LEGO-Anything: Coding Agents for 3D Scene Reconstruction

Xirui Li[1,2,*], **Peng Shi**[2], **Mingwen Dong**[2], **Sheng Zhang**[2], **Zhuoyan Xu**[2], **Dongkyu Lee**[2], **Shuaichen Chang**[2], **Yi Xiang**[2], **Lin Pan**[2], **Jiarong Jiang**[2]

[1]University of Maryland, College Park, [2]AWS
[*]Work done during an internship at AWS

A 3D scene reconstructed from a single image is most useful when represented not as a rendering or a fixed 3D output, but as an explicit scene program whose execution yields a scene that can be inspected, edited, and queried. We present **LEGO-Anything**, an *Image-to-Code* framework in which a coding agent builds such a program by iteratively writing and executing Blender code, inspecting the evolving scene and its renderings, and revising the program. To evaluate how well such agents recover scenes end to end, we introduce **LEGO-Bench**, a simulator-grounded benchmark with 208 images from 104 diverse scenes spanning indoor and outdoor environments. LEGO-Bench separately scores artifact validity, visible-surface geometry, and rendered appearance, and its simulator-grounded design makes the benchmark naturally extensible while retaining precise automatic evaluation. Across the evaluated coding agents, GPT-6-astra achieves the strongest overall results, with 53.4% indoor and 39.6% outdoor scores, yet substantial gaps remain between delivering valid scene artifacts and faithfully recovering scene geometry and appearance. To understand these failures, we analyze agent construction trajectories and identify three recurring issues: weak scene initialization, regressive edits during iteration, and unreliable self-evaluation. These findings motivate **LEGO-Plugin**, a training-free harness plugin for more controlled iterative scene construction with relative gains of up to 62.7% in overall score. Finally, we ask whether reconstructed scenes are faithful enough to represent natural images and support vision tasks. In **LEGO-World**, we take scenes reconstructed by GPT-6-astra and derive object detections, instance masks, and relative depth as deterministic queries on each scene. These readouts show non-trivial performance across all three tasks but fall well short of specialized vision models, suggesting that program-constructed scenes from current coding agents are a promising but not yet sufficiently precise representation of natural images.

Date: September 2026
Author E-mails: xiruili@umd.edu, penshi@amazon.com
Project Page: https://lego-anything.com

1 Introduction

The representation a 3D reconstruction takes determines what can be done with it. Compared with meshes (Gkioxari et al., 2020), point maps (Wang et al., 2024), or object sets (Ardelean et al., 2025), an executable scene program makes objects, geometry, layout, and camera explicit, and can be run, inspected, edited, and queried like any other code. Yet existing approaches rarely adopt this form. Modular pipelines compose specialized components for perception, reconstruction, asset retrieval, and scene assembly (Kim et al., 2026; Kang et al., 2026), while learned models map images directly to fixed 3D outputs (Team et al., 2026; Cao et al., 2025; Dahnert et al., 2021; Zadaianchuk et al., 2026). Neither offers a direct way to check the result against the input and revise it. Building on recent program-based approaches (Hu et al., 2024; Li et al., 2026a; Yin et al., 2026), we treat single-image scene reconstruction as the iterative construction of an explicit, executable scene program, in which each step is executed, compared with the input image, and refined.

Coding agents (Anthropic, 2026; OpenAI, 2026) offer a natural way to instantiate this representation: rather than predicting a scene in one shot, they can build it incrementally, using execution feedback to guide each revision (Yin et al., 2026). We study this setting as **LEGO-Anything**, an *Image-to-Code (Image2Code)* framework in which a general-purpose coding agent writes and executes Blender programs, inspects the

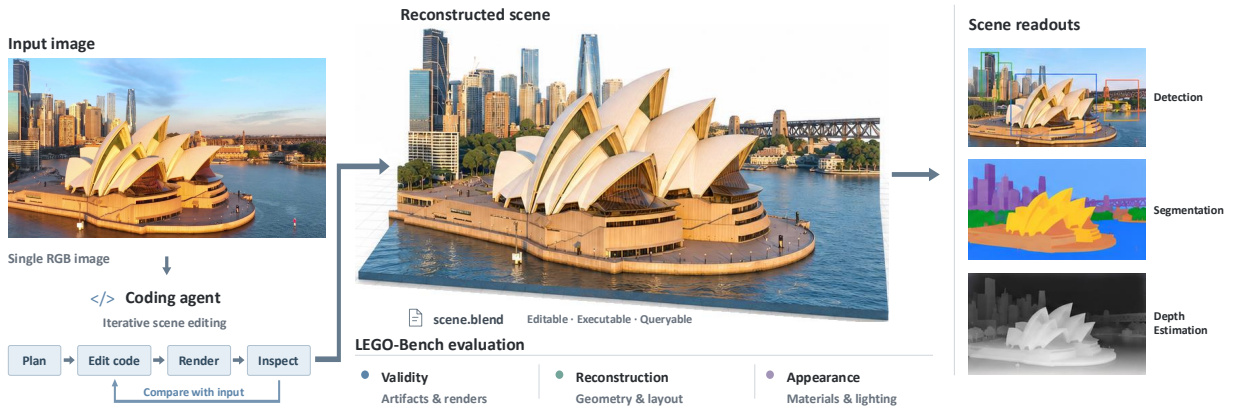


Figure 1 From image to scene program. Given a single RGB image, a coding agent iteratively plans, edits Blender code, renders, and inspects the result against the input, producing an editable and executable scene program. The resulting scene program can also serve as a representation that can be queried directly for detection, segmentation, and depth estimation.

evolving scene and its renderings, and revises the construction to match a single reference image. This formulation lets us study three questions: (1) how faithfully coding agents reconstruct scenes from a single image, (2) what limits their iterative construction process, and (3) whether the resulting scene programs are precise enough to serve as representations of natural images. Figure 1 illustrates the formulation, from iterative scene construction to querying the reconstructed scene for detection, segmentation, and depth.

To evaluate agentic end-to-end reconstruction, we need a benchmark whose inputs resemble the natural and diverse scenes that users would actually ask a system to reconstruct, while still supporting precise automatic evaluation. Existing benchmarks cover parts of this setting but not all of it: they focus on object- or part-level modeling (Gao et al., 2026; Yang et al., 2026), specify worlds through text rather than images (Lu et al., 2026a), or rely on calibrated multi-view indoor observations (Zhao et al., 2026). We therefore introduce **LEGO-Bench**, a simulator-grounded diagnostic benchmark with 208 images rendered from 104 diverse indoor and outdoor simulator scenes. Rendering from fully specified 3D scenes yields natural-image-style inputs while retaining private geometry, depth, and instance masks for evaluation, and allows scene complexity to be varied in a controlled way. LEGO-Bench separately scores artifact validity, visible-surface geometry, and rendered appearance, distinguishing failures to deliver valid scene artifacts from failures of geometric and visual fidelity. Among the evaluated coding agents and scene-construction baselines, `GPT-6-astra` achieves the strongest overall results, with 53.4% indoor and 39.6% outdoor scores.

To diagnose why end-to-end scene reconstruction still falls short, we analyze agent construction trajectories and identify three recurring failures: weak scene initialization, regressive edits during iteration, and unreliable self-evaluation. These findings motivate **LEGO-Plugin**, a training-free harness plugin with three modules, each targeting one failure: *Enhanced Initialization* grounds the starting scene in the reference image, *Grounded Refinement* replaces unreliable self-judgment with evidence from the reference, and *Version Control* protects previously correct progress from regressive edits. Rather than introducing a separate planner, LEGO-Plugin plugs into existing harnesses as tools, skills, and runtime hooks. It improves all six evaluated models on the LEGO-Bench Office subset, by up to 62.7% relative, with the largest gains for weaker agents.

We next ask whether reconstructed scenes are precise enough to serve as structured representations of natural images (Zhang et al., 2025; Wang et al., 2026; Zhu et al., 2022) for downstream vision tasks. To this end, we introduce **LEGO-World**, an evaluation setting that takes the program reconstructed from a natural image and derives object detections, instance masks, and relative depth through deterministic readouts. Without any task-specific training, these readouts achieve non-trivial performance on all three tasks but remain well below SOTA vision models, suggesting that executable scenes from current coding agents are a promising but not yet sufficiently precise representation of natural images.

Table 1 Comparison of related 3D benchmarks. ✓: supported; ✗: not supported. LEGO-Bench uniquely targets end-to-end evaluation of executable scene reconstruction across diverse, realistic scenes with controlled difficulty.

Benchmark	Image Input	Scene-level Eval.	Executable / Interactive	Indoor/Outdoor Coverage	Natural-Image Style	Controlled Difficulty
3DCodeBench (Gao et al., 2026)	✓	✗	✗	✗	✗	✗
WorldCoder-Bench (Lu et al., 2026b)	✗	✓	✓	✗	✗	✗
P3D-Bench (Yang et al., 2026)	✓	✗	✗	✗	✗	✗
SEIG (He et al., 2026)	✓	✗	✓	✗	✗	✗
SceneActBench (Zhao et al., 2026)	✓	✓	✓	✗	✗	✗
LEGO-Bench **(Ours)**	✓	✓	✓	✓	✓	✓

2 Related Work

2.1 Coding Agents

Coding agents combine language models with tools for code editing, execution, and iterative verification (Anthropic, 2026; OpenAI, 2026; Liu et al., 2026b). Their harnesses organize tool access, execution context, and environmental feedback, allowing code to serve not only as an output, but also as an interface for planning and interaction (Ning et al., 2026). Recent work extends this paradigm to visual and 3D settings. VIGA reconstructs and edits visual programs through a code–render–inspect loop (Yin et al., 2026); SEIG reconstructs images as editable Blender programs through staged executable inverse graphics (He et al., 2026); ArtiCraft (Zhou et al., 2026a) develops an agentic coding interface for articulated 3D assets; and Code-CoT (Liu et al., 2026c) uses executable 3D programs as intermediate representations for spatial reasoning. In contrast, we study general-purpose coding agents as a testbed for single-image executable scene construction, emphasizing scene fidelity, iterative failure modes, and the perceptual sufficiency of frozen reconstructed scenes.

2.2 3D Scene Construction and Evaluation

Prior work addresses important components of scene construction under different assumptions. Procedural systems generate natural, indoor, and urban environments from structured specifications and reusable assets (Deitke et al., 2022; Raistrick et al., 2023, 2024; Deng et al., 2024), while image-conditioned methods recover geometry and appearance from visual observations (Cao et al., 2026; Team et al., 2026). Other work targets editable indoor scenes from RGB-D scans (Huang et al., 2026) or simulation-ready articulated assets (Jiang et al., 2022; Liu et al., 2023; Wu et al., 2026; Zhang et al., 2026; Le et al., 2025; Cao et al., 2025). These directions provide increasingly capable components, but do not directly answer whether a single natural image can be reconstructed as a faithful executable scene.

Existing benchmarks leave different parts of this setting untested. 3DCodeBench (Gao et al., 2026) focuses on procedural object modeling, P3D-Bench (Yang et al., 2026) targets parametric parts and assemblies, and WorldCoder-Bench (Lu et al., 2026a) evaluates text-specified interactive worlds rather than fidelity to a reference image. SEIG is closely related in its executable Blender-program formulation, but its quantitative evaluations are object-centric. SceneActBench (Zhao et al., 2026) comes closest, but its reconstruction track uses calibrated multi-view indoor observations and excludes room structure and texture recovery. **LEGO-Bench** instead evaluates single-image, end-to-end executable scene reconstruction across diverse and realistic environments, separating artifact validity, visible-surface reconstruction, and rendered appearance. See Table 1 and Appendix A.5 for additional comparisons.

3 LEGO-Anything: Scene Reconstruction as Image-to-Code

LEGO-Anything formulates single-image scene reconstruction as an *Image-to-Code (Image2Code)* problem. Given a single RGB image I, a general-purpose coding agent π interacts with a 3D editing environment $\mathcal{E}$ to produce a scene program P, whose execution constructs a 3D scene S:

$$P = \pi(I; \mathcal{E}), \qquad S = \mathrm{Exec}_{\mathcal{E}}(P). \tag{1}$$

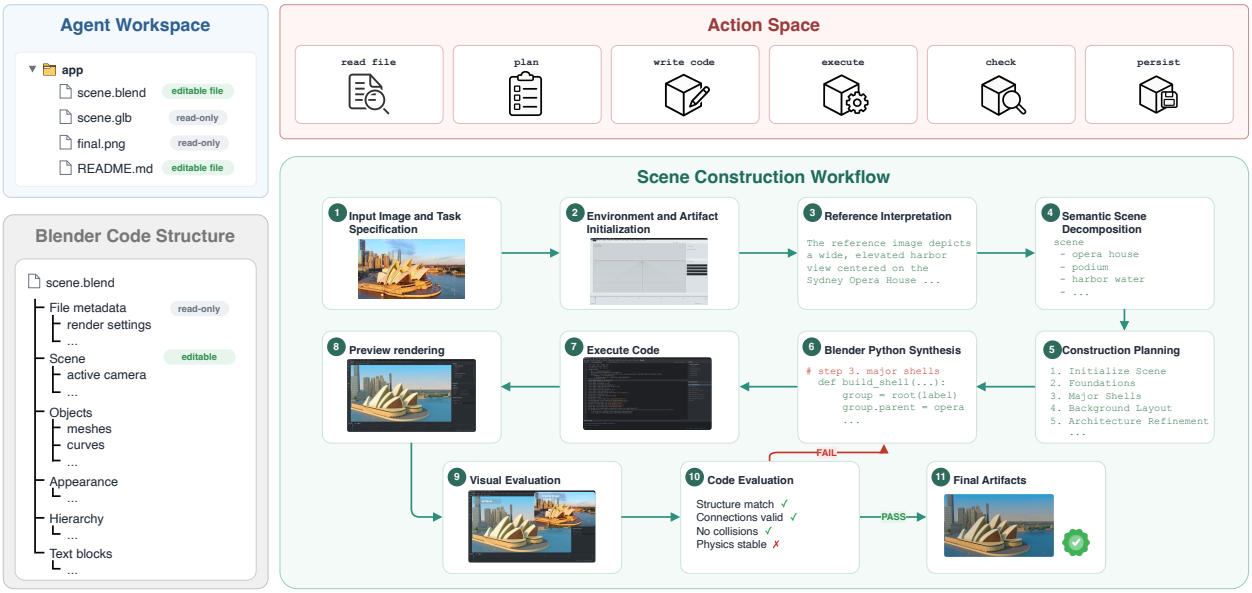


Figure 2 Agentic scene construction, abstracted from a `GPT-6-astra` trajectory. Four blocks define the process: the Agent Workspace manages artifacts, the Action Space exposes agent operations, the Blender Code Structure organizes editable scene state, and the Scene Construction Workflow iterates from reference interpretation to validated final artifacts. Together, they enable executable, feedback-driven scene reconstruction.

