TACER: VERIFIED 3D TABLETOP SCENE GENERATION FOR SIMULATION AND INTERACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating 3D tabletop scenes from images or text supports robot learning, but visual and geometric plausibility alone do not establish simulation validity or interaction usability. Existing methods largely operate on static meshes, with limited support for functional structures. Evaluation also emphasizes static geometry and visual judgments, offering limited evidence of dynamic validity and task executability. To address these limitations, we present *TACER*, which connects asset construction, scene assembly, and verification-guided refinement through hierarchical executable programs. Asset-level programs construct geometry and physical representations, including articulation; scene-level programs organize placements and spatial relations. Geometric checks and simulation tests guide asset or layout revisions, which are re-executed and verified before acceptance. To independently assess scene capabilities, we introduce *TabletopAudit*, a **500**-case benchmark covering input fidelity, 3D scene quality, simulation validity, and robot task execution. TACER leads input fidelity on both input tracks. On Embodied-100, it achieves **71.67%** rigid-manipulation and **60.00%** articulated task success, compared with **20.00%** for TabletopGen and **5.00%** for TabletopGen + Articraft, respectively. To demonstrate scalability, we construct *TACER-11K*: **11,399** scenes across nine environments with physical representations, interactive assets, and available verification evidence. A downstream study demonstrates their utility for demonstration collection and task-specific policy learning. Together, these results support *TACER*'s effectiveness and establish *TabletopAudit* as a basis for assessing simulation readiness and interaction usability.

1 INTRODUCTION

Generating 3D tabletop scenes from images or texts is valuable for virtual environment creation and robot learning. Robotic applications require both simulation validity and interaction usability: the former demands valid physical representations and plausible dynamics, while the latter requires object functionality and spatial arrangements that support the intended operations. Constructing scenes that meet both requirements remains a central challenge.

Recent methods advance tabletop generation through asset retrieval and layout reasoning (MesaTask, Hao et al., 2025), physics-aware layout correction (STABLE, Luo et al., 2026; PhyScene3D, Chen et al., 2026), and image-based asset generation and simulation assembly (TabletopGen, Wang et al., 2025). However, these methods primarily operate on static meshes and their configurations, with limited support for internal functional structures and articulated mechanisms. Therefore, a gap remains between visually and geometrically plausible scenes and environments that simulate reliably and support the intended interactions. Existing evaluations provide limited evidence about this gap: principal metrics emphasize static collisions, geometric support, and scene-level visual judgments (Hao et al., 2025; Luo et al., 2026; Chen et al., 2026; Wang et al., 2025). Although some methods report dynamic settling or robot experiments (Wang et al., 2025; Chen et al., 2026), systematic verification spanning physical validity and task execution remains limited.

Closing this gap requires establishing conditions for valid simulation and intended interactions at both the asset and scene levels, with multiple forms of verification guiding repair. VIGA (Yin et al., 2026) demonstrates how a code–render–inspect loop supports program generation and revision through visual feedback, motivating executable code as a practical medium for construc-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

tion and repair. We introduce *TACER*, which connects asset construction, scene assembly, and verification-guided repair through a hierarchical executable program. Asset-level programs make objects physically and functionally executable, while scene-level programs organize them into layouts that preserve support, clearance, and interaction access. Relation checks and simulation tests assess contact, stability, and articulation clearance, with shared object identities linking failures to construction operations. The system revises assets or layouts, replays affected steps, and renews downstream checks, accepting revisions only when verification improves without regressions.

To better assess simulation validity and interaction usability, we introduce *TabletopAudit*, an independent benchmark extending scene evaluation to dynamic simulation and robot task execution. It measures input fidelity and 3D scene quality on **500** cases with paired Text and Image tracks, and simulation validity and task execution on Embodied-100. TACER leads input fidelity on both tracks and all reported image-track metrics. Under a shared execution protocol, it achieves **71.67%** rigid-manipulation task success versus **20.00%** for TabletopGen, and **60.00%** articulated task success versus **5.00%** for TabletopGen augmented with Articraft-generated assets. Ablations assess the contributions of the program representation, asset routing, verification, and refinement.

To assess scalability, we build *TACER-11K*, a dataset of **11,399** tabletop scenes across nine environments. Built for physical simulation, it includes physical representations, interactive assets, and available verification evidence, providing diverse environments for robot evaluation and interaction data collection. To assess utility, we conduct a downstream screwdriver-to-tray study, showing that generated workbench scenes support demonstration collection and task-specific policy learning. Interactive demo: `https://anonymousauthor282.github.io/cancer_demo/`

Our contributions are threefold:

- We introduce *TACER*, which connects asset construction and scene assembly through hierarchical executable programs, using verification of physical and interaction requirements to guide asset and layout repair for tabletop simulation and robotic interaction.
- We establish *TabletopAudit*, extending tabletop generation evaluation from input fidelity and static scene quality to dynamic simulation and robot task execution, providing an independent, layered assessment of simulation readiness and interaction usability.
- We construct *TACER-11K*, comprising **11,399** tabletop scenes across nine environments built for physical simulation, demonstrating the framework's scalability and validating the utility of generated scenes for demonstration collection and task-specific policy learning.

2 Related Work

Tabletop Scene Generation MesaTask (Hao et al., 2025) generates tabletop layouts through task-relevant object and relation reasoning. STABLE (Luo et al., 2026) and PhyScene3D (Chen et al., 2026) introduce physics-aware layout correction, while TabletopGen (Wang et al., 2025) combines image-based asset generation with interactive simulation. These methods primarily adjust the spatial configurations of object meshes, with limited support for constructing internal functional structures and articulated mechanisms or correcting asset-level physical defects. These limitations constrain the usability of generated scenes for the intended robot interactions, since layout correction alone cannot resolve missing or unsuitable asset structures. *TACER* incorporates both asset construction and scene assembly into verification and repair, while *TabletopAudit* separately assesses scene quality, simulation validity, and robot task execution.

