulation data so that both sources train the same models.

7 Scalable Online Post-Training

Demonstrations alone hit a well-understood ceiling. Behavior cloning treats imitation as supervised learning, but a deployed policy’s own mistakes carry it into states absent from the demonstration distribution; errors compound, growing quadratically with the task horizon [7]. The remedy is equally well understood—collect corrective labels on the state distribution induced by the learner itself [9], with a human deciding when to take over and when to hand back control [10]. What has never been solved is the economics: human-gated correction has traditionally required one human continuously supervising one robot, an arrangement whose cost scales linearly with data volume and which has confined correction datasets to laboratory scale, even as recent fleet-based systems show how valuable on-policy correction becomes once a base policy exists [11–13].

Axis removes this constraint by observing that all three ingredients—the robot, the environment, and the supervisor’s control channel—already exist in the browser. The platform deploys the *policy itself* to the contributor’s machine, turning every participating client into a self-contained rollout-and-correction node.

7.1 Client-Side Policy Deployment

The task’s current policy checkpoint—a diffusion-based action denoiser [14]—is exported to ONNX and fetched by the client via a per-policy manifest; inference runs on ONNX Runtime’s WebAssembly backend [15] in the same process as the MuJoCo physics. In-browser inference is established practice for vision and language workloads [16]; to our knowledge, Axis is the first robot-learning platform to execute the policy under correction client-side. The consequences are architectural, not cosmetic: there is no observation–action round trip to a server, no server-side GPU fleet to provision, and rollout capacity scales with the number of participating browsers.

7.2 Seeded Rollouts, Human Gating, and the Intervention Budget

Each session is initialized from a seeded simulation state—initial object poses, robot configuration, and physics state are drawn once and recorded with the trajectory—so any submitted episode can be re-simulated and verified, and every correction anchors to a precise, recover-

able failure state. Seeding also lets rollouts start in regimes the policy has not yet mastered, directing contributor effort toward correction rather than re-demonstrating solved behavior.

Following the human-gated protocol [10], the contributor—not an uncertainty estimate—decides when to intervene. With π_θ the deployed policy and π_H the contributor’s teleoperation, the executed action is

$$a_t = g_t \pi_H(s_t) + (1 - g_t) \pi_\theta(o_t), \quad (1)$$

where the gating indicator $g_t \in \{0, 1\}$ is toggled through three interface states: *Run Policy* starts autonomous execution, *Take Over* transfers control to the contributor’s full teleoperation scheme, and *Return to Policy* hands control back once the robot re-enters a state the policy can handle. The interface coaches the central skill—intervening *at* the failure, neither before (which wastes effort and re-specifies solved behavior) nor after (which lets the failure contaminate the episode).

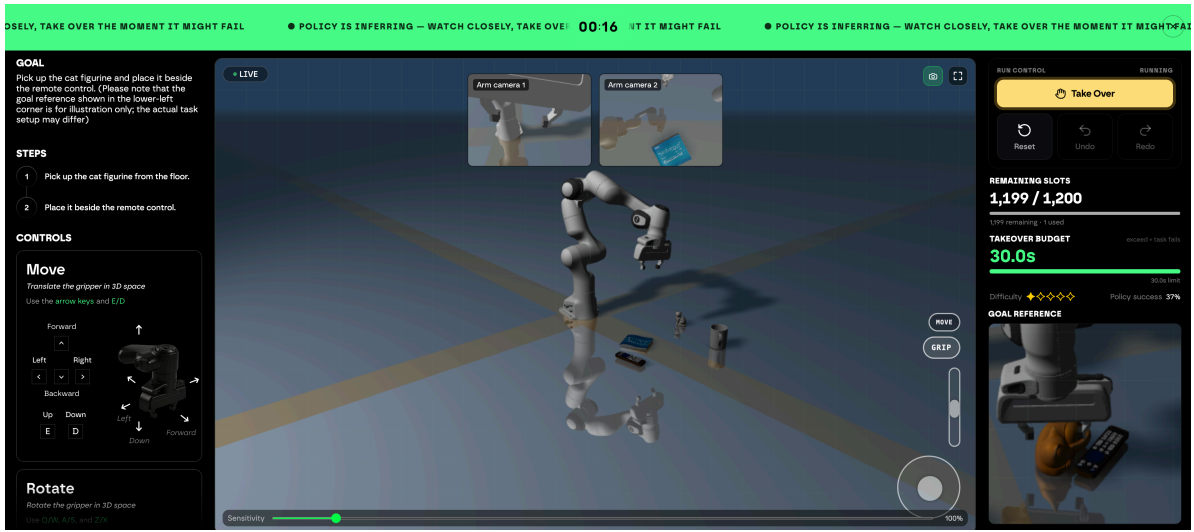


Figure 4. A post-training session during autonomous execution. The policy drives the robot while the run-control panel offers *Take Over*; a status banner cues the contributor to intervene the moment the policy is about to fail.