In our setting, $\mathcal{E}$ is Blender. Rather than producing P in one shot, the agent alternates between editing code, executing it, and inspecting the resulting scene and renderings, yielding a *construction trajectory* $\tau = \{(P_t, S_t, o_t)\}_{t=1}^{T}$ of intermediate programs, scenes, and observations, with $P = P_T$. The agent submits the final scene, its export, and a rendered view, which together form an executable artifact that can be evaluated for fidelity and queried for downstream perception. Figure 2 illustrates this workflow as abstracted from a GPT-6-astra trajectory. We evaluate final scenes in LEGO-Bench (Section 4), analyze trajectories to diagnose failures and design LEGO-Plugin (Section 5), and query frozen scenes in LEGO-World (Section 6).

4 LEGO-Bench: A Simulator-Grounded Benchmark for End-to-End Scene Reconstruction

LEGO-Bench evaluates end-to-end 3D scene reconstruction in a setting that mirrors real use: given a single natural image, an agent must reconstruct the scene as a complete, executable 3D artifact, jointly recovering its contents, geometry, spatial layout, and appearance from one view. Because every case is rendered from a fully specified simulator scene, ground truth comes for free, and the benchmark scales naturally: users can extend it with their own scenes and assets through the same construction pipeline while retaining precise automatic evaluation.

4.1 Benchmark Design and Construction

Real photographs lack precise 3D ground truth, while simple synthetic scenes lack realism. LEGO-Bench resolves this tension by rendering inputs from professionally authored simulator scenes, which (1) provides realistic, natural-image-style observations, (2) retains private geometry, object identities, camera parameters, depth, and instance masks for automatic evaluation, and (3) allows scene complexity to be controlled directly.

As shown in Figure 3, we build LEGO-Bench in LychSim (Ma et al., 2026a) using scene and asset packs from Fab, and organize indoor and outdoor scenes into themes with nested object sets (Easy $\subset$ Medium $\subset$ Hard). We select only Fab assets whose usage terms allow AI use. Within each theme, architecture, materials, lighting, and camera remain fixed while visible scene content increases, allowing matched comparisons across difficulty

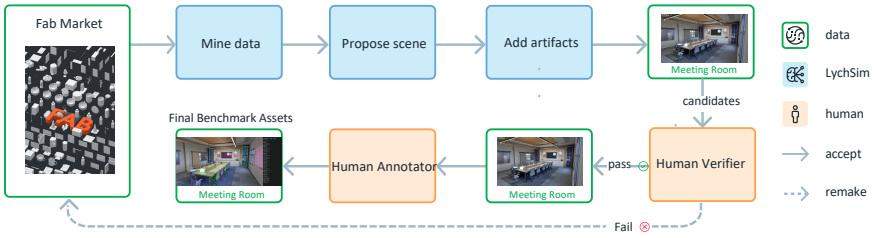


Figure 3 Extensible benchmark-construction pipeline. Fab assets are mined and assembled into candidate scenes; human verification triggers either remaking or annotation before acceptance. This human-in-the-loop process scales benchmark construction while retaining simulator ground truth and scene quality.

Table 2 Statistics of LEGO-Bench. The benchmark spans 104 scenes, eight environments, 17 themes, and 443 assets.

Statistic	LEGO-Bench
# RGB Inputs	208
# Environments	8
# Themes	17
# Logical Scenes	104
# Registered Assets	443

levels. Candidate scenes undergo collision and stability checks and human review before inclusion. Because this pipeline only requires registered assets and a scene specification, LEGO-Bench scales with available asset libraries: new environments, difficulty levels, and views can be added without changing the evaluation protocol. As an example, we include an NYC aerial split without difficulty pairing as a large-scale reconstruction stress test (Appendix B.3).

LEGO-Bench contains 208 RGB inputs from 104 scenes across 8 environments and 17 themes, using 443 registered assets (Table 2). Through the construction pipeline in Figure 3, users can convert their own scenes and assets into new evaluation cases with the same ground truth and scoring protocol. Each trial provides an RGB image and public metadata, including image dimensions, horizontal field of view, a category taxonomy, and an output schema, while reference geometry, depth, instance masks, object correspondences, and scene transforms remain private to the evaluator.

4.2 Evaluation Metrics

We evaluate each trial along three complementary axes: submission **V**alidity V_i, visible-surface **R**econstruction R_i, and rendered **A**ppearance A_i. Together, these metrics distinguish failure to deliver an evaluable scene from failures of geometric and visual fidelity. Appendix E provides implementation details, auxiliary metric protocols, and a human-validation study of the headline metrics (Appendix E.5).

Validity. Validity checks whether the submission delivers usable scene artifacts: `scene.blend` must open and contain renderable geometry and an active camera, `scene.glb` must be a well-formed, non-empty export, and `final.png` must be a parseable, non-degenerate image. Submissions that fail these checks, or for which the headline evaluation cannot be completed, receive $V_i = 0$ and $R_i = A_i = 0$. Low reconstruction quality alone does not make a submission invalid.

Reconstruction. Reconstruction measures geometric agreement on object surfaces visible in the reference view. We extract visible surfaces from the submitted scene under its active camera and derive reference-visible surfaces from private depth. Both point sets are projected onto the image plane and partitioned into scored objects by the private instance masks, then compared directly in camera coordinates without alignment or rescaling. For each object o, precision is the fraction of submitted points within a depth-scaled tolerance $\tau(g) = 0.05\, z(g)$ of the nearest reference point g, where $z(g)$ is its forward depth, and recall is defined symmetrically. The Reconstruction score (F@5%) averages object-level F1 scores:

$$F_{i,o} = \frac{2\,\mathrm{Prec}_{i,o}\,\mathrm{Rec}_{i,o}}{\mathrm{Prec}_{i,o} + \mathrm{Rec}_{i,o}}, \qquad R_i = \frac{1}{|\mathcal{O}_i|} \sum_{o \in \mathcal{O}_i} F_{i,o}, \tag{2}$$

where $\mathcal{O}_i$ contains the annotated scored objects (Appendix E.2), and $F_{i,o} = 0$ when its denominator is zero. Missing surfaces reduce recall, while extra geometry within a scored mask reduces precision. Appendix E.2 also reports stricter F@2% and relaxed F@10% variants.

Table 3 Performance (%) on LEGO-Bench. Higher is better for every metric. As a group, general-purpose coding agents reliably produce valid artifacts across both Indoor and Outdoor splits, with `GPT-6-astra` achieving the strongest overall performance.

	Indoor				Outdoor			
Model (+ Harness)	**V** ↑	**R** ↑	**A** ↑	**S** ↑	**V** ↑	**R** ↑	**A** ↑	**S** ↑
General-Purpose Coding Agents								
GPT-6-astra + Codex	**100.0** (0.0)	52.4 (1.1)	**54.4** (0.7)	**53.4** (0.8)	98.0 (1.0)	34.0 (1.5)	**45.5** (0.5)	**39.6** (0.8)
GPT-6-sol + Codex	99.4 (1.1)	22.2 (2.4)	42.5 (0.3)	32.3 (1.0)	99.7 (0.6)	19.8 (1.0)	28.7 (0.5)	24.2 (0.4)
GPT-6-luna + Codex	99.7 (0.5)	15.8 (1.2)	30.5 (0.3)	23.2 (0.7)	99.7 (0.6)	13.2 (0.6)	21.4 (0.5)	17.3 (0.5)
GPT-5.6-sol + Codex	97.8 (2.3)	9.8 (2.1)	19.8 (0.6)	14.8 (1.0)	98.7 (0.6)	12.8 (1.8)	17.7 (0.4)	15.3 (0.8)
GPT-5.6-terra + Codex	99.7 (0.5)	9.5 (1.0)	21.3 (0.5)	15.4 (0.6)	98.0 (1.7)	8.1 (0.7)	15.0 (0.9)	11.5 (0.4)
GPT-5.6-luna + Codex	99.1 (0.0)	10.2 (2.1)	18.0 (0.7)	14.1 (1.3)	97.0 (1.0)	7.8 (1.5)	16.6 (0.8)	12.1 (1.0)
Task-Specific LLM Systems								
VIGA (Yin et al., 2026)	**100.0** (0.0)	10.8 (1.5)	20.3 (0.7)	15.6 (1.1)	**100.0** (0.0)	8.0 (0.4)	17.1 (0.0)	12.6 (0.2)
SceneConductor (Kim et al., 2026)	12.3 (2.3)	5.0 (2.1)	0.0 (0.0)	0.3 (0.2)	–	–	–	–
Single-Image Scene Construction Baselines								
3D-RE-GEN (Sautter et al., 2025)	**100.0** (0.0)	1.2 (0.1)	11.3 (0.2)	6.3 (0.1)	–	–	–	–
Gen3DSR (Ardelean et al., 2025)	92.0 (1.1)	**65.4** (0.7)	0.0 (0.0)	30.1 (0.4)	80.0 (1.0)	**35.0** (0.5)	0.0 (0.0)	14.0 (0.1)
REST3D (Ma et al., 2026b)	77.5 (13.0)	7.2 (0.7)	0.0 (0.0)	2.8 (0.4)	–	–	–	–
SceneGen (Xia et al., 2025)	71.6 (3.0)	9.7 (1.8)	0.0 (0.0)	3.4 (0.5)	38.7 (1.5)	2.6 (1.1)	0.0 (0.0)	0.5 (0.2)

Appearance. Appearance evaluates visual similarity between the reconstructed scene and the reference image. Rather than scoring the agent's own `final.png`, the evaluator re-renders the submitted scene, preserving its camera, geometry, materials, lighting, and color settings while standardizing the rendering engine and output resolution. Let $\hat{I}_i$ and I_i denote the evaluator render and reference image in 8-bit sRGB, with pixel domain Ω_i. Appearance is the fraction of pixels whose error does not exceed a threshold δ in any channel:

$$A_i = \frac{1}{|\Omega_i|} \sum_{u \in \Omega_i} \mathbf{1}\left[\left\| \hat{I}_i(u) - I_i(u) \right\|_\infty \leq \delta\right]. \tag{3}$$

This metric captures the combined effect of scene geometry, materials, lighting, and camera configuration on the rendered view. We set $\delta = 30$ (on a 0-255 scale) based on our sensitivity analysis (Appendix E.3).

Overall score. Valid submissions receive the mean of Reconstruction and Appearance. Artifact-invalid submissions and unresolved evaluator failures receive zero. The benchmark score averages over all N attempted cases, including failures:

$$S = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2}(R_i + A_i) * V_i. \tag{4}$$

4.3 Main Results

Experimental setup. We evaluate GPT-series coding agents under the Codex harness (Bolin, 2026), using each model's native harness behavior. All methods run in a shared execution environment based on the Harbor Framework (Harbor Framework Team, 2026), with `Blender 5.0.1` (Blender Foundation) and `Blender-MCP` (ahujasid, 2025). We report mean and standard deviation over three runs; Appendix F details baseline-specific configurations.

How well do coding agents reconstruct 3D scenes? Table 3 reveals a clear gap between artifact delivery and faithful scene reconstruction. All six GPT configurations achieve near-saturated Validity, yet fidelity differs sharply by model: overall scores range from 53.4%/39.6% for `GPT-6-astra` to 15.4%/15.3% for the strongest `GPT-5.6` configurations. Among baselines, Gen3DSR attains higher Reconstruction than `GPT-6-astra` (65.4% vs. 52.4% indoors) but produces no evaluable appearance, while VIGA and 3D-RE-GEN deliver valid artifacts with low fidelity. Three of six baselines do not support outdoor scenes, whereas all coding agents run on both splits with stable Validity, although fidelity drops outdoors for every model.

Finding 1: GPT-6-astra achieves the best overall score, surpassing task-specific and single-image scene-construction baselines. Coding agents deliver valid artifacts across indoor and outdoor scenes.

How does scene complexity affect performance? Increasing scene complexity degrades fidelity rather than executability. Averaged over all GPT-6 and GPT-5.6 configurations, Validity stays at 99.5% across tiers, while the overall score drops from 24.6% (Easy) to 21.3% (Medium) and 20.5% (Hard), with consistent declines in Reconstruction and Appearance on both splits (Table 4). The drop is steepest for outdoor Reconstruction (18.4% → 12.7%).

Table 4 Performance across scene-complexity tiers.

	Indoor		Outdoor	
Tier	R	A	R	A
Easy	23.3 (0.1)	30.4 (0.6)	18.4 (0.9)	25.3 (0.4)
Medium	19.7 (1.2)	27.9 (0.6)	14.2 (1.0)	22.8 (0.9)
Hard	18.8 (0.4)	27.4 (0.5)	12.7 (1.0)	22.4 (0.3)

Finding 2: Greater scene complexity lowers both Reconstruction and Appearance while leaving Validity intact, and outdoor scenes are harder than indoor scenes at every complexity tier.

Does more reasoning improve scene reconstruction? We run a test-time scaling experiment on a fixed 42-case Office subset, varying the harness-native reasoning effort (Low, Medium, High, XHigh) while keeping inputs, prompts, execution budgets, and scoring fixed. For all three GPT-6 models, the overall score rises with effort (Figure 4): astra from 32.3% to 61.8%, sol from 21.3% to 39.7%, and luna from 14.4% to 21.2%. Gains grow with model strength, and astra saturates toward XHigh while sol keeps improving. By contrast, GPT-5.6 models show weak or non-monotonic changes (Appendix F.4).

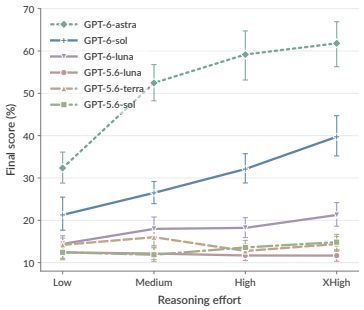


Figure 4 Test-time scaling improves GPT-6 performance.

Finding 3: Higher reasoning effort improves all three GPT-6 models, with larger gains for stronger models, while GPT-5.6 models show no consistent improvement.

5 LEGO-Plugin: From Failure Diagnosis to Reliable Scene Reconstruction

LEGO-Bench scores the submitted scene but collapses the construction process into a single outcome. A poor result may stem from a weak or delayed initial scene, edits that undo earlier progress, or an agent's failure to judge whether its scene is improving. We therefore analyze intermediate artifacts and test whether agents can reliably evaluate their own artifacts. These diagnoses motivate **LEGO-Plugin**, a training-free harness plugin whose three modules, *Enhanced Initialization*, *Version Control*, and *Grounded Refinement*, each target one failure.

5.1 Diagnosing Construction Failures

Trajectory analysis. We re-evaluate every renderable intermediate artifact in the trajectories from our main experiments with the LEGO-Bench metrics. This yields three trajectory-level quantities: time to the first evaluable scene, the frequency of score-decreasing edits, and the gap between the best intermediate score and the final submission. We contrast GPT-6-astra and GPT-5.6-sol here and report all four models in Appendix B.2. As shown in Figure 5, GPT-6-astra reaches its first evaluable scene within roughly the first tenth of its budget, whereas GPT-5.6-sol needs about one fifth, leaving less budget for render-based refinement. After initialization, GPT-6-astra improves more steadily, while GPT-5.6-sol often stagnates or declines: 29.6% of its edits decrease the score, and its final submission trails its best intermediate scene by 3.2 points. These regressions stem from later edits that damage geometry, camera settings, or a previously stronger scene.