Code-Native and Agentic 3D Scene Construction LayoutGPT (Feng et al., 2023), Holodeck (Yang et al., 2024b), and LayoutVLM (Sun et al., 2025) generate and refine structured layouts. I-Design (Çelen et al., 2024), LLPlace (Yang et al., 2024a), OptiScene (Yang et al., 2025b), and InstructScene (Lin & Mu, 2024) support personalized design, editing, preferences, or semantic-graph priors. Chat2Layout (Wang et al., 2024), AnyHome (Fu et al., 2024), and Imaginarium (Zhu et al., 2025) support conversational, open-vocabulary, or vision-guided construction. SceneWeaver (Yang et al., 2025a), SceneSmith (Pfaff et al., 2026), and SAGE (Xia et al., 2026) use tools, self-reflection, or simulator feedback. Code-as-Room (Yang et al., 2026) uses staged executable Blender programs. SceneCode (Wang et al., 2026) represents editable articulated scenes with executable programs, asset routing, and a persistent state registry. Joint verification of asset

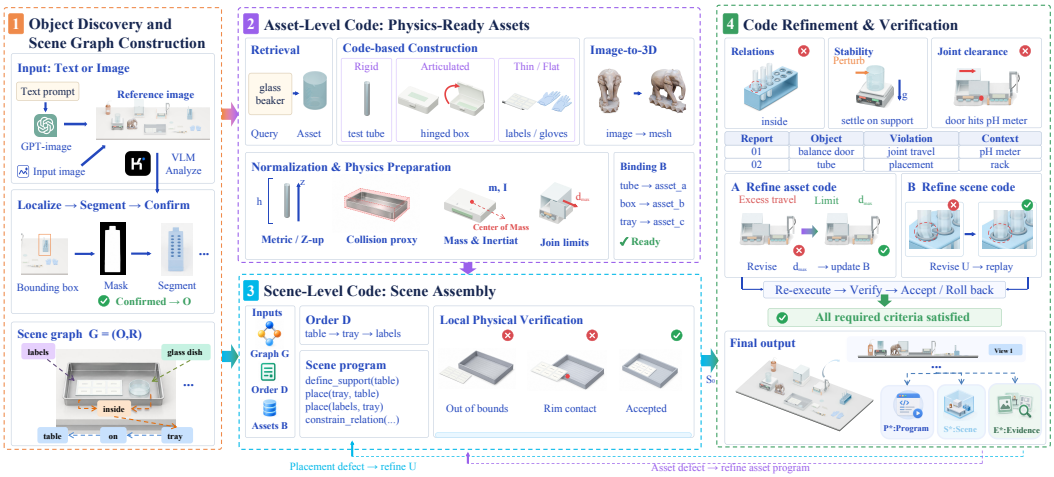

Figure 1: **TACER overview.** (1) Discover objects and relations; derive construction dependencies. (2) Build physics-ready assets. (3) Assemble assets with scene-level code in dependency order and check placements. (4) Verify relations, stability, and motion clearance; repair and replay the responsible code. Export the program, scene, and evidence only if all required checks pass.

structures and spatial conditions for tabletop interaction remains limited. *TACER* links physical and interaction checks to asset- and scene-level programs, using failures to guide repair and re-execution with renewed downstream verification.

3 Method

3.1 Problem Formulation and Overview

TACER formulates tabletop scene generation as a simulator-in-the-loop construction problem. Given an image or text description x, it uses the provided or text-generated image as reference. A verified run returns an executable program $P^\star$, the 3D scene $S^\star$, and verification evidence $E^\star$:

$$(P^\star, S^\star, E^\star) = \mathrm{TACER}(x), \qquad S^\star = \mathrm{Execute}(P^\star), \qquad \mathrm{Verify}(P^\star, S^\star, E^\star) = 1. \tag{1}$$

The superscript $\star$ denotes results that satisfy all required verification criteria. We define $P = (P^a, P^s)$, where $P^a = \{p_i^{\mathrm{asset}}\}_{o_i \in O}$ is the set of asset-level programs, p_i^{asset} specifies the construction of object o_i, and P^s assembles the assets into a scene.

A controller coordinates multi-stage generation, while a verifier evaluates intermediate outputs to guide acceptance, refinement, or rollback. As shown in Figure 1, TACER identifies objects and relations from the reference image, constructs physics-ready assets, and assembles them with scene-level code. Verification requires valid physical representations for all required objects, the intended spatial, support, and containment relations to hold, and the scene to remain physically stable; articulated objects additionally require valid joints and sufficient motion clearance. Failures are traced to the responsible asset or placement instruction, whose code is revised and re-executed. TACER exports the program, scene, and evidence only when all criteria are satisfied, i.e., $\mathrm{Verify} = 1$.

3.2 Object Discovery and Scene Graph Construction

An off-the-shelf vision-language model (VLM) proposes open-vocabulary objects from the reference image. After confirmation, TACER forms the object set O and infers relations R, obtaining the scene graph $G = (O, R)$ and dependency graph $D = \mathrm{Dep}(R)$. Relations in R describe support, containment, and relative position; D derives from support and containment so that supporting objects precede their dependents during assembly.

Each object is assigned a stable identifier. The VLM predicts its semantic category, articulation properties, likely supporting object, and coarse image region, then localizes it with a bounding box.

For small objects, the VLM performs localization within a crop defined by the supporting object's bounding box and the predicted coarse region. TACER uses the localized boxes to guide segmentation with SAM 3 (Carion et al., 2025) and associates the resulting masks with their objects. Candidates satisfying the confirmation criteria are marked as `confirmed` and included in O; unresolved candidates remain `pending` and are excluded from construction. Appendix A.1 provides details on object confirmation, relation inference, and dependency construction.