Each rollout carries an intervention budget of $T_{\max} = 30$ s: the cumulative wall-clock time under human control may not exceed it, and an episode that does is marked failed. Where robot-gated systems budget how often the robot may *request* help [17], this budget bounds how much help the human may *supply*. It is what keeps crowd-collected corrections *corrections*: without it, a struggling rollout degenerates into a full demonstration, which the pre-training stage already collects at scale. Completed episodes are scored inversely to total intervention time, and incomplete episodes—including budget overruns—earn nothing; a rollout the policy completes with zero intervention is the best outcome and doubles as an evaluation datum. Appendix A gives the full session protocol.

7.3 Correction Recording and Model-Conditioned Collection

Every control tick is labeled by control mode $c_t \in \{\text{policy}, \text{human}\}$; the full simulator state is captured at the adaptive rate and trajectories are resampled to a uniform training frequency (Appendix B). Intervention onsets are themselves supervision: each marks a state the contributor judged to be leaving the policy’s reliable regime. Verified corrections are aggregated following the DAgger recipe [9], and aggregate rollout statistics—per-task autonomous success rates and intervention-onset distributions—drive failure mining, weakness-targeted task construction, and distribution reweighting. The retrained policy re-enters the platform and

the loop repeats: contributors supply what the policy lacks, and the policy keeps what it already does well.

This instantiates the scalable-online-post-training paradigm [11] with different resources. Where fleet-based systems stream on-policy experience from physical robots supervised by on-site operators, Axis sources rollouts from a global contributor community operating simulated rollouts in their own browsers, with distributed human-gated takeovers as the corrective signal. The scalable source of rollouts is a community rather than a robot fleet, and the operating environment is simulation rather than the physical world.

7.4 Related Work in Interactive Imitation Learning

Behavior cloning’s compounding-error pathology [7] motivated DAgger [9], which aggregates expert labels on the learner’s induced state distribution. Interactive imitation learning [18] refines this around interventions: HG-DAgger [10] lets the human decide when to seize and return control; ThriftyDAgger [17] studies robot-gated variants under an intervention budget; and intervention-weighted objectives [19,20] exploit the fact that intervention onsets mark exactly the states where a policy leaves a reliable regime. A complementary line scales the *supply* of on-policy experience: Fleet-DAgger [13] allocates scarce human attention across many robots, and SOP [11] and related experience-driven systems [12] report that once a base policy exists, the marginal utility of on-policy correction exceeds that of additional offline demonstrations. Crowdsourced collection has a long lineage—RoboTurk [21] streamed cloud simulation to browsers, and later systems recruit or gamify contributors [22]—but these crowdsource *demonstrations*, or at most remote takeovers of server-side execution. Axis differs in both artifact and architecture: the crowd supplies *corrections* on autonomous rollouts, and both physics and policy inference run client-side.

8 The Data Processing Pipeline

Raw trajectories are not training data. Axis treats processing as model-centric data engineering, not raw accumulation: every submission is verified, cleaned, and expanded into many training-ready samples.

8.1 Verification and Cleaning

Every submission—demonstration or correction—is re-simulated from its seeded initial state and checked for physical consistency before acceptance; episodes that cannot be reproduced within tolerance, or that fail the task’s success checker on replay, never reach training. Accepted trajectories are then cleaned: a *filter* stage rejects corrupted or incomplete episodes, extreme state discontinuities, and implausible transitions by comparing adjacent-frame state differences against physically plausible thresholds; *static-frame removal* drops frames whose joint variation falls below a small threshold; a *Savitzky–Golay filter* smooths action trajectories while excluding gripper joints to preserve sharp open/close transitions; and *cubic-spline re-sampling* aligns every trajectory, arriving at an adaptive control rate, to a uniform training frequency. Object-state channels bypass the smoothing filter and are interpolated separately. Across benchmark tasks from the LIBERO suite [23], cleaning reduces mean acceleration and jerk while preserving motion shape.

8.2 Realistic Augmentation via Domain Randomization

Cleaned trajectories are replayed on GPU servers running IsaacSim, where each original trajectory is multiplied into many augmented samples under domain randomization [24]. The

randomization spans visuals (lighting direction, intensity, and color temperature; surface materials and textures; backgrounds and distractors; camera intrinsics and extrinsics) and physics (mass, friction, damping, and compliance), and generates multi-view RGB with optional depth and segmentation. A layered scene hierarchy—environment, workspace, and objects—separates macro backgrounds from the manipulation surface and from tabletop clutter, and independent seeded random-number generators make every perturbation reproducible even under distributed training. The objective is to force the model to learn task-relevant structure rather than memorize environment-specific detail.

8.3 Annotation

For real-world and long-horizon data, an annotation stage couples semantics with physical motion: *segmentation* slices long videos into discrete, meaningful steps; *context-aware description* generates a language annotation for each segment; and frame-by-frame *atomic labeling* synchronizes text, video, and 4D trajectories. This is what makes ego-centric capture model-ready and what lets language supervision ride alongside motion.

9 Results & Validation

9.1 Noisy Crowdsourcing Trains Strong Robots

The central empirical question behind Axis is whether large-scale, noisy, crowd-sourced simulation data can serve as a sufficient substrate for embodied pretraining. The answer is a qualified yes: dirty at the trajectory level, clean in the aggregate, provided diversity and coverage clear a critical bar. Every condition starts from the released $\pi_{0.5}$ checkpoint, optionally continues pretraining on a simulation corpus, then fine-tunes on LIBERO under an identical recipe and is evaluated on **LIBERO-Plus**, a perturbation benchmark spanning camera, lighting, sensor-noise, background, layout, language, and robot variations. Continual pretraining on the full Axis set lifts overall success rate from **83.9% to 88.8%**—a 5.8% relative improvement—and clears a volume-matched RoboCasa365 control by **31.3 points** (Table 4).