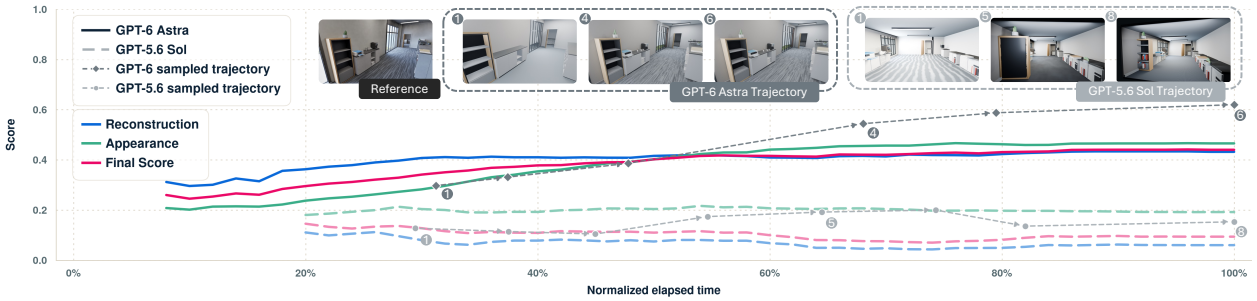


Figure 5 Scene quality over construction time. `GPT-6-astra` reaches an evaluable scene earlier and refines it steadily, whereas `GPT-5.6-sol` starts later and regresses more often. The gap shows that reliable construction depends on fast initialization and protection against regressive edits.

Finding 4: Reconstruction quality is non-monotonic over the construction trajectory: weaker agents take longer to produce an evaluable scene, and subsequent edits can still degrade scene fidelity.

Can coding agents judge their own scenes? We next test whether agents can tell when a scene improves. Given two renderings of the same case, a judge model picks the better one, and we measure agreement with the direction given by deterministic LEGO-Bench metrics (chance: 50%). Across all 36 builder–judge pairs of six models, self-judgments agree 45.8% of the time on Reconstruction and 62.2% on Appearance, versus 45.4% and 63.9% for cross-model judgments. Judges are thus near or below chance on geometry, and self-judging beats the mean of the other five judges for only 3 of 6 builders on Reconstruction and 2 of 6 on Appearance (Figure 6; protocol in Appendix F.3).

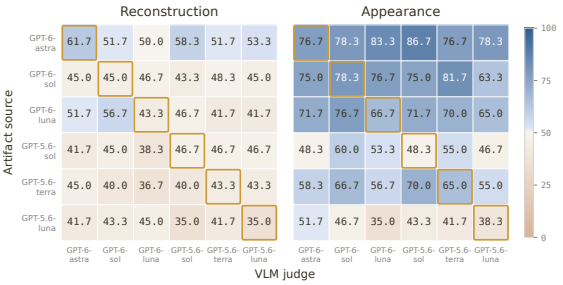


Figure 6 Judge agreement with metric directions. Reconstruction judgments remain near or below chance, and self-judging provides no consistent advantage over cross-model judging.

Finding 5: Coding agents cannot reliably judge scene quality, especially geometry, and self-judging offers no advantage over cross-model judging. Refinement should therefore be grounded in deterministic evidence rather than self-assessment.

5.2 LEGO-Plugin

LEGO-Plugin is a training-free control layer exposed as MCP tools, workflow skills, and runtime hooks, rather than a separate planner or memory module. The agent remains the planner and Blender MCP remains the scene editor, so LEGO-Plugin stays compatible with evolving harnesses while preserving native agent behavior. All plugin signals are derived from the reference image alone; private benchmark ground truth is never accessed. LEGO-Plugin comprises three modules, each targeting one failure mode from Section 5.1. We illustrate the workflow in Figure 7 .

Enhanced Initialization. LEGO-Plugin recovers the scene frame and camera from the reference image with VGGT (Wang et al., 2025), then derives visible-object layout cues to form an explicit initialization proposal. This gives the agent a reference-aligned starting scene instead of an unconstrained first placement.

Grounded Refinement. Since agents judge their own scenes unreliably (Section 4), Grounded Refinement replaces free-form self-correction with tool-grounded measurement. It extracts reference object regions with SAM 3 (Carion et al., 2026) and relative depth with Depth Anything V2 (Yang et al., 2024), then compares

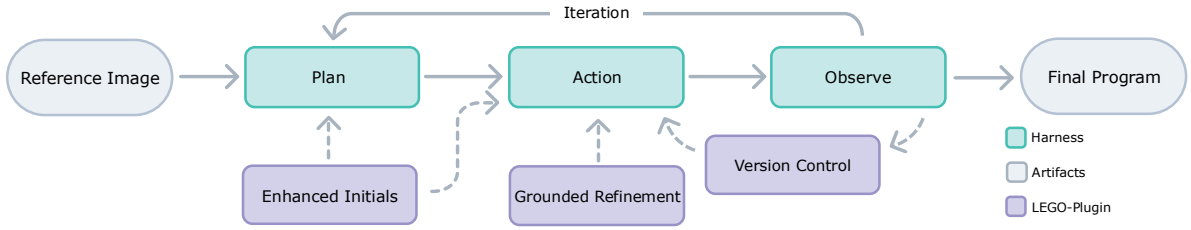


Figure 7 LEGO-Plugin. A harness-compatible plugin adds Enhanced Initialization, Grounded Refinement, and Version Control to the vanilla coding-agent harness. These modules directly address weak initialization, unreliable self-evaluation, and regressive edits without changing the underlying agent.

Figure 9 Qualitative LEGO-Plugin examples. Relative to the best of three GPT-5.6-sol base runs, LEGO-Plugin better matches both references and more than doubles overall scores (0.147→0.325; 0.109→0.278).

the executed scene against these targets through explicit residuals, such as projected extent and relative depth, rather than global visual judgment.

Version Control. LEGO-Plugin treats each bounded update as a candidate revision and scores the resulting scene against reference-grounded residuals. Each update is accepted, repaired, or rolled back to the last accepted state, preventing later edits from silently overwriting earlier progress.

5.3 LEGO-Plugin Evaluation

Protocol. We evaluate LEGO-Plugin on the 42-case Office subset (Section 4), comparing each model with and without the plugin under identical inputs, prompts, execution budgets, reasoning effort, and evaluator. As noted in Section 5, the plugin sees only the reference image and the current Blender scene, never private ground truth or LEGO-Bench scores, so it guides construction without optimizing against the evaluator.

Main comparison. We compare each model with and without LEGO-Plugin on the Office subset, reporting the overall LEGO-Bench score for six model settings (Figure 8). LEGO-Plugin improves all six models, with gains inversely related to base performance: the three GPT-5.6 models improve by 55–63% relative (about 7–8 points), GPT-6-luna and GPT-6-sol by 27.5% and 12.1%, and GPT-6-astra by only 2.1%. This pattern suggests that the plugin mainly compensates for the initialization, regression, and self-evaluation failures that weaker agents exhibit, while the strongest agent already avoids most of them.

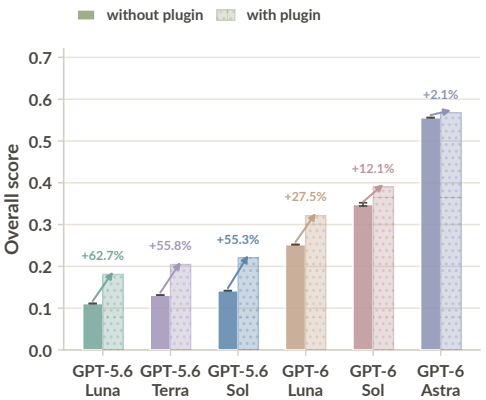


Figure 8 LEGO-Plugin comparison. LEGO-Plugin improves all six models.

Finding 6: LEGO-Plugin improves every model without training, with the largest gains for weaker agents, narrowing but not closing the gap to the strongest model.

6 LEGO-World: Reconstructed Scenes as Visual Representations

We next ask whether reconstructed scenes can serve as representations of natural images. Because each reconstructed artifact is an executable scene program, object detection, instance segmentation, and relative depth can be obtained as deterministic readouts of the same frozen scene rather than as separate task-specific predictions (Figure 1). For task k, a deterministic readout ρ_k produces the prediction:

$$I \xrightarrow{\pi, \mathcal{E}} P \xrightarrow{\text{Exec}_{\mathcal{E}}} S \xrightarrow{\rho_k} \hat{y}_k. \tag{5}$$

We test whether a single reconstructed scene supports three standard vision tasks. We reconstruct scenes with `GPT-6-astra`, the strongest agent on LEGO-Bench, without LEGO-Plugin, and let the agent label each object from the target dataset's category set. From each frozen scene, visible object instances are projected into 2D boxes for detection and instance masks for segmentation, and camera-space depth renderings provide relative depth. We evaluate on 100 randomly sampled images each from COCO val2017 (Lin et al., 2015), LVIS v1 val (Gupta et al., 2019), and ETH3D (Schops et al., 2017). Since scene exports provide no calibrated confidences, box and mask AP follow an equal-confidence protocol (Appendix H.2); coverage is reported in Table 9. As shown in Table 5, scene readouts reach 30.1 box AP, 14.8 mask AP, and 0.155 AbsRel, compared with 59.9, 54.0, and 0.078 for DINO (Siméoni et al., 2025), SAM 3 (Carion et al., 2026), and Depth Anything 3 (Lin et al., 2025).

Table 5 Task readout from frozen reconstructed scenes on three 100-image tracks. Without task-specific training, the same scenes support detection, segmentation, and depth, but remain substantially behind specialist models.

Task	Metric	Methods	Score
Detection			
COCO BBox	AP ↑	DINO	59.88
		LEGO-Anything	30.14
Segmentation			
LVIS Mask	AP ↑	Segment Anything 3	53.96
		LEGO-Anything	14.75
Depth Estimation			
ETH3D Depth	AbsRel ↓	Depth Anything 3	0.0783
		LEGO-Anything	0.1554

Finding 7: Scenes reconstructed by current coding agents already support detection, segmentation, and depth without task-specific training, reaching about half of specialist box AP, but fall well short of specialized models. Executable scenes from current coding agents are thus a promising but not yet sufficiently precise representation of natural images.

7 Conclusion

We presented **LEGO-Anything**, an Image-to-Code framework in which coding agents iteratively reconstruct 3D scenes from a single image as editable, executable scene programs. **LEGO-Bench** separately measures artifact validity, visible-surface geometry, and rendered appearance, revealing that current agents reliably deliver valid scene artifacts but remain limited in geometric fidelity. Trajectory analysis traces this gap to weak initialization, regressive edits, and unreliable self-evaluation, which **LEGO-Plugin** addresses through Enhanced Initialization, Version Control, and Grounded Refinement, improving all six evaluated models without training. Finally, **LEGO-World** shows that a single frozen scene already supports detection, segmentation, and depth readouts, though well below specialized models. Together, these results position executable scene programs as a promising, measurable, and diagnosable target for general-purpose coding agents, and we hope LEGO-Bench's extensible design supports future progress toward faithful scene reconstruction.

Acknowledgment

We thank Longxuan Yu, Bale Chen, Jiyeon Kim, Janet Wang, Tianyi Zhou, Mustafa Kaba, and Hideo Kobayashi for their invaluable support and discussion.

References

ahujasid. Blendermcp: Blender model context protocol integration, 2025. https://github.com/ahujasid/blender-mcp. GitHub repository, accessed 2026-08-06.

Anthropic. Claude code by anthropic: Ai coding agent, terminal, ide. https://claude.com/product/claude-code, 2026. Accessed: 2026-06-27.

Andreea Ardelean, Mert Özer, and Bernhard Egger. Gen3dsr: Generalizable 3d scene reconstruction via divide and conquer from a single view, 2025. https://arxiv.org/abs/2404.03421.

Blender Foundation. Blender. https://www.blender.org/.

Clémentin Boittiaux and Vincent Lepetit. Buildee: A 3d simulation framework for scene exploration and reconstruction with understanding. In *NeurIPS 2025 Workshop on Synthetic Data for Computer Vision*, 2025. https://openreview.net/forum?id=1LmsiOaMTy.

Michael Bolin. Unrolling the codex agent loop. https://openai.com/index/unrolling-the-codex-agent-loop/, January 2026. Accessed: 2026-08-06.

Ziang Cao, Fangzhou Hong, Zhaoxi Chen, Liang Pan, and Ziwei Liu. Physx-anything: Simulation-ready physical 3d assets from single image, 2025. https://arxiv.org/abs/2511.13648.

Ziang Cao, Zhaoxi Chen, Liang Pan, and Ziwei Liu. Collaborative multi-modal coding for high-quality 3d generation, 2026. https://arxiv.org/abs/2508.15228.

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, 2026. https://arxiv.org/abs/2511.16719.

Dongping Chen, Xuanao Huang, Zhihan Hu, Qingyuan Shi, Dianqi Li, and Tianyi Zhou. Sandboxed coding agents are competitive omni-modal task solvers, 2026. https://arxiv.org/abs/2606.00579.

Jiacheng Chen, Ramin Mehran, Xuhui Jia, Saining Xie, and Sanghyun Woo. Blenderfusion: 3d-grounded visual editing and generative compositing, 2025. https://arxiv.org/abs/2506.17450.

Manuel Dahnert, Ji Hou, Matthias Nießner, and Angela Dai. Panoptic 3d scene reconstruction from a single rgb image, 2021. https://arxiv.org/abs/2111.02444.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation, 2022. https://arxiv.org/abs/2206.06994.

Jie Deng, Wenhao Chai, Junsheng Huang, Zhonghan Zhao, Qixuan Huang, Mingyan Gao, Jianshu Guo, Shengyu Hao, Wenhao Hu, Jenq-Neng Hwang, Xi Li, and Gaoang Wang. Citycraft: A real crafter for 3d city generation, 2024. https://arxiv.org/abs/2406.04983.

Max Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi "Jim" Fan. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation, 2026. https://arxiv.org/abs/2603.22435.

Yipeng Gao, Lei Shu, Genzhi Ye, Xi Xiong, Ameesh Makadia, Meiqi Guo, Laurent Itti, and Jindong Chen. 3dcodebench: Benchmarking agentic procedural 3d modeling via code, 2026. https://arxiv.org/abs/2606.01057.

Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn, 2020. https://arxiv.org/abs/1906.02739.

Agrim Gupta, Piotr Dollár, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation, 2019. https://arxiv.org/abs/1908.03195.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, 2026. https://doi.org/10.5281/zenodo.20953922.

Guangzhao He, Rundong Luo, Wei-Chiu Ma, and Hadar Averbuch-Elor. Thinking in blender: Staged executable inverse graphics with vision-language models, 2026. https://arxiv.org/abs/2606.02580.

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. https://arxiv.org/abs/2403.01248.

Zhening Huang, Xiaoyang Wu, Fangcheng Zhong, Hengshuang Zhao, Matthias Nießner, and Joan Lasenby. Litereality: Graphics-ready 3d scene reconstruction from rgb-d scans, 2026. https://arxiv.org/abs/2507.02861.

Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction, 2022. https://arxiv.org/abs/2202.08227.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. https://arxiv.org/abs/2310.06770.

Haoqiang Kang, Xiaokang Ye, Yuhan Liu, Siddhant Hitesh Mantri, Lingjun Mao, James Fleming, Drishti Regmi, and Lianhui Qin. Simworld studio: Automatic environment generation with evolving coding agent for embodied agent learning, 2026. https://arxiv.org/abs/2605.09423.

Andrej Karpathy. autoresearch: Autonomous pretraining research swarm, 2026. https://github.com/karpathy/autoresearch. GitHub repository.

Jeonghwan Kim, Yushi Lan, Yongwei Chen, Hieu Trung Nguyen, Chuanyu Pan, and Xingang Pan. Sceneconductor: 3d scene generation from a single image with multi-agent orchestration, 2026. https://arxiv.org/abs/2606.08402.

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, 2025. https://arxiv.org/abs/2410.13882.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. https://arxiv.org/abs/2603.28052.

Kai Li, Lutao Jiang, Zhenyang Li, Jiayu Dong, Jierui Zhang, Yingda Yin, Runze Zhang, Kai Yan, Xiaoyang Huang, Keyang Luo, Xin Wang, Xiangyu Zhao, and Weikai Chen. Scenix: Sparse-view 3d scene reconstruction via executable scene programs, 2026a. https://arxiv.org/abs/2608.07012.

Xirui Li, Ming Li, Ion Stoica, Cho-Jui Hsieh, and Tianyi Zhou. Clawenvkit: Automatic environment generation for claw-like agents, 2026b. https://arxiv.org/abs/2604.18543.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views, 2025. https://arxiv.org/abs/2511.10647.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. https://arxiv.org/abs/1405.0312.

Haowen Liu, Xirui Li, Shaoxiong Yao, Peng Shi, Tianyi Zhou, Jia-Bin Huang, Furong Huang, and Jiayuan Mao. Guava: An effective and universal harness for embodied manipulation, 2026a. https://arxiv.org/abs/2606.18363.

Jiacheng Liu, Xiaohan Zhao, Xinyi Shang, and Zhiqiang Shen. Dive into claude code: The design space of today's and future ai agent systems, 2026b. https://arxiv.org/abs/2604.14228.

Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. Paris: Part-level reconstruction and motion analysis for articulated objects, 2023. https://arxiv.org/abs/2308.07391.

Junze Liu, Kun Qian, Florian Dubost, Kai Zhong, Arvind Srinivasan, Nan Chen, Anping Wang, Sam Zhang, Alejandro Mottini, Qingjun Cui, and Tian Wang. 3d primitives are a spatial language for vlms, 2026c. https://arxiv.org/abs/2605.12586.

Shuo Lu, Yinuo Xu, Kecheng Yu, Siru Jiang, Yongcan Yu, Yubin Wang, Haitao Yang, Yuxiang Zhang, Bin Wang, Ran He, and Jian Liang. Worldcoder-bench: Benchmarking physically grounded 3d world synthesis, 2026a. https://arxiv.org/abs/2606.01869.

Shuo Lu, Yinuo Xu, Kecheng Yu, Siru Jiang, Yongcan Yu, Yubin Wang, Haitao Yang, Yuxiang Zhang, Bin Wang, Ran He, et al. Worldcoder-bench: Benchmarking physically grounded 3d world synthesis. *arXiv preprint arXiv:2606.01869*, 2026b.

Sining Lu, Guan Chen, Nam Anh Dinh, Itai Lang, Ari Holtzman, and Rana Hanocka. LL3M: Large language 3d modelers, 2025. https://arxiv.org/abs/2508.08228.

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. https://arxiv.org/abs/2509.06784.

Wufei Ma, Chloe Wang, Siyi Chen, Jiawei Peng, Patrick Li, and Alan Yuille. Lychsim: A controllable and interactive simulation framework for vision research, 2026a. https://arxiv.org/abs/2605.12449.

Xiaoxuan Ma, Jiashun Wang, Nicolas Ugrinovic, Yehonathan Litman, and Kris Kitani. Rest3d: Reconstructing physically stable 3d scenes from a single image, 2026b. https://arxiv.org/abs/2605.30338.

Xuying Ning, Katherine Tieu, Dongqi Fu, Tianxin Wei, Zihao Li, Yuanchen Bei, Jiaru Zou, Mengting Ai, Zhining Liu, Ting-Wei Li, Lingjie Chen, Yanjun Zhao, Ke Yang, Bingxuan Li, Cheng Qian, Gaotang Li, Xiao Lin, Zhichen Zeng, Ruizhong Qiu, Sirui Chen, Yifan Sun, Xiyuan Yang, Ruida Wang, Rui Pan, Chenyuan Yang, Dylan Zhang, Liri Fang, Zikun Cui, Yang Cao, Pan Chen, Dorothy Sun, Ren Chen, Mahesh Srinivasan, Nipun Mathur, Yinglong Xia, Hong Li, Hong Yan, Pan Lu, Lingming Zhang, Tong Zhang, Hanghang Tong, and Jingrui He. Code as agent harness, 2026. https://arxiv.org/abs/2605.18747.

Nous Research. Hermes agent. https://github.com/NousResearch/hermes-agent, 2026. Pinned revision aa6f77596b, accessed 2026-08-20.

OpenAI. Codex: Ai coding partner from openai. https://openai.com/codex/, 2026. Accessed: 2026-06-27.

OpenClaw contributors. OpenClaw: Personal ai assistant, 2026. https://github.com/openclaw/openclaw. GitHub repository.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation, 2023. https://arxiv.org/abs/2306.09310.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation, 2024. https://arxiv.org/abs/2406.11824.

Tobias Sautter, Jan-Niklas Dihlmann, and Hendrik P. A. Lensch. 3d-re-gen: 3d reconstruction of indoor scenes with a generative framework, 2025. https://arxiv.org/abs/2512.17459.

Thomas Schops, Johannes L Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3260–3269, 2017.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025. https://arxiv.org/abs/2508.10104.

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, 2026. https://arxiv.org/abs/2511.16624.

Hanyang Wang, Yimo Cai, Weiliang Chen, Jiawei Chi, Haowen Sun, Qiyu Dai, Yi-Hsin Hung, Xingzhuo Guo, Jinshan Ren, Runmao Yao, Ziwei Liu, Mingsheng Long, Yueqi Duan, Jun Gao, Jiangran Lyu, Fangfu Liu, and Jialong Wu. Code as worlds: Agentic discovery of executable world representations for physical reasoning, 2026. https://arxiv.org/abs/2608.27549.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer, 2025. https://arxiv.org/abs/2503.11651.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy, 2024. https://arxiv.org/abs/2312.14132.

Zhuangzhe Wu, Yue Xin, Chengkai Hou, Minghao Chen, Yaoxu Lyu, Jieyu Zhang, and Shanghang Zhang. Urdf-anything+: End-to-end generation for simulation-ready articulated assets, 2026. https://arxiv.org/abs/2603.14010.

Xiao Xia, Dan Zhang, Zibo Liao, Zhenyu Hou, Tianrui Sun, Jing Li, Ling Fu, and Yuxiao Dong. Scenegenagent: Precise industrial scene generation with coding agent, 2025. https://arxiv.org/abs/2410.21909.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

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. https://arxiv.org/abs/2509.08643.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2, 2024. https://arxiv.org/abs/2406.09414.

Yikang Yang, Zhanpeng Hu, Youtian Lin, Mengqi Zhou, Jingxi Xu, Feihu Zhang, Jiaheng Liu, and Yao Yao. P3d-bench: Benchmarking mllms for parametric 3d generation and structural reasoning, 2026. https://arxiv.org/abs/2606.11152.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, Qi Liu, Zhifang Sui, and Tong Yang. Claw-eval: Towards trustworthy evaluation of autonomous agents, 2026. https://arxiv.org/abs/2604.06132.

Shaofeng Yin, Jiaxin Ge, Zora Zhiruo Wang, Chenyang Wang, Xiuyu Li, Michael J. Black, Trevor Darrell, Angjoo Kanazawa, and Haiwen Feng. Vision-as-inverse-graphics agent via interleaved multimodal reasoning, 2026. https://arxiv.org/abs/2601.11109.

Cunxi Yu, Chenhui Deng, Nathaniel Pinckney, and Brucek Khailany. Agentic hardware design as repository-level code evolution, 2026. https://arxiv.org/abs/2606.28279.

Andrii Zadaianchuk, Leonardo Barcellona, Lennard Schuenemann, Christian Gumbsch, Zehao Wang, Muhammad Zubair Irshad, Fabien Despinoy, Rahaf Aljundi, Stratis Gavves, and Sergey Zakharov. Reconstruction by generation: 3d multi-object scene reconstruction from sparse observations, 2026. https://arxiv.org/abs/2604.27106.

Chuanrui Zhang, Minghan Qin, Yuang Wang, Baifeng Xie, Hang Li, and Ziwei Wang. Simart: Decomposing monolithic meshes into sim-ready articulated assets via mllm, 2026. https://arxiv.org/abs/2603.23386.

Yunzhi Zhang, Zizhang Li, Matt Zhou, Shangzhe Wu, and Jiajun Wu. The scene language: Representing scenes with programs, words, and embeddings. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24625–24634. IEEE, 2025.

Yifei Zhao, Xiangxin Zhou, Wenhao Yang, Jiaqi Tang, Pu Jian, Huanjin Yao, Jiarui Yao, Haowei Lin, Chunchao Guo, Zhuo Chen, Wenkai Lyu, Jianzhu Ma, Xueqian Wang, and Wenxi Zhu. Sceneactbench: Can agents act on the 3d scenes they see?, 2026. https://arxiv.org/abs/2607.22393.

Matt Zhou, Ruining Li, Xiaoyang Lyu, Zhaomou Song, Zhening Huang, Chuanxia Zheng, Christian Rupprecht, Andrea Vedaldi, and Shangzhe Wu. Articraft: An agentic system for scalable articulated 3d asset generation, 2026a. https://arxiv.org/abs/2605.15187.

Rong Zhou, Dongping Chen, Zihan Jia, Yao Su, Yixin Liu, Yiwen Lu, Dongwei Shi, Yue Huang, Tianyang Xu, Yi Pan, et al. Digital twin ai: Opportunities and challenges from large language models to world models. *arXiv preprint arXiv:2601.01321*, 2026b.

Guangming Zhu, Liang Zhang, Youliang Jiang, Yixuan Dang, Haoran Hou, Peiyi Shen, Mingtao Feng, Xia Zhao, Qiguang Miao, Syed Afaq Ali Shah, and Mohammed Bennamoun. Scene graph generation: A comprehensive survey, 2022. https://arxiv.org/abs/2201.00443.

Contents

Appendix

A Additional Related Work

The main paper focuses on work most directly related to end-to-end image-conditioned 3D scene reconstruction. Here, we briefly situate LEGO-Anything within adjacent lines on procedural scene generation, object and asset reconstruction, coding agents, and executable 3D benchmarks.

A.1 Procedural Scene Generation and Simulation Environments

Procedural systems construct large scene collections from rules and reusable asset libraries. ProcTHOR (Deitke et al., 2022) targets embodied indoor environments; Infinigen and Infinigen Indoors (Raistrick et al., 2023, 2024) generate photorealistic natural and indoor scenes; and CityCraft (Deng et al., 2024) extends procedural construction to cities. These systems provide scalable generation once structured specifications are available. LEGO-Anything instead studies whether an agent can infer such an executable specification from a single reference image.

A.2 Object Reconstruction and Asset Generation

Object-level image-to-3D methods recover geometry and appearance from visual observations. Representative systems include TriMM (Cao et al., 2026), SAM 3D (Team et al., 2026), and LiteReality (Huang et al., 2026). Such models provide increasingly capable geometric and appearance priors, making them useful components of larger scene-construction pipelines. However, reconstructing an isolated object does not by itself determine how multiple objects should be selected, positioned, articulated, and integrated into a coherent executable scene.

Related work also studies articulated and simulation-compatible assets. Ditto (Jiang et al., 2022) and PARIS (Liu et al., 2023) recover movable parts and articulation from observations. More recent systems generate or refine structured articulated assets, including URDF-Anything+ (Wu et al., 2026), SIMART (Zhang et al., 2026), and Articulate-Anything (Le et al., 2025). PhysX-Anything (Cao et al., 2025) further targets physical attributes and simulation-ready representations. ArtiCraft (Zhou et al., 2026a) develops an agentic coding interface for articulated-object generation, illustrating how executable programs can expose structured modeling operations to a general-purpose coding agent. These lines are complementary to our setting: they strengthen object-level geometry, appearance, or functionality, whereas LEGO-Anything evaluates image-conditioned assembly of full executable scenes.

A.3 Scene Reconstruction and Editable 3D Intermediates

Holistic scene-reconstruction methods recover multiple objects, geometry, semantics, or layout from partial visual evidence. Panoptic 3D scene reconstruction from a single RGB image jointly reasons about geometry, semantics, and instances in a fixed scene representation (Dahnert et al., 2021). RecGen reconstructs multi-object scenes from sparse RGB-D observations through a generative framework, directly addressing ambiguity, occlusion, and uncertainty under incomplete views (Zadaianchuk et al., 2026). These works are closely related in task motivation, but they differ from LEGO-Anything in output interface: we ask whether a general coding agent can produce an executable Blender scene program from a single RGB image, rather than predicting a fixed panoptic or generative scene representation.

Editable 3D intermediates are also useful beyond reconstruction. BlenderFusion uses 3D-grounded Blender scenes for visual editing and generative compositing, showing that explicit scene state, camera control, and object manipulation can support image-conditioned visual tasks (Chen et al., 2025). This is adjacent to our LEGO-World motivation: executable scenes are valuable not only as reconstruction artifacts, but also as inspectable representations that can be queried, edited, and reused by downstream procedures.

A.4 Coding Agents and Harnesses

Coding-agent systems increasingly use code as the operational interface for planning, tool use, environment interaction, and execution-based verification (Jimenez et al., 2024; Anthropic, 2026; OpenAI, 2026; OpenClaw contributors, 2026; Liu et al., 2026b). Agent harnesses organize these interactions by exposing tools, maintaining execution state, and returning environmental feedback to the underlying model (Ning et al., 2026). Recent work has studied automatic harness optimization (Lee et al., 2026) and extended coding agents to scientific, hardware, simulated, and physical environments (Karpathy, 2026; Li et al., 2026b; Yu et al., 2026; Chen et al., 2026; Fu et al., 2026; Liu et al., 2026a; Kang et al., 2026). LL3M is especially relevant to our formulation because it uses multiple language-model agents to write, debug, and refine Blender Python scripts for 3D modeling (Lu et al., 2025). LEGO-Anything differs in the task and evaluation target: it evaluates end-to-end image-conditioned scene reconstruction under hidden scene ground truth, rather than focusing on language-driven asset generation and editing.