3.3 Asset-Level Code: Constructing Physics-Ready Assets

TACER uses three routes: asset retrieval, code-based construction, and image-to-3D generation. For each confirmed object $o_i \in O$, routing rules filter unsuitable routes, and the VLM ranks the rest. TACER attempts them in order until one succeeds, recording each route, outcome, and failure reason. The selected asset program p_i^{asset} uses object cues to generate a 3D asset:

$$a_i = \text{ExecuteAsset}(p_i^{\text{asset}}), \qquad b_i = \text{Bind}(o_i, a_i), \qquad B = \{b_i\}_{o_i \in O}. \tag{2}$$

Here, b_i binds object o_i to asset a_i and records its source and construction parameters. The bindings B are passed to the scene-level program P^s; assets are constructed independently without assigning their final scene poses.

Asset retrieval selects matching 3D assets from a preprocessed MesaTask-10K (Hao et al., 2025) collection, indexed by category, textual description, material, and dimensions. Code-based construction has three branches: rigid, articulated (e.g., hinged), and thin/flat. Rigid programs assemble fixed components, including internal structures such as tray dividers. Articulated programs use Articraft (Zhou et al., 2026) to generate Python programs specifying part geometry and joints, which are compiled into GLB and URDF representations. Thin/flat programs construct shallow meshes and extract surface textures from the reference image. Image-to-3D generation uses TRELLIS (Xiang et al., 2025) to generate textured assets from segmented object images when the geometry or appearance is difficult to reproduce procedurally.

Before assembly, assets use a metric Z-up coordinate system, canonical orientation, render geometry, collision geometry preserving required functional structures, and physical properties; articulated assets additionally require a valid URDF with joint definitions and motion limits. An asset is accepted only if its geometry, canonical pose, and required simulation representations are valid, i.e., $\text{VerifyAsset}(b_i) = 1$ for every $o_i \in O$. Placeholder geometry is rejected regardless of object size. Appendix A.2 details routing, physical preparation, and budgets.

3.4 Scene-Level Code: Assembling Assets into a Scene

Given the scene graph G, dependency graph D, and asset bindings B, TACER generates and executes the scene-level program $P^s = \text{GenerateSceneCode}(G, D, B)$ to obtain $S_0 = \text{ExecuteScene}(P^s)$. The program contains three instruction types: `define_support` specifies a root support and its surface; `place` binds an object to its asset and specifies its supporting object, target position, and orientation; and `constrain_relation` encodes R as layout constraints. TACER executes these instructions in D order, allowing the controller to revise and re-execute the responsible instruction when verification fails.

TACER initializes object positions from image-space bounding boxes. For object o_i and its supporting object s, let (c_i^x, c_i^y) and (c_s^x, c_s^y) denote their bounding-box centers, and let w_s, h_s denote the support-box width and height in pixels. The initial position is

$$x_i = \frac{c_i^x - c_s^x}{w_s} W_s, \qquad y_i = \frac{c_i^y - c_s^y}{h_s} D_s, \qquad z_i = z_s^{\text{top}} + \frac{H_i}{2}. \tag{3}$$

where W_s, D_s are the physical width and depth of the supporting object, z_s^{top} is its surface height, and H_i is the object's vertical bounding-box extent. The normalized image offsets provide approximate physical offsets from the support center, while z_i places the object on its surface; this is an initialization rather than perspective reconstruction. TACER uses the asset's canonical orientation by default and stores available rotation estimates in the corresponding `place` instruction. If bounding boxes are unavailable, relations in R define an initial placement region, while unresolved relations remain explicit constraints in P^s.

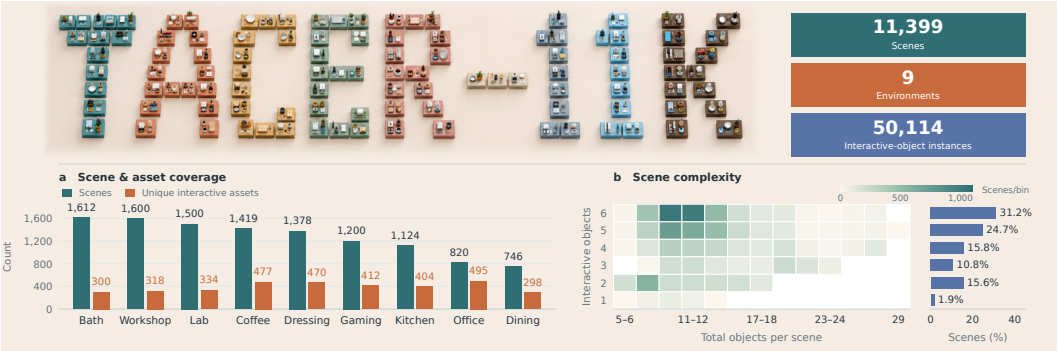

Figure 2: **TACER-11K composition.** Top: dataset name and totals. (a) Scenes and distinct interactive assets by environment; assets may be shared. (b) Scene counts by interactive and total objects (two-object bins; final bin: 29). White indicates zero; right bars show percentages.

During execution, out-of-bounds targets are moved into the valid support region. If a target pose intersects an already placed object, TACER searches nearby positions in increasing distance from the initial target. Candidate placements are locally checked using the asset representations in Section 3.3 and the verification criteria in Section 3.5. After all required objects are placed, execution returns S_0, object poses, and a record for verification and refinement.

3.5 Simulation-Guided Code Refinement and Verification

TACER checks spatial relations, physical stability under settling and perturbations, and articulation clearance, linking violations and affected objects through their identifiers to asset bindings B and scene-level program P^s. Based on this evidence, the controller localizes each failure to the asset or scene level: asset-level failures trigger refinement of p_i^{asset} or replacement of the asset and binding b_i, while scene-level failures are addressed by revising placement instructions in P^s. As illustrated in Fig. 1, TACER follows these two refinement paths before re-execution and verification.

Each refined program is re-executed according to D and re-verified. A candidate is accepted only if verification improves without regressions; otherwise, TACER restores the last accepted program. Revisiting an earlier stage invalidates and recomputes downstream checks. Within a fixed budget, this process continues until the required criteria are satisfied. TACER then re-executes the accepted program and performs final scene-level verification before returning $(P^\star, S^\star, E^\star)$; violations remaining at budget exhaustion leave the scene unverified. Appendices A.3–A.4 specify verification thresholds, articulation sampling, and acceptance rules; Algorithm 1 formalizes refinement and final verification.