Table 4. LIBERO-Plus success rate (%) by continual-pretraining condition. N is the full Axis demonstration set; the RoboCasa-matched control uses the same trajectory count. Overall performance rises with Axis volume, while the matched-volume control falls far below vanilla.

Pretraining	Demos	Overall	Cam.	Light	Noise	B.G.	Layout	Lang.
$\pi_{0.5}$ (vanilla)	0	83.9	72.5	98.2	82.5	94.4	82.9	89.6
+ AXIS-25%	0.25 N	84.7	77.5	91.2	86.2	100.0	85.5	81.8
+ AXIS-50%	0.50 N	85.7	68.8	98.2	91.2	96.3	88.2	84.4
+ AXIS-100% (ours)	N	88.8	83.8	96.5	96.2	98.1	85.5	88.3
+ RoboCasa-matched (ctrl.)	N	57.5	35.2	79.5	63.2	81.7	68.0	49.2

Overall performance rises with Axis volume—84.7, 85.7, then 88.8 across 25%, 50%, and 100% of the set—evidence that the crowd’s aggregate signal is coherent rather than noise that washes out. The control settles the alternative explanation: at an identical trajectory count, RoboCasa-matched reaches only 57.5, well below vanilla, so the gain is a property of the Axis pipeline rather than of additional simulation data. (Robot-axis scores, omitted above for width, run 74.4 vanilla, 76.9, 79.5, 78.2 for Axis, and 39.4 for the control.)

9.2 Where the Gains Land

Broken out by perturbation axis (Table 5), the largest improvements fall where the augmentation is aimed—sensor noise and camera. Lighting and language land within noise of vanilla, which is what one expects from augmentation that does not target them. Against the matched-volume control, however, every axis improves by a wide margin, and the axes where Axis pulls furthest ahead—camera, language, robot—indicate a better underlying manipulation representation rather than surface-level robustness alone.

Table 5. Per-axis gain of AXIS-100% in points, against vanilla $\pi_{0.5}$ and against the volume-matched RoboCasa365 control.

Axis	vs. vanilla	vs. RoboCasa-matched
Sensor Noise	+13.7	+33.0
Camera	+11.3	+48.6
Robot	+3.8	+38.8
Background	+3.7	+16.4
Layout	+2.6	+17.5
Language	−1.3	+39.1
Light	−1.7	+17.0
Overall	+4.9	+31.3

9.3 What This Changes

Egocentric data already made this case for perception and world models; these results extend it to *control*. Individual trajectories may be imperfect so long as the distribution they compose is diverse and well covered, because what a policy learns from is the distribution rather than any single demonstration. The practical consequence is a change in the unit of quality control: stop grading the trajectory and grade the distribution, then let closed-loop refinement finish the last mile—the policy’s own failures reopen the tasks it is weakest at, and human-gated correction (Section 7) supplies precisely the data the distribution lacks.

9.4 The Dataset Behind the Result

The Axis dataset used here spans **207 manipulation tasks, 50,000+ human demonstrations, and 60,000+ scene variants** across **7 scene categories** (kitchen, toys, personal care, office, household, tools, tabletop). Each task carries a language instruction, a parameterized scene, its assets, and a structured success checker; each trajectory carries synchronized RGB-D from two camera views, robot and object states, actions, metadata, and success labels. Every trajectory passes format verification, physical-consistency and success checking, smoothing, and resampling before IsaacSim rendering—the same pipeline that runs in production (Section 8), which cuts mean acceleration by 63.9% and mean jerk by 80.8% relative to raw teleoperation. Full methodology, ablations, and the held-out evaluation protocol are in the paper [1].

9.5 Sim-to-Real and Published Foundations

The dataset and benchmark target simulated Franka tabletop manipulation, and sim-to-real transfer remains an open problem rather than a settled result; real-robot evaluation on a Franka Research 3 arm across pick-and-place, drawer opening and closing, pouring, and button pressing runs with a typical fine-tuning budget of 40–60 demonstrations per task. The

platform builds on published research by the core team and collaborators, including RoboVerse [8] (the unified cross-simulator framework), GraspVLA [25] (models trained purely on simulation data achieving high zero-shot success on real robots), SkillBlender [26] (fusing atomic skills across embodiments including humanoids), and DIPOLE [27] (a diffusion policy that fuses vision and geometry, holding up under lighting, texture, viewpoint, and object-placement shift). Together with the interactive-imitation-learning literature (Section 7), this work grounds each component of the engine in published results.

