

Awomo-SimDataEngine: Agentic Simulation-Ready World Generation

Awomo-PhysicalRSI Team



Generating useful robot-training data requires more than visually plausible scenes: objects must support interaction, placements must remain physically valid, and tasks must admit repeatable execution. We present **Awomo-SimDataEngine**, an agentic system that connects asset and scene generation to robot demonstration synthesis. Shared asset services provide rigid and articulated objects, including structure-grounded part and joint generation with ISArt. Scene generation supports two complementary routes: Unravel reconstructs editable scenes from images, while SimForge builds single-room and multi-room environments from text. A graph-native harness coordinates construction, validation, and bounded repair, routing failures to the responsible module while retaining unaffected scene state. PolicyForge binds validated worlds to tasks and robot embodiments to produce replayable demonstrations. Evaluations cover asset geometry, scene quality, and downstream policy learning. On MuJoCo-based LIBERO-Plus, co-training with Isaac Sim demonstrations improves the overall success rate of a World-Action Model (WAM) from 77.17% to 89.43%. Goal and spatial success improve by 31.66 and 6.25 percentage points, respectively. These results support the utility of the generated data for cross-simulator policy training, with more limited gains on long-horizon tasks.

Keywords: agent harness, graph-native execution, multimodal world compilation, simulation-ready worlds, embodied AI

1. Introduction

Robot learning depends on demonstrations that cover varied objects, environments, and manipulation goals. Collecting such coverage in the real world is costly, while manually preparing each simulated environment limits the scale and diversity of synthetic data. Generative models offer a way to expand

this training distribution, but their outputs must support interaction: a plausible image or textured mesh can still hide invalid collision geometry, inaccessible interiors, incorrect joints, unstable placements, or an unreplayable initial state. These defects affect both whether demonstrations can be collected and what policies learn from them.

Recent progress spans image-conditioned 3D assets, agentic scene construction, and automated demonstration synthesis [Tencent Hunyuan3D Team, 2025a; SAM 3D Team et al., 2025; Pfaff et al., 2026; Xia et al., 2026; Mandlekar et al., 2023; Ranawaka et al., 2026]. Connecting these capabilities requires agreement on object identities, coordinate frames, physical representations, task requirements, and validation results. An asset replacement changes the geometry available for registration; a revised scene changes the reset and interaction conditions seen by a controller. An integrated system must preserve these dependencies and return failures to the stage that can resolve them.

We introduce **Awomo-SimDataEngine**, an agentic system for constructing simulation-ready worlds and converting them into robot demonstrations. The system connects asset generation, scene generation, and demonstration synthesis through a *Graph-Native Harness* for execution and repair. Scene generation supports both visual references and language descriptions: *Unravel* reconstructs editable scenes from single-view images, whereas *SimForge* builds single-room and multi-room environments from text. Both routes use shared asset services and produce worlds that can be checked, edited, and bound to robot tasks. Figure 1 illustrates the shared execution framework and the image-driven route.

From assets to interactive scenes. Scene construction begins with assets that are usable beyond rendering. The asset module combines retrieval and generation with common geometry, collision, and physical checks (Section 4). Articulated objects add a structural requirement: their parts and joints must describe an executable mechanism. *ISArt* addresses this through geometric-kinematic planning, structure-conditioned part generation, and articulation binding.

The two scene-generation routes use these assets differently (Section 5). *Unravel* progressively removes objects from a reference image to expose occluded regions, then uses accumulated point-map evidence to register and reassemble independently editable assets. *SimForge* instead plans room geometry, object requirements, and spatial arrangements from language, supporting both individual rooms and multi-room houses. For both routes, plausible appearance alone does not establish valid support, collision geometry, or articulation.

From validated worlds to demonstrations. Construction failures need not be repaired where they are detected. A missing asset calls for acquisition; an intersecting object calls for a placement change; an incorrect joint may require revising the articulated asset. The *Graph-Native Harness* (Section 6) makes these handoffs explicit through shared state, validation evidence, bounded retries, and checkpoints. It retains unaffected work and revalidates the dependencies of each edit before construction proceeds.

PolicyForge then binds an accepted world to a task, robot embodiment, and reset specification (Section 7). Scripted or motion-planned teachers execute in Isaac Sim, producing synchronized images, proprioception, and actions; successful validated episodes are retained for training. Scene variants, object substitutions, and initial-state and language augmentation diversify the resulting data. World validity, demonstration success, and downstream policy performance remain distinct outcomes.

Experimental evidence. Module evaluations support the geometry and reconstruction methods: *ISArt* achieves part-geometry CD 0.163 versus 0.268 for the combined TRELIS.2 + P3-SAM + X-Part

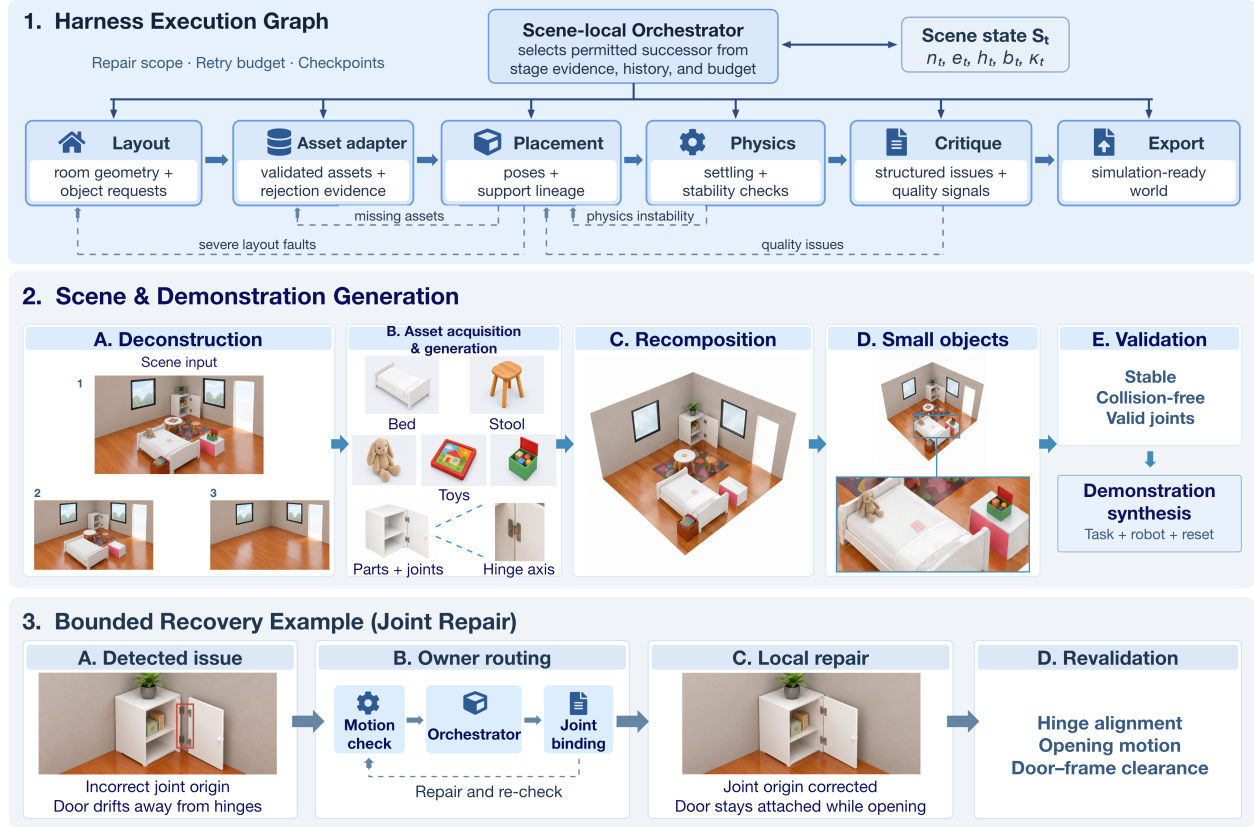


Figure 1: Awomo-SimDataEngine overview. (1) The graph-native harness coordinates execution, validation, and bounded repair. (2) The illustrated image-driven route uses Unravel to reconstruct scenes using acquired or generated assets, including articulated assets generated by ISArt. PolicyForge synthesizes demonstrations from validated worlds with task, robot, and reset specifications. (3) A schematic example illustrates failure detection and owner routing, followed by joint-origin correction and motion revalidation.

baseline on 60 objects, while Unravel improves blind human asset-fidelity scores from 4.9 to 8.6 on 160 objects. These protocols are detailed in Sections 4.1 and 5.1.3. Text-driven scene quality is evaluated separately on 179 room prompts and 31 multi-room prompts using the SceneEval and SceneSmith protocols (Section 5.2). These evaluations test complementary properties rather than a single pooled quality score. On the MuJoCo-based LIBERO-Plus benchmark, SimData-Ultra co-training increases World-Action Model success from 77.17% to 89.43% when comparing each configuration’s best evaluated checkpoint up to and including 13K steps, selected by overall success, with gains of 31.66 and 6.25 percentage points in goal and spatial generalization (Section 8). These results support the training utility of the combined data-generation system across simulator boundaries; they do not isolate each generator or augmentation’s contribution. The smaller long-horizon gain motivates richer sequential task supervision.

This report makes four contributions:

1. **Simulation-ready asset generation**, including ISArt’s structure-grounded articulated-object

pipeline and shared candidate validation, canonicalization, collision construction, and physical augmentation.

2. **Complementary scene-generation routes:** image-driven reconstruction with Unravel and text-driven single-room and multi-room construction with SimForge.
3. **Integrated execution and demonstration synthesis:** a Graph-Native Harness coordinates attributed repair and acceptance, while PolicyForge generates replayable demonstrations across scenes, tasks, and robot embodiments.
4. **Cross-simulator policy-learning evidence,** showing a 12.26-point overall gain on LIBERO-Plus at the best evaluated checkpoints within a 13K-step budget, with the largest improvement in goal generalization.

2. Related Work

Awomo-SimDataEngine connects interactive asset generation, text- and image-driven scene construction, agentic execution, and demonstration synthesis. We position these components against the corresponding literature and distinguish world-construction capabilities from downstream policy measurements.

2.1. 3D assets and interactive environments

Hunyuan3D, TRELIS, and SAM 3D generate geometry and appearance from visual conditions [Tencent Hunyuan3D Team, 2025a,b; Xiang et al., 2024; SAM 3D Team et al., 2025]. Their outputs supply candidate assets for Awomo-SimDataEngine, which additionally checks canonical orientation, measured extents, collision compatibility, and physical attributes before simulator use. For articulated objects, Section 4 introduces ISArt: structural planning conditions part generation, and agentic binding and motion verification connect the parts to an executable kinematic representation.

PartCrafter and OmniPart address part-level geometry generation, while Articulate AnyMesh, Articulate-Anything, and SPARK estimate or reconstruct articulated representations [Lin et al., 2025; Yang et al., 2025b; Qiu et al., 2025; Le et al., 2025; He et al., 2026]. ISArt connects explicit structural planning to a learned multi-part generator and then refines articulation through simulation feedback. Its fixed-geometry binding comparison separates joint estimation from the quality of generated meshes; the part-generation comparison evaluates the latter independently.

HSSD-200, RoboCasa, and BEHAVIOR-1K provide reusable assets, scenes, or household interaction settings [Khanna et al., 2023; Nasiriany et al., 2024; Li et al., 2024a]. They motivate the interaction requirements that generated assets must satisfy. Awomo-SimDataEngine exposes retrieval and generation through a common asset lifecycle, preserving source provenance separately from instance poses, articulation states, and simulator-specific packaging.

2.2. Scene synthesis and single-view construction

ATISS and DiffuScene learn distributions over indoor object arrangements; InstructScene, LayoutGPT, and PhyScene incorporate instructions, structured relations, or physical constraints [Paschalidou et al., 2021; Tang et al., 2023; Lin and Mu, 2024; Feng et al., 2023; Yang et al., 2024]. These approaches provide useful priors for scene composition. Awomo-SimDataEngine’s image-driven path follows Unravel (Section 5), using object removal to expose occluded regions and fixed-camera point evidence to guide reconstruction and recomposition. Its execution graph records stage dependencies; removal order provides local occlusion evidence rather than a global support or depth order.

Text2Room, GraphDreamer, DreamScene, AnyHome, WonderWorld, and MIDI explore text-, graph-, or image-conditioned 3D scene construction [Höllein et al., 2023; Gao et al., 2023; Li et al., 2024b; Fu

Table 1: System positioning. Conditions, construction procedures, and evaluation settings differ; this comparison describes scope rather than a matched performance ranking.

System	World construction	Connection to policy learning
EmbodiedGen V2	Simulation-ready assets, affordances, task-driven assembly, and editable worlds	Simulator deployment and embodied policy training
SimFoundry	Video-based digital twins and object, scene, and task cousins	Real-to-sim policy evaluation and sim-to-real learning
Awomo-SimDataEngine	ISArt-assisted assets, Unravel image reconstruction, and SimForge text-driven scenes, coordinated by a shared harness	PolicyForge demonstrations in Isaac Sim; measured co-training utility on MuJoCo benchmarks

et al., 2023; Yu et al., 2025; Huang et al., 2024]. Awomo-SimDataEngine uses SimForge for text-driven single-room and multi-room construction, and Unravel for single-image reconstruction. The latter retains object identities, masks, point provenance, and registration evidence. Settling and visual audit check the recomposed scene before packaging. This image-driven route does not imply multi-view reconstruction or uniquely recoverable metric geometry from an unconstrained image.

2.3. Agentic generation and execution harnesses

ReAct, Reflexion, Tree of Thoughts, and Graph of Thoughts make tool use, feedback, or structured reasoning explicit [Yao et al., 2022; Shinn et al., 2023; Yao et al., 2023; Besta et al., 2023]. Holodeck, SceneCraft, SceneWeaver, SceneSmith, SAGE, and SceneCode connect language-model reasoning to 3D scene tools or executable representations [Yang et al., 2023; Hu et al., 2024; Yang et al., 2025a; Pfaff et al., 2026; Xia et al., 2026; Wang et al., 2026a].

Awomo-SimDataEngine coordinates specialized modules through the execution contract in Section 6. Each repair records typed state, evidence, the responsible module, affected workset, remaining budget, and checkpoint decision. The contract spans ISArt articulation refinement, Unravel editing and reconstruction, and world-owned PolicyForge feedback. Preserved state is reused only when its input dependencies remain valid. These mechanisms support recovery within a run; the policy experiments do not independently establish harness efficiency or learning in the orchestrator.

2.4. Integrated worlds and policy-data synthesis

EmbodiedGen V2 connects simulation-ready assets, affordances, task-driven worlds, editing, and simulator deployment [Wang et al., 2026b]. SimFoundry reconstructs digital twins from video and generates object, scene, and task cousins for policy learning and evaluation [Ranawaka et al., 2026]. WorldComposer also connects visual reconstruction and digital cousins to robot learning [Lu et al., 2026]. These systems demonstrate the importance of linking world representations to interaction and downstream training.

MimicGen expands demonstrations from a small set of human trajectories, RoboMimic studies learning from offline demonstrations, and Diffusion Policy models visuomotor actions through diffusion [Mandlekar et al., 2023, 2021; Chi et al., 2023]. PolicyForge complements this work by preparing worlds and task contracts for scripted or motion-planned teachers, recording replayable resets and synchronized observations and actions, and retaining successful demonstrations. Its contribution is the generation and validation pipeline rather than a new policy optimizer.

SceneEval and RelScene evaluate semantic coherence and spatial relations [Tam et al., 2025; Ye et al.,

2024]. World-level checks and policy success answer different questions. Section 8 evaluates whether Awomo-SimDataEngine data improves a WAM policy across simulator boundaries; it does not rank generators under a common scene benchmark or establish real-world transfer. The system connects asset structure, image-grounded scene evidence, bounded repair, and validated demonstration synthesis in one workflow.

3. Overview

Awomo-SimDataEngine connects simulation-ready asset generation, text- and image-driven scene generation, and automated demonstration synthesis through a Graph-Native Harness. Sections 4–5 describe ISArt, Unravel, and SimForge; Sections 6–7 describe their orchestration and the PolicyForge interface. Module-level asset and scene evaluations are reported separately from downstream policy co-training results.

Input and scene-generation routes. The input is $C = \{T_{\text{text}}, I, Q\}$: a scene description, one image, and an optional task request. At least one of text or image is supplied. SimForge constructs scenes from text without requiring a reference image. Unravel reconstructs scenes from one single-view image, optionally accompanied by a description; this description may also be derived from the image. Object-centric and optional open-state references used inside ISArt are derived guidance for asset generation, not extra input scene views. Without Q , the output is a scene-only world; otherwise a task contract can be bound for demonstration synthesis.

Asset generation. Object requests are served by retrieval, image-to-3D models, specialized generation, or procedural modeling. ISArt converts a single object image into textured part meshes, a kinematic tree, and joint parameters through structural planning, structure-grounded generation, articulation binding, and simulation validation. All acquisition branches share candidate validation, canonical transforms, collision construction, physical augmentation, and asset-source provenance.

Scene generation. Unravel’s training-free plan–edit–verify loop progressively removes objects from the input scene image. Before each removal, crops and masks guide object extraction. Fixed-camera point maps are aligned using unedited anchors, fused only where new surfaces are exposed, and attributed to object instances by pixel and removal-round provenance. Recomposition starts from the room shell and proceeds in reverse removal order, followed by constrained registration, support-aware settling, and visual audit. The reconstructed scene contains independently editable assets and their instance transforms. SimForge instead plans room geometry and object placement from text, supporting single-room and multi-room scenes. Both routes use shared asset services and undergo validation before downstream use.

Graph-Native Harness. The execution graph tracks typed state, validation evidence, the responsible module, repair scope, attempt budget, and checkpoints. Stage dependencies determine which records become invalid after an edit. Hard validation and simulator packaging govern world acceptance; a saved render or mesh is insufficient. The harness coordinates module-specific repair while retaining the provenance needed for replay.