A.5 3D Construction Benchmarks

Existing benchmarks study the generation of executable 3D content through code. 3DCodeBench (Gao et al., 2026) evaluates agentic procedural modeling, while WorldCoder-Bench (Lu et al., 2026b) and P3D-Bench (Yang et al., 2026) examine code-based generation of structured, parametric, or physically grounded 3D content. These benchmarks establish code generation as a practical interface between foundation models and graphics environments. Buildee provides a Blender-based simulation framework for scene exploration, reconstruction, and understanding with simulator-grounded geometry and semantics (Boittiaux and Lepetit, 2025). It is complementary to LEGO-Bench: Buildee emphasizes interactive exploration and simulator tasks, whereas LEGO-Bench fixes a single RGB observation, hides evaluator geometry and semantics, and evaluates final executable scene artifacts across diverse scene families and controlled difficulty levels.

Compared with prior executable-3D benchmarks, LEGO-Bench is *image-conditioned*, *scene-level*, and *end-to-end*: the agent must infer a complete executable scene program from visual evidence rather than generate code from a textual or parametric specification.

B Supplementary Analyses

This section collects supplementary analyses on the scope and limits of LEGO-Anything beyond the paper's main evaluations. We place the strongest benchmark-adjacent evidence first and retain more exploratory extensions only as secondary context.

B.1 Cost–Performance Pareto Frontier

Figure 10 summarizes the six coding-agent configurations with one point per model. The horizontal axis uses a standardized token-cost proxy reconstructed from logged uncached input, cached input, and output tokens for the retained source attempts; the vertical axis reports the corresponding benchmark-wide validity-gated Final score.

Finding 8: `GPT-5.6-luna` is the absolute cheapest point, but most of the quality–cost frontier is defined by the GPT-6 family: `GPT-6-luna` improves sharply over the lowest-cost regime, `GPT-6-sol` adds another quality tier at moderate cost, and `GPT-6-astra` reaches the highest score at substantially higher spend. `GPT-5.6-sol` and `GPT-5.6-terra` lie off the frontier.

B.2 Construction Trajectory Diagnostics

We analyze the first archived native, high-reasoning-effort runs of `GPT-6-astra`, `GPT-5.6-sol`, `GPT-5.6-terra`, and `GPT-5.6-luna`. Each cohort targets 198 Indoor/Outdoor tasks; the 10 Bird's-Eye tasks are excluded. An eligible checkpoint has a valid Blender scene, geometry export, and both Reconstruction and Appearance scores. Checkpoint quality is $Q = (R + A)/2$, computed offline and never exposed to the actor. Figure 11 summarizes matched common-task statistics for first-scene timing, trajectory regression, and final regret.

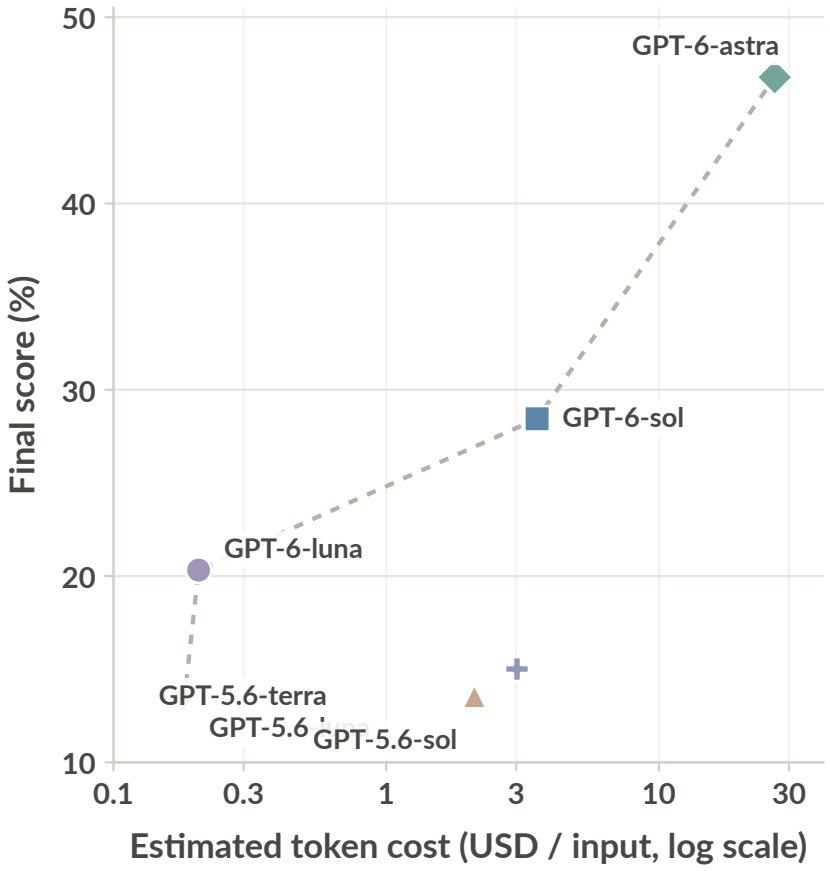


Figure 10 Cost–performance frontier for coding agents on LEGO-Bench. Each point is one coding-agent configuration, using full-benchmark mean Final score and a standardized token-cost proxy per input. The dashed line connects the non-dominated frontier.

> ***Finding 9:*** `GPT-6-astra` reaches its first evaluable scene later in absolute time, but earlier relative to its own runtime, and it shows the lowest regression rate and best-to-final regret. Construction remains non-monotonic for all four models, so final submissions should be interpreted as trajectory endpoints rather than guaranteed best states.

Severity-selected failure example. To make this non-monotonicity concrete, Figure 12 shows the common task with the largest mean best-to-last regret across the four models. We use this case only as a severity example. It shows that even the strongest model can still regress sharply late in the construction process.

B.3 Bird's-Eye Reconstruction Stress Test

The unpaired NYC Bird's-Eye split contains 10 overhead-view cases per complete run. The main result table pools these cases with the 90 ground-level Outdoor views, but we report them separately here only as a stricter large-scale reconstruction stress test rather than as a matched benchmark split. It is therefore excluded from the paired scene-complexity analysis. Table 6 reports the three GPT-6 agents under artifact validity, Reconstruction F@2%, Appearance, and the corresponding validity-gated Final score.

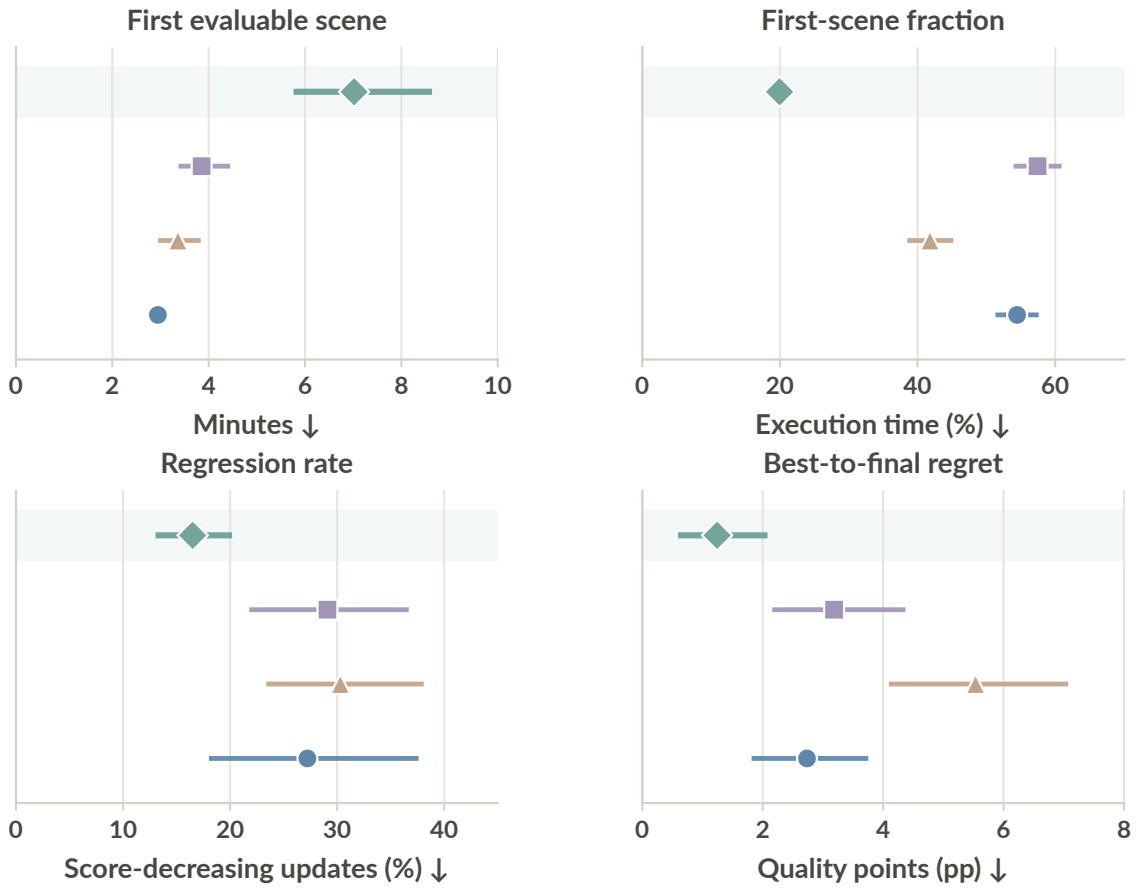


Figure 11 Trajectory diagnostics on the common-task intersection. GPT-6-astra reaches the first evaluable scene later in wall-clock time but earlier relative to its own runtime, and it has the lowest regression rate and best-to-final regret. Points and bars denote means and scene-family bootstrap 95% intervals.

Finding 10: GPT-6-astra is the strongest agent on the Bird's-Eye stress test, and the GPT-6 family substantially outperforms the GPT-5.6 family. However, all models still show a clear gap between artifact delivery and much weaker strict-geometry reconstruction under F@2%.

B.4 Pilot Scene-Level Articulation Evaluation on LEGO-Bench

Motivation. Interactive 3D scenes must capture not only geometry and appearance, but also which objects can move and how they move (Zhou et al., 2026b; Xiang et al., 2020). Prior work such as Articulate-Anything and ArtiCraft (Le et al., 2025; Zhou et al., 2026a) focuses on individual objects. Scene-level evaluation is harder because it must identify articulatable instances, recover their movable structure, and avoid assigning false motion to static objects.

Canonical GT coverage. This pilot uses the canonical articulation registry described in Appendix D.4. LEGO-Bench provides one human-validated reference articulation for each articulated asset together with confirmed-static controls; Table 7 summarizes the resulting coverage of the canonical manifest.

Pilot diagnostic and result. As an initial scene-level diagnostic, each eligible canonical joint requirement is classified as *object missing*, *part missing*, *joint missing*, *wrong joint type*, or *correct*. For one articulated object,

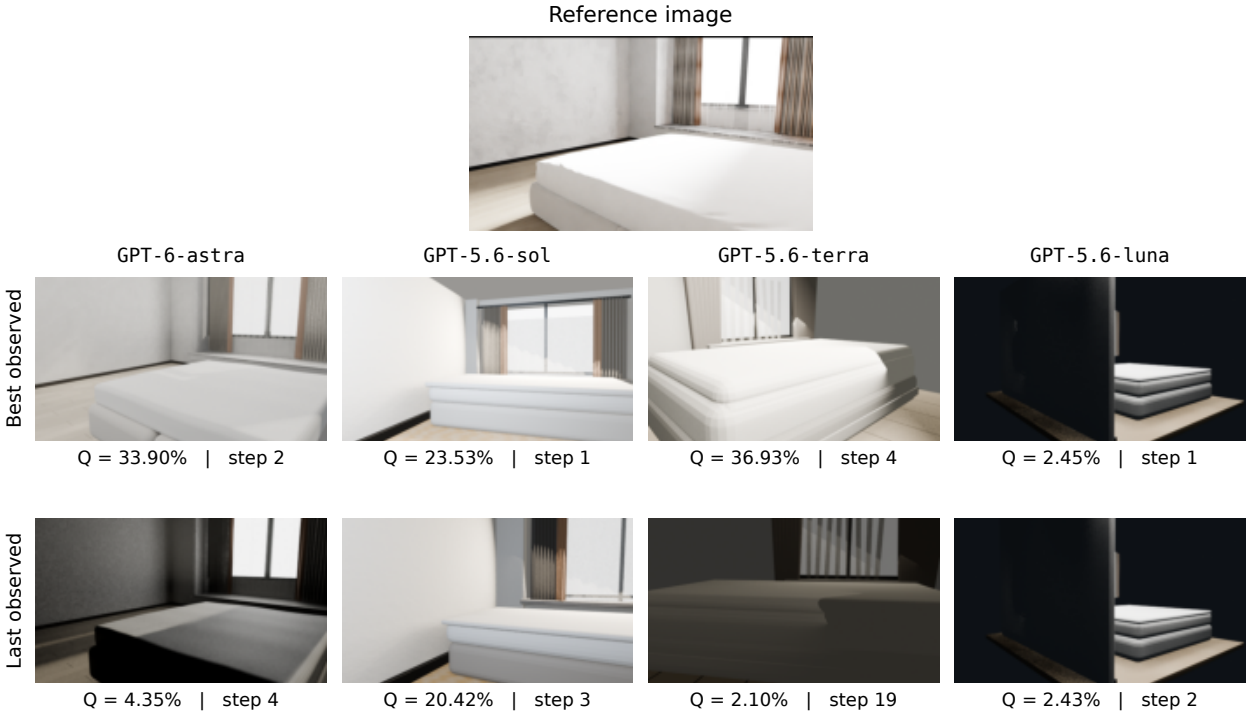


Figure 12 A severe best-to-last regression example. Each model is shown on the same House bedroom task at its best and last observed checkpoints. Even the strongest model can regress late in the trajectory. Displayed Q is the ungated average of Reconstruction and Appearance.

Table 6 Performance (%) on the Bird's-Eye reconstruction stress split. Higher is better for every metric.

Model + Harness	V ↑	R ↑	A ↑	S ↑
GPT-5.6-sol + Codex	93.3 (5.8)	11.2 (7.9)	24.3 (3.3)	16.4 (3.6)
GPT-5.6-terra + Codex	86.7 (5.8)	0.9 (1.3)	23.0 (4.2)	10.7 (2.8)
GPT-5.6-luna + Codex	80.0 (10.0)	7.6 (6.0)	21.1 (2.9)	11.4 (2.8)
GPT-6-astra + Codex	**100.0** (0.0)	**24.0** (5.0)	**46.6** (0.3)	**35.3** (2.5)
GPT-6-sol + Codex	100.0 (0.0)	17.8 (2.6)	40.0 (0.8)	28.9 (1.0)
GPT-6-luna + Codex	100.0 (0.0)	6.7 (4.7)	34.4 (0.7)	20.0 (2.7)

the provisional score is

$$S_{\text{Art}}^{\text{pilot}} = \frac{2C}{2C + O + P + J + 2W + E}, \tag{6}$$

where C, O, P, J, W, and E denote correct requirements, missing objects, missing parts, missing joints, wrong joint types, and extra joints, respectively. Scene scores macro-average articulated target objects and predictions that place movable joints on matched human-confirmed static objects receive zero.