10 The Evaluation Oracle and Provenance Layer

The data engine of Sections 3–8 closes a *technical* loop. Opening that loop to a globally distributed, permissionless contributor base raises a question the engine itself cannot answer: how do you turn contributions that are noisy, individually suboptimal, and possibly adversarial into a *trustworthy training set*, while keeping an accurate record of what each contributor produced? Axis’s answer is **Proof of Quality Work**: a two-tier evaluation oracle whose single quality verdict both gates admission to training and determines how much a run counts toward its contributor’s record, so that what the system credits is data value by construction. Better contribution earns a better score, and a better score counts for more—which is what makes the incentive point at genuine human contribution rather than at volume or time spent. On-chain provenance makes those verdicts auditable. Crypto here is infrastructure for provenance, transparency, and coordination—not a narrative layer.

10.1 The Two-Tier Evaluation Oracle

The oracle scores contribution at two levels: the micro level of a single trajectory, and the macro level of a contributor’s standing over time.

Tier 1 — Scoring System (per trajectory). Every submission passes a multi-stage evaluation before it counts for anything or enters training (Table 6). The stages escalate from cheap structural checks to expensive semantic judgment: schema and replay checks reject anything unreproducible; a rule-based checker confirms the task was actually completed; and a vision-language model judges whether the motion was reasonable rather than a lucky or degenerate success. Every stage is algorithmic—no contributor grades another’s work, which keeps the verdict cheap enough to run on every submission and removes the collusion surface a human-review tier would open. The combined score $Q(\text{traj})$ does double duty—it gates admission to the training pool *and* sets how much the run counts toward its contributor’s points—which is exactly what ties credit to data value. The replay check here is the same deterministic admission control described for the processing pipeline (Section 8); the oracle wraps it in a quality verdict. Post-training runs add one rule aligned with the intervention budget of Section 7: the task must be completed, less takeover time scores higher, and a run exceeding the 30-second budget fails and receives no credit. Failed attempts, spam, and duplicate trajectories receive none either.

Tier 2 — Point System (per contributor). Above the per-trajectory score sits a global Point System that aggregates a contributor’s standing over a settlement cycle along five factors: *volume* (valid trajectories submitted), *difficulty* (the tiers of tasks taken on), *score* (the Tier-1 verdicts those runs earned), *diversity* (spread across both scenarios and skills), and *referrals* (what directly invited contributors themselves produce, so inviting inactive accounts yields nothing). The first four are interdependent rather than substitutable: strength on one factor cannot offset neglect of another, which is what makes a focused set of hard, varied, well-executed submissions worth more than a large volume of trivial ones in a single scenario.

Table 6. Tier 1: the staged evaluation that produces $Q(\text{traj})$. Stages escalate from structural to semantic; failing any early stage stops the pipeline.

Stage	What it verifies / catches
Sanity & replay	Schema integrity and scene reloadability; re-simulation from the recorded seed for physical consistency — rejects corrupted or unreproducible submissions.
Rule-based success	The task’s generated success checker, evaluated on replay — rejects runs that do not actually complete the task.
VLM motion quality	A vision-language model judges key frames against the language goal — catches completions achieved through unreasonable or random motion.

Per-task caps complement the diversity factor. The platform now limits each task to five completions per account; for contribution recorded before that limit existed, when a task could be repeated without bound, only the first ten submissions from one address that pass evaluation are counted—failed submissions do not consume the allowance—and addresses that ran far past it carry a penalty factor. Task slots are finite, so an address filling them with near-duplicate repeats crowds out the variety the dataset is collected for. Issuance has two phases, split by a fixed snapshot: a one-time retroactive grant covering everything signed before it, and per-epoch emission after it, on a two-week *epoch*. Points accumulate without reset or expiry. Admission to an epoch’s accounting is gated on provenance: only runs signed on-chain, with a transaction recorded on Base (Section 10), are counted, and a submission signed in a later epoch counts in the epoch of its signature rather than the one in which it was collected. The same condition governed the one-time retroactive grant for pre-launch contribution: unsigned work was never recorded and did not enter it—so the contribution record and the on-chain record cannot drift apart. They are a non-transferable record of qualifying contribution rather than a balance or a claim; Section 11.3 states how they relate to the Community Contributor Airdrop.

One signal, two consumers. The oracle is the accounting analogue of the deployment-feedback loop: where failure signals tell the engine *what data to collect next*, the quality verdict tells it *what data to trust and what to credit*. Because the same $Q(\text{traj})$ both admits data to training and drives the contributor’s record, what the crowd is pulled toward is corrective and data value rather than volume or time-on-platform—the platform counts data, not presence.

10.2 On-Chain Provenance

Provenance is the trust substrate beneath the oracle’s verdicts. Every accepted run on the Axis Hub can be signed on-chain: a unique Data ID is recorded on Base (Coinbase’s L2), binding a contributor’s wallet, a timestamp, the quality score, and a task reference into a permanent, tamper-proof record, while the full trajectory data resides off-chain in the Axis backend. This makes the intelligence-production pipeline auditable end to end—which datasets trained which policies, and which contributions they trace back to—and makes contribution attribution verifiable rather than asserted. The record is a provenance trail; it does not by itself create ownership or financial rights in the datasets, the models, or the platform. The model is chain-agnostic: contributions made through partner alliances are verified on the partner’s chain (for the BitRobot subnet, Solana), with the same Data-ID and quality-score structure everywhere.