Policy generation and evaluation. PolicyForge binds an accepted world to a task, robot embodiment, and reset specification. Scripted or motion-planned teachers produce synchronized observations, states,

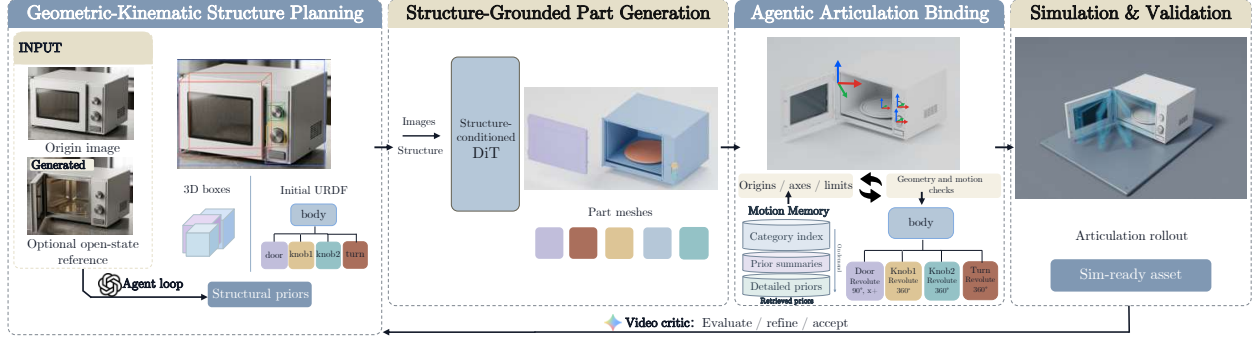


Figure 2: ISArt articulated-asset generation. Structural planning supplies part and kinematic priors for learned geometry generation. Agentic binding constructs a URDF, and rendered simulation feedback guides parameter repair, structural re-planning, or geometry regeneration.

and actions; successful episodes form demonstrations. Scene and task cousins, with initial-state and language augmentation, diversify policy-training data. The measured downstream endpoint in this report is WAM co-training in Isaac Sim-generated environments and evaluation on MuJoCo-based LIBERO benchmarks. World construction, episode success, and benchmark policy success remain distinct validation outcomes.

4. Asset Generation

Asset generation converts an object request into a validated simulator asset. Given text, an image, or both, the module obtains candidates through retrieval, image-to-3D generation, specialized generation, or procedural modeling. Object-centric references remove distracting scene content and provide appearance and structural cues. Each branch returns geometry and available semantics, orientation, scale, articulation, and physical information to the shared validation process in Section 6.

Acquisition branches. CLIP-based retrieval selects candidates from reusable asset databases. Image-to-3D models such as Hunyuan3D, TRELIS.2, and SAM 3D provide static geometry from object-centric references or scene images and masks [Tencent Hunyuan3D Team, 2025a; Xiang et al., 2026; SAM 3D Team et al., 2025]. Specialized branches handle assets requiring additional structure. For articulated objects, our *ISArt* pipeline generates part meshes and executable kinematics jointly with agentic structural reasoning. Retrieved or procedurally modeled objects remain alternatives when generation fails the required checks.

4.1. Single-Image Articulated Assets with ISArt

Given one object-centric image I_0 , ISArt produces

$$\mathcal{A} = F(I_0) = (\mathcal{M}, \mathcal{K}, \Theta), \quad \mathcal{M} = \{M_i\}_{i=1}^N, \quad (1)$$

where $\mathcal{M}$ contains textured part meshes, $\mathcal{K}$ is the URDF kinematic tree, and Θ contains joint origins, axes, and ranges. The pipeline has four stages: geometric–kinematic structure planning, structure-grounded part generation, agentic articulation binding, and simulation validation (Figure 2).

Geometric–kinematic structure planning. A VLM predicts object category, part count, semantics, parent links, joint types, coarse axes, part bounding boxes, and cavity boxes. The joint vocabulary

comprises fixed, revolute, prismatic, and continuous joints; coarse axes use six canonical directions. A synthesized open-state reference exposes possible concealed interiors when needed. Coarse URDF prediction is iteratively critiqued, while part and cavity boxes are refined through estimate–project–critique updates with the static-body box held fixed. The resulting structure remains editable for later fallback.

Structure-grounded part generation. ISArt adapts a pretrained TripoSG geometry prior with a multi-part DiT [Li et al., 2025]. Each predicted part has its own latent group. Frozen visual encoders process the original and optional open-state image; seven part-aligned tokens encode category, semantics, joint attributes, parent relation, box center, box size, and cavity attributes. Two mechanisms condition denoising: *E* injects a pooled part-structure embedding into every DiT layer, and *C* cross-attends from geometry tokens to the full structural token sequence. The backbone retains within-part and across-part attention and visual cross-attention. A pretrained decoder produces each part mesh, followed by part-level texture synthesis. Structural modules and attention LoRA parameters are optimized with the rectified-flow objective, with structural residuals zero-initialized. Appendix B details the tokenization and conditioning equations.

Articulation binding and simulation validation. Part identifiers bind meshes to URDF links. Parent relations and coarse joint predictions initialize the tree; geometry and retrieved category-level motion priors guide refinement of axes, pivots, and ranges. ISArt’s hierarchical articulation memory can expand a motion summary into detailed joint guidance. This module-specific memory is separate from the harness’s run-local checkpoint state. In Isaac Sim, joint motions are rendered and a video critic checks direction, pivot, range, interference, and collisions. Feedback drives parameter edits, rollback, and retry; persistent failures escalate to structural planning or geometry regeneration. Invalid candidates remain rejected rather than silently becoming static objects.

Module-level evidence. On 60 articulated objects from two sources spanning 12 categories, ISArt reports part-geometry Chamfer Distance 0.163, compared with 0.268 for the TRELIS.2 + P3-SAM + X-Part pipeline [Xiang et al., 2026; Ma et al., 2025; Yan et al., 2025], and F-Scores 0.604 and 0.789 at thresholds 0.05 and 0.1. With fixed ground-truth part geometry, ISArt’s URDF binding has AxisErr 0.07, PivotErr 0.05, TypeErr 13.33%, and reported binding success 100%. This last measurement evaluates binding with supplied geometry, not end-to-end asset generation. Complete comparisons and training details are in Appendices B and E.1.

4.2. Standardization, Physical Augmentation, and Reuse

Accepted geometry is transformed into a declared canonical frame, retaining the source-to-canonical transform. Orientation and scale are estimated from the request and reference evidence; placement extents are measured from the standardized mesh. Collision construction uses convex decomposition and interaction-clearance checks. Small objects use VHACD; larger objects may be segmented and decomposed at part level. Mass, friction, and other available physical attributes remain associated with the asset.

Visual meshes, collision proxies, articulated link frames, and simulator representations retain auditable scale and transform relationships. The Asset Registry records accepted and rejected candidates, sources, validation outcomes, and reusable geometry separately from instance poses (Appendix B.4). Scene registration and support validation operate on the instantiated assets in Section 5; target-specific load and package checks complete the acceptance process.

5. Scene Generation

Awomo-SimDataEngine supports two parallel scene-generation routes. *Unravel* reconstructs an editable scene from a single image, optionally accompanied by a description; *SimForge* constructs single-room scenes and multi-room houses from text. Neither route requires the other as a preprocessing stage. Both use the asset services in Section 4, retain object and articulation identities, and produce worlds for validation and downstream demonstration synthesis.

5.1. Image-to-Scene Generation: Unravel

Unravel is a training-free scene-reconstruction procedure that deconstructs a single-view scene image before recomposing it as an interactive world. Its input is one scene image, with an optional description and task constraints as specified in Section 3. Unravel’s output is

$$\mathcal{W} = (\mathcal{E}, \{(a_i, T_i)\}_{i=1}^N), \quad (2)$$

where $\mathcal{E}$ is the room shell, a_i is an independent asset, and $T_i = (\mathbf{s}_i, R_i, t_i)$ stores scale, rotation, and translation. Figure 3 shows the two-stage pipeline.

5.1.1. Verified Layer-Wise Deconstruction

A VLM planner selects exposed objects for removal, protects the remaining objects, and preserves walls and the floor. Contents are removed before containers. An image editor removes targets and inpaints exposed regions; a verifier checks the edit before the next round. Right before removal, each object’s best-observed crop, semantic description, bounding box, instance mask, and removal round are recorded. A canonical object image guides the asset-generation branch in Section 4. Box localization uses a global prediction with independent local cross-checks (Appendix C.1).

The fixed camera is retained across rounds. Newly inpainted content is a generated hypothesis about hidden surfaces, not a newly acquired physical measurement; verification and downstream geometry checks remain necessary. Deconstruction ends when only the room shell remains or the harness records an unresolved failure.

Anchored point-map fusion. For round k , a monocular estimator returns point map P_k and confidence mask M_k , excluding unreliable depth-boundary pixels. Let E_j denote pixels edited in round j . Stable anchors are valid in both the first and current round and have never been edited:

$$\mathcal{A}_k = \{p : M_k(p) \wedge M_1(p), p \notin \bigcup_{j < k} E_j\}. \quad (3)$$

A similarity transform aligns P_k directly to P_1 using these anchors, with residual-based outlier rejection. Unedited pixels keep their existing points; fusion adds valid points only from the regions exposed by the preceding edit. This retains earlier object geometry while accumulating newly revealed surfaces.

Instance attribution. Every fused point retains its pixel and reconstruction round. For a point from round k , candidate objects are those whose masks cover its pixel and whose removal rounds satisfy $r_i \geq k$. The earliest surviving removal candidate identifies the frontmost object along that ray; an empty candidate set identifies the room shell. Object points form clusters C_i for registration. This is a per-ray rule, not a global depth ordering.

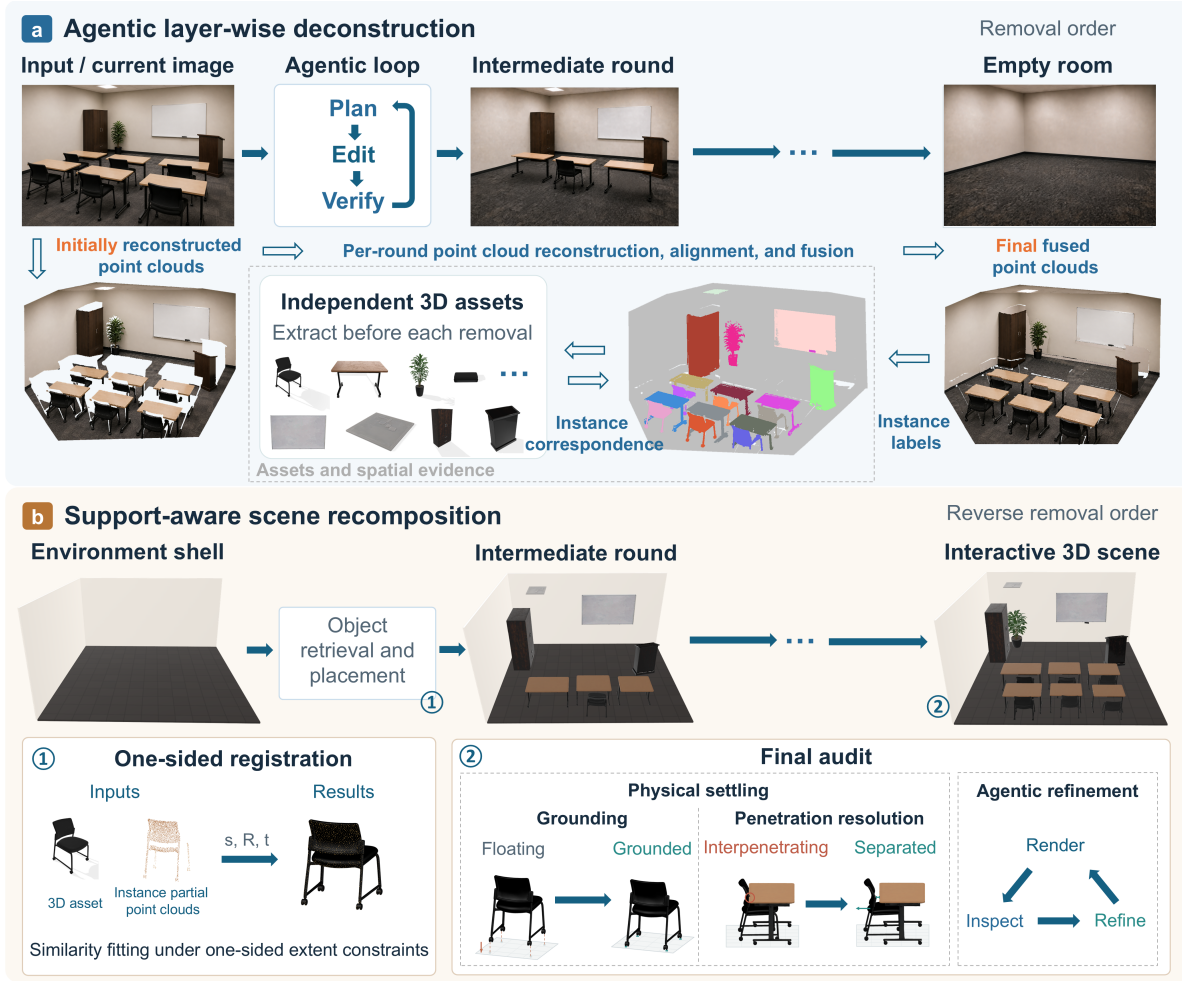


Figure 3: Unravel single-view scene generation. Verified removals expose objects and collect assets and point evidence. Reverse-order recomposition registers each asset to its attributed cluster, followed by support-aware settling and a final audit.

5.1.2. Registration and Support-Aware Recomposition

Assets are restored in descending removal round, beginning with the room shell. Contents-before-container planning favors placing supporting geometry before its contents. This order provides an assembly prior; support is checked against local geometry during settling rather than assumed from occlusion alone.

Registration from partial observations. For asset surface samples X_i and cluster C_i , transformations are ranked by a bidirectional median Chamfer distance:

$$D_i(T) = \frac{1}{2} \left[\text{med}_{x \in T(X_i)} \min_{y \in C_i} \|x - y\|_2 + \text{med}_{y \in C_i} \min_{x \in T(X_i)} \|y - x\|_2 \right]. \quad (4)$$

Truncated similarity ICP refines reliable correspondences. The median and truncation reduce the influence of neighboring-object fragments and spurious generated geometry. Along under-observed

Table 2: Scene-level image similarity against the reference photo. Mean over 20 scenes. Renders use the recovered photo viewpoint. **Bold** marks the better value per metric. Δ is the relative change of Unravel over GPT-6 Astra, computed from the displayed means.

Method	CLIP-I $\uparrow$	LPIPS $\downarrow$	Edge IoU $\uparrow$
GPT-6 Astra (Ultra/ExtraHigh)	0.803	0.649	0.107
Unravel (w/ finetuned GPT-6)	0.854	0.632	0.156
Δ (rel.)	+6.4%	−2.6%	+45.8%

axes $\mathcal{U}_i$, visible extent is treated as a lower bound:

$$\text{span}_u(T_i(X_i)) \geq \text{span}_u(C_i), \quad u \in \mathcal{U}_i. \quad (5)$$

A global scale and bounded object-frame axis adjustments preserve this constraint. Well-observed axes may contract, but partial observations cannot justify shrinking an asset below their measured span.

Physical settling and audit. Semantic relations and local support geometry distinguish mounted objects, which stay fixed, from movable objects, which may undergo observation-constrained translations δ_i . Settling minimizes total squared displacement subject to support, non-interpenetration, and observation-consistency constraints. Contact and observation-fit tolerances are specified in Appendix C.2. A render-and-inspect audit proposes local corrections, followed by renewed settling. Failed extraction, registration, or physical checks return to the responsible bounded stage in Section 6; the accepted scene retains independently editable assets and their observation and validation records.

5.1.3. Module-Level Reconstruction Evidence

A controlled comparison on five scenes containing 160 objects uses the same SAM 3 [Carion et al., 2025], TRELLIS.2, and MoGe3 [Kong et al., 2026] components for direct reconstruction and Unravel. Deconstruction increases mean VLM asset-fidelity scores from 4.7 to 8.2 and human scores from 4.9 to 8.6 on a 1–10 scale. Mean scene point count increases from 1.53M to 2.81M through fusion of four to eight rounds. Direct masks cover 77.2% of the Unravel mask area on average for objects removed after the first round. These coverage measurements use Unravel’s masks and clusters as references, and additional points may come from generated image completion; they are not independent geometric ground truth. Shared-asset qualitative comparisons also separate registration quality from mesh-generation quality. Appendix E.2 gives the full measurements, scoring protocol, and representative comparisons.

We further assess full-scene fidelity at the image level. For each generated scene, we first recover the photo’s camera pose. We run truncated ICP between surface samples of the generated geometry and the first-round MoGe3 point cloud. We then render the scene with the estimated intrinsics and compare the render with the photo. We use three metrics. **CLIP-I** is the cosine similarity of CLIP ViT-L/14 image embeddings and captures semantic content. **LPIPS** is a learned perceptual distance over VGG features. **Edge IoU** is the intersection-over-union of Canny edge maps with a two-pixel tolerance. It captures object layout and composition. We evaluate on **20 scenes**. Table 2 shows that Unravel beats GPT-6 Astra on all three metrics.

5.2. Text-to-Scene Generation: SimForge

SimForge translates a scene description into a single-room or multi-room world. The request specifies room types, required objects, spatial relations, and optional interaction requirements. Layout planning establishes room geometry, openings, and object requests; an asset adapter retrieves or generates compatible candidates. Placement instantiates these assets on measured supports, followed by physics checks, critique, and simulator packaging. A reference scene image is not required. In multi-room mode, room topology and inter-room connections provide the architectural context for room-specific furnishing and object placement.