3.6 TACER-11K Dataset

To assess scalability, we construct TACER-11K with 11,399 tabletop scenes across nine tabletop environments, including office desks, laboratory benches, and workshop tables, and 50,114 interactive-object instances. Scenes contain 5–29 objects, averaging 12.50 objects and 4.40 interactive objects. Physical asset representations, interactive objects, and available simulation evidence support robot evaluation and interaction data collection. Programs P^a and P^s, asset bindings B, dependencies D, and asset sources are retained for inspection and further refinement. Figure 2 summarizes the statistics; Appendix B provides detailed definitions.

4 TabletopAudit: Evaluating Simulation Readiness and Interaction Usability

Existing tabletop evaluations emphasize visual judgments and static geometric measures (Hao et al., 2025; Wang et al., 2025), providing limited evidence of simulation readiness and interaction usability. Building on the requirement-driven perspective of SceneEval (Tam et al., 2026), we introduce

TabletopAudit, an independent benchmark spanning input fidelity, 3D scene quality, simulation validity, and robot task execution.

TabletopAudit contains 500 cases across nine tabletop domains: 333 from six indoor domains in MesaTask (Hao et al., 2025) and 167 newly constructed laboratory, office, and workshop cases. Each case pairs a text specification with a reference image; the 167 new cases use GPT-image-2. All pairs are manually verified, with evaluation requirements retained (Appendix C.1). The Text track uses the full specification; the Image track uses only the reference image.

Input Fidelity and 3D Scene Quality are evaluated on all 500 cases. For Simulation Validity and Embodied Execution, we select 100 cases across all nine domains as **Embodied-100**: **RigidManip-60** for rigid-object tasks and **Articulation-40** for articulated-object tasks. Each case specifies a robot task and the annotations required for execution and success evaluation, and all scene–task pairs are manually reviewed. Appendix D details the annotations and execution protocol; Figure 3 summarizes the benchmark.

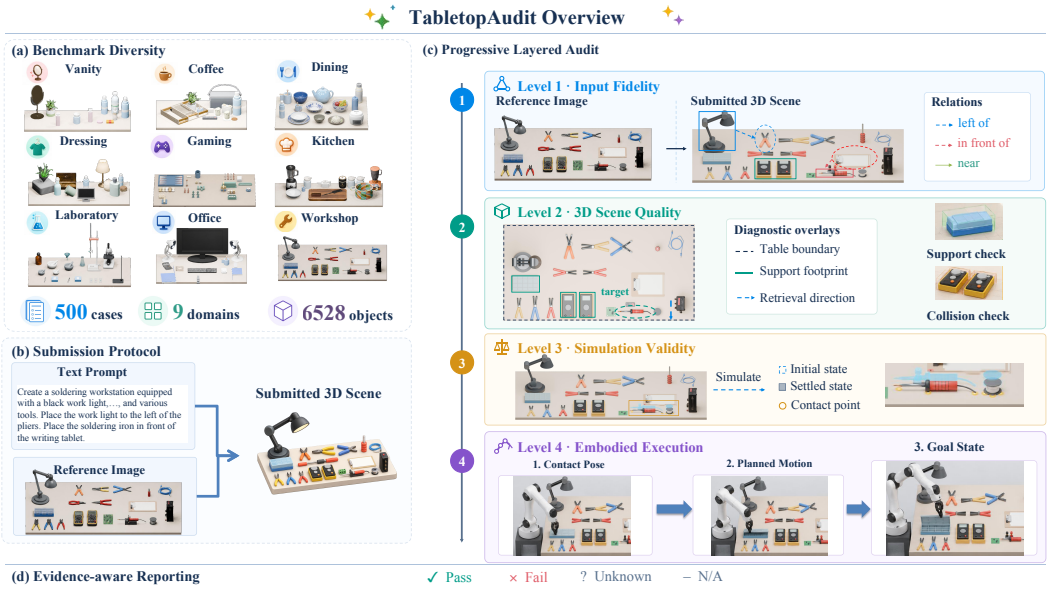

Figure 3: **TabletopAudit overview.** (a) Nine domains. (b) Shared Text/Image submissions. (c) Four audit levels. (d) Evidence-aware outcomes.

Input Fidelity covers count (`CNT`), relation (`REL`), placement (`PLC`), and attribute (`ATR`) requirements. **3D Scene Quality** measures collision (`COL`), support (`SUP`), and, on the Text track, workspace boundary (`EDGE`) and access clearance (`CLR`); the Image track additionally reports LPIPS (Zhang et al., 2018), DINOv2 (Oquab et al., 2024), and CLIP (Radford et al., 2021). **Simulation Validity** covers loading (`LOAD`), physics advancement (`PHYS`), and target settling (`STAB`). **Embodied Execution** covers grasp or interaction-pose generation (`GRASP`/`POSE`), reachability (`REACH`), planning (`PLAN`), physical execution (`PICK`/`MOVE`), and task completion (`TASK`).

Methods submit final scenes through a common interface. Evaluation uses frozen requirements and shared simulation and robot protocols, independently of each method's construction process. Outcomes are `pass`, `fail`, or `unknown` , with `N/A` reserved for inapplicable units; results are reported separately by track and level. Appendices C.2–C.4 specify metrics, aggregation, and submission requirements; Appendix D gives execution protocols.

5 Experiments

We assess input fidelity and construction quality (§5.1–5.2), simulation readiness and interaction usability (§5.3), and component contributions (§5.4). Downstream studies examine practical utility for data collection and policy learning, with an additional multi-support extension (§5.5).