We reevaluate one frozen Office submission without changing its predicted Blender scene. Under the earlier partial GT adapter, only two articulated targets were scoreable and 12 were unavailable. The complete canonical manifest makes all 14 articulated targets scoreable, retains 12 human-confirmed static targets as false-motion controls, and reduces the unavailable count to zero. The submission still obtains $S_{\text{Art}}^{\text{pilot}} = 0$, so the zero reflects the prediction rather than missing articulation GT.

Finding 11: Under the complete canonical articulation manifest, the pilot Office submission still obtains $S_{\text{Art}}^{\text{pilot}} = 0$. A more concrete scenario for tenable scene-level articulation needs further exploration.

C Demos on LEGO-Bench

Figure 13 shows qualitative examples from the same `run_01` evaluation split used in the main LEGO-Bench experiments. Each row uses one reference image on the left and the evaluator-rendered final artifacts from six coding-agent configurations on the right.

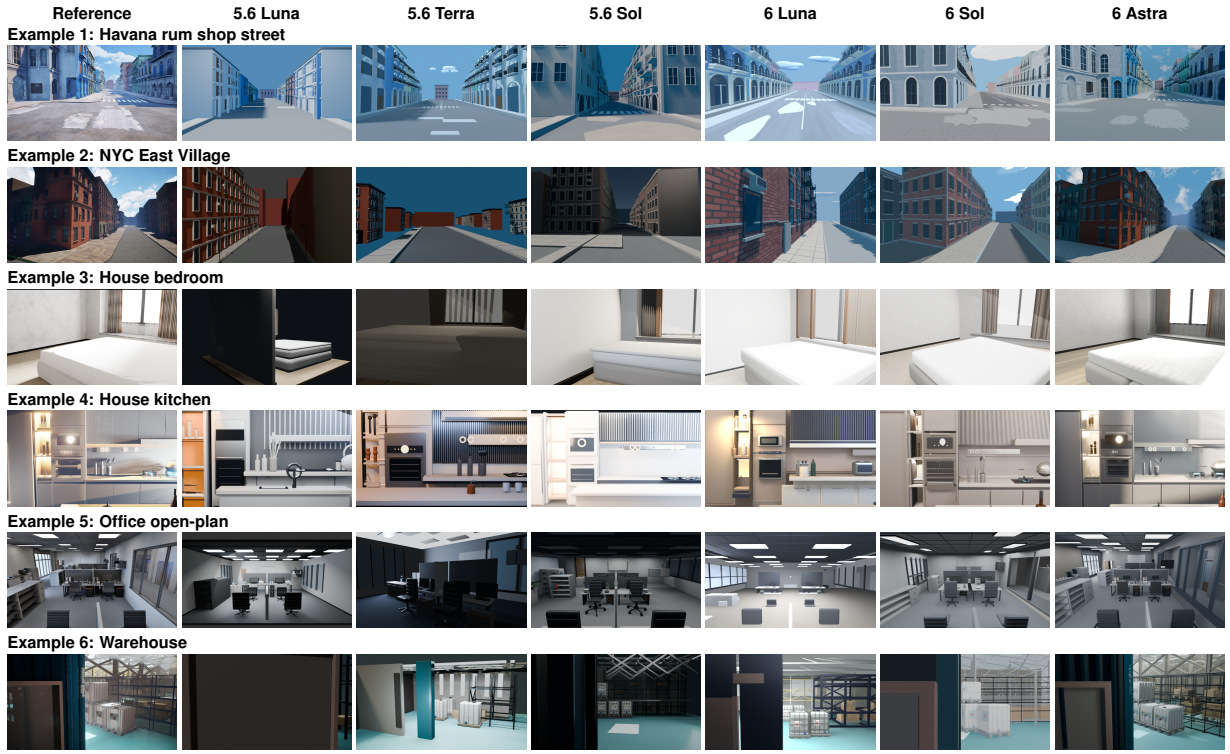


Figure 13 Qualitative LEGO-Bench examples. Columns show the reference image followed by final evaluator rerenders from `GPT-5.6-luna`, `GPT-5.6-terra`, `GPT-5.6-sol`, `GPT-6-luna`, `GPT-6-sol`, and `GPT-6-astra`.

D LEGO-Bench Construction and Statistics

D.1 Scene Construction and Capture

LychSim provides the scene-construction, simulation, and ground-truth capture backend for LEGO-Bench. We use it to assemble scene geometry, object layout, support relations, lighting, and materials, then render benchmark inputs and record the hidden annotations used for evaluation. Candidate scenes are included only after collision and stability checks and human review of the resulting scene composition and capture quality.

Task inputs and ground truth. Each task exposes only the public input needed for end-to-end reconstruction, including the reference image, basic camera metadata, and the prediction schema. The geometric and correspondence-level ground truth used for evaluation remains private to the benchmark.

Evaluator coordinate convention. For evaluation, simulator outputs are converted from LychSim's left-handed centimetre coordinates to the evaluator's right-handed metric coordinates with $+Z$ up:

$$\mathbf{p}_B = \mathbf{D}\mathbf{p}_U/100, \qquad \mathbf{R}_B = \mathbf{D}\mathbf{R}_U\mathbf{D}, \qquad \mathbf{D} = \mathrm{diag}(1, -1, 1). \tag{7}$$

Subscripts U and B denote the simulator and evaluation coordinate systems, respectively.

D.2 Controlled Scene Complexity

LEGO-Bench controls difficulty within matched scene families rather than by a global object-count threshold. For each paired family, object membership is nested, with $\mathcal{O}_E \subset \mathcal{O}_M \subset \mathcal{O}_H$, so higher tiers add visible scene content while keeping shared content fixed. Across the three tiers, architecture, lighting, materials, camera, and the asset identity, transform, scale, and bounding box of every shared object are held constant. Every retained object must preserve support consistency and be visible in at least one reference view; families that violate these conditions are rejected. Difficulty is therefore defined by controlled additions within a scene family rather than by raw object count alone. The Aerial XHard split has no matched family and is excluded from the paired complexity analysis.

D.3 Dataset Statistics

Figure 14 shows representative benchmark inputs. Figures 15 and 16 report scene inventory and object-size distributions.

D.4 Articulated-Asset Annotation

Annotated assets. The frozen benchmark registry contains 443 canonical assets, each of which received a human articulation judgment. The archived review log additionally contains 597 historical records, including superseded asset identifiers kept for provenance. We map these records back to the canonical registry before computing statistics, so all counts below are reported over canonical assets rather than raw historical entries. This is the same 443-asset registry summarized in Table 2. Execution records for the planned `GPT-5.6-terra` proposal workflow are tracked separately in the source manifest and do not affect the reported benchmark statistics.

Proposal and verification workflow. Annotation is human-verified throughout. For each candidate asset, annotators inspect the reference observation together with the available source geometry. `P3-SAM` (Ma et al., 2025) initializes the part segmentation, while `X-Part` (Yan et al., 2025) provides candidate part decompositions and articulation structures. Before joint inference, an object-first stage groups mesh components into logical rigid parts and labels each part as fixed, movable, or uncertain. Candidate joints then specify parent–child relations, joint type, axis, origin, motion limits, and rest state. Deterministic checks validate the URDF structure, mesh references, and executable joint motion. A reviewer then compares the complete object, part decomposition, candidate structure, and rendered motion evidence. Acceptance requires complete coverage of the object's functional movable components, correct rigid-part grouping and mechanism geometry, and

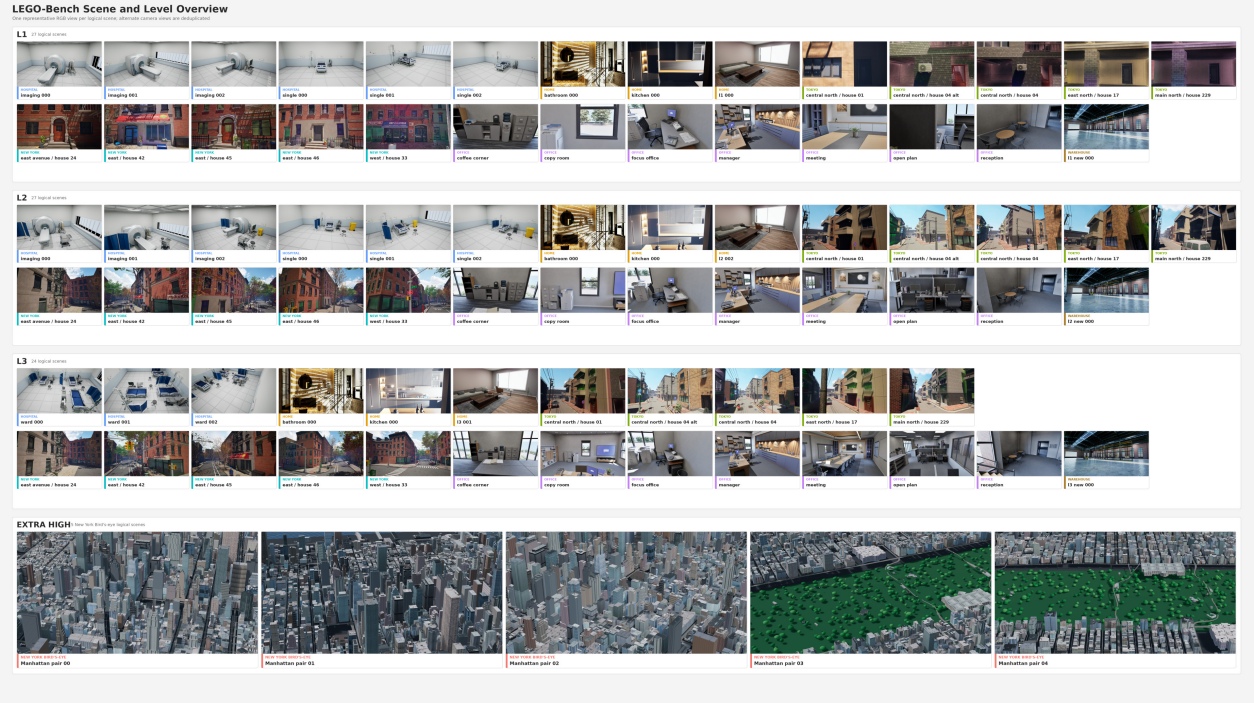


Figure 14 Representative LEGO-Bench inputs. LEGO-Bench includes indoor, outdoor, and Bird's-Eye reconstruction settings with controlled scene complexity.

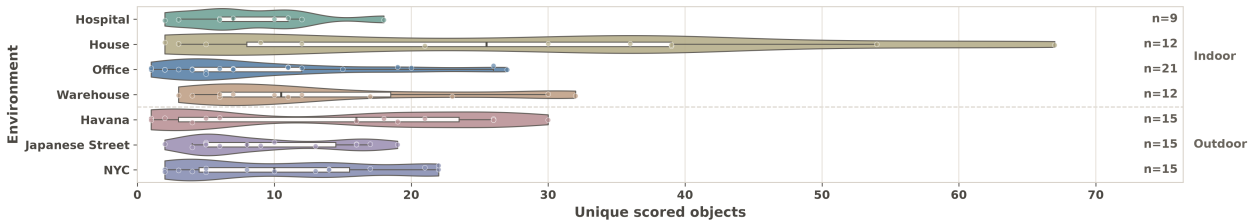


Figure 15 Scored-object count per logical scene. Counts increase within paired families, but global count is not used as the formal difficulty definition.

individually observable and executable joint motion. Reviewers may mark an object as static, validate its articulation, or assign repair labels such as missing joint, incorrect grouping, joint type, axis, pivot, or motion range. Before acceptance, each motion is executed in the simulator and visually checked, including the asset's return to its rest state. All automatically generated outputs remain proposals and are never promoted directly to benchmark ground truth without human review.

Annotation results. Figure 17 provides a Sankey-style view of the round-by-round annotation flow, showing how assets move from initial review to static acceptance, articulated acceptance, or regeneration as evidence is refined across rounds. After 15 human annotation rounds, all 443 canonical assets are resolved: 368 (83.1%) are confirmed static and 75 (16.9%) have validated articulation, with no unspecified or unresolved assets remaining. Each accepted annotation is linked to the reviewed asset revision and its motion-validation evidence. Only human-reviewed and execution-verified annotations enter Articulation evaluation, using the matching and aggregation rules in Appendix E.

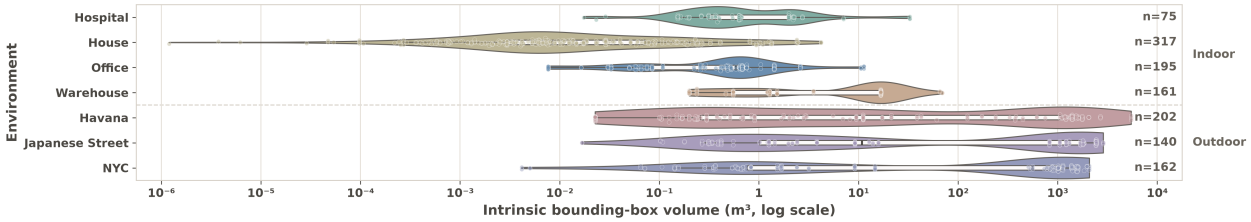


Figure 16 Distribution of scored-object size. Object scale spans small manipulable items through architectural and city-scale structures.

Table 7 Coverage of the canonical articulation GT manifest. Joint requirements are parsed from the exact human-accepted candidate URDFs.

Manifest statistic	Count
Canonical assets	443
Confirmed static	368
Validated articulated	75
Joint requirements	327
Continuous	75
Revolute	55
Prismatic	197
GT unavailable	0

E Metric Implementation and Validation

The primary comparison uses Validity, Reconstruction, Appearance, and their validity-gated Final score. Layout and articulation are auxiliary protocols and are documented separately after the headline metrics and validation plan.

E.1 Artifact Checks and Reported Validity

The raw artifact-validity check measures deliverability rather than reconstruction quality. Artifacts pass this check iff the submission contains a reloadable non-empty `scene.blend` with at least one mesh and an active camera, a non-empty `scene.glb`, and a decodable non-degenerate render. This check is determined from the submitted artifacts themselves rather than from process status: a timeout or interrupted run may still leave a valid submission, while a clean process exit does not by itself establish validity. Artifact-invalid attempts remain in the denominator and receive zero validity-gated Final score. For the main table, reported Validity additionally requires completed headline evaluation: an unresolved verifier error receives zero penalties for all four columns. Historical diagnostic analyses retain their stated artifact-validity and missingness conventions. This distinction is important in agent evaluation more broadly: environment- construction and agent-benchmark work increasingly treats usable artifact delivery as a first-class outcome, rather than assuming that nominal task completion or process exit is sufficient evidence of success (Li et al., 2026b; Ye et al., 2026).

E.2 Reconstruction Alignment and Scoring

The headline Reconstruction metric performs no registration. Submitted geometry is evaluated directly in the camera frame induced by its active camera, and the evaluator does not fit or apply a global or per-object transform. In particular, it does not absorb errors through fitted translation, rotation, scale, reflection, anisotropic scaling, or a Layout transform. The evaluator rasterizes the submitted triangle geometry to obtain submitted visible surfaces, back-projects private depth to obtain reference-visible surfaces, and uses private instance masks to select the scored surface scopes.