5.2.1. Language-Guided Construction

The language model specifies object requirements and spatial arrangement intent; geometry-aware routines instantiate and validate candidate poses. Repeated small-object groups use procedural placement, while individually constrained items retain separate pose and support assignments.

Construction uses the registered layout, asset-adapter, placement, physics, critique, and export stages of the Graph-Native Harness (Section 6). Missing assets return to acquisition; placement or support faults return to the responsible placement scope; severe layout faults require architectural revision. Bounded routing and checkpoints retain unaffected work while the revised candidate is revalidated. These are shared execution mechanisms, not an additional image-reconstruction stage.

5.2.2. Reusable Assets and Scene Variants

Processed assets store shared geometry, appearance, collision, and articulation metadata. Scene instances separately retain pose, support, joint state, and task role. This separation supports local object insertion, appearance changes, and rigid or articulated substitution. Changed dependencies are rebound and affected supports, neighbors, and joints are rechecked before the edited world is accepted. Accepted worlds can then be bound to tasks and robots by PolicyForge; scene generation does not itself imply a successful robot demonstration.

5.2.3. Module-Level Scene-Quality Evidence

Room-level evaluation uses 179 prompts under SceneEval, while house-level evaluation uses 31 multi-room prompts under SceneSmith’s house-level protocol [Tam et al., 2025; Pfaff et al., 2026]. The two scales are reported separately. Table 3 summarizes semantic and physical quality. Relative to SceneSmith, SimForge improves count agreement by 6.5 percentage points at room level and 6.4 at house level, and accessibility by 14.7 and 13.3 points, respectively. Detected collision is 0% and stability is 100% under the stated checks. These are scene-quality measurements, not generation acceptance rates or isolated causal estimates of Harness effectiveness.

6. Graph-Native Harness for Execution and Repair

The Graph-Native Harness coordinates asset generation, Unravel reconstruction, SimForge scene construction, simulator packaging, and PolicyForge demonstration synthesis. Each execution edge specifies its inputs, responsible module, validation requirements, and records needed to resume or repair the result. Figure 1 shows these system connections. The execution graph organizes module calls and recovery for both SimForge’s text-driven construction and Unravel’s image-grounded recomposition.

Shared state and inputs. The input is $C = \{T_{\text{text}}, I, Q\}$ as defined in Section 3. SimForge receives a scene description; Unravel receives one scene image with an optional description. When Q is present, its entities, interaction requirements, reset assumptions, and success predicate are bound to the PolicyForge

Table 3: Text-to-scene quality under SceneEval and SceneSmith protocols [Tam et al., 2025; Pfaff et al., 2026]. Room-level and house-level cohorts are kept separate. Values are reported means $\pm$ 95% confidence intervals where available. SAGE[†] uses a separate 30-scene cohort and native collision/stability definitions; it is a reference, not a matched ranking against the other rows [Xia et al., 2026]. #Obj: object count; CNT: count agreement; ATR: attribute correctness; OOR: object-object relations; OAR: object-architecture relations; ACC: accessibility; NAV: navigability; COL: collision; STB: stability; OOB: out-of-bounds. SimForge is the text-driven scene module of Awomo-SimDataEngine.

Method	#Obj	CNT $\uparrow$	ATR $\uparrow$	OOR $\uparrow$	OAR $\uparrow$	ACC $\uparrow$	NAV $\uparrow$	COL $\downarrow$	STB $\uparrow$	OOB $\downarrow$
<i>Room-level generation (179 prompts)</i>										
HSM [Pun et al., 2025]	22.7 $\pm$ 2.6	60.6 $\pm$ 4.2	61.5 $\pm$ 7.3	30.6 $\pm$ 5.7	66.2 $\pm$ 6.6	88.9 $\pm$ 1.7	99.4 $\pm$ 0.4	20.6 $\pm$ 3.0	45.2 $\pm$ 3.9	5.5 $\pm$ 1.1
Holodeck [Yang et al., 2023]	23.0 $\pm$ 1.4	44.2 $\pm$ 4.3	44.4 $\pm$ 7.2	16.8 $\pm$ 4.6	38.0 $\pm$ 7.1	84.9 $\pm$ 1.4	99.6 $\pm$ 0.2	12.3 $\pm$ 2.3	31.9 $\pm$ 2.7	0.8 $\pm$ 0.4
SceneWeaver [Yang et al., 2025a]	13.5 $\pm$ 1.0	41.8 $\pm$ 3.9	29.1 $\pm$ 6.7	15.3 $\pm$ 4.2	31.6 $\pm$ 6.5	77.5 $\pm$ 2.7	98.1 $\pm$ 1.2	12.5 $\pm$ 2.5	37.3 $\pm$ 4.4	0.0 $\pm$ 0.0
I-Design [Çelen et al., 2024]	13.0 $\pm$ 1.0	69.3 $\pm$ 4.0	50.3 $\pm$ 7.2	28.6 $\pm$ 5.5	66.6 $\pm$ 6.7	70.1 $\pm$ 2.8	95.9 $\pm$ 1.4	3.0 $\pm$ 1.5	60.8 $\pm$ 4.6	4.3 $\pm$ 2.0
LayoutVLM (Cur.) [Sun et al., 2025]	11.2 $\pm$ 1.6	41.0 $\pm$ 4.0	25.6 $\pm$ 6.6	14.1 $\pm$ 4.0	22.4 $\pm$ 6.1	93.8 $\pm$ 1.9	99.7 $\pm$ 0.2	25.9 $\pm$ 4.4	19.4 $\pm$ 3.6	6.2 $\pm$ 2.1
LayoutVLM (Obj.) [Sun et al., 2025]	14.2 $\pm$ 1.7	55.5 $\pm$ 4.7	34.3 $\pm$ 7.0	20.7 $\pm$ 5.0	17.4 $\pm$ 5.6	91.5 $\pm$ 1.6	98.7 $\pm$ 0.8	28.9 $\pm$ 3.5	8.1 $\pm$ 1.7	5.1 $\pm$ 1.4
SceneSmith [Pfaff et al., 2026]	71.1 $\pm$ 13.0	82.9 $\pm$ 3.4	74.4 $\pm$ 6.2	67.6 $\pm$ 5.8	80.6 $\pm$ 5.5	83.4 $\pm$ 1.5	97.6 $\pm$ 1.1	1.2 $\pm$ 0.6	95.6 $\pm$ 1.7	0.2 $\pm$ 0.4
SimForge	67.6 $\pm$ 5.1	89.4 $\pm$ 4.2	76.8 $\pm$ 7.1	73.1 $\pm$ 5.4	79.3 $\pm$ 5.1	98.1 $\pm$ 1.4	98.4 $\pm$ 1.2	0.0 $\pm$ 0.0	100.0 $\pm$ 0.0	0.0 $\pm$ 0.0
SAGE [†] [Xia et al., 2026]	48.2	—	—	—	—	—	—	1.9 [†]	99.9 [†]	—
<i>House-level generation (31 prompts)</i>										
Holodeck [Yang et al., 2023]	81.3 $\pm$ 18.3	64.6 $\pm$ 6.8	43.2 $\pm$ 27.6	30.1 $\pm$ 10.1	58.4 $\pm$ 6.3	79.4 $\pm$ 3.0	99.6 $\pm$ 0.2	3.8 $\pm$ 3.1	17.9 $\pm$ 5.6	0.3 $\pm$ 0.2
SceneSmith [Pfaff et al., 2026]	214.1 $\pm$ 60.9	83.3 $\pm$ 6.8	66.8 $\pm$ 24.6	76.6 $\pm$ 9.2	82.6 $\pm$ 5.8	79.9 $\pm$ 2.0	96.5 $\pm$ 2.1	0.9 $\pm$ 0.5	79.8 $\pm$ 9.1	0.6 $\pm$ 0.3
SimForge	242.3 $\pm$ 32.9	89.7 $\pm$ 6.4	70.3 $\pm$ 26.7	78.9 $\pm$ 10.3	81.9 $\pm$ 6.1	93.2 $\pm$ 2.7	98.1 $\pm$ 1.3	0.0 $\pm$ 0.0	100.0 $\pm$ 0.0	0.0 $\pm$ 0.0

task contract $\mathcal{T}$ (Section 7.2). Scene-only requests carry no policy task. State records include asset candidates and canonical transforms, ISArt part and joint identities, SimForge layouts and placement records, Unravel masks and removal rounds, point-cloud provenance, registrations, physical checks, export manifests, retry budgets, and checkpoints. World versions link these records to downstream reset and rollout records.

Validation at module boundaries. Asset validation checks semantic match, usable geometry, orientation and scale, collision clearance, and required articulation (Section 4). For ISArt, rendered motion supplies evidence about joint axes, pivots, ranges, and interference between parts. Unravel verifies each removal edit and checks registration against attributed point clusters and the observed extent of partially occluded objects (Section 5). Settling and visual audit assess the assembled world. Packaging checks coordinate consistency, required files, and simulator loadability. Each boundary returns a candidate and evidence, retaining failed alternatives even when a later attempt succeeds.

Explicit execution and repair transitions. Each attempt separates observation, diagnosis, owner attribution, action, validation, candidate selection, and routing. The transition contract is formalized in Eq. (10). The selected module returns a revised candidate for validation; the decision advances execution, retries within budget, restores a checkpoint, or records terminal failure. Visual critique can guide refinement but cannot override a failed required geometric, physical, task, or export check.

Evidence-driven owner selection. An unsuitable mesh returns to acquisition or geometry generation; an incorrect joint returns to ISArt binding or structural planning. An inconsistent removal plan or edit returns to the affected Unravel round. Point attribution or alignment errors return to reconstruction or registration, while penetration or unstable placement returns to recomposition and settling. Packaging errors remain with export unless evidence identifies an upstream cause. Records name affected entities, violated requirements, supporting artifacts, and proposed owners. Ambiguous causes trigger further

inspection and remain explicit rather than receiving an unsupported diagnosis.

Repair scope and dependent-state invalidation. A repair affects an asset, part, instance, image region, removal round, point cluster, or local placement workset. Replacing a mesh invalidates its registration and physical checks on dependent instances. Revising an early image edit can alter later removal images and fused point evidence, requiring the affected downstream rounds to be reconstructed and checked again. A pose change requires renewed collision, settling, and task-clearance checks for affected objects. Other state is preserved only when its inputs remain valid. Nested dispatch applies this rule within specialized modules and across their boundaries.

Checkpoints and bounded recovery. The last validated checkpoint is retained before repair. A candidate is selected only after required checks pass; improved appearance cannot compensate for a broken joint, penetration, or missing task object. Failed attempts consume the budget for their scope and may restore the checkpoint. Exhaustion terminates the affected request with failure evidence. Optional-object removal or another relaxation is recorded as degradation and cannot satisfy an unchanged requirement. These mechanisms describe recovery within a run rather than weight updates or guaranteed cross-run improvement.

World acceptance and demonstration admission. A world is accepted after its applicable semantic, geometric, physical, and task requirements pass and a valid simulator package is produced (Appendix A.11). Scene-only requests produce reconstructed scenes; promotion to an Accepted World additionally requires a task contract and successful task validation. PolicyForge binds the accepted world to an embodiment and reset specification, executes a teacher, and records observations, states, and actions. World acceptance and episode success are separate: a valid world can still yield a failed controller rollout. Only episodes satisfying the task-success predicate enter the dataset.

Feedback, provenance, and reuse. PolicyForge returns world and task identities, the seed, affected entities, and first blocker with a failed rollout (Section 7.4). Missing affordances, invalid geometry, or world-owned physical failures request repair of the corresponding world module; controller-owned failures remain with PolicyForge. A world revision invalidates dependent resets and rollouts and must pass the same acceptance gates before fresh demonstrations are admitted. Asset sources, world versions, robot adapters, resets, and augmentation configurations remain associated with accepted episodes so scene and task cousins can be traced to their collection conditions.

Appendix A supplies extended state and transition definitions. Section 8 measures the utility of the co-training data; it does not independently measure repair efficiency or the harness contribution.

7. Policy Generation via Automated Demonstration Synthesis

Given a validated Awomo-SimDataEngine world and a task contract, we propose **PolicyForge**, which provides a fully automated pipeline that converts a user-level task specification into large-scale, multi-embodiment demonstration data ready for policy learning. The pipeline operates within NVIDIA Isaac Sim and is designed to be embodiment-agnostic: it supports a growing roster of robotic platforms including Franka, Piper dual-arm, Agibot G1, and Agibot G2, each interacting with diverse objects across procedurally generated scenes.

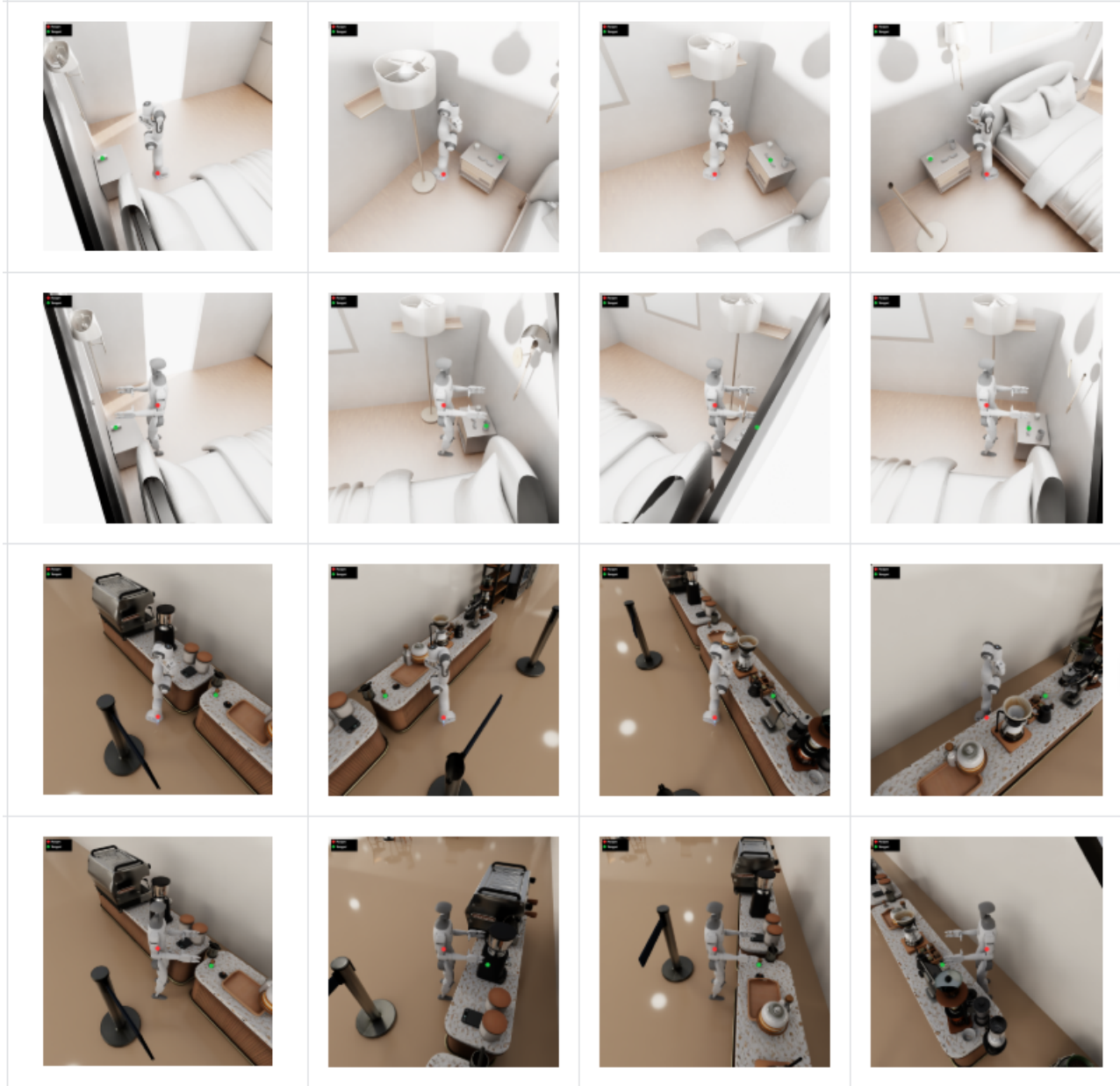


Figure 4: Qualitative examples of PolicyForge robot interactions across different embodiments and scene configurations. The 4×4 grid shows bedroom scenes in the top two rows and coffee-shop scenes in the bottom two rows. Examples illustrate variation in robot embodiment, object arrangement, and viewpoint.

7.1. Pipeline overview

PolicyForge accepts three inputs: (i) a validated scene generated through either Unravel or SimForge with retrieved, generated, or procedurally modeled assets, (ii) a task description specifying goal predicates, interaction requirements, and success/termination conditions, and (iii) an embodiment record that fixes the robot model, sensor suite, and action parameterization. A target episode count N specifies the requested generation scale. Figure 5 summarizes four stages:

1. **Task analysis and decomposition.** An LLM analyzes the scene and task description and, when needed, decomposes the task into an ordered sequence of atomic tasks $A_1, \dots, A_k$. An already atomic task is represented by a single element ($k = 1$).
2. **Controller routing.** A task manager selects an execution backend for each atomic task according to the task requirements and robot embodiment. Available backends include an inverse-kinematics solver, cuRobo motion planning, an RL policy, and custom scripts. The selected backend is specific to the data-generation configuration.
3. **Execution and episode assembly.** The scene graph is instantiated in Isaac Sim with physics-consistent placements, support/containment relationships, and randomized nuisance parameters (lighting, textures, object poses). A reset controller samples a robot and object configuration satisfying the task preconditions and records the simulator state for replay. The selected backends execute the atomic-task sequence while recording multi-camera RGB observations, proprioceptive state, and end-effector actions at the control rate. These synchronized streams are assembled into an episode.
4. **Validation and export.** Each episode is validated against the task-success predicate; only successful episodes are retained. The accepted episodes are exported in LeRobot format with aligned image, state, and action streams for downstream policy learning.