Setup. We follow TabletopAudit (Section 4) on the 500-case Text and Image tracks and use Embodied-100 for simulation validity and embodied execution. Text-track baselines are Holodeck-table, adapted from Holodeck (Yang et al., 2024b), MesaTask (Hao et al., 2025), and TabletopGen (Wang et al., 2025); Image-track baselines are MIDI (Huang et al., 2025), SAM3D (Team et al., 2025), and TabletopGen. On Embodied-100, we compare against TabletopGen, adding an Articraft (Zhou et al., 2026) route for articulated assets. For simulation and embodied evaluation, all methods use the same robot, pose generators, cuRobo (Sundaralingam et al., 2023) configuration, Isaac Sim (NVIDIA, 2026) settings, and evaluation budgets (Appendix D). TACER synthesizes intermediate reference images for text inputs with GPT-image-2 and uses Kimi-K3 for all VLM components; baselines otherwise retain their default configurations.

5.1 TEXT-CONDITIONED CONSTRUCTION

Table 1 shows TACER leads overall input fidelity. Holodeck-table scores well on COL and EDGE but poorly on CNT and REL, and MesaTask remains weak in input fidelity and collision avoidance. TabletopGen improves CNT and leads SUP, but reaches only 24.20% REL and 26.14% PLC; TACER raises these to 78.65% and 95.82% while leading COL and CLR. TACER better preserves objects, relations, and access clearance (Figure 4); Figure 12 provides further comparisons.

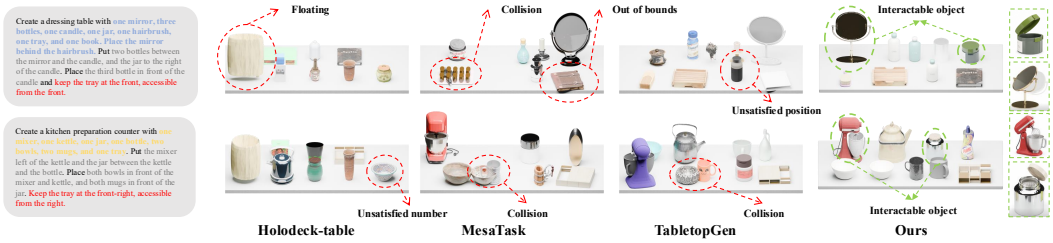

Figure 4: **Text-conditioned results.** Red callouts mark baseline failures.

Table 1: **Text-conditioned results on TabletopAudit.** Pass rates (%); best in bold.

Method	Input Fidelity				3D Scene Quality			
	CNT ↑	REL ↑	PLC ↑	ATR ↑	COL ↑	SUP ↑	EDGE ↑	CLR ↑
Holodeck-table	18.43	7.20	17.98	19.76	97.20	94.87	**98.71**	58.40
MesaTask	23.83	9.80	21.95	22.86	9.40	83.99	97.49	10.00
TabletopGen	67.80	24.20	26.14	56.68	67.60	**99.05**	96.54	63.60
Ours	**99.86**	**78.65**	**95.82**	**68.35**	**99.40**	98.65	98.61	**75.80**

5.2 IMAGE-CONDITIONED CONSTRUCTION

Table 2 shows that TACER leads all reported image-track metrics. Relative to SAM3D, REL/PLC increase from 63.25/53.25% to 82.36/97.88%, with the highest COL and SUP. Relative to TabletopGen, LPIPS decreases from 0.3896 to 0.2589 and DINOv2 similarity increases from 0.6463 to 0.8466. Figure 5 shows better preservation of small objects, local arrangements, and scene layout, including a real laboratory reconstruction; Figure 13 provides additional examples.

Table 2: **Image-track results.** Pass rates and appearance metrics; bold marks best.

Method	Input Fidelity				3D Scene Quality				
	CNT ↑	REL ↑	PLC ↑	ATR ↑	COL ↑	SUP ↑	IMG-LPIPS ↓	IMG-DINO ↑	IMG-CLIP ↑
MIDI	49.29	20.55	12.05	28.68	44.80	19.29	0.4126	0.6542	0.8741
SAM3D	98.95	63.25	53.25	60.13	81.20	99.35	0.3594	0.6063	0.7684
TabletopGen	76.42	31.60	34.94	61.83	74.80	99.23	0.3896	0.6463	0.8982
Ours	**99.67**	**82.36**	**97.88**	**84.62**	**98.80**	**99.55**	**0.2589**	**0.8466**	**0.9199**

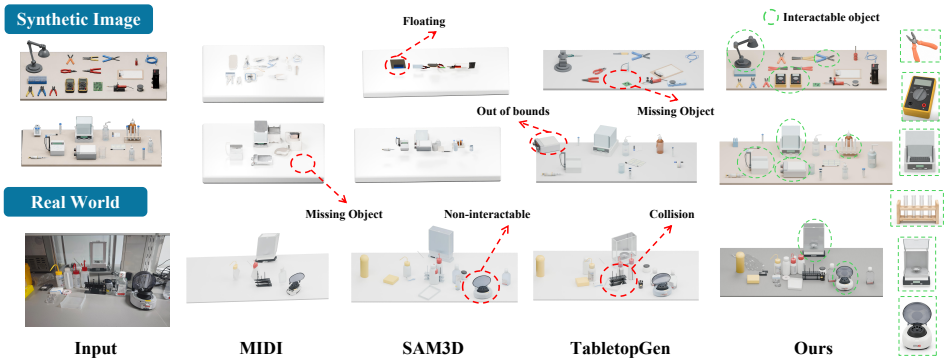

Figure 5: **Image-conditioned results.** Synthetic and real-image comparisons; red marks failures.

5.3 Simulation Validity and Embodied Execution

Table 3 reports simulation validity and embodied execution on Embodied-100. On RigidManip-60, TACER improves STAB/PLAN/TASK from 56.67/33.33/20.00% to 93.33/85.00/71.67%. On Articulation-40, TACER achieves 85.00% STAB and 60.00% TASK, compared with 32.50% and 5.00% for TabletopGen + Articraft. The gaps between stability and task completion highlight that simulation validity alone does not ensure interaction success, even with articulated assets. Figure 6 contrasts successful placement and lid opening in TACER scenes with failed baseline attempts; Figures 15 and 16 provide further examples.