11 Tokenomics

\$AXIS is the native token of Axis Robotics. The majority of its fixed supply is allocated to the network: the Ecosystem allocation funds network growth, and the Contributor Rewards allocation is distributed to data contributors. Contributing to Axis Robotics does not require holding \$AXIS. Throughout this section, TGE is the token generation event and $T+n$ denotes n months after TGE; every percentage is a share of total supply unless stated otherwise.

11.1 Supply and Allocation

Total supply is fixed at **1,000,000,000 \$AXIS**, with no inflation. Circulating supply at TGE is **165,000,000 \$AXIS** (16.50%), and the full supply is unlocked by $T+35$. Table 7 sets out the eight allocations in four groups, and Figure 5 shows the same split proportionally. The Network group, Ecosystem and Contributor Rewards, accounts for **61.08%**; Team & Investors for 31.67%; Liquidity for 6.00%; and Community Sale & Campaigns for 1.25%.

Table 7. Allocation of the fixed 1,000,000,000 \$AXIS supply.

Allocation	\$AXIS	% of supply	Purpose
<i>Network (61.08%)</i>			
Ecosystem	510,833,333	51.08%	Growth, partnerships, research, community programs, treasury
Contributor Rewards	100,000,000	10.00%	Community Contributor Airdrop
<i>Team & Investors (31.67%)</i>			
Core Team	152,000,000	15.20%	Long-term team alignment
Private A	98,000,000	9.80%	Private round
Private B	66,666,667	6.67%	Private round
<i>Liquidity (6.00%)</i>			
Liquidity	60,000,000	6.00%	Exchange liquidity
<i>Community Sale & Campaigns (1.25%)</i>			
Sonar	10,000,000	1.00%	Community sale
Kaito Reward Pool	2,500,000	0.25%	Kaito campaign rewards
Total	1,000,000,000	100.00%	

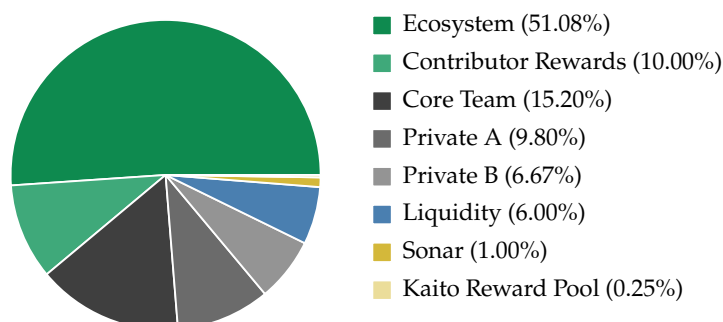


Figure 5. Allocation of the fixed \$AXIS supply. Greens mark the Network group, grays Team & Investors, blue Liquidity, and yellows Community Sale & Campaigns.

11.2 Release Schedule

Table 8 gives the release terms of each allocation. No Core Team or private-round tokens unlock at TGE. Supply is fixed, so no further tokens are issued after T+35.

Table 8. Release terms. The At TGE column sums to the 165,000,000 \$AXIS circulating at TGE.

Allocation	At TGE (% of supply)	Cliff	Vesting
Ecosystem	80,000,000 (8.00%)	—	Vests monthly from T+1 to T+35; T+1 and T+2 each include an additional 8,250,000
Contributor Rewards	20,000,000 (2.00%)	—	30,000,000 (3.00%) at T+3; 50,000,000 (5.00%) at T+6
Core Team, Private A, Private B	0	12 months	Vests monthly over 24 months, T+12 to T+35
Liquidity	60,000,000 (6.00%)	—	Fully unlocked at TGE
Sonar	2,500,000 (0.25%)	6 months	Vests monthly over 6 months, T+6 to T+11
Kaito Reward Pool	2,500,000 (0.25%)	—	Fully unlocked at TGE
Total	165,000,000 (16.50%)		

Aggregated, these terms produce the supply curve of Figure 6. Monthly unlocks are about 20.09 million \$AXIS at T+1 and T+2, 41.84 million at T+3, and 11.84 million at T+4 and T+5. T+6 is the largest month at 63.09 million, when the final Contributor Rewards tranche and the first Sonar tranche coincide. Unlocks are about 13.09 million per month from T+7 to T+11 and about 25.03 million per month from T+12 to T+35, when the Core Team and private-round allocations vest.