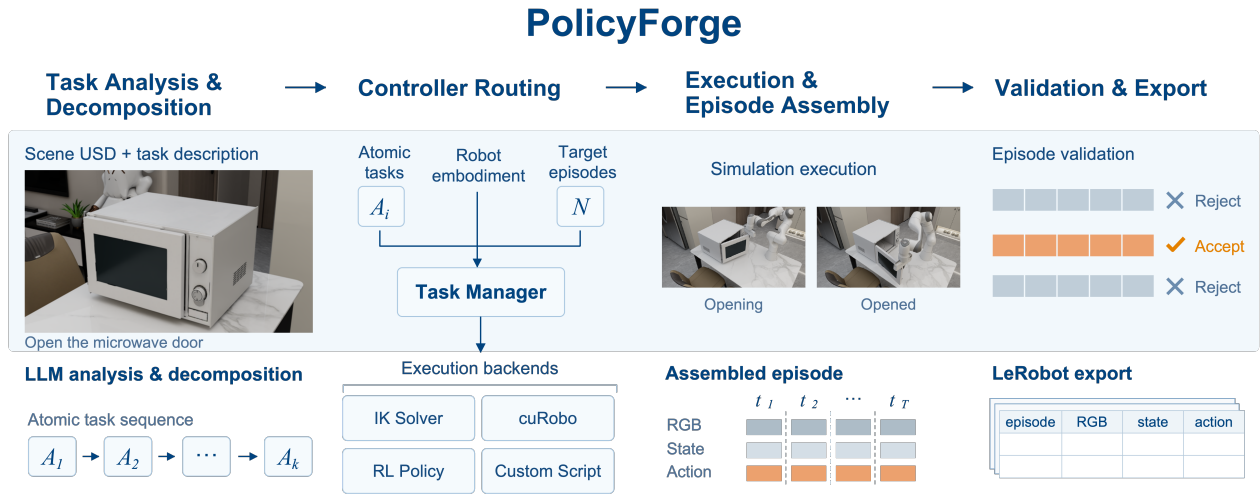


Figure 5: PolicyForge pipeline. Given a scene, task description, robot embodiment, and target episode count N , an LLM analyzes the task and decomposes it into atomic tasks when needed. A task manager selects suitable execution backends. Simulation execution in Isaac Sim produces synchronized observations, states, and actions, which are assembled into episodes. Episodes that pass task-success validation are exported in LeRobot format. The microwave images illustrate an ISArt input asset and its simulated manipulation.

7.2. Task contract

The pipeline consumes a validated tuple

$$(\mathcal{W}, \mathcal{T}, R_{\text{bot}}, \Sigma),$$

where $\mathcal{W}$ is the compiled scene, $\mathcal{T}$ encodes the task goal, entities, interaction requirements, and success/termination predicates, R_{bot} specifies the embodiment and observation/action interface, and Σ specifies reset state, randomization ranges, episode horizon, and seed. The contract is expressed over exported object, support, container, and articulation identities rather than hidden generator internals, ensuring full reproducibility and downstream verifiability.

7.3. Multi-embodiment adaptation

A key design principle of PolicyForge is embodiment decoupling: the scene and task specification are independent of the robot. An embodiment adapter binds kinematics, sensors, gripper type, control rate, and action parameterization to the shared world and task record. This enables the same scene–task pair to generate demonstrations for heterogeneous platforms—from industrial Franka arm setups to humanoid-scale systems—without modifying the upstream scene graph. The adapter also normalizes action and proprioceptive representations so that data produced for different embodiments can be mixed at training time after a lightweight convention alignment step.

7.4. Failure feedback

The failure schema classifies rollout failures into six categories—task specification, missing affordance, geometry/support, physics, reset, and controller—when the registered manifest supplies sufficient evidence. A world-owned failure feeds back the affected entities, first blocker, and a requested change to the smallest responsible graph edge to the Awomo-SimDataEngine harness for automated scene repair. Every requested revision must pass the same validation and strict export gates before contributing new rollouts, preventing regression loops.

7.5. Data cousins: scaling diversity along scene and task axes

Inspired by the digital-cousin paradigm, PolicyForge systematically generates data variants—*cousins*—along two orthogonal axes:

- **Scene cousins.** The same task executed in visually and physically distinct environments (e.g., kitchen → bedroom → workshop), varying backgrounds, lighting, and distractor objects.
- **Task cousins.** Variations of the manipulation goal within the same scene (e.g., *pick mug* → *pick bowl* → *stack plates*), diversifying the interaction semantics while sharing the visual context.

This cousin-based decomposition is the basis for the data families evaluated in Section 8: SIMDATA-EASY (straightforward scenes and tasks), SIMDATA-LITE (simple scenes with challenging manipulation tasks), and SIMDATA-HARD (complex scenes paired with demanding task specifications), each representing a distinct point in the scene×task difficulty space. The SIMDATA-ULTRA co-training configuration additionally mixes data sources and applies robot initial-state perturbations and language-instruction augmentation, as specified in Section 8.1.

8. Experiments and Results

We evaluate whether PolicyForge demonstrations generated in Isaac Sim improve a World-Action Model (WAM) on the MuJoCo-based LIBERO-Plus benchmark [Fei et al., 2025]. The main comparison selects each configuration’s best evaluated checkpoint up to and including 13K steps by overall success, using the

same evaluation set of 1,003 episodes across object, spatial, goal, and long-horizon suites. This measures the downstream utility of the co-training data across simulator boundaries. Detailed checkpoint results and additional LIBERO-Pro [Zhou et al., 2025] evaluations are provided in Appendix E.3.

8.1. Setup

Policy and training protocol. WAM uses a video and action diffusion-transformer architecture initialized from Wan2.2-T12V-5B [Team Wan et al., 2025]. Inputs comprise front and wrist RGB views plus proprioception; outputs encode end-effector motion and gripper state. LIBERO-Plus experiments use 32 GPUs, global batch size 768, and the training recipe in Appendix D.3.2. The baseline uses only the benchmark demonstrations. Co-training configurations use weighted sampling with a reported 1:1 effective-frame ratio between benchmark and additional data. All headline comparisons use the best evaluated checkpoint up to and including 13K steps for each configuration. Selection uses overall success on the reported evaluation set; this is a best-observed comparison, not selection on a separate validation set. Suite scores come from the same selected checkpoint rather than separate per-suite maxima.

Data configurations. SIMDATA-EASY uses straightforward scenes and pick-and-place tasks; SIMDATA-LITE combines simpler scenes with more demanding manipulation tasks; SIMDATA-HARD combines richer scenes and task specifications. SIMDATA-ULTRA mixes SIMDATA-HARD, additional in-house demonstrations from complex scenes, and the benchmark demonstration source, with robot initial-state perturbations and language-instruction augmentation. It is a combined data configuration: the experiment does not isolate these interventions individually. Data-source records and sampling configuration are given in Appendix D.3.3.

8.2. Cross-Simulator Co-training Improves Overall Success

Among evaluated checkpoints up to and including 13K steps, SIMDATA-ULTRA achieves its best overall success of **89.43%** at 13K, compared with the baseline’s best **77.17%** at 9K: an improvement of **12.26 percentage points**. It also exceeds SIMDATA-HARD by **8.37 points**. All three other Awomo-SimDataEngine configurations improve on the baseline under this selection rule (Table 4 and Figure 6).

Table 4: WAM success on LIBERO-Plus at each configuration’s best evaluated checkpoint up to and including 13K steps, selected by overall success. The gain is measured against benchmark-only training in percentage points. All configurations use the same 1,003-episode evaluation set.

Co-training data	Selected step	Success (%)	Gain (pp)
Baseline	9K	77.17	—
SimData-Easy	13K	78.76	+1.59
SimData-Lite	13K	78.27	+1.10
SimData-Hard	13K	81.06	+3.89
SimData-Ultra	13K	89.43	+12.26

8.3. Goal Generalization Shows the Largest Gain

Figure 7 compares the baseline, Hard, and Ultra at their respective best-overall checkpoints up to and including 13K steps (9K, 13K, and 13K, respectively). With SIMDATA-ULTRA, goal generalization increases from **52.51%** to **84.17%** (**+31.66 points**), spatial generalization from **87.08%** to **93.33%** (**+6.25 points**), and object generalization from **84.52%** to **92.86%** (**+8.34 points**). Relative to Hard,

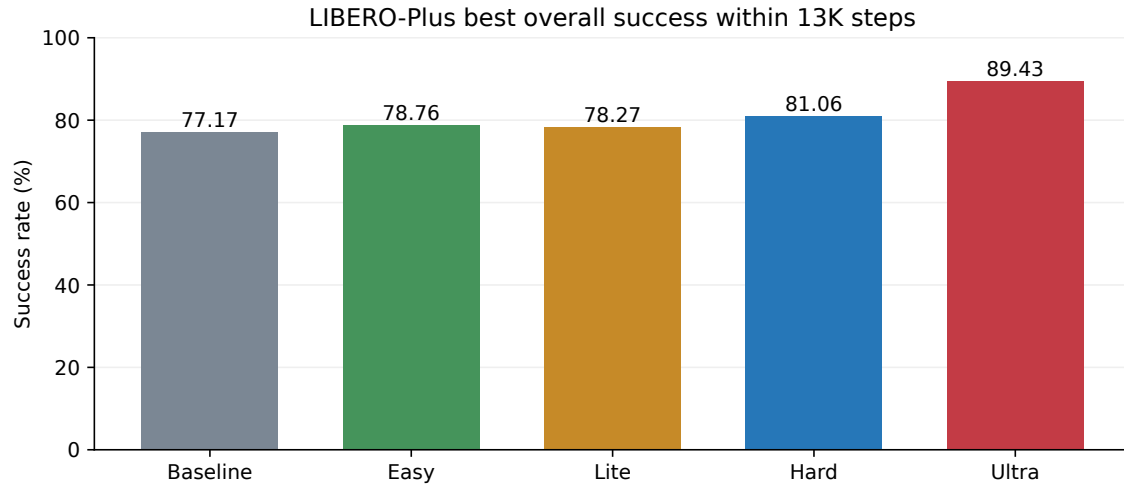


Figure 6: Best-checkpoint co-training comparison. Best observed overall WAM success on LIBERO-Plus up to and including 13K steps. SimData-Ultra gives the highest score.

Ultra adds **20.08 points** on the goal suite. The pattern is consistent with a richer training distribution across objects, layouts, goals, initial states, and instruction wording. Establishing the independent contribution of each factor requires controlled ablations.

Long-horizon LIBERO-10 improves from 85.71% to 87.70% (+1.99 points), a much smaller change. The synthetic data provides limited long-horizon sequential supervision, so the result supports stronger object, spatial, and goal generalization rather than a broad claim that sequential manipulation has been solved.

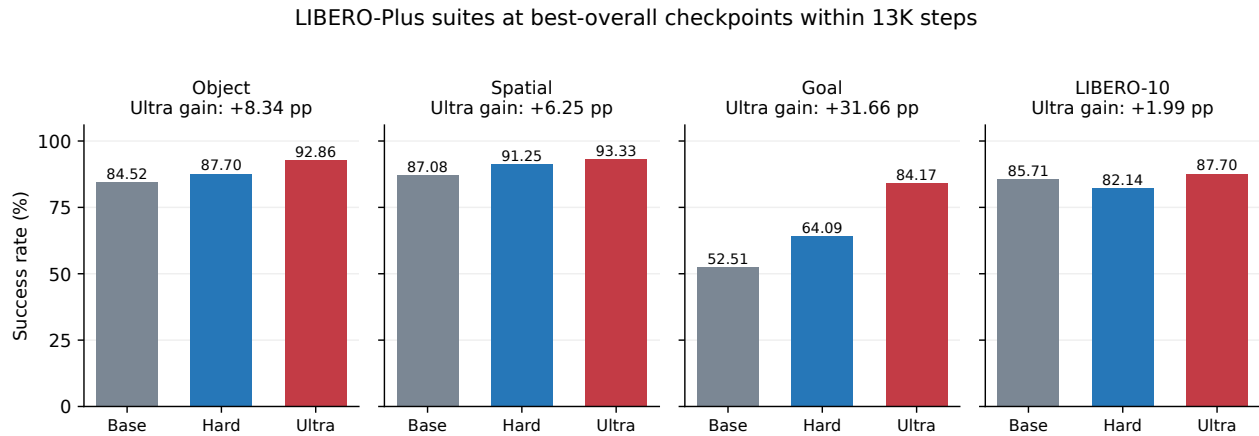


Figure 7: Generalization at the selected best-overall checkpoints. Each configuration is selected by its highest overall success up to and including 13K steps; all four suites use that selected checkpoint. Gains are largest for goals, followed by objects and spatial configurations; the long-horizon improvement is limited.

8.4. Training Progress and Best-Checkpoint Performance

The learning curves in Figure 8 show that Ultra reaches 84.75% at 5K steps, exceeding Hard’s highest listed score (81.06% at 13K). Within the 13K-step selection budget, Ultra’s best evaluated checkpoint reaches **89.43%** at 13K, whereas the baseline’s best score is **77.17%** at 9K, an **12.26-point** difference. Later checkpoints are retained in the appendix but excluded from the main comparison. The curves indicate earlier attainment of high success; they do not measure wall-clock savings or data-generation cost.

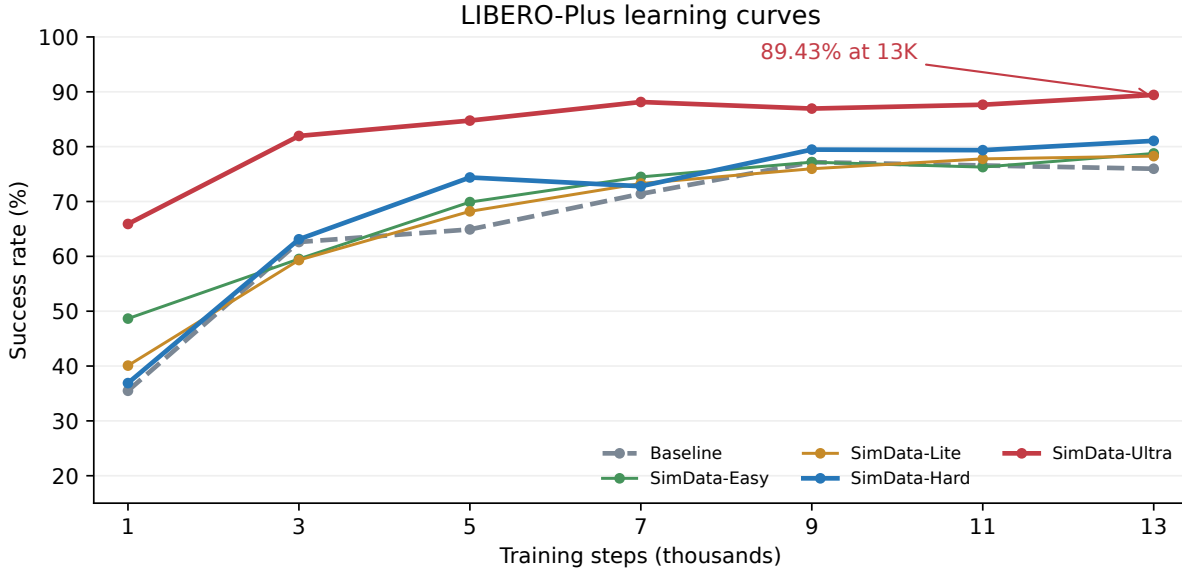


Figure 8: WAM learning curves on LIBERO-Plus. Points correspond to the reported evaluated checkpoints up to and including 13K steps. Ultra reaches 84.75% at 5K and its best score within this budget is 89.43% at 13K. Main comparisons select each configuration’s highest-overall-success checkpoint within the same budget.

8.5. Summary and Scope

The main LIBERO-Plus comparison uses the best-overall evaluated checkpoints up to and including 13K steps: **+12.26 points overall**, **+31.66 on goals**, and **+6.25 on spatial generalization**. These measurements support the utility of the integrated PolicyForge co-training data across simulator boundaries. Additional LIBERO-Pro results are provided in the appendix. The reported tables do not include repeated-training-seed confidence intervals, and the combined Ultra configuration is not an ablation of ISArt, Unravel, the harness, or individual augmentations. Physical-world transfer and world-generation acceptance rates require separate evaluation.

9. Limitations and Conclusion

9.1. Limitations

Single-view ambiguity and structural priors. ISArt and Unravel infer geometry and interaction structure from incomplete visual evidence. ISArt’s part geometry depends on predicted semantics, boxes, cavities, and kinematic relations; incorrect priors can require structural re-planning or geometry regeneration. Unravel’s image edits hypothesize occluded surfaces, and fused point maps inherit uncertainty from both image completion and monocular reconstruction. Motion validation, registration,

and settling check candidate consistency but cannot establish that hidden geometry or metric scale matches the physical scene. The current asset and physics representations primarily support rigid objects and articulated mechanisms; deformable interaction requires additional modeling.

Validation and bounded recovery. The Graph-Native Harness makes evidence, dependencies, repair scopes, and checkpoint decisions explicit. Its validators and critics can still miss errors or attribute them incorrectly, and a finite repair budget can end with an unresolved request. World acceptance is conditional on the applicable checks and simulator contract. It does not guarantee successful execution of every downstream task. The reported experiments do not quantify harness repair efficiency, generation cost, or end-to-end accepted-world yield.

Demonstration coverage and sequential tasks. PolicyForge depends on scripted or motion-planned teachers, feasible resets, and task-success predicates. Demonstrations therefore cover interactions that these teachers can execute and validate. Contact-rich manipulation, dexterous control, and extended task sequences remain challenging. On LIBERO-Plus, Ultra’s gain on LIBERO-10 is 1.99 percentage points at each configuration’s best-overall evaluated checkpoint up to and including 13K steps, compared with 31.66 on goals and 6.25 on spatial generalization. This limited sequential-task gain motivates richer task composition and long-horizon demonstration coverage; the current comparison does not isolate which data factors explain the difference.