Table 3: **Embodied-100 results.** Cumulative pass rates (%).

(a) RigidManip-60

Method	Simulation Validity			Embodied Execution				
	LOAD ↑	PHYS ↑	STAB ↑	GRASP ↑	REACH ↑	PLAN ↑	PICK ↑	TASK ↑
TabletopGen	81.67	63.33	56.67	46.67	40.00	33.33	23.33	20.00
Ours	**100.00**	**96.67**	**93.33**	**90.00**	**86.67**	**85.00**	**78.33**	**71.67**

(b) Articulation-40

Method	Simulation Validity			Embodied Execution				
	LOAD ↑	PHYS ↑	STAB ↑	POSE ↑	REACH ↑	PLAN ↑	MOVE ↑	TASK ↑
TabletopGen + Articraft	87.50	67.50	32.50	25.00	20.00	10.00	5.00	5.00
Ours	**100.00**	**95.00**	**85.00**	**77.50**	**72.50**	**70.00**	**62.50**	**60.00**

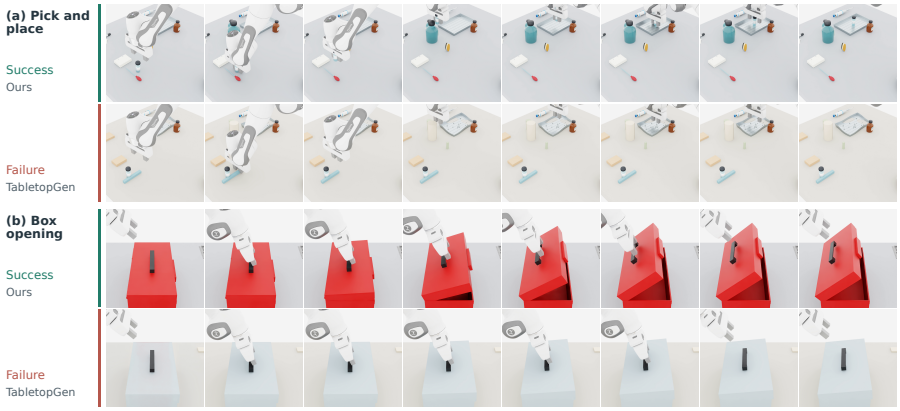

Figure 6: **Embodied execution.** (a) Pick and place. (b) Box opening. Top: TACER successes; bottom: baseline failures. Time progresses left to right.

5.4 Ablation Studies

Table 4 compares variants (Appendix A.5) under matched conditions. Removing the hierarchical program lowers REL/PLC by 14.13/11.75 points. Single-route construction lowers articulated TASK from 60.00% to 10.00%. Without staged verification or local refinement, rigid STAB falls from 93.33% to 68.33% or 51.67%, and TASK from 71.67% to 30.00% or 23.33%, respectively.

Table 4: **Ablations.** Pass rates (%) except IMG-DINO; bold marks best.

Method	Input Fidelity				3D Scene Quality			Simulation Validity			Embodied Execution	
	Image track				Image track			RigidManip-60		Articulation-40	RigidManip-60	Articulation-40
	CNT ↑	REL ↑	PLC ↑	ATR ↑	COL ↑	SUP ↑	IMG-DINO ↑	PHYS ↑	STAB ↑	STAB ↑	TASK ↑	TASK ↑
Full TACER	**99.67**	**82.36**	**97.88**	**84.62**	**98.80**	**99.55**	**0.8466**	**96.67**	**93.33**	**85.00**	**71.67**	**60.00**
w/o Hierarchical Program	99.61	68.23	86.13	84.51	92.40	88.63	0.8178	**96.67**	78.33	72.50	55.00	42.50
w/o Heterogeneous Asset Routing	95.18	79.46	95.32	60.68	97.20	98.13	0.6132	93.33	85.00	55.00	63.33	10.00
w/o Staged Verification	99.11	76.85	92.46	83.31	85.60	89.42	0.8304	86.67	68.33	55.00	30.00	15.00
w/o Local Refinement	99.52	62.16	80.46	84.29	77.40	81.27	0.7748	80.00	51.67	37.50	23.33	7.50

5.5 Downstream Applications

Embodied data generation and policy learning. Using M2T2 (Yuan et al., 2023), cuRobo, and Isaac Sim in TACER scenes, we collect 303 successful screwdriver-to-tray demonstrations, including recovery trajectories (Figure 7). On ten held-out augmented instances (Table 5), fine-tuning $\pi_{0.5}$ (Physical Intelligence et al., 2025) raises completion from 0/10 to 5/10 and lifting from 4/10 to 8/10; BC-RNN, trained using robomimic (Mandlekar et al., 2022) with recovery data, completes 1/10 tasks. These results support task-specific policy learning; Appendices E.2–E.4 detail data collection, splits, training, and evaluation.

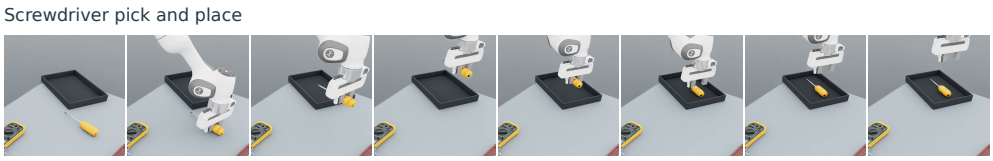

Figure 7: **Screwdriver-to-tray demonstration.** From grasp to placement.

Table 5: **Policy results.** Successes/10 with separate evaluators (Appendix E.4).

(a) BC-RNN policy adaptation					*(b) $\pi_{0.5}$ fine-tuning*				
Model	**Success ↑**	**Reach ↑**	**Closure ↑**	**Lift ↑**	Model	**Success ↑**	**Reach ↑**	**Closure ↑**	**Lift ↑**
Fine-tuned	0/10	4/10	8/10	1/10	Original	0/10	7/10	8/10	4/10
FT + recovery	**1/10**	**8/10**	**10/10**	**3/10**	Fine-tuned	**5/10**	**8/10**	**9/10**	**8/10**

Multi-support-surface extension. TACER arranges objects across shelves, raised platforms, and tabletops (Appendix E.5).