Point-to-instance assignment. The implementation assigns surfaces to object scopes in the reference image plane, not by matching predicted objects to ground-truth objects. Each scored reference instance has a private

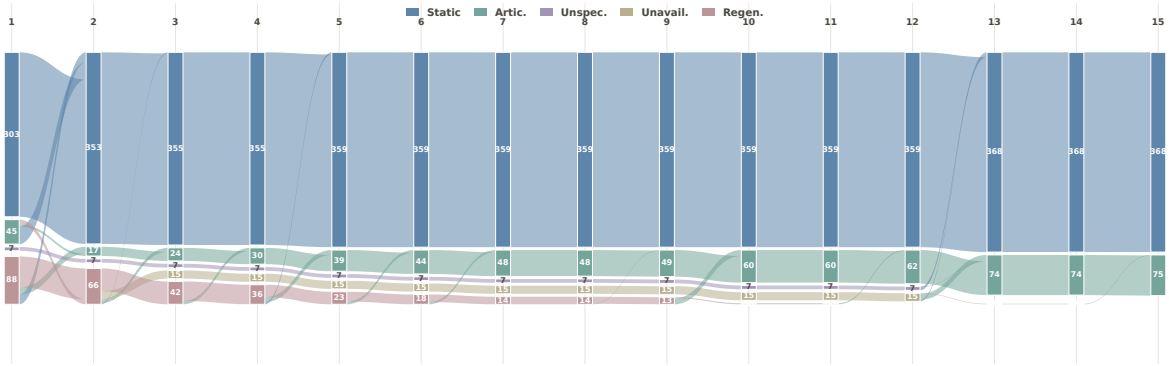


Figure 17 Human-in-the-loop articulation annotation process. The figure traces how the 443 canonical assets move across review states over 15 human annotation rounds, from initial review to static acceptance, articulated acceptance, or regeneration. The second round deliberately re-examines both first-pass articulation positives and unresolved assets, so the articulated count need not increase monotonically. After 15 rounds, all assets are resolved as 368 static and 75 articulated.

RGB segmentation color, which defines a binary mask M_o. The reference set G_o is obtained by back-projecting private depth samples whose pixels lie in M_o, after invalid/far-depth filtering and reference-camera visibility. The submitted set is obtained by rasterizing the submitted triangles from the fixed camera frame, retaining the visible submitted surface, projecting each visible submitted point into the same reference image plane, and keeping it in P_o iff its rounded projected pixel lies in M_o.

The evaluator therefore does not use predicted object names, categories, mesh grouping, or a Hungarian/IoU matching step to establish object correspondence. A camera, scale, boundary, occlusion, duplicate-object, or category error is penalized through where the submitted surface projects: it may be assigned to the wrong private instance mask, fall outside all scored object masks, or fail the geometric nearest-neighbor test below. Visible submitted geometry outside the scored object masks is excluded from the headline per-object macro score, while companion object-union/visible-scene diagnostics and the evaluator-rendered Appearance score expose broader-scope errors.

Nearest-neighbor tests. For each scored instance o, let G_o be the reference-visible point set and P_o be the predicted visible point set assigned by the rule above. Recall is computed in the reference-to-prediction direction:

$$\mathrm{Rec}_o = \frac{1}{|G_o|} \sum_{g \in G_o} \mathbf{1}\left[\min_{p \in P_o} \|g - p\|_2 \le \tau(g)\right],$$

with zero recall when G_o is non-empty and P_o is empty. Precision uses the opposite nearest-neighbor direction. For each $p \in P_o$, define $g^\star(p) = \arg\min_{g \in G_o} \|p - g\|_2$; the reference-dependent tolerance is then evaluated at this nearest reference point:

$$\mathrm{Prec}_o = \frac{1}{|P_o|} \sum_{p \in P_o} \mathbf{1}[\|p - g^\star(p)\|_2 \le \tau(g^\star(p))]\,.$$

If P_o is empty, precision is defined as zero for the F1 denominator. These rules make unmatched visible predicted geometry inside a scored mask count as false-positive surface mass for that instance, and missing reference-visible surface count as false-negative mass.

For each eligible object, precision and recall use a pointwise tolerance $\tau(g) = 0.05z(g)$, where $z(g)$ is the reference point's positive forward depth in the reference-camera frame. There is no fixed metre floor or cap in the headline tolerance. The object score is the harmonic mean of precision and recall, and the trial score is the equal-weight macro average over eligible scored objects. Objects with no target points after sampling are

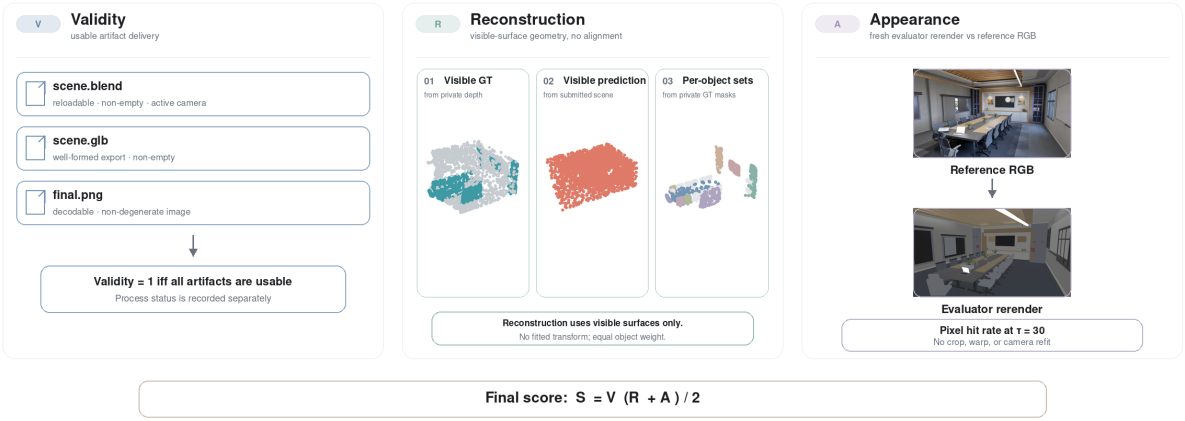


Figure 18 Overview of the LEGO-Bench metrics. Validity checks usable scene artifacts; the main table also treats unresolved headline-evaluation failures as invalid. Reconstruction is illustrated with step-wise point-cloud states from the evaluator: visible GT surfaces from private depth, visible prediction surfaces from the submitted scene, and per-object sets induced by private masks. Appearance compares a fresh evaluator rerender against the reference image without geometric warping or camera refitting.

marked ineligible rather than included in the macro average; objects with target points but no submitted points receive zero precision, recall, and F1. The whole-scene scoring representation is deterministically capped at 100,000 points before the object masks are applied. Missing visible surfaces therefore reduce recall, extra visible geometry inside a scored mask reduces precision, and camera or scale errors are penalized directly rather than corrected after the fact.

The evaluator also reports stricter 2% and relaxed 10% depth-relative variants, fixed-distance and globally aligned diagnostics, normalized symmetric Chamfer-L_1, and visible-surface voxel IoU. These are auxiliary diagnostics only; the headline metric remains the no-alignment per-object F@5% score.

Auxiliary reconstruction diagnostics. The strict and relaxed depth-relative variants keep the same visible-surface point sets and per-object macro aggregation, but replace the headline tolerance by

$$\tau_\alpha(g) = \alpha\, z(g), \qquad \alpha \in \{0.02, 0.10\}, \tag{8}$$

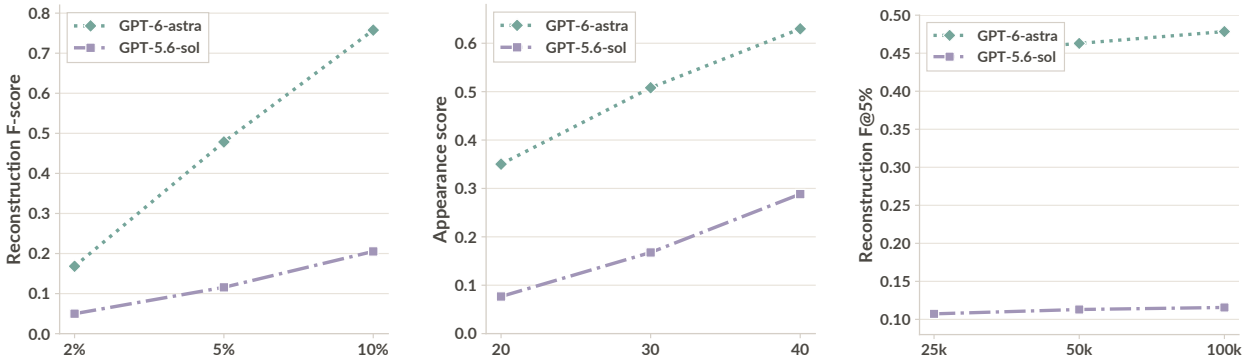
yielding F@2% and F@10%, respectively. Historical fixed-distance diagnostics instead use a constant threshold $\tau_\delta(g) = \delta$ with $\delta \in \{0.05, 0.10\}$ m.

E.3 Evaluator-Rendered Appearance

Appearance uses a fresh render of the submitted Blender scene. The evaluator preserves its active camera, geometry, materials, lights, and color settings, while fixing the rendering engine, output resolution, and full-frame output. The reference is resized to that resolution without fitting a spatial warp, crop, camera transform, or photometric correction. For 8-bit sRGB images, a pixel is correct only if its largest absolute channel difference is at most 30. Appearance is the fraction of correct pixels, reported as a percentage. The submitted `final.png` is checked for artifact delivery but does not substitute for this evaluator render.

This measure tests agreement with the reference rendering under a fixed protocol. It is sensitive to lighting, materials, camera error, and geometry, and is not a perceptual similarity model or an independent measure of geometric correctness. Reconstruction and Appearance are therefore reported separately as well as through their validity-gated average.

Figure 19 reports a small-scale sensitivity check on a fixed six-task subset shared by `GPT-6-astra` and `GPT-5.6-sol`. Varying reconstruction thresholds, appearance tolerances, and reconstruction point budgets changes absolute values as expected, but does not change the model ordering on this subset.



(a) Reconstruction threshold. **(b)** Appearance tolerance. **(c)** Point-budget sensitivity.

Figure 19 Small-scale metric sensitivity on a fixed shared subset. Left: no-alignment per-object Reconstruction under F@2%, F@5%, and F@10%. Middle: evaluator-rendered Appearance under RGB hit tolerances 20, 30, and 40. Right: Reconstruction F@5% under 25k, 50k, and 100k sampled-point budgets. Across all three perturbations, `GPT-6-astra` remains ahead of `GPT-5.6-sol`; the conclusions are therefore not driven by a single threshold or point-sampling choice in this subset analysis.

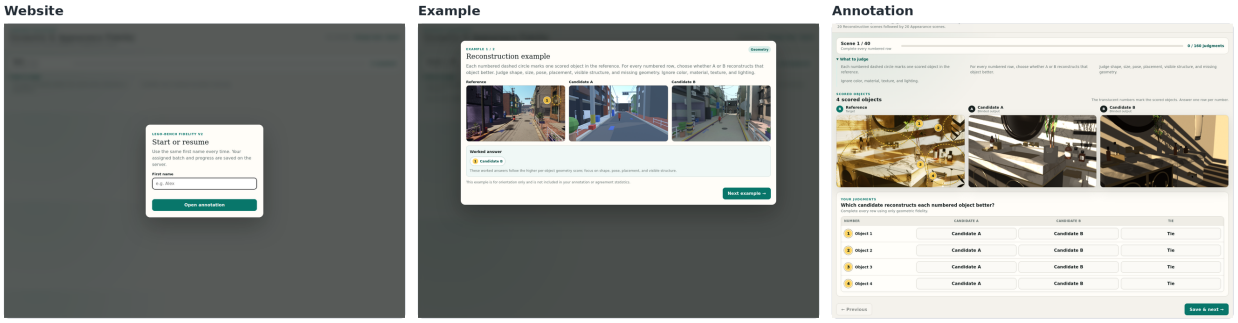


Figure 20 Human study interface used for metric validation. Left: annotator login. Middle: a worked example shown before annotation. Right: the main annotation page, where the annotator compares two blinded full-scene reconstructions against a shared reference and records one A/B/Tie judgment for each marked object or region.

E.4 Aggregation and Failure Accounting

The main table averages over all attempted benchmark views within each declared Indoor or Outdoor split. For completed evaluations, $S_i = V_i(R_i + A_i)/2$; artifact-invalid trials receive $S_i = 0$. In the main table, submissions with usable artifacts but unresolved headline-evaluation failures are also assigned $V_i = R_i = A_i = S_i = 0$ and remain in the denominator as explicit failure penalties rather than measured scores. Conditional diagnostic analyses may use restricted evaluated subsets, but must report both eligible and evaluated counts.

E.5 Human-Metric Alignment

To assess whether the headline metrics align with human judgment, we conduct a blinded A/B/Tie study on fixed candidate pairs. Each task presents one natural reference image and two full-scene candidate reconstructions. Reconstruction tasks ask annotators to compare four numbered objects using shape, size, pose, placement, visible structure, and missing geometry, while disregarding color, material, texture, and lighting. Appearance tasks ask annotators to compare four numbered circular regions using only local visual similarity. Each annotator first completes one worked example for each task type and then annotates 40 assigned scenes from controlled-overlap batches: 20 Reconstruction and 20 Appearance.

We report results over six completed annotators in total 240 samples and 960 local area on those samples.

Our primary inter-annotator measure is *compatibility agreement*, which counts `Tie-A`, `Tie-B`, and `Tie-Tie` as agreement and only a direct `A-B` conflict as disagreement. Compatibility agreement is 86.6% overall, with 88.9% on Reconstruction and 84.3% on Appearance.

For human-metric comparison, human labels are aggregated per scored target, and an equal number of A and B votes is treated as `Both OK`. On the metric side, we likewise assign `Both OK` whenever the absolute score gap between the two candidates falls below a fixed uncertainty threshold, $|s_A - s_B| < \varepsilon$, with $\varepsilon = 0.05$. Under this rule, human-metric compatibility is 83.7%, indicating that the headline metrics track human preferences in most cases.

E.6 Articulation Protocol

Articulation evaluation asks whether a predicted scene recovers the benchmark's canonical articulated structure for each matched object, rather than whether it admits any visually plausible mechanism. Because global scene pose is not part of the functional target, movable parts are first matched in an object-centric frame. The articulation score then combines three factors: movable-part structure F1, matched-part surface quality, and a quality-weighted joint F1 that evaluates parent–child structure, joint type, motion trajectory, and observed state:

$$S_{\text{Art}} = \text{HMean}(S_{\text{struct}}, S_{\text{geom}}, S_{\text{kin}}). \tag{9}$$

Unmatched predicted or ground-truth parts contribute false positives or false negatives, and assigning motion to a human-confirmed static object is counted as an explicit error. We additionally report Worst-Mover, movable recall, false-move rate, axis/state error, and complete-object success as diagnostics.