Experimental scope and attribution. The module and policy evaluations answer different questions. ISArt’s quantitative binding experiment supplies ground-truth part geometry, so its reported binding success does not measure end-to-end asset generation. Unravel’s five-scene coverage comparison uses its own masks and point clusters as references; increased coverage and point count do not establish accuracy against independent geometric ground truth. Its asset-fidelity scores average 160 objects, without reported confidence intervals or rater-agreement statistics. The WAM experiments evaluate the combined Ultra data configuration, without repeated-training-seed confidence intervals or controlled ablations of ISArt, Unravel, harness repair, and individual augmentations. The WAM evaluations show that gains vary between LIBERO-Plus and LIBERO-Pro. Evaluation across Isaac Sim and MuJoCo supports cross-simulator training utility; real-robot transfer remains unmeasured.

9.2. Conclusion

Awomo-SimDataEngine connects interactive assets, text- and image-driven scene generation, and robot demonstration synthesis through a shared Graph-Native Harness. ISArt couples geometric-kinematic planning with structure-conditioned part generation, articulation binding, and simulation feedback. Unravel links verified deconstruction to point-map fusion, object registration, reverse-order recomposition, and physical settling. SimForge constructs single-room and multi-room scenes from text. The harness coordinates these modules through typed state, validation evidence, bounded repair, and checkpoints. PolicyForge binds accepted worlds to tasks, embodiments, and replayable resets, then admits successful teacher rollouts as demonstrations. Together these components provide a traceable asset-world-demonstration workflow for expanding robot-training data.

The experiments support this workflow at complementary levels. On the 60-object geometry benchmark, ISArt achieves CD 0.163, compared with 0.268 for TRELIS.2 + P3-SAM + X-Part. Unravel improves blind human asset-fidelity scores from 4.9 to 8.6 over 160 objects. For downstream learning, Isaac Sim demonstrations improve WAM success on MuJoCo-based LIBERO-Plus from 77.17% to 89.43% at each configuration’s best evaluated checkpoint up to and including 13K steps, selected by overall

success. This is a 12.26-point overall gain, with improvements of 31.66 points on goals and 6.25 on spatial generalization. These results establish module-level geometry and fidelity improvements and the utility of the combined co-training data under their respective protocols. Priorities for further work include independent scene-geometry evaluation, measured end-to-end readiness and repair costs, controlled module and augmentation ablations, and broader sequential-task and real-robot evaluation.

Contributors

All core contributors listed below contributed **equally** to this work and are listed **alphabetically by first name**:

Data Infra & Model Training: Danjiao Ma, Haohan Liu, Heng Jia, Hui Shan, Jiahuan Zhang, Jiangdi Xu, Kaiwen Guo, Linwei Zhang, Liyang Jin, Maochun Luo, Pengyao Niu, Shiwen Li, Shuangyu Feng, Tianheng Wang, Xin Wang, Yongqiang Huang, Zhaozhi Wang, Zijian Ma

Project Leads: Jianhua Xu, Tong Zhang, Xiangru Huang

Project Sponsors: Jianhua Xu, Kaicheng Yu

Acknowledgments

We thank the contributors who supplied the image-conditioned scene and robot-policy modules. Their detailed algorithmic descriptions and experimental results are maintained by the respective module owners.

References

- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models, 2023. URL <https://arxiv.org/abs/2308.09687>.
- Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts, 2025. URL <https://arxiv.org/abs/2511.16719>.
- Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-Design: Personalized LLM interior designer, 2024. URL <https://arxiv.org/abs/2404.02838>.
- Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023. URL <https://arxiv.org/abs/2303.04137>.
- Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models, 2025. URL <https://arxiv.org/abs/2510.13626>.
- Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric

- Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models, 2023. URL <https://arxiv.org/abs/2305.15393>.
- Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. Anyhome: Open-vocabulary generation of structured and textured 3d homes, 2023. URL <https://arxiv.org/abs/2312.06644>.
- Gege Gao, Weiyang Liu, Anpei Chen, Andreas Geiger, and Bernhard Schölkopf. Graphdreamer: Compositional 3d scene synthesis from scene graphs, 2023. URL <https://arxiv.org/abs/2312.00093>.
- Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7081–7091, June 2023.
- Yumeng He, Ying Jiang, Jiayin Lu, Yin Yang, and Chenfanfu Jiang. Spark: Sim-ready part-level articulated reconstruction with vlm knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7233–7243, June 2026.
- Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Niessner. Text2room: Extracting textured 3d meshes from 2d text-to-image models, 2023. URL <https://arxiv.org/abs/2303.11989>.
- Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scene as blender code, 2024. URL <https://arxiv.org/abs/2403.01248>.
- Zehuan Huang, Yuan-Chen Guo, Xingqiao An, Yunhan Yang, Yangguang Li, Zi-Xin Zou, Ding Liang, Xihui Liu, Yan-Pei Cao, and Lu Sheng. MIDI: Multi-instance diffusion for single image to 3D scene generation, 2024. URL <https://arxiv.org/abs/2412.03558>.
- Mukul Khanna, Yongsan Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation, 2023. URL <https://arxiv.org/abs/2306.11290>.
- Lingyu Kong, Ruicheng Li, Ruicheng Wang, Sicheng Xu, Chengtang Yao, Jianfeng Xiang, and Jiaolong Yang. MoGe-3: Fine-detail monocular geometry estimation with self-guided sparse volumetric refinement, 2026. URL <https://arxiv.org/abs/2607.17967>.
- Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and ERIC EATON. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations (ICLR)*, volume 2025, pages 17578–17602, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/2c8047bf3ed8ef6905351608d641f02f-Paper-Conference.pdf.
- Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon,

- C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation, 2024a. URL <https://arxiv.org/abs/2403.09227>.
- Haoran Li, Haolin Shi, Wenli Zhang, Wenjun Wu, Yong Liao, Lin Wang, Lik-hang Lee, and Pengyuan Zhou. Dreamscene: 3d gaussian-based text-to-3d scene generation via formation pattern sampling, 2024b. URL <https://arxiv.org/abs/2404.03575>.
- Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, and Yan-Pei Cao. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models, 2025. URL <https://arxiv.org/abs/2502.06608>.
- Chenguo Lin and Yadong Mu. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior, 2024. URL <https://arxiv.org/abs/2402.04717>.
- Yuchen Lin, Chenguo Lin, Panwang Pan, Honglei Yan, Feng Yiqiang, Yadong Mu, and Kate-rina Fragkiadaki. Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, Main Conference, pages 35387–35415. Curran Associates, Inc., 2025. doi: 10.52202/085713-1189. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/32cc61322f1e2f56f989d29ccc7cfbb7-Paper-Conference.pdf.
- Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL <https://arxiv.org/abs/2306.03310>.
- Jasper Lu, Zhenhao Shen, Yuanfei Wang, Shugao Liu, Shengqiang Xu, Shawn Xie, Jingkai Xu, Feng Jiang, Jade Yang, Chen Xie, and Ruihai Wu. From seeing to simulating: Generative high-fidelity simulation with digital cousins for generalizable robot learning and evaluation, 2026. URL <https://arxiv.org/abs/2604.15805>.
- Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-sam: Native 3d part segmentation, 2025. URL <https://arxiv.org/abs/2509.06784>.
- Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation, 2021. URL <https://arxiv.org/abs/2108.03298>.
- Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023. URL <https://mimicgen.github.io/docs/introduction/overview.html>.
- Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots, 2024. URL <https://arxiv.org/abs/2406.02523>.
- Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu

- Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=a68SUt6zFt>.
- Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis, 2021. URL <https://arxiv.org/abs/2110.03675>.
- Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. Scenesmith: Agentic generation of simulation-ready indoor scenes, 2026. URL <https://arxiv.org/abs/2602.09153>.
- Hou In Derek Pun, Hou In Ivan Tam, Austin T. Wang, Xiaoliang Huo, Angel X. Chang, and Manolis Savva. HSM: Hierarchical scene motifs for multi-scale indoor scene generation, 2025. URL <https://arxiv.org/abs/2503.16848>.
- Xiaowen Qiu, Jincheng Yang, Yian Wang, Zhehuan Chen, Yufei Wang, Tsun-Hsuan Wang, Zhou Xian, and Chuang Gan. Articulate anymesh: Open-vocabulary 3d articulated objects modeling. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 4585–4606. PMLR, 27–30 Sep 2025. URL <https://proceedings.mlr.press/v305/qiu25b.html>.
- Nadun Ranawaka, Josiah Wong, Wei-Lin Pai, Wei-Teng Chu, Tianyuan Dai, Masoud Moghani, Hang Yin, Yunfan Jiang, Wesley Durbano, Brandon Huynh, Yu Fang, Danfei Xu, Ruohan Zhang, Li Fei-Fei, Linxi Fan, Bowen Wen, Ajay Mandlekar, and Yuke Zhu. Simfoundry: Modular and automated scene generation for policy learning and evaluation, 2026. URL <https://arxiv.org/abs/2606.28276>.
- SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. Sam 3d: 3dfy anything in images, 2025. URL <https://arxiv.org/abs/2511.16624>.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL <https://arxiv.org/abs/2303.11366>.
- Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. LayoutVLM: Differentiable optimization of 3D layout via vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL <https://arxiv.org/abs/2412.02193>.
- Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. Sceneeval: Evaluating semantic coherence in text-conditioned 3d indoor scene synthesis, 2025. URL <https://arxiv.org/abs/2503.14756>.
- Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Niessner. Diffusescene: Denoising diffusion models for generative indoor scene synthesis, 2023. URL <https://arxiv.org/abs/2303.14207>.
- Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Fei Wu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang,

- Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wenten Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL <https://arxiv.org/abs/2503.20314>.
- Tencent Hunyuan3D Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation, 2025a.
- Tencent Hunyuan3D Team. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details, 2025b. URL <https://arxiv.org/abs/2506.16504>.
- Puyi Wang, Yuhao Wang, Linjie Li, Zhengyuan Yang, Kevin Qinghong Lin, Yangguang Li, and Yu Cheng. Scenecode: Executable world programs for editable indoor scenes with articulated objects, 2026a. URL <https://arxiv.org/abs/2605.19587>.
- Xinjie Wang, Liu Liu, Taojun Ding, Andrew Choi, Chaodong Huang, Mengao Zhao, Ziang Li, Jackson Jiang, Chunlei Yu, Shengxiang Liu, Wei Xu, and Zhizhong Su. Embodiedgen v2: An agentic, simulation-ready 3d world engine for embodied ai, 2026b. URL <https://arxiv.org/abs/2607.07459>.
- Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. Sage: Scalable agentic 3d scene generation for embodied ai, 2026. URL <https://arxiv.org/abs/2602.10116>.
- Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.
- Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14419–14429, June 2026.
- Xinhao Yan, Jiachen Xu, Yang Li, Changfeng Ma, Yunhan Yang, Chunshi Wang, Zibo Zhao, Zeqiang Lai, Yunfei Zhao, Zhuo Chen, and Chunchao Guo. X-part: high fidelity and structure coherent shape decomposition, 2025. URL <https://arxiv.org/abs/2509.08643>.
- Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai, 2024. URL <https://arxiv.org/abs/2404.09465>.
- Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent, 2025a. URL <https://arxiv.org/abs/2509.20414>.
- Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3d embodied ai environments, 2023. URL <https://arxiv.org/abs/2312.09067>.

- Yunhan Yang, Yufan Zhou, Yuan-Chen Guo, Zi-Xin Zou, Yukun Huang, Ying-Tian Liu, Hao Xu, Ding Liang, Yan-Pei Cao, and Xihui Liu. Omnipart: Part-aware 3d generation with semantic decoupling and structural cohesion. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, SA Conference Papers '25, New York, NY, USA, 2025b. Association for Computing Machinery. ISBN 9798400721373. doi: 10.1145/3757377.3763872. URL <https://doi.org/10.1145/3757377.3763872>.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022. URL <https://arxiv.org/abs/2210.03629>.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL <https://arxiv.org/abs/2305.10601>.
- Zhaoda Ye, Xinhan Zheng, Yang Liu, and Yuxin Peng. Relscene: A benchmark and baseline for spatial relations in text-driven 3d scene generation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 10563–10571, 2024. doi: 10.1145/3664647.3681653. URL <https://doi.org/10.1145/3664647.3681653>.
- Yootta. WorldSimReady-Home. <https://huggingface.co/datasets/Yootta/World-SimReady-Home>, 2026.
- Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T. Freeman, and Jiajun Wu. Wonderworld: Interactive 3d scene generation from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yu_WonderWorld_Interactive_3D_Scene_Generation_from_a_Single_Image_CVPR_2025_paper.html.
- Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2025. URL <https://arxiv.org/abs/2510.03827>.

A. Graph-Native Harness

World state. The constructed world is represented by

$$S = (R, O, A, X, H, J, P, M). \quad (6)$$

R denotes rooms, architectural structure, and opening geometry. O denotes object instances with stable scene identities. A denotes visual, collision, and articulated assets. X stores world poses for objects and reference poses for articulated links. H stores support, containment, room ownership, and parent lineage. J stores joints, types, axes, limits, dynamics, and serialized articulation state. P stores mass, friction, inertia, stability, and collision semantics. M stores materials, textures, and appearance attributes.

The world state is intentionally separate from evidence. The same S may be retained, repaired, or rejected under different validation histories, so the evidence plane must be explicit rather than implicit.

Evidence state. The evidence state is

$$E = (E_{\text{sem}}, E_{\text{geo}}, E_{\text{phy}}, E_{\text{vis}}, E_{\text{exp}}). \quad (7)$$

The components denote semantic, geometric, physical, visual, and export evidence. A hard failure in any of these components can block promotion even if the rendered scene looks plausible.

Harness state. The execution state at step k is

$$H_k = (S_k, T, E_k, R_k, B_k, \kappa_k), \quad (8)$$

where R_k is the selected graph edge and causal owner, B_k is the remaining retry budget, and κ_k is the checkpoint selected for preservation or rollback. This is the normative Harness state used to define the transition semantics; module implementations retain stage snapshots and artifact records; the schema specifies their common audit meaning rather than a universal serialization.

Accepted-world gate. A world is simulation-ready only when the compiled task and the constructed world satisfy the evidence gate

$$\Phi(S, T, E) = 1[\text{semantic} \wedge \text{geometric} \wedge \text{physical} \wedge \text{task} \wedge \text{export}]. \quad (9)$$

We call the result an *Accepted World* exactly when $\Phi(S, T, E) = 1$. The typed request enters the Harness at the input boundary. Scene-only requests return reconstructed scenes without Accepted World status; task-conditioned requests return an Accepted World only after the task contract and all required checks pass. Downstream policy evaluation and demonstration synthesis consume only Accepted Worlds. A saved render, partial layout, or best-effort export is therefore not an accepted-world output. Module-specific image reconstruction and demonstration synthesis are described in Sections 5 and 7 respectively.

A.1. Transition and Repair Semantics

Awomo-SimDataEngine treats iterative generation as an explicit harness, not as a hidden retry loop. The graph does not merely host stages; it atomizes adaptation into typed transitions, evidence-bearing ownership, and bounded re-entry. The transition is the method’s smallest unit: models, critics, solvers, and simulators are executors attached to it, not competing control abstractions.

Observation	Owner	Repair scope	Evidence
Incomplete or inconsistent object-removal plan or image edit	Unravel Planning / Editing	Affected removal round and image region	Plan, mask, and edit-verification evidence
Missing, rejected, or semantically wrong asset	Asset	Named request and dependent scene instances	Acquisition and semantic evidence
Mismatch between an asset and its observed point cluster	Registration	Named object and attributed point cluster	Point-cloud, mask, and alignment evidence
Residual penetration, physical instability, or simulator infeasibility	Physics / Settling	Affected object instances and local recomposition workset	Physical feasibility evidence
Validation regression without a hard geometry failure	Critique	Structured modification plan with a bounded attempt budget	Perceptual and validation evidence
Missing or invalid simulator package, frame, or strict-export predicate	Export / Packaging	Package manifest and coordinate/export artifacts; return upstream only when evidence identifies a source-geometry or physics cause	Packaging evidence

Table 5: Owner-aware repair contract. The repair scope follows the causal owner when it can be identified; unrelated validated state remains protected by the checkpoint policy.

A.2. Control Plane

The control plane is the role of the orchestrator. It inspects typed state and structured evidence, applies a bounded budget, and chooses the owner and next edge. An executor may be learned, deterministic, or tool-backed; the control contract remains the same.

Structured evidence names an owner, a bounded scope, and the checkpoint decision needed before the next transition.

A.3. Loop-to-Graph Atomicization

The point of the Harness is not that the system has a loop. The point is that the loop has been decomposed into first-class transitions:

$$\begin{aligned} o_k &= \text{Observe}(H_k), & d_k &= \text{Diagnose}(o_k), & u_k &= \text{Attribute}(d_k, H_k), & a_k &= \text{Act}_{u_k}(H_k), \\ v_k &= \text{Validate}(a_k), & H_{k+1} &= \text{Select}(H_k, a_k, v_k), & r_{k+1} &= \text{Route}(v_k, H_{k+1}). \end{aligned} \quad (10)$$

Here u_k is the causal owner, H_{k+1} is the selected candidate or checkpointed state, and r_{k+1} is the next explicit graph edge. The atomic transition therefore carries typed state, structured evidence, repair scope, a retry budget, and a checkpoint decision. This is the single control semantics used by every domain subgraph.

A.4. Evidence Routing

The transition semantics above are evidence-driven, not score-driven. Visual, geometric, physical, and packaging observations are normalized into a common evidence record before an owner is selected. When the cause is ambiguous, the Harness preserves the ambiguity and checkpoint rather than inventing a precise owner; unresolved cases remain explicit non-successes.