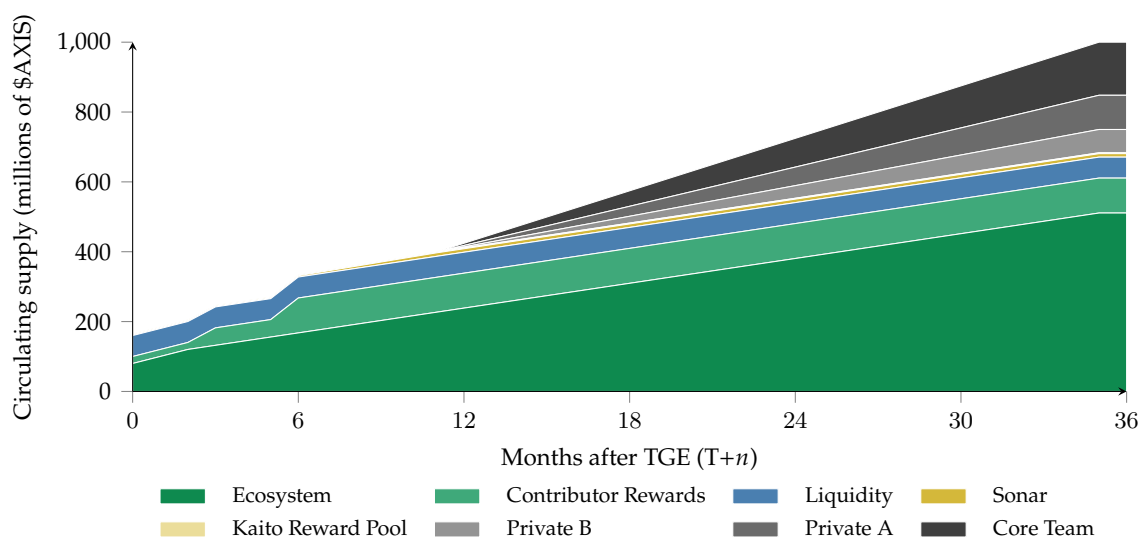


Figure 6. Circulating supply by allocation from TGE to T+36: 165,000,000 \$AXIS (16.50%) at TGE, steps at T+3 and T+6, team and investor releases from T+12, fully unlocked at T+35.

11.3 Community Contributor Airdrop

The Contributor Rewards allocation, 10% of total supply, is distributed through the Community Contributor Airdrop in two seasons (Table 9).

Table 9. The two airdrop seasons.

Season	Focus	Allocation (% of supply)
Season 1 · Pre-TGE	Simulation	40,000,000 (4.00%)
Season 2 · Post-TGE	Sim-to-real	60,000,000 (6.00%)

Season 1 unlocks 50% at TGE and the remaining 50% in two equal tranches at T+3 and T+6 (Table 10). The staged release aligns Season 1 rewards with the network’s development over the first six months after TGE. Season 2 terms will be announced when the season opens.

Table 10. Season 1 unlock.

	At TGE	T+3	T+6
\$AXIS	20,000,000	10,000,000	10,000,000
% of Season 1	50%	25%	25%
% of supply	2.00%	1.00%	1.00%

Accepted contributions earn Axis Points (Section 10.1). Points may be considered as one input for eligibility and reward allocation under separately published program terms.

11.4 Utility

Ecosystem development. The Ecosystem allocation funds growth, external partnerships, academic research, community programs, and treasury.

Contributor rewards. Distributed through the Community Contributor Airdrop (Section 11.3). Only verified, accepted contributions qualify.

Staking and Governance (Proposed). Enable clients, developers, researchers, and model builders to participate in an open Axis ecosystem. Staking links token utility to participation, including access to high-value tasks. Terms will be published before launch.

11.5 Disclaimer

This section does not constitute an offer to sell, or a solicitation of an offer to buy, any token or security in any jurisdiction in which such an offer or sale would be unlawful, and does not constitute investment advice. Figures may change before TGE.

12 Traction, Customers, and Roadmap

12.1 Deployed Scale

The platform is live and in the hands of a global community. As of 10 October 2026 it had accumulated **7.3M+ trajectories** from **229,000+ contributors** across **1,600+ published tasks**, with **7.0M+ on-chain transactions** recording their provenance. Current simulation throughput runs at **10,000 valid trajectories per hour**; ego-centric capture is in development, targeting

10,000+ valid hours per month. These figures describe the deployed platform today; the volume and revenue targets in the roadmap below are forward-looking.

12.2 Who Uses It

The engine serves three classes of customer, each consuming a different part of the loop:

- **Robotics hardware** partners receive their embodiment in simulation, tailored data, and end-to-end training and deployment—using high-fidelity simulation and ego-centric data to reduce dependence on expensive real-robot data—in exchange for robot specifications (URDF/MJCF) and hardware for validation. Working with AgiBot, Booster Robotics, and others including Zeroth, Deep Robotics, EngineAI, and Unitree.
- **Foundation model** labs receive premium trajectory packages, structured task bundles, and joint co-training. Working with Manycore Tech and other foundation-model labs.
- **Industrial automation** customers receive end-to-end, learning-based automation—sorting solutions included—powered by simulation-driven pipelines, bridging advanced robotics intelligence with real production environments. Working with Geely Auto and Lotus Cars.

12.3 Roadmap

The near-term roadmap advances three tracks in parallel—the data engine, the global ecosystem, and commercialization. Unless marked as reached, the figures below are targets.

Table 11. Near-term roadmap. Figures are forward-looking targets unless marked as reached.