F Experimental Setup and Supplementary Results

F.1 Common Runtime and Artifacts

All formal experiments run in isolated Harbor tasks with Blender 5.0.1 and Blender-MCP. For each campaign, we freeze the agent, harness, prompt, evaluator, timeout policy, and container image. The actor receives a single public RGB image and no private depth, segmentation, or metric feedback. The main LEGO-Bench coding-agent campaigns use three independent runs per model configuration; main tables report mean and standard-deviation summaries over these repeated runs.

All coding-agent campaigns use the same base user prompt shown in Figure 21.

Vanilla user prompt

> You are given a single RGB reference image.
> Reconstruct the depicted scene in Blender using the available Blender MCP tools. Create a valid, editable 3D scene that represents the visible environment and major objects in the reference image.
> The submission must contain actual 3D scene geometry rather than a rendered-image approximation. Major visible objects should exist as separate scene objects where appropriate.
> Save the final Blender scene and export all required submission artifacts before finishing.

Figure 21 The vanilla user prompt for LEGO-Bench.

Evaluation is performed on the submitted artifacts, including `scene.blend`, `scene.glb`, and `final.png` when available. Failed or incomplete attempts remain in scope. Main-table aggregation follows Appendix E.4; conditional diagnostic subsets are reported separately.

F.2 Method Configurations and Information Access

The main table compares end-to-end systems: general-purpose coding agents, task-specific LLM systems, and single-image scene-construction baselines. These systems need not share the same tools, native outputs, or recovery procedures. The shared component is the evaluation endpoint: each method is scored by the same

frozen evaluator on each supported task after any declared artifact conversion. This equalizes scoring, but not inference budget or information access.

For the non-Codex rows, we preserve each method's native generation pipeline and report only the tasks it supports. VIGA is evaluated in its procedural-only setting with external asset generation disabled. SceneConductor and REST3D are evaluated only on Indoor tasks in our setup, while SceneGen and Gen3DSR are evaluated on both Indoor and Outdoor splits. 3D-RE-GEN is also Indoor-only; because it outputs calibrated PLY geometry rather than a Blender scene, we convert its outputs to Blender using public camera intrinsics, a fixed camera convention, and fixed area lighting before Appearance evaluation.

F.3 VLM Judge Setup and Reference Results

The VLM judge is read-only: it sees only the reference image and candidate renders, not system identity, transcripts, private depth, segmentation, or LEGO-Bench scores. We evaluate `GPT-6-astra`, `GPT-6-sol`, `GPT-6-luna`, and the three `GPT-5.6` models as both builders and judges. The comparison contains 360 checkpoint pairs, 60 from each builder, and 2,160 primary judgments from the full 6×6 matrix.

Each builder contributes one pair from each of 60 distinct, artifact-valid main-experiment trajectories from runs two and three. We select 20 pairs each with sequence gaps of 1–2, 3–5, and at least 6, balancing run, difficulty, and environment deterministically. Both checkpoint scenes and renders must exist, and their scene signatures and image hashes must differ. Pairs with unavailable deterministic endpoint metrics are replaced under the declared validity rule before judging, preserving builder and gap bucket. Selection does not compare score values or judge preferences.

Agreement is an exact match to the deterministic metric direction, using $\epsilon = 10^{-6}$ for ties. `both_bad` and terminal judge failures count as incorrect; all 60 pairs remain in each cell's denominator. There are 0 terminal failures among the 2,160 primary judgments. An additional 36 pairs are evaluated in reversed order by every judge, yielding 216 audit judgments. Order-swap consistency ranges from 69.4% to 100.0% for Reconstruction and 61.1% to 83.3% for Appearance.

Table 8 Builder–judge directional agreement (%). Each cell reports Reconstruction / Appearance agreement on 60 archived checkpoint pairs. Rows are builders; columns are judges. `both_bad` and failed judgments count as incorrect.

Builder / Judge	GPT-6-astra	GPT-6-sol	GPT-6-luna	GPT-5.6-sol	GPT-5.6-terra	GPT-5.6-luna
GPT-6-astra	61.7 / 76.7	51.7 / 78.3	50.0 / 83.3	58.3 / 86.7	51.7 / 76.7	53.3 / 78.3
GPT-6-sol	45.0 / 75.0	45.0 / 78.3	46.7 / 76.7	43.3 / 75.0	48.3 / 81.7	45.0 / 63.3
GPT-6-luna	51.7 / 71.7	56.7 / 76.7	43.3 / 66.7	46.7 / 71.7	41.7 / 70.0	41.7 / 65.0
GPT-5.6-sol	41.7 / 48.3	45.0 / 60.0	38.3 / 53.3	46.7 / 48.3	46.7 / 55.0	46.7 / 46.7
GPT-5.6-terra	45.0 / 58.3	40.0 / 66.7	36.7 / 56.7	40.0 / 70.0	43.3 / 65.0	43.3 / 55.0
GPT-5.6-luna	41.7 / 51.7	43.3 / 46.7	45.0 / 35.0	35.0 / 43.3	41.7 / 41.7	35.0 / 38.3

Judge Prompt

All judges use the same prompt template:
You are a blind evaluator of a rendered 3D reconstruction. Image 1 is the original reference. Image 2 is candidate A (left) and image 3 is candidate B (right). Judge only visible evidence in these images. Model identity, run identity, critique text, and benchmark metrics are unavailable.
Reconstruction: Which candidate better matches the reference geometry? Judge camera/viewpoint, object presence, layout, scale, shape, silhouette, and occlusion. Ignore color, material, texture, and lighting as much as possible.
Appearance: Which candidate better matches the reference appearance? Judge color, material, texture, lighting, shadows, and overall visual character. Do not reward geometry differences except where they prevent appearance judgment.
For each question choose exactly one of: `left`, `right`, `tie`, or `both_bad`. Return only the requested JSON object. Do not include an explanation, score, or confidence.

F.4 Reasoning-Effort Protocol

We study test-time scaling with the native Low, Medium, High, and XHigh reasoning settings. The sweep covers `GPT-5.6-luna`, `GPT-5.6-terra`, `GPT-5.6-sol`, `GPT-6-astra`, `GPT-6-sol`, and `GPT-6-luna` under the Codex harness on the 42-task Office subset, comprising seven room types, three complexity levels, and two paired views per room–level group. This produces a balanced $6 \times 4 \times 42$ design with one scored outcome per task cell. When infrastructure failures occur, we rerun the same model–effort–task configuration, but do not use retries for best-of selection. Within each model, task inputs, prompts, execution budgets, and scoring are held fixed across effort settings. All 1,008 outcomes are rescored with the same frozen evaluator and container image. Figure intervals are pointwise 95% percentile bootstrap intervals for the mean over the 42 task inputs, using 10,000 resamples; they do not measure variation across repeated runs.

The new `GPT-6-sol` and `GPT-6-luna` sweeps contribute 336 outcomes, all with completed evaluations. Their High results are generated specifically for this sweep, whose prompts disable the extra camera guidance used in the main-table campaigns. Thirteen Luna trials retain native nonzero-exit flags but yield artifacts that are successfully rescored; these outcomes remain in the fixed denominator. Terminal evaluation failures would receive zero scores rather than being dropped. Worker concurrency and shared-host disk contention varied during this campaign, so its elapsed times are not treated as a controlled measure of reasoning cost.

G LEGO-Plugin: Implementation and Controlled Evaluation

G.1 Runtime Boundary

LEGO-Plugin runs alongside the standard Blender MCP. Blender MCP remains the only general scene editor and the agent remains the planner; the plugin exposes only bounded operations for measurement, validation, edit transactions, and evidence retrieval. It adds no unrestricted code-execution path and never modifies the scene without an explicit agent call. All services communicate through fixed JSON schemas, and large payloads are stored by content hash.

G.2 Shared Reference Evidence

Both Enhanced Initialization and Grounded Refinement draw on evidence extracted from the reference image only: camera and scene geometry from `VGGT` (Wang et al., 2025), and optionally object masks from `SAM 3` (Carion et al., 2026) and relative depth from `Depth Anything V2` (Yang et al., 2024). Providers return image regions and depth maps; the agent, not the provider, decides which Blender entity each region corresponds to.

G.3 Enhanced Initialization

The initialization service estimates a gravity-aligned, Z-up Manhattan frame from `VGGT`, constructs a landscape and installs the room and camera together after validation. It then produces a construction plan recording, for each visible object, its normalized centroid, projected extent, relative and ordinal depth, and observability flags. The plan deliberately avoids metric depth or absolute object size, which cannot be reliably recovered from a single monocular image. Outputs include the `VGGT` predictions, the recovered room and camera, a rendered room shell, and an initialization report.

G.4 Version Control

Each edit is wrapped in a transaction that declares editable entities, protected entities, allowed types of change, and acceptance requirements. Before the edit, the plugin snapshots the relevant scene state, including transforms, meshes, materials, hierarchy, visibility, active camera, and render settings. After the Blender MCP edit, it rejects any change to protected or undeclared entities. A transaction commits only if it produces a non-empty allowed change and all hard requirements pass on the current scene ; otherwise it is rolled back by removing new entities and restoring the snapshot. Evaluation runs allow at most eight transactions and two rollbacks per case to prevent unbounded repair loops.

G.5 Grounded Refinement

Validation checks named objects or collections against explicit requirements and returns *pass*, *fail*, or *unknown*. Supported requirements cover existence, valid transforms and dimensions, support, containment, collision, facing direction, repeated layout, camera visibility, projected centroid and extent, reference depth, depth order, and rendered luminance. Failures of hard requirements, or missing evidence for them, block acceptance, while other failures are reported as warnings. Each report records the requirements, measurements, and residuals, and is tied to the exact scene state: any later scene change invalidates it. Each candidate receives at most two correction rounds before it must be committed or rolled back.

G.6 Execution Records

As supporting infrastructure rather than a separate module, every plugin call is logged with its request, response, Blender execution status, transaction state, and resulting scene signature, so that a failed backend call can never be recorded as a successful edit. Runtime hooks prevent the agent from finishing while validations are stale, transactions are unresolved, or renders are unreviewed. All camera renders are logged independently of the prompt. At the end of each run, the plugin exports `scene.blend`, `scene.glb`, and a 1920×1080 render without modifying scene geometry.

H LEGO-Anything: Natural-Image Task Protocols

H.1 Task Inputs and Frozen-Scene Readouts

The main paper evaluates `GPT-6-astra` without LEGO-Plugin on three fixed 100-image tracks: COCO val2017 detection, LVIS v1 validation instance segmentation, and ETH3D relative depth. The agent constructs a scene before evaluation; task predictions are read from the frozen artifact. These experiments test a common executable-scene interface across datasets, not whether a single scene has already been evaluated against all three annotation types on the same image.

The readout projects semantic/instance geometry into visible-object boxes and masks and renders camera-space depth. Ground-truth annotations and metric feedback belong to the verifier and must not be returned to the agent. The intended category vocabulary and task schema are public inputs; exact camera metadata and task-dependent prompt differences must be recorded.

H.2 Detection and Segmentation: Confidence and Missing Predictions

The archived direct baselines are `DINO-R50-4scale-24ep` for boxes and `SAM 3` for instance masks. The COCO comparison uses `pycocotools.cocoeval.COCOeval` with at most 100 detections per image; the LVIS comparison uses `lvis.LVISEval` with at most 300 detections and its federated-label evaluation.

LEGO-Anything's scene schema does not provide a calibrated confidence score. The AP export therefore assigns every predicted instance a confidence of 1.0 and uses a deterministic tie order based on image ID, category ID, and the mask/box payload. The main table reports this *equal-confidence AP*. It measures the resulting ranking under that declared policy, not calibrated detector confidence. Missing scene predictions are exported as no detections rather than dropping their images. Because LEGO-Anything emits deterministic scene-derived predictions without calibrated confidence scores, AP should be read as a compatibility diagnostic under a fixed ordering rather than as a calibrated ranking comparison against detectors or segmenters with learned confidence scores.

Both scene-based AP tracks have 94 scored trials out of 100 attempted images. All 100 images remain in the official subset AP evaluation. The separate score-free matched-IoU diagnostic has 93 paired quality-valid images: one image in each subset contains no scored ground-truth instances, and the remaining missing predictions further restrict paired coverage.

H.3 Relative Depth and Coverage

The direct depth baseline is DA3MONO-LARGE. Evaluation uses the frozen inverse-depth scale-and-shift alignment protocol, so this is a relative-depth comparison rather than a test of absolute metric scale. The main table's direct AbsRel is averaged over 100 images, whereas LEGO-Anything has 99 quality-valid depth outputs. A missing output cannot be assigned zero error for a lower-is-better metric.

On the same 99 valid images, the archived direct and scene-based AbsRel means are 0.07845 and 0.15540, respectively. This paired diagnostic addresses the coverage difference while retaining the separate 99/100 completion statement. Table 9 makes the distinct denominators explicit.

Table 9 Coverage of the natural-image comparisons. Official box/mask AP retains all 100 subset images, including missing scene predictions. Matched diagnostics use only jointly valid pairs.

Track	Attempted images	Scene outputs scored	Matched diagnostic pairs
COCO boxes	100	94	93
LVIS masks	100	94	93
ETH3D depth	100	99	99

The raw standard-evaluator metrics, equal-confidence exports, and paired comparison summaries are preserved with source hashes in the ancillary file `appendix_verified_evidence.json`.

I Evaluation Infrastructure and Reproducibility

This section summarizes the evaluation infrastructure used by the reported experiments. Adapter support alone does not imply that a corresponding model configuration appears in the paper; reported runs are identified by the frozen manifests in Appendix F.

Harbor extension. We build on Harbor (Harbor Framework Team, 2026) for task expansion, repetitions, logging, artifact collection, verifier execution, and aggregation. Our `lego_bench_harbor` package adds the dataset converter, isolated Blender runtime, agent adapters, verifier, and run manifest. Public inputs are mounted for the actor, while private geometry, masks, depth, and correspondences remain verifier-only.

Common runtime. Formal runs use isolated Docker/Compose environments with Blender 5.0.1, Xvfb, and one Blender MCP listener per trial. Every harness receives the same image, prompt, taxonomy, tool interface, limits, and artifact contract. The runtime collects the submitted scene artifacts, transcripts, and verifier reports, and applies the same prompt-independent finalization policy on timeouts.

Codex. `ContainerCodex` subclasses Harbor's Codex adapter (Bolin, 2026), registers the MCP tools in an isolated `CODEX_HOME`, and attaches the input image. Final campaigns set `model_provider=amazon-bedrock`; GPT-family identifiers therefore do not by themselves specify the serving path.

Hermes. Our wrapper uses Harbor's generic adapter and the upstream Nous Research Hermes Agent (Nous Research, 2026), pinned to revision `aa6f77596b`. `ContainerHermes` supports local-model diagnostics, while `ContainerBedrockHermes` uses the Bedrock Converse provider with the same MCP tools and session export. Only configurations with frozen run manifests are reported.