A.5. Scoped Re-entry

Re-entry returns to the smallest owner that can change the failing state. The owner table supplies the canonical mapping, and unaffected validated state remains eligible for reuse while the repair consumes only the remaining budget for that scope.

A.6. Checkpoint, Rollback, and Degradation

Checkpoints are not decorative saves. A candidate is selected only after validation; if it improves one metric but violates a hard predicate, the Harness restores the last valid checkpoint or records the actual degradation instead of pretending the run succeeded.

Degradation records capture observed execution failures, never success. They remain part of the audit trail and cannot substitute for validation.

A.7. Nested Subgraphs

The global Harness composes owner-local subgraphs around the world compiler described in Section 3. Each subgraph may contain local transitions, but all of them use the single contract above. Strict export closes a local validation attempt by promoting only an accepted candidate; multimodal adapters and policy interfaces connect at that boundary without introducing additional control mechanisms.

A.8. Multimodal Condition and Typed Specification

The condition contract makes modality an input adapter, not a separate control logic. SimForge uses text requests directly; Unravel uses a single scene image with an optional description. Object-centric references provide the separate ISArt input. The table lists conceptual groups rather than a prescribed serialization format.

Table 6: Conceptual condition and typed-specification contract.

Group	Required meaning
Condition identity	Stable cohort identifier, modality label, source provenance, calibration status, and adapter/version owner.
Scene request	Rooms, architectural elements, object requests, attributes, dimensions, relations, and material/style constraints.
Task request	Goal, ordered subgoals, operable entities, interaction type, robot constraints, and success predicates.
Compilation evidence	Conflicts between modalities, unresolved requirements, confidence, and the rule used to resolve each conflict.
Construction requirements	Asset, articulation, support, clearance, containment, and export prerequisites derived from the task.
Output contract	Typed scene-task specification consumed by the ordered planning, acquisition, construction, validation, and export stages; an optional policy handoff attaches only after the Accepted World.

A.9. Harness State Fields (implementation: PipelineState)

PipelineState is the implementation name for the Harness state exchanged by graph nodes. The state is defined by semantic responsibilities, not by a public serialization shape. Snapshots may be partial during diagnosis, but a partial snapshot is never a strict success claim.

Table 7: Conceptual Harness-state field groups (implementation name: PipelineState).

Field group	Required audit meaning
Condition and configuration	Source identity, modality, adapter/version, model choices, seeds, strictness, and declared budgets.
Scene and layout	Current typed scene state, room topology, relation diagnostics, and the stage at which the snapshot was taken.
Acquisition	Asset requests, candidate outcomes, accepted sources, rejection reasons, and terminal failures keyed by request identity.
Placement and physics	Poses, support relations, collision and stability evidence, hard violations, and object removals or restorations.
Critique and evidence	Structured visual, geometric, physical, and export observations with severity, entities, and candidate owner.
Control and recovery	Selected graph edge, retry budget, checkpoint choice, rollback history, regression status, and next edge.
Degradation and provenance	Observed execution outcome, unresolved limitations, responsible owner, and versioned provenance. Degradation is never success evidence.

A.10. SceneState and Object Fields

SceneState is the candidate-world checkpoint. It distinguishes an asset template from a scene instance and preserves the lineage needed for support-aware repair and strict export.

Table 8: Conceptual SceneState and object-instance contract.

Group	Required audit meaning
World identity	Stable scene identity, condition linkage, stage, iteration, and checkpoint provenance.
Topology	Rooms, boundaries, openings, architecture, object membership, and typed spatial relations.
Object instance	Unique instance identity, semantic role, placement role, room ownership, dimensions, pose, and measured geometry.
Asset evidence	Non-placeholder visual and collision representations, material/physical properties, source provenance, and validation status.
Support lineage	Parent instance, concrete support surface, local pose, footprint, clearance, and sibling compatibility for every supported child.
Articulation	Links, joint type, axis, origin, limits, and actuation metadata for movable objects.
Constraints and history	Requested relations, repair markers, validator outcomes, and prior accepted checkpoint references.

Table 9: Conceptual organization of the evidence package.

Layer	Contents	Strictness role
Condition	Modality, source identity, adapter/version, task, seeds, and split	Immutable cohort key
Scene state	Rooms, objects, poses, relations, assets, joints, support lineage, and materials	Candidate world state
Execution trace	Transitions, observations, owner decisions, budgets, checkpoints, and rollbacks	Harness audit trail
Asset evidence	Candidate sources, geometry, collision, material, and validation outcomes	Acquisition gate
Validation evidence	Semantic, visual, geometric, physical, export, and degradation records	Hard/soft gate inputs
Simulator package	Coordinate contract, dependencies, load result, and reset metadata	Strict export gate
Policy handoff	Accepted world, task contract, robot/reset adapter, roll-out status, and first blocker	Downstream interface

A.11. Strict Export Predicate

The report uses a conjunctive export gate:

$$\text{StrictSuccess} \iff \text{terminal export stage} \wedge \text{successful execution} \wedge \text{hard validation passes} \wedge \text{validated simulator package.} \quad (11)$$

File existence, a render, or an intermediate package cannot establish any conjunct. Accepted-scene full success additionally requires experiment-level loadability, visual/collision alignment, task-entity coverage, and reset replay. Unresolved degradation, incomplete execution, and best-effort output remain diagnostic failures.

A.12. Evidence Package Organization

The evidence plane is organized by meaning rather than by storage location. Later layers may be absent when an earlier stage fails; absence is recorded as a failure state, not treated as an empty success.

A.13. Per-Run Evidence Checklist

Before a run contributes to accepted-scene or policy-facing metrics, the following conceptual evidence must be present:

1. A condition record binds modality, source identity, adapter/version, task, seeds, split, and protocol version.
2. The final state records stage, warnings, errors, transitions, checkpoints, rollback decisions, and every degradation with its actual owner and reason.
3. Rooms, objects, poses, dimensions, relations, and placement evidence are present in the scene checkpoint.
4. Every accepted object has non-placeholder visual/collision evidence, provenance, canonical geometry, and validation status.
5. Rejected and failed candidates remain in the negative-evidence ledger.
6. Every supported child has concrete parent/surface lineage, local pose, measured hull, clearance, and sibling-collision evidence.

7. Geometry, physics, articulation, and export validators report hard violations and first blockers; visual critique remains a soft observation.
8. The strict conjunctive export gate passes; a render, intermediate package, or best-effort artifact alone never qualifies.
9. Policy episodes, when evaluated, join world, task, robot, simulator, seed, checkpoint, terminal status, first blocker, and recovery evidence.
10. Environment and resource records distinguish generation, export, physics, and policy costs and preserve the declared version/budget contract.

B. ISArt

B.1. Structural Encoding and Geometry Conditioning

ISArt introduces a structure-conditioned DiT-based geometry generation framework that explicitly injects semantic, spatial, and kinematic priors into the multi-part denoising process. The overall architecture is illustrated in Fig. 9. To acquire the geometry and kinematic structure of the target object, we first use a VLM to predict the global structural information

$$\mathcal{S} = (N, T_{\text{category}}), \quad (12)$$

where N denotes the number of parts and T_{category} denotes the object category. We then jointly predict structural information $\{S_i\}_{i=1}^N$ for all parts. For part i , we organize part-wise structural information into several text descriptions and vectors:

$$S_i = (T_{\text{semantic},i}, T_{\text{conn},i}, l_i, J_i, \mathbf{c}_{\text{box},i}, \mathbf{s}_{\text{box},i}, \mathbf{c}_{\text{cav},i}, \mathbf{s}_{\text{cav},i}) \quad (13)$$

where $T_{\text{semantic},i}$ denotes the semantic label of part i ; $T_{\text{conn},i}$ denotes its parent and child connection in the kinematic tree; $l_i \in \{0, 1\}^4$ denotes the one-hot joint-type representation of part i , corresponding to four predefined categories: fixed, revolute, prismatic, and continuous; $J_i \in \mathbb{R}^3$ denotes the coarse joint-axis direction of part i ; $\mathbf{c}_{\text{box},i}, \mathbf{s}_{\text{box},i} \in \mathbb{R}^3$ denote the center and size of the part bounding box, and $\mathbf{c}_{\text{cav},i}, \mathbf{s}_{\text{cav},i} \in \mathbb{R}^3$ denote the center and size of the associated cavity bounding box. The proposed method only handles the case of one predicted cavity (one hole). If no cavity is predicted, we set $\mathbf{c}_{\text{cav},i} = \mathbf{s}_{\text{cav},i} = \mathbf{0}$. We encode the textual attributes $\{T_{\text{semantic},i}, T_{\text{conn},i}\}$ using a frozen BERT-base encoder followed by a learnable projection to the structural feature dimension d_s , while the discrete joint-type attribute l_i is mapped through a learned embedding. The continuous canonical-frame attributes $\{J_i, \mathbf{c}_{\text{box},i}, \mathbf{s}_{\text{box},i}, \mathbf{c}_{\text{cav},i}, \mathbf{s}_{\text{cav},i}\}$ are independently projected by lightweight MLPs into the same d_s -dimensional space; attribute-specific role embeddings are added to distinguish their semantics, and learned missing-value embeddings are used whenever an attribute is unavailable.

Given the structural attributes above, we encode them into $K = 7$ part-aligned structural tokens: T_{category} , $T_{\text{semantic},i}$, joint token (including l_i, J_i), $T_{\text{conn},i}$, $\mathbf{c}_{\text{box},i}, \mathbf{s}_{\text{box},i}$, cavity token (including $\mathbf{c}_{\text{cav},i}, \mathbf{s}_{\text{cav},i}$).

Let $E_i^{\text{str}} \in \mathbb{R}^{7 \times d_s}$ denote these structural tokens. Meanwhile, we employ a shared frozen DINOv2 encoder [Oquab et al., 2024] to extract visual features from the input image I_0 and the optional open-state reference I_{open} generated by GPT-image-2, which provides complementary cues for occluded internal structures, yielding $\mathbf{E}^{\text{vis}} = [\mathcal{E}_{\text{img}}(I_0); \mathcal{E}_{\text{img}}(I_{\text{open}})]$.

Generation proceeds through four steps with two complementary conditioning mechanisms:

(1) *Multi-Part Latent Initialization.* Given the predicted number of parts N , we maintain one group of geometry latent tokens for each part. At sampling timestep t , the latent representations of all parts are organized as

$$\mathbf{Z}_t = [\mathbf{Z}_t^1; \mathbf{Z}_t^2; \dots; \mathbf{Z}_t^N], \quad \mathbf{Z}_t^i \in \mathbb{R}^{M \times D}, \quad (14)$$

where each $\mathbf{Z}_t^i$ remains aligned with part i throughout the denoising process. The latent tokens of each part are first projected into the DiT hidden space and combined with the timestep embedding:

$$\mathbf{H}_0^i = f_{\text{in}}(\mathbf{Z}_t^i) + \mathbf{e}_t, \quad (15)$$

where $\mathbf{e}_t$ denotes the timestep embedding.

(2) *Per-Layer Structural Bias Injection (E)*. At each DiT layer l , the seven part-aligned structural tokens $\mathbf{E}_i^{\text{str}} \in \mathbb{R}^{7 \times d_s}$ are first aggregated into a part-level structural summary:

$$\bar{\mathbf{e}}_i^{\text{str}} = \frac{1}{K} \sum_{k=1}^K \mathbf{E}_{i,k}^{\text{str}}, \quad K = 7. \quad (16)$$

The summary feature is then projected into the DiT hidden space and added to the current hidden geometry tokens:

$$\tilde{\mathbf{H}}_l^i = \mathbf{H}_l^i + f_E(\bar{\mathbf{e}}_i^{\text{str}}). \quad (17)$$

This provides a persistent part-level structural bias at every DiT layer.

(3) *Local, Global, and Image Attention*. The structurally biased geometry tokens are subsequently processed by the original Multi-Part DiT backbone. Specifically, local self-attention first models interactions among geometry tokens within each part, global self-attention enables information exchange across different parts, and image cross-attention conditions the geometry tokens on the visual features $\mathbf{E}^{\text{vis}}$ extracted from I_0 and the optional I_{open} . We compactly denote these backbone operations as

$$\hat{\mathbf{H}}_l^i = \mathcal{B}_l(\tilde{\mathbf{H}}_l^i, \mathbf{E}^{\text{vis}}), \quad (18)$$

where $\mathcal{B}_l$ denotes the original l -th DiT block containing local self-attention, global self-attention, and image cross-attention.

(4) *Structural Cross-Attention (C)*. After the original backbone operations, we further introduce fine-grained structural conditioning by directly attending to the full structural token sequence. The updated geometry tokens $\hat{\mathbf{H}}_l^i$ serve as queries, while $\mathbf{E}_i^{\text{str}}$ provides the keys and values:

$$\mathbf{Q}_i^C = \hat{\mathbf{H}}_l^i \mathbf{W}_Q^C, \quad \mathbf{K}_i^C = \mathbf{E}_i^{\text{str}} \mathbf{W}_K^C, \quad \mathbf{V}_i^C = \mathbf{E}_i^{\text{str}} \mathbf{W}_V^C. \quad (19)$$

The structural attention weights and the resulting hidden representation are computed as

$$\mathbf{A}_i^C = \text{softmax}\left(\frac{\mathbf{Q}_i^C (\mathbf{K}_i^C)^\top}{\sqrt{d}}\right), \quad \mathbf{H}_{l+1}^i = \hat{\mathbf{H}}_l^i + \mathbf{A}_i^C \mathbf{V}_i^C, \quad (20)$$

where d is the attention-head dimension. In contrast to E , which provides a shared part-level structural bias, C allows individual geometry tokens to dynamically retrieve the most relevant semantic, spatial, and kinematic information from $\mathbf{E}_i^{\text{str}}$.

These operations are repeatedly applied across DiT layers and sampling timesteps, progressively transforming the initial noisy part latents into structured part-level geometry representations.

B.2. Training and Failure-Driven Data Expansion

The structural encoder, E/C modules, and attention LoRA parameters are trained with the rectified-flow velocity loss

$$\mathcal{L}_{\text{RF}} = \mathbb{E}[\|V_\theta(Z_t, \Phi, t) - \mathbf{U}^*\|_2^2], \quad (21)$$

Here, t denotes the sampled flow timestep, $\mathbf{Z}_t$ is the interpolated latent at timestep t , Φ denotes the combined visual and structural conditions, V_θ is the predicted velocity field, and $\mathbf{U}^*$ is the target velocity defined by the Rectified Flow interpolation path. The learning rate is 10^{-4} and effective batch size is 8; timesteps follow a logit-normal sample with $\mu = 0$, $\sigma = 1$. The initial approximately 1,100 programmatically modeled assets support 10,000 training steps. Targeted modeling of failure categories adds approximately 1,300 assets, followed by 15,000 more steps. The reported total is approximately 2,400 assets and 25,000 steps, taking about 80 hours on four NVIDIA A100 GPUs.

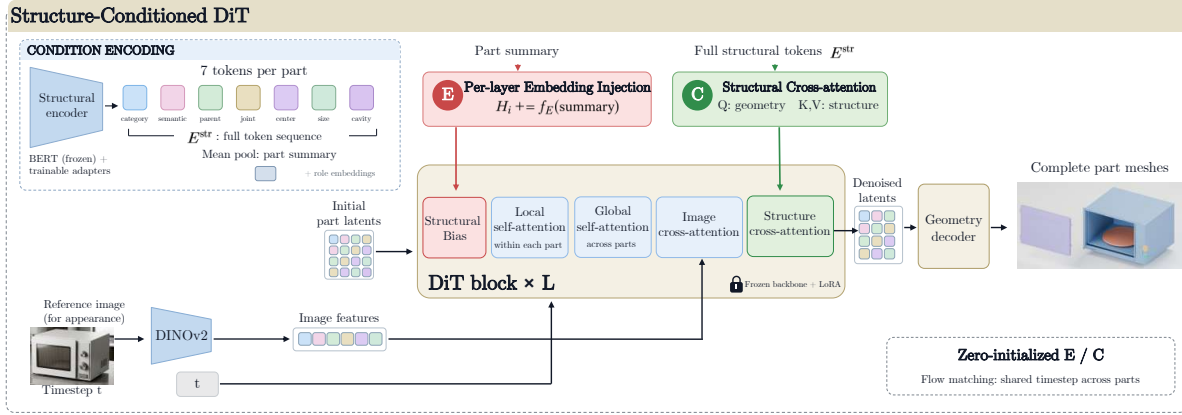


Figure 9: ISArt’s structure-grounded multi-part DiT. E and C condition the pretrained geometry backbone while retaining visual conditioning.

B.3. Binding Memory and Validation Records

The binding agent retrieves category-level motion summaries and may expand them into joint-specific axes, pivots, types, and ranges. This hierarchical articulation memory supplies priors rather than proof of correctness. Part identifiers and the predicted tree bind geometry to URDF links. Isaac Sim motion renders are inspected for axis, direction, pivot, range, and interference errors. Parameter repair and rollback precede fallback to structure planning or geometry generation. Records retain the candidate structure, geometry, motion render, critique, and adoption or rejection reason. Quantitative comparisons are collected in Appendix E.1.

B.4. Asset Registry Fields

The Asset Registry preserves both adoption and negative evidence. A rejected candidate, placeholder, or missing collision representation remains visible in the audit even when another candidate is accepted.

Table 10: Conceptual asset-registry contract.

Group	Required audit meaning
Request identity	Stable request, candidate, and instance identities that survive acquisition, placement, and export.
Semantic request	Description, object role, dimensions, attributes, style context, and articulation requirements.
Lifecycle	Planned, attempted, adopted, rejected, failed, and superseded states with terminal reason.
Provenance	Retrieval or generation source, query/prompt summary, back-end/version, and cache status.
Geometry contract	Canonical orientation, scale, visual geometry, collision geometry, decomposition audit, and alignment evidence.
Material and physics	Material references, mass/friction properties, and simulator-readiness status.

Group	Required audit meaning
Validation history	Warnings, hard failures, candidate comparisons, and evidence used for the adoption decision.