6 Conclusion

TACER combines hierarchical programs with simulation-guided refinement to generate tabletop scenes for interaction. TabletopAudit evaluates fidelity, geometry, simulation validity, and robot execution, showing improved rigid and articulated task success. TACER-11K and the downstream study support demonstration collection and policy learning. Future work includes complex mechanisms, deformable objects, and real-robot transfer.

AI use statement

Generative AI tools were used primarily to assist with language polishing and the implementation and debugging of the code framework. The use of generative models as components of the proposed method is described in Sections 3.1–3.5, with model choices specified in Section 5. The authors reviewed and revised all AI-assisted content and take responsibility for the final manuscript, code, experiments, citations, and claims.

REFERENCES

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, et al. SAM 3: Segment Anything with Concepts. *arXiv preprint arXiv:2511.16719*, 2025. URL https://arxiv.org/abs/2511.16719.

Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-Design: Personalized LLM Interior Designer. *arXiv preprint arXiv:2404.02838*, 2024. URL https://arxiv.org/abs/2404.02838.

Weixing Chen, Zhuoqian Feng, Yang Liu, Yexin Zhang, Yifan Wen, Yinghong Liao, Weichao Qiu, Guanbin Li, and Liang Lin. PhyScene3D: Physically Consistent Interactive 3D Tabletop Scene Generation. *arXiv preprint arXiv:2606.01649*, 2026. URL https://arxiv.org/abs/2606.01649.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision Transformers Need Registers. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16588.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional Visual Planning and Generation with Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 18225–18250, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/3a7f9e485845dac27423375c934cb4db-Abstract.html.

Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. AnyHome: Open-vocabulary Generation of Structured and Textured 3D Homes. In *European Conference on Computer Vision*, pp. 52–70, 2024. URL https://arxiv.org/abs/2312.06644.

Jinkun Hao, Naifu Liang, Zhen Luo, Xudong Xu, Weipeng Zhong, Ran Yi, Yichen Jin, Zhaoyang Lyu, Feng Zheng, Lizhuang Ma, and Jiangmiao Pang. MesaTask: Towards Task-Driven Tabletop Scene Generation via 3D Spatial Reasoning. *arXiv preprint arXiv:2509.22281*, 2025. URL https://arxiv.org/abs/2509.22281.

Zehuan Huang, Yuan-Chen Guo, Xingqiao An, Yunhan Yang, Yangguang Li, Zi-Xin Zou, Ding Liang, Xihui Liu, Yan-Pei Cao, and Lu Sheng. MIDI: Multi-Instance Diffusion for Single Image to 3D Scene Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23646–23657, 2025. URL https://arxiv.org/abs/2412.03558.

Chenguo Lin and Yadong Mu. InstructScene: Instruction-driven 3D Indoor Scene Synthesis with Semantic Graph Prior. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2402.04717.

Zhen Luo, Yixuan Yang, Xudong Xu, Jinkun Hao, Zhaoyang Lyu, Feng Zheng, Jiangmiao Pang, and Yanwei Fu. STABLE: Simulation-Ready Tabletop Layout Generation via a Semantics–Physics Dual System. *arXiv preprint arXiv:2605.16137*, 2026. URL https://arxiv.org/abs/2605.16137.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690, 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

NVIDIA. Isaac Sim Documentation. https://docs.isaacsim.omniverse.nvidia.com/latest/index.html, 2026. Software documentation. Accessed September 8, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien

Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=a68SUt6zFt.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. SceneSmith: Agentic Generation of Simulation-Ready Indoor Scenes, 2026. URL https://arxiv.org/abs/2602.09153.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. LayoutVLM: Differentiable Optimization of 3D Layout via Vision-Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2412.02193.

Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. cuRobo: Parallelized Collision-Free Minimum-Jerk Robot Motion Generation. *arXiv preprint arXiv:2310.17274*, 2023. URL https://arxiv.org/abs/2310.17274.

Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. SceneEval: Evaluating Semantic Coherence in Text-Conditioned 3D Indoor Scene Synthesis. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2026. URL https://arxiv.org/abs/2503.14756.

SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. Sam 3d: 3dfy anything in images. 2025. URL https://arxiv.org/abs/2511.16624.

Can Wang, Hongliang Zhong, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Chat2Layout: Interactive 3D Furniture Layout with a Multimodal LLM. *arXiv preprint arXiv:2407.21333*, 2024. URL https://arxiv.org/abs/2407.21333.

Puyi Wang, Yuhao Wang, Linjie Li, Zhengyuan Yang, Kevin Qinghong Lin, Yangguang Li, and Yu Cheng. SceneCode: Executable World Programs for Editable Indoor Scenes with Articulated Objects, 2026. URL https://arxiv.org/abs/2605.19587.

Ziqian Wang, Yonghao He, Licheng Yang, Wei Zou, Hongxuan Ma, Liu Liu, Wei Sui, Yuxin Guo, and Hu Su. TabletopGen: Tabletop Scene Generation and Interactive Simulation for Robotic Manipulation. *arXiv preprint arXiv:2512.01204*, 2025. URL https://arxiv.org/abs/2512.01204.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. SAGE: Scalable Agentic 3D Scene Generation for Embodied AI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22358–22368, 2026. URL https://arxiv.org/abs/2602.10116.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D Latents for Scalable and Versatile 3D Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21469–21480, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Xiang_Structured_3D_Latents_for_Scalable_and_Versatile_3D_Generation_CVPR_2025_paper.html.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. SceneWeaver: All-in-One 3D Scene Synthesis with an Extensible and Self-Reflective Agent. In *Advances in Neural Information Processing Systems*, 2025a. URL https://papers.nips.cc/paper_files/paper/2025/hash/cd2b3c429c8a2ca57656970e010b4b60-Abstract-Conference.html.