When	Data engine	Ecosystem & commercialization
Sep 2026	Ego-centric pipeline (in development): maximize in-the-wild diversity	100k+ contributor network <i>reached</i> (229,000+ as of 10 October 2026); expanding to LATAM & Eastern Europe
Oct 2026	Sim Dataset V2 (in progress): target 10,000+ hours, millions of trajectories, more embodiments and skills	—
EoY 2026	First large-scale human-gated DAgger dataset: failure-recovery at scale	High-velocity yield: target 500+ hrs/day ego, 50+ hrs/day sim; 5+ top-tier lab partnerships. Target \$1M+ exit ARR; 2–3 new paid pilots
Q1 2027	Continuous scaling of all data streams	Target entry to a Tier-1 Physical AI foundation-model supply chain
Mid 2027	—	Target 3× year-over-year revenue

Longer term, the direction is open infrastructure for robotic intelligence: foundation models for embodied intelligence, interoperable data and model markets (a data marketplace, simulation- and augmentation-as-a-service) where qualified participants publish under defined licensing terms, and a verifiable coordination layer built on the provenance record of Section 10—a shared audit trail across task creators, data contributors, and model trainers.

13 Limitations & Open Problems

The engine is deployed and its headline result is reproducible, but several problems remain open, and we state them plainly.

- **Corrections are collected in simulation.** The post-training loop improves policies in the domain where corrections are gathered. Whether crowd-collected corrections transfer to physical deployment at the same rate as in-lab corrections is an open question; real-robot validation (Section 9) begins to address it but does not settle it.
- **Cross-simulator replay is lossy.** Replaying a trajectory across MuJoCo and IsaacSim exposes differences in contact models, actuator dynamics, and solver settings. Replay is therefore treated as a scalable augmentation mechanism verified within physical tolerances, not as exact reproduction; stronger tracking controllers and tighter simulator calibration are active work.
- **Crowd gating quality varies.** Contributors differ in when they choose to intervene, and some takeovers will be early, late, or unnecessary. The scoring rule, replay verification, and intervention-weighted training objectives mitigate this variance but do not eliminate it.
- **Supervision remains sparse.** Teleoperation data carries no dense per-step labels. Intervention onsets provide temporally localized failure labels that demonstrations cannot, and the annotation stage (Section 8) densifies ego-centric data, but automatic dense labeling of demonstrations at scale is unsolved.
- **Cross-embodiment transfer is unsolved at the infrastructure level.** The engine generates and normalizes data across embodiments, but transferring a learned policy from one embodiment to another may still require embodiment-aware retargeting, canonical action spaces, or learned motion translation.
- **The intervention budget is a blunt instrument.** A fixed thirty-second budget suits tabletop-scale tasks; long-horizon tasks may warrant stage-indexed budgets, and the right budget is ultimately a function of policy competence rather than a constant.

14 Conclusion

Physical AI’s bottleneck is data, and the bottleneck is structural: it will not yield to a better model, only to a better way of producing data. Axis is that production system—a compounding data engine in which diversity is generated on demand, collected in the browser without hardware, expanded into model-ready training data, and continuously re-targeted by the failures of the policies it trains. One platform and one community produce both of the datasets modern robot learning needs: diverse demonstrations to establish competence, and on-policy corrections to establish reliability. The headline result—LIBERO-Plus success rising from 83.9% to 88.8%, scaling with data and clearing a volume-matched control by 31.3 points—is early evidence that the loop works. As it runs, each contribution makes the next one more valuable. That is what it means for data to compound.

A Post-Training Session Protocol

Algorithm 1 gives the client-side post-training session, corresponding to the gating rule of Equation (1) and the intervention budget of Section 7.

B Correction Episode Record

A correction episode record contains the seeded initial state and full initial simulator snapshot; per-tick robot joint positions and velocities and full simulator state at the adaptive control rate; the per-tick executed action a_t ; a per-tick control-mode label $c_t \in \{policy, human\}$; and metadata (task identifier, embodiment, timestamps) in the unified trajectory format. Records

Algorithm 1 Post-training session (client-side).

Require: task τ with deployed policy π_θ ; budget $T_{\max}$

- 1: fetch π_θ via the task’s policy manifest (ONNX)
- 2: initialize simulator from seeded state s_0 ; record seed and initial snapshot
- 3: $B \leftarrow T_{\max}$; $g \leftarrow 0$ ▷ remaining budget; gate
- 4: **while** not terminated (success $\Phi=1$, failure, or timeout) **do**
- 5: $g \leftarrow$ contributor’s gate (TAKE OVER / RETURN TO POLICY)
- 6: **if** $g = 1$ **then** ▷ human in control
- 7: $a_t \leftarrow \pi_H(s_t)$; $B \leftarrow B - \Delta t$
- 8: **if** $B \leq 0$ **then** fail episode; **break**
- 9: **end if**
- 10: **else** ▷ policy in control
- 11: $a_t \leftarrow \pi_\theta(o_t)$ ▷ in-browser inference
- 12: **end if**
- 13: $s_{t+1} \leftarrow f(s_t, a_t)$; record (o_t, a_t, c_t)
- 14: **end while**
- 15: package episode (seed, snapshot, segments, metadata); upload
- 16: **server:** re-simulate from seed; accept iff physically consistent and Φ verified on replay

are resampled to a uniform frequency for policy training.