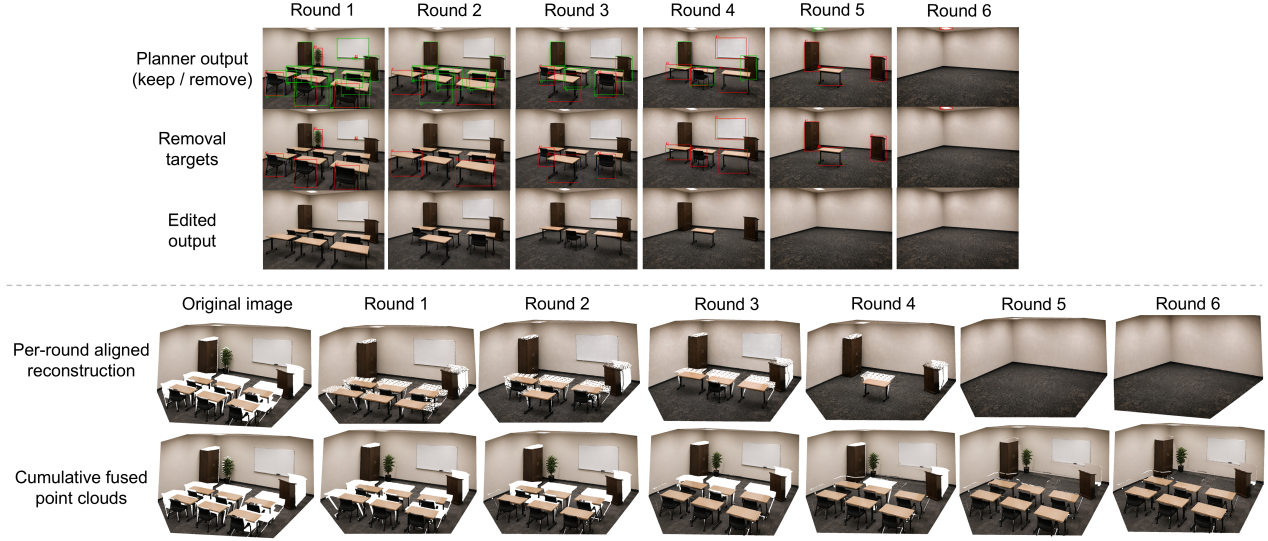


Figure 10: Layer-wise deconstruction and point-cloud fusion in Unravel. The upper block shows planner outputs, removal targets, and edited images across six rounds. Red boxes mark removal targets; green boxes mark protected objects. The lower block shows aligned per-round reconstructions and cumulative fused point clouds.

C. Unravel

C.1. Verified Localization and Deconstruction

The planner proposes boxes for all targets in a global call. Boxes of the same category with IoU above 0.85 are treated as duplicates. Each target receives an independent local probe; IoU of at least 0.5 confirms the global box. Otherwise, three independent local detections are sampled, and detections agreeing with another sample at IoU of at least 0.7 are retained. Their element-wise median replaces the drifting box. Failed probes or local detections preserve the global estimate. A further inspection adds missed movable objects to the protected set, preventing accidental deletion.

C.2. Point Fusion, Registration, and Settling

Point-map validity uses confidence threshold 0.5 and excludes unreliable boundary pixels. Every round aligns to the first round through never-edited anchors. Only newly exposed valid points enter fusion, retaining per-point pixel and round provenance. Per-ray removal order assigns the fused points to independent object clusters or the room shell.

Registration ranks transformations by bidirectional median Chamfer distance, then refines trusted correspondences with truncated similarity ICP. A global scale and bounded object-frame axis adjustments avoid interpreting occluded extent as true object size. Under-observed axes may expand but cannot shrink below observed span. Well-observed axes may shrink if fit does not worsen. When all axes have insufficient observed extent, uniform scaling preserves shape proportions.

Mounted objects remain fixed. Movable objects receive translations that minimize total squared displacement subject to non-interpenetration, support, and consistency with their attributed observation clusters. A supporting surface lies beneath the object’s local footprint. With bottom height b_i and

eligible support heights H_i , contact requires

$$0 \leq b_i - \max H_i \leq \epsilon_s, \quad \epsilon_s = 0.03 \text{ m.} \quad (22)$$

The floor is always an eligible support. Observation-consistent translations satisfy $d(a'_i, C_i) \leq \max(d(a_i, C_i), \epsilon)$, with tolerance $\epsilon = 0.3$ m horizontally and 0.15 m vertically. These are tolerances relative to observation fit, not a universal bound on every displacement. Audit edits are followed by renewed feasibility checks.

C.3. Support and Coordinate Records

Asset-local, camera, world, collision, and simulator frames retain their transform relations and declared scale. Support uses measured local geometry under each object footprint. Where a supporting parent exists, its instance identity and supporting surface are recorded; semantic occlusion order alone cannot establish contact. Mounted objects and articulation links preserve their fixed or link-local transforms. Registration and settling records retain observation fit, support gaps, penetrations, and accepted pose corrections. Module results and blind scoring details are in [Appendix E.2](#).

D. PolicyForge

PolicyForge generates robot demonstrations from validated worlds and task contracts. Source-specific robot, camera, action, and reset adapters preserve the provenance required to admit trajectories into co-training. World acceptance and episode success are evaluated separately.

D.1. World-to-Policy Contract

The policy interface is intentionally algorithm-neutral. A policy module may consume an accepted world only after the scene, task, simulator, and reset contracts have been validated. The following setup documents the evaluated WAM co-training configuration; real-robot transfer is not measured by the reported experiments.

Table 11: Conceptual world-to-policy handoff.

Component	Required contents
World	Accepted scene state, strict validation summary, simulator package, coordinate contract, and provenance.
Task	Task family, goal, ordered subgoals, target entities, affordances, joint targets, and success predicate.
Robot adapter	Embodiment, sensors, observation/action spaces, control horizon, and reset assumptions.
Randomization	Scene, asset, reset, and rollout seeds plus declared domain-randomization policy.
Rollout feedback	Terminal status, first blocker, owner, evidence, and recovery decision returned to the Harness.

D.2. Rollout Metadata

Each rollout is joinable to one world, task, policy, robot, simulator, seed, and checkpoint. The minimum conceptual record is summarized below; exact serialization is left to the policy owner.

Table 12: Conceptual rollout record.

Group	Required meaning
Identity	World, task, policy, robot, simulator, cohort split, and checkpoint identifiers.
Randomness and budget	Scene/reset/rollout seeds, horizon, environment steps, and declared resource budget.
Outcome	Success predicate, terminal status, return/length/latency summaries, and infrastructure validity.
Failure attribution	First blocker, causal owner, affected entity or surface, secondary failures, and recovery evidence.

D.3. Policy-Learning Setup

D.3.1. WAM Architecture

WAM adopts a dual-stream Video DiT + Action DiT architecture, initialized from the Wan2.2-TI2V-5B pretrained video generation model.

Table 13: WAM architecture configuration.

Component	Configuration
Video DiT	30 layers, $d_{\text{hidden}} = 3072$, $d_{\text{ffn}} = 14336$, 24 heads ($d_{\text{head}} = 128$)
Action DiT	30 layers, $d_{\text{hidden}} = 1024$, $d_{\text{ffn}} = 4096$, 24 heads ($d_{\text{head}} = 128$)
Text Encoder	UMT5-XXL (frozen, precomputed embeddings, context_len = 128)
Input Cameras	2 cameras (front + wrist), 224×224 each, concat $\rightarrow 224 \times 448$
Video Context	33 frames, action_video_freq_ratio = 4
Action Space	7-DoF: 6 delta ($xyz + rpy$) + 1 absolute gripper (0=close, 1=open)
Proprioception	8-dim state vector
Normalization	min/max per-dimension
Diffusion	Flow matching, train_shift = 5.0, infer_shift = 5.0, 1000 timesteps
Loss	$\lambda_{\text{action}} = 1.0$

D.3.2. Training Hyperparameters

LIBERO-Pro experiments are trained on a single node with 8 GPUs; LIBERO-Plus experiments scale to 4 nodes (32 GPUs), with the learning rate adjusted according to the square-root scaling rule.

Table 14: Training hyperparameters. The LIBERO-Plus configuration uses $\sqrt{6} \times 10^{-4} \approx 2.449 \times 10^{-4}$ to compensate for the $6 \times$ larger global batch size.

Hyperparameter	LIBERO-Pro (1-node)	LIBERO-Plus (4-node)
GPUs	8 (1×8)	32 (4×8)
Batch size (per GPU)	16	24
Global batch size	128	768
Learning rate	1×10^{-4}	2.449×10^{-4} ($= \sqrt{6} \times 10^{-4}$)
LR schedule	Cosine	Cosine
Epochs	10	10
Max steps	—	15K
Optimizer	AdamW (DeepSpeed ZeRO-1)	AdamW (DeepSpeed ZeRO-1)
Weight decay	1×10^{-2}	1×10^{-2}
Gradient accumulation	1	1
Precision	bf16 mixed	bf16 mixed
Checkpoint save interval	2,000 steps	1,000 steps

D.3.3. Co-training Data Configuration

Data sources. LIBERO [Liu et al., 2023] is an open-source benchmark dataset based on MuJoCo. Our synthetic demonstrations are generated from Awomo-SimDataEngine scenes using PolicyForge in Isaac Sim and organized into three data families: SIMDATA-EASY, with simple scenes and tasks; SIMDATA-LITE, with simple scenes and more challenging tasks; and SIMDATA-HARD, with richer scenes and challenging tasks. SIMDATA-ULTRA denotes a co-training configuration combining multiple data sources with robot initial-state and language augmentation.

Co-training mixes benchmark demonstrations and additional data using sampling weights, with a reported 1:1 effective-frame ratio. SimData-Ultra combines SimData-Hard, additional in-house demonstrations from complex scenes, and LIBERO-Plus data with robot initial-state and language augmentation. Exact source weights and augmentation settings must be retained in the run manifest; we do not enumerate them fully in this report.

Table 15: Co-training data mixing for LIBERO-Plus experiments. LIBERO-Plus provides the benchmark demonstrations.

Config	Data Description	Sampling Weight	Frame Ratio
Baseline	LIBERO-Plus only	—	—
+ SimData-Hard	Richer scenes with challenging tasks	0.308	1 : 1
+ SimData-Ultra	Multi-source mixture with robot initial-state and language augmentation	varied	1 : 1

D.3.4. Evaluation Protocol

Table 16: Evaluation protocol summary for all benchmarks.

Benchmark	Simulator	Scale	Details
LIBERO-Plus	MuJoCo	1,003 episodes	4 suites (Obj/Spa/Goal/L-10), 8 GPU / 16 workers
LIBERO-Pro	MuJoCo	1,000 episodes	200 configs $\times$ 5 trials, 4 suites $\times$ 5 dimensions

The tables list available evaluated checkpoints. The main LIBERO-Plus comparison selects each configuration’s highest-overall-success evaluated checkpoint up to and including 13K steps; all suite scores use that same selected checkpoint. Baseline runs finish at 15K and the recorded Ultra runs at 14,580; unlisted final results are not imputed. WAM evaluations on LIBERO-Plus use 1,003 episodes. On LIBERO-Pro, Ultra peaks at 48.7% at 13K steps and finishes at 48.0% at 14,580 steps. The final per-dimension and per-suite results in Tables 25 and 26 use Ultra’s 14,580-step checkpoint rather than its peak checkpoint.

E. Experimental Results

The module evaluations measure asset geometry, articulation binding, and single-view reconstruction. Policy evaluations measure downstream utility of synthetic demonstrations. These are separate protocols and are not pooled into a single system score. The tables preserve evaluation conditions and checkpoint identities; unreported values are not inferred.

E.1. ISArt Results

The benchmark contains 60 objects: 30 from World-SimReady-Home [Yotta, 2026] and 30 from GPartNet_PhyScene [Geng et al., 2023], spanning 12 categories. Geometry methods use the same input images and are compared with corresponding ground-truth part meshes. ISArt adopts object-level normalization, where the bounding-box center of the entire object is translated to the origin and all parts are uniformly scaled using a shared scale factor. We evaluate geometry using F-Score at thresholds 0.05 and 0.1 (F@0.05 and F@0.1) and Chamfer Distance (CD). For each semantically corresponding component pair, we sample 10,000 surface points per side and compute the geometry metrics. Scores are averaged over three sampling repeats, component pairs, and objects. The results are reported in in Table 17.

Table 17: ISArt part geometry on the 60-object benchmark.

Method	CD ↓	F@0.05 ↑	F@0.1 ↑
PartCrafter	0.421	0.229	0.395
OmniPart	0.305	0.306	0.540
SPARK	0.278	0.314	0.505
TRELLIS.2 + P3-SAM + X-Part	0.268	0.431	0.627
ISArt	0.163	0.604	0.789

Binding is evaluated with ground-truth part geometry (using the same dataset as above) to isolate kinematic estimation from reconstruction. Following the ISArt binding evaluation protocol, AxisErr measures joint-axis deviation, PivotErr measures pivot-position error, and TypeErr denotes the percentage of incorrectly predicted joint types. The reported success rate concerns binding with supplied geometry. The results are reported in in Table 18.

Table 18: ISArt URDF binding with fixed ground-truth geometry.

Method	AxisErr ↓	PivotErr ↓	TypeErr ↓	Success ↑
Articulate AnyMesh	0.72	0.02	46.67%	80%
Articulate-Anything	0.12	0.17	21.11%	90%
SPARK	0.38	0.19	23.61%	100%
ISArt	0.07	0.05	13.33%	100%

Conditioning ablations. We qualitatively examine the effects of the structural conditioning branches E and C . At inference time, we independently enable E , C , or $E+C$ from the same trained checkpoint to visualize their effects on generated part geometry. We further conduct a controlled training-time comparison by training C -only and $E+C$ variants from the same backbone for 1,000 steps on 100 trash-bin samples. As shown in Figure 11, C mainly improves global part layout and structural consistency, while E provides complementary local-detail refinement.

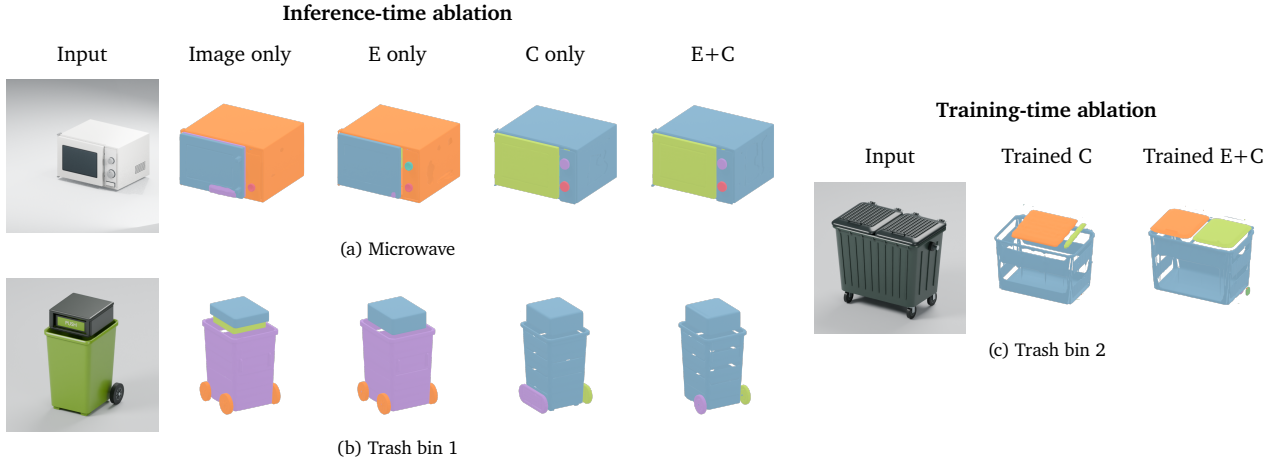


Figure 11: ISArt inference and training ablations for structural conditioning E and C. The examples support complementary effects; no aggregate ablation metric is reported.

E.2. Unravel Results

The controlled deconstruction comparison uses five scenes with 160 objects. Direct reconstruction and Unravel share SAM 3 segmentation, TRELIS.2 asset generation, and MoGe3 point estimation; the direct variant only uses the input image. Unravel fuses four to eight deconstruction rounds. The pipeline components are used off the shelf without fine-tuning.

Table 19: Unravel deconstruction comparison. Mask area and under-coverage rates (UCR, coverage below 70%) apply to objects removed after the first round, using Unravel masks and clusters as references. Fidelity scores average all 160 objects; point counts average scenes.

Method	Masks		Assets (1–10)		Geometry	
	Area ratio	UCR (%)	VLM	Human	Points (M)	UCR (%)
Direct	0.772	11.3	4.7	4.9	1.53	10.1
Unravel	1.000	0.0	8.2	8.6	2.81	0.0

Coverage interpretation. Objects removed after the first round account for approximately one third of the test set. Their direct mask area averages 77.2% of the Unravel reference, and direct point coverage averages 76.9%. The zero UCR of Unravel follows from using its own masks and clusters as the reference. Additional points include geometry predicted from inpainted surfaces. Consequently these statistics measure relative coverage, not accuracy against independent ground-truth geometry.

Blind asset-fidelity scoring. Each object supplies two meshes, rendered from identical four views separated by 90°. Paired renders are randomly ordered without revealing the pipeline. Both are judged against the crop from the object’s best-observed round. Human raters and GPT-5.5 score independently on the same rubric: 1–3 for wrong shape or missing essential parts; 4–6 for recognizable but deformed or incomplete objects; 7–8 for correct overall shape with minor artifacts; and 9–10 for faithful shape and structure. Scores are means over all 160 objects. We do not report confidence intervals or a rater-agreement statistic, and the reference may include generated completion.

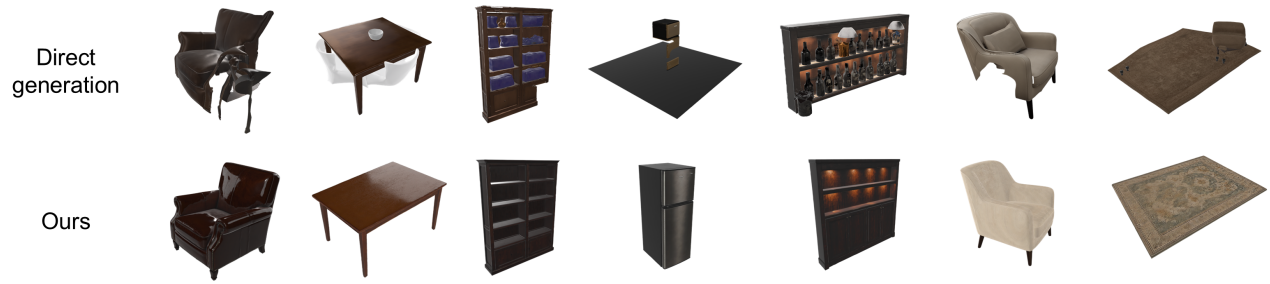


Figure 12: Asset examples generated directly from the input masks (top) and from the best-observed deconstruction round (bottom).

Placement and scene examples. The shared-asset agentic baseline receives the same generated meshes and 2D masks as Unravel, estimating only scale and pose. This comparison isolates placement from mesh-generation quality (Figure 13). We also present Unravel reconstructions from six real indoor photographs (Figure 14); these examples are qualitative and are not included in the five-scene quantitative aggregate.

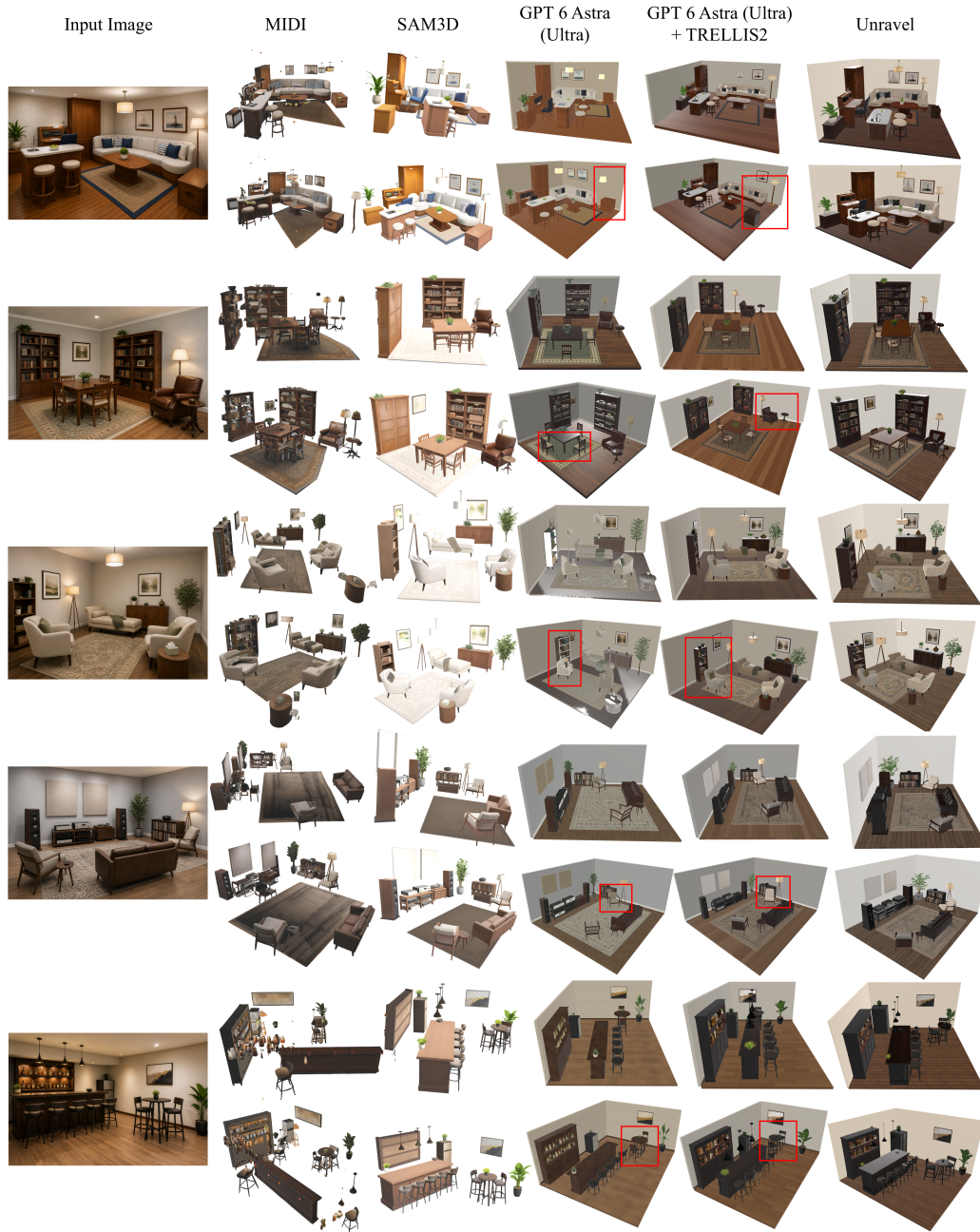


Figure 13: Single-image scene comparisons. The shared-asset agentic variant receives Unravel’s meshes and masks, isolating pose and scale estimation.



Figure 14: Unravel reconstructions from six real indoor photographs. Each row shows the input and three reconstructed views.

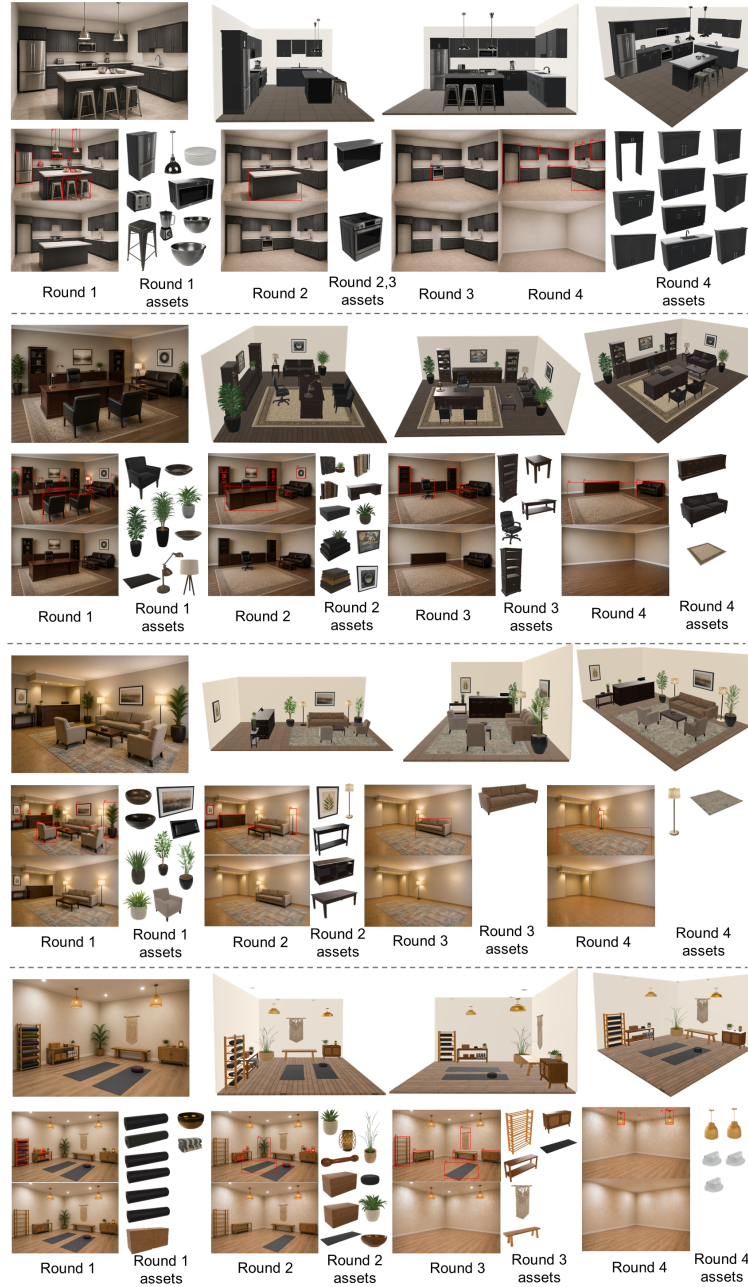


Figure 15: Additional Unravel scenes with input images, reconstructed views, per-round edits, and extracted assets. These traces are qualitative examples.

E.3. Policy Co-training Results

The tables below retain the recorded checkpoint results, including fixed-13K and final-checkpoint summaries. The main-text LIBERO-Plus comparison instead selects each configuration’s highest-overall-success evaluated checkpoint up to and including 13K steps (Table 4); its suite scores all come from that selected checkpoint.

Table 20: LIBERO-Plus overall success rate (%) across training steps. Co-training models mix the corresponding additional data family at equal effective-frame ratio; the baseline uses benchmark data only. Evaluation: 1,003 episodes, 4 generalization suites. The final column uses 15K for the baseline and 14,580 for Ultra; final results for the other sources are not listed. Best per-step result in **bold**.

Co-training Data	1K	3K	5K	7K	9K	11K	13K	Final
Baseline (no synthetic)	35.49	62.61	64.91	71.39	77.17	76.57	75.97	74.28
+ SimData-Easy	48.65	59.52	69.89	74.48	77.17	76.27	78.76	—
+ SimData-Lite	40.08	59.32	68.20	73.28	75.97	77.77	78.27	—
+ SimData-Hard	36.89	63.11	74.38	72.78	79.46	79.36	81.06	—
+ SimData-Ultra	65.90	81.95	84.75	88.14	86.94	87.64	89.43	89.63

Table 21: LIBERO-Plus per-suite success rate (%) at training step 13,000. All configurations evaluated at the same checkpoint step for fair comparison. Best per-suite result in **bold**.

Co-training Data	Overall	Object	Spatial	Goal	LIBERO-10
Baseline (no synthetic)	75.97	84.13	81.25	54.83	84.52
+ SimData-Easy	78.76	86.51	85.83	58.30	85.32
+ SimData-Lite	78.27	85.71	83.33	61.78	82.94
+ SimData-Hard	81.06	87.70	91.25	64.09	82.14
+ SimData-Ultra	89.43	92.86	93.33	84.17	87.70

Table 22: LIBERO-Plus per-suite success rate (%) for all configurations across training steps. Suites: Object generalization (Obj), Spatial generalization (Spa), Goal generalization (Goal), Long-horizon sequential tasks (L-10).

Co-training Data	Step	Overall	Obj	Spa	Goal	L-10
Baseline	1K	35.49	55.16	41.67	24.32	21.43
	3K	62.61	84.52	71.25	37.07	58.73
	5K	64.91	79.37	72.08	39.77	69.44
	7K	71.39	83.73	81.25	51.74	69.84
	9K	77.17	84.52	87.08	52.51	85.71
	11K	76.57	80.16	85.42	61.39	80.16
	13K	75.97	84.13	81.25	54.83	84.52
	15K	74.28	83.73	78.33	54.44	81.35
+ SimData-Easy	1K	48.65	74.60	61.25	29.73	30.16
	3K	59.52	74.60	69.58	47.10	47.62
	5K	69.89	86.51	81.25	45.95	67.06
	7K	74.48	86.51	81.67	50.19	80.56
	9K	77.17	87.30	85.42	51.35	85.71
	11K	76.27	83.73	83.75	55.21	83.33
	13K	78.76	86.51	85.83	58.30	85.32
+ SimData-Lite	1K	40.08	61.51	54.58	27.80	17.46
	3K	59.32	72.22	71.67	45.17	49.21
	5K	68.20	84.92	73.75	44.79	70.24
	7K	73.28	88.10	78.75	54.44	72.62
	9K	75.97	87.70	82.92	55.60	78.57
	11K	77.77	87.30	81.25	59.46	83.73
	13K	78.27	85.71	83.33	61.78	82.94
+ SimData-Hard	1K	36.89	59.92	54.17	22.01	12.70
	3K	63.11	84.92	75.42	43.24	50.00
	5K	74.38	86.90	85.83	58.30	67.46
	7K	72.78	87.30	82.92	51.35	70.63
	9K	79.46	87.30	86.67	62.93	81.75
	11K	79.36	86.90	87.50	60.62	83.33
	13K	81.06	87.70	91.25	64.09	82.14
+ SimData-Ultra	1K	65.90	92.46	76.25	57.14	38.49
	3K	81.95	90.08	87.92	78.38	71.83
	5K	84.75	90.87	90.83	77.61	80.16
	7K	88.14	92.06	96.25	83.01	81.75
	9K	86.94	91.67	93.75	81.08	81.75
	11K	87.64	92.06	91.67	78.38	88.89
	13K	89.43	92.86	93.33	84.17	87.70
	14580	89.63	92.46	92.50	84.56	89.29

Table 23: Improvement (Δ pp) over Baseline at step 13,000 on LIBERO-Plus. Positive values indicate gains from co-training with the corresponding synthetic data.

Co-training Data	Δ Overall	Δ Object	Δ Spatial	Δ Goal	Δ LIBERO-10
+ SimData-Easy	+2.79	+2.38	+4.58	+3.47	+0.80
+ SimData-Lite	+2.30	+1.58	+2.08	+6.95	−1.58
+ SimData-Hard	+5.09	+3.57	+10.00	+9.26	−2.38
+ SimData-Ultra	+13.46	+8.73	+12.08	+29.34	+3.18

Table 24: LIBERO-Pro overall success rate (%) across training steps. Evaluation: 200 task configurations $\times$ 5 trials = 1,000 episodes per checkpoint, spanning 4 base suites $\times$ 5 generalization dimensions. Baseline final step is 15,000; all co-training configs final step is 14,580. Best per-step result in **bold**.

Co-training Data	1K	5K	9K	13K	Final
Baseline (no synthetic)	13.8	40.6	45.3	45.9	46.7
+ SimData-Easy	15.8	42.2	47.7	47.6	47.9
+ SimData-Lite	14.6	40.8	46.6	46.8	46.9
+ SimData-Hard	13.1	43.3	45.7	46.9	47.1
+ SimData-Ultra	—	—	45.9	48.7	48.0

Table 25: LIBERO-Pro per generalization dimension success rate (%) at the final training step. Five dimensions: Object (novel objects), Swap (position-swap), Language (semantic-language), Task (novel tasks), Environment (novel environments). 200 episodes per dimension per config. Best per-dimension result in **bold**.

Co-training Data	Object	Swap	Language	Task	Environ.
Baseline	79.50	3.00	93.50	18.00	39.50
+ SimData-Easy	77.00	4.00	93.50	21.00	44.00
+ SimData-Lite	77.00	3.00	90.50	18.50	45.50
+ SimData-Hard	80.50	4.00	96.00	18.50	36.50
+ SimData-Ultra	80.50	1.00	97.00	16.50	45.00

Table 26: LIBERO-Pro per base suite success rate (%) at the final training step. Each suite contains 250 episodes (50 task configs $\times$ 5 trials). Best per-suite result in **bold**.

Co-training Data	Overall	Object	Spatial	Goal	LIBERO-10
Baseline	46.70	47.60	55.20	44.00	40.00
+ SimData-Easy	47.90	46.80	57.60	44.40	42.80
+ SimData-Lite	46.90	48.80	53.60	46.00	39.20
+ SimData-Hard	47.10	48.80	57.60	45.60	36.40
+ SimData-Ultra	48.00	45.60	60.40	48.80	37.20

Table 27: LIBERO-Pro per base suite success rate (%) for all configurations across training steps. Each suite contains 250 episodes per checkpoint.

Co-training Data	Step	Overall	Obj	Spa	Goal	L-10
Baseline	1K	13.80	16.80	20.80	12.00	5.60
	5K	40.60	46.00	49.60	37.20	29.60
	9K	45.30	47.60	53.20	40.80	39.60
	13K	45.90	48.40	53.60	44.80	36.80
	15K	46.70	47.60	55.20	44.00	40.00
+ SimData-Easy	1K	15.80	24.00	25.20	12.80	1.20
	5K	42.20	46.00	53.60	35.20	34.00
	9K	47.70	50.00	58.00	40.80	42.00
	13K	47.60	47.20	56.80	44.40	42.00
	14580	47.90	46.80	57.60	44.40	42.80
+ SimData-Lite	1K	14.60	23.20	22.80	10.80	1.60
	5K	40.80	46.40	48.00	40.00	28.80
	9K	46.60	50.00	53.20	45.60	37.60
	13K	46.80	48.00	54.80	43.60	40.80
	14580	46.90	48.80	53.60	46.00	39.20
+ SimData-Hard	1K	13.10	16.80	21.20	12.00	2.40
	5K	43.30	47.20	56.80	40.80	28.40
	9K	45.70	46.00	55.60	46.40	34.80
	13K	46.90	48.00	56.40	44.40	38.80
	14580	47.10	48.80	57.60	45.60	36.40
+ SimData-Ultra	9K	45.90	45.20	57.60	47.20	33.60
	13K	48.70	46.00	60.80	48.80	39.20
	14580	48.00	45.60	60.40	48.80	37.20

Table 28: Best listed WAM checkpoint per configuration on LIBERO-Plus and LIBERO-Pro. Ultra’s LIBERO-Pro peak is at 13K, not its final checkpoint.

Co-training data	LIBERO-Plus (WAM)	LIBERO-Pro (WAM)
Baseline	77.17	46.7
SimData-Easy	78.76	47.9
SimData-Lite	78.27	46.9
SimData-Hard	81.06	47.1
SimData-Ultra	89.63	48.7