References

- [1] M. Zhao, D. Huang, Y. Tang, P. Li, M. Yan, R. Zhuang, Y. Huang, J. Wang, H. Zhai, T. Zhou, R. Zhang, Z. Luo, Y. Huang, J. Yang, and J. Li, “AXIS: A growable community-driven data engine for scalable robot manipulation,” *arXiv preprint arXiv:2607.21588*, 2026. [Online]. Available: <https://arxiv.org/abs/2607.21588>
- [2] A. Brohan *et al.*, “RT-2: Vision-language-action models transfer web knowledge to robotic control,” in *Conference on Robot Learning (CoRL)*, 2023, arXiv:2307.15818.
- [3] M. J. Kim *et al.*, “OpenVLA: An open-source vision-language-action model,” in *Conference on Robot Learning (CoRL)*, 2024, arXiv:2406.09246.
- [4] K. Black *et al.*, “ π_0 : A vision-language-action flow model for general robot control,” *arXiv preprint arXiv:2410.24164*, 2024.
- [5] Open X-Embodiment Collaboration, “Open X-Embodiment: Robotic learning datasets and RT-X models,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6892–6903, arXiv:2310.08864.
- [6] A. Brohan *et al.*, “RT-1: Robotics transformer for real-world control at scale,” in *Robotics: Science and Systems (RSS)*, 2023, arXiv:2212.06817.
- [7] S. Ross and J. A. Bagnell, “Efficient reductions for imitation learning,” in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 661–668.
- [8] Axis Robotics Team and collaborators, “RoboVerse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning,” in *arXiv preprint*, 2025, arXiv:2504.18904.

- [9] S. Ross, G. J. Gordon, and J. A. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, arXiv:1011.0686.
- [10] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, “HG-Dagger: Interactive imitation learning with human experts,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, arXiv:1810.02890.
- [11] M. Pan, S. Feng, Q. Zhang, X. Li, J. Song, C. Qu, Y. Wang, C. Li, Z. Xiong, Z. Chen, Y. Liu, and J. Luo, “SOP: A scalable online post-training system for vision-language-action models,” *arXiv preprint arXiv:2601.03044*, 2026.
- [12] Physical Intelligence, “ $\pi_{0.6}^*$: A VLA that learns from experience,” *arXiv preprint arXiv:2511.14759*, 2025.
- [13] R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg, “Fleet-Dagger: Interactive robot fleet learning with scalable human supervision,” in *Conference on Robot Learning (CoRL)*, 2022, arXiv:2206.14349.
- [14] C. Chi, Z. Xu, S. Feng *et al.*, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, 2024, arXiv:2303.04137. RSS 2023; extended journal version.
- [15] ONNX Runtime developers, “ONNX Runtime,” <https://onnxruntime.ai>, 2021.
- [16] C. F. Ruan *et al.*, “WebLLM: A high-performance in-browser LLM inference engine,” *arXiv preprint arXiv:2412.15803*, 2024.
- [17] R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg, “ThriftyDagger: Budget-aware novelty and risk gating for interactive imitation learning,” in *Conference on Robot Learning (CoRL)*, 2021, arXiv:2109.08273.
- [18] C. Celemin, R. Pérez-Dattari *et al.*, “Interactive imitation learning in robotics: A survey,” *Foundations and Trends in Robotics*, vol. 10, no. 1–2, pp. 1–197, 2022, arXiv:2211.00600.
- [19] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, “Human-in-the-loop imitation learning using remote teleoperation,” *arXiv preprint arXiv:2012.06733*, 2020.
- [20] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu, “Robot learning on the job: Human-in-the-loop autonomy and learning during deployment,” in *Robotics: Science and Systems (RSS)*, 2023, arXiv:2211.08416.
- [21] A. Mandlekar, Y. Zhu, A. Garg *et al.*, “RoboTurk: A crowdsourcing platform for robotic skill learning through imitation,” in *Conference on Robot Learning (CoRL)*, 2018, arXiv:1811.02790.
- [22] S. Mirchandani, D. D. Yuan, K. Burns, M. S. Islam, T. Z. Zhao, C. Finn, and D. Sadigh, “RoboCrowd: Scaling robot data collection through crowdsourcing,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, arXiv:2411.01915.
- [23] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, “LIBERO: Benchmarking knowledge transfer for lifelong robot learning,” in *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023, arXiv:2306.03310.
- [24] 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,”

- in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, arXiv:1703.06907.
- [25] S. Deng, M. Yan, S. Wei, H. Ma, Y. Yang, J. Chen, Z. Zhang, T. Yang, X. Zhang, W. Zhang, H. Cui, Z. Zhang, and H. Wang, “GraspVLA: a grasping foundation model pre-trained on billion-scale synthetic action data,” *arXiv preprint arXiv:2505.03233*, 2025.
- [26] Y. Kuang, H. Geng, A. Elhafsi, T.-D. Do, P. Abbeel, J. Malik, M. Pavone, and Y. Wang, “SkillBlender: Towards versatile humanoid whole-body loco-manipulation via skill blending,” *arXiv preprint arXiv:2506.09366*, 2025.
- [27] Y. Tang, H. Geng, J. Jia, Y. Hu, S. Zang, J. Yang, P. Abbeel, and J. Malik, “DIPOL: Fusing vision and geometry for robust visuomotor generalization,” *arXiv preprint arXiv:2511.22445*, 2025, revised May 2026 (v2). [Online]. Available: <https://arxiv.org/abs/2511.22445>