Yixuan Yang, Junru Lu, Zixiang Zhao, Zhen Luo, James J. Q. Yu, Victor Sanchez, and Feng Zheng. LLplace: The 3D Indoor Scene Layout Generation and Editing via Large Language Model. *arXiv preprint arXiv:2406.03866*, 2024a. URL https://arxiv.org/abs/2406.03866.

Yixuan Yang, Zhen Luo, Tongsheng Ding, Junru Lu, Mingqi Gao, Jinyu Yang, Victor Sanchez, and Feng Zheng. OptiScene: LLM-driven Indoor Scene Layout Generation via Scaled Human-aligned Data Synthesis and Multi-stage Preference Optimization. In *Advances in Neural Information Processing Systems*, 2025b. URL https://papers.neurips.cc/paper_files/paper/2025/hash/3ca6d336ddaa316a6ae953a20b9477cf-Abstract-Conference.html.

Yixuan Yang, Zhen Luo, Wanshui Gan, Jinkun Hao, Junru Lu, Jinghao Yan, Zhaoyang Lyu, and Xudong Xu. Code-as-Room: Generating 3D Rooms from Top-Down View Images via Agentic Code Synthesis. *arXiv preprint arXiv:2605.18451*, 2026. URL https://arxiv.org/abs/2605.18451.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language Guided Generation of 3D Embodied AI Environments. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16227–16237, 2024b. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yang_Holodeck_Language_Guided_Generation_of_3D_Embodied_AI_Environments_CVPR_2024_paper.html.

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. *arXiv preprint arXiv:2601.11109*, 2026. URL https://arxiv.org/abs/2601.11109.

Wentao Yuan, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. M2T2: Multi-Task Masked Transformer for Object-centric Pick and Place. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3619–3630, 2023. URL https://proceedings.mlr.press/v229/yuan23a.html.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. URL https://arxiv.org/abs/1801.03924.

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. *arXiv preprint arXiv:2605.15187*, 2026. URL https://arxiv.org/abs/2605.15187.

Xiaoming Zhu, Xu Huang, Qinghongbing Xie, Zhi Deng, Junsheng Yu, Yirui Guan, Zhongyuan Liu, Lin Zhu, Qijun Zhao, Ligang Liu, and Long Zeng. Imaginarium: Vision-Guided High-Quality 3D Scene Layout Generation. *ACM Transactions on Graphics*, 2025. URL https://arxiv.org/abs/2510.15564.

APPENDIX OVERVIEW

Appendices A–E cover method details, TACER-11K, benchmark metrics, robot evaluation, and downstream policy learning. Appendix F provides additional comparisons and interaction sequences.

A ADDITIONAL METHOD DETAILS

A.1 OBJECT CONFIRMATION AND SCENE RELATIONS

Small objects receive a focused localization pass over their predicted support region and, when needed, one full-image retry. Reconciliation matches candidates to unused segmentation records by semantic identifier, then name or alias. Confirmation requires a matched numeric bounding box with positive width and height; confidence and localization/retry provenance are recorded without an additional confidence threshold at this step. The controller checks mask usability and identity consistency. Confirmed objects enter O; pending identities are excluded from downstream inventories and the scene graph.

Relation inference and interpretation. The VLM infers relations from the image, confirmed objects, and supports. Self-relations, unknown endpoints, and unsupported types are rejected; ambiguous directions remain unresolved. Image left/right map to horizontal directions and bottom/top to front/back, with $+x$ right and $+y$ front. This is a layout initialization convention, not calibrated camera reconstruction. Table 6 lists the core relations.

Table 6: **Core scene relations.** Relations are directed from subject to target.

Relation	Meaning	Assembly order
On top of	Subject rests on the target's supporting surface.	Target first
Inside	Subject is contained within the target.	Target first
Left of	Subject lies to the left of the target.	Unconstrained
In front of	Subject lies toward the front of the target.	Unconstrained
Near	Subject and target are spatially close.	Unconstrained

The construction-time relation verifier evaluates these requirements using object bounding boxes. Let $\epsilon = 0.015$ m for metric scenes. For `left_of`, it requires $x_s^{\max} \le x_t^{\min} + \epsilon$; for `in_front_of`, $y_s^{\min} \ge y_t^{\max} - \epsilon$. For `near`, the planar distance between the two boxes must not exceed $\max(2\epsilon, 0.75 \min(d_s, d_t))$, where d_s, d_t are their footprint diagonals. The coarse `on_top_of` predicate requires a vertical contact gap of at most ϵ in magnitude and footprint overlap exceeding 5% of the smaller footprint. The coarse `inside` predicate requires the subject's planar center to lie within the target bounds, allowing ϵ, and its bottom height to lie between $z_t^{\min} - \epsilon$ and $z_t^{\max} + \max(\epsilon, H_s)$, where H_s is the subject's height. These coarse diagnostics require separate physical and kinematic checks and differ from the benchmark predicates in Appendix C.

Construction dependencies. Support, containment, and attachment relations induce provider-to-dependent edges in D; other relations remain constraints in R. Missing endpoints and cycles are rejected, while geometric conflicts are handled during layout refinement.

A.2 ASSET CONSTRUCTION AND PHYSICAL PREPARATION

Rules select eligible asset routes from object semantics, dimensions, articulation, and scene context; VLM ranking orders them, falling back to rule order if invalid. At most three routes are tried, with TRELLIS (Xiang et al., 2025) last when enabled (Table 7). The chain stops at the first produced candidate, whose acceptance still requires the asset checks. Attempts retain their outcomes and failure reasons.

An optional procedural fallback after route exhaustion remains subject to the same checks. Assets use metric Z-up coordinates, converting centimeter-valued layout poses, and retain scale, canonical transform, mass, center of mass, inertia, and collision geometry. Containers may use five-wall

Table 7: **Asset routes by object requirement.** Eligible routes are ranked per object.

Object requirement	Candidate routes	Fallback
Common object	Retrieval	Rigid code when applicable, then image-to-3D
Irregular or organic object	Retrieval	Image-to-3D
Regular rigid geometry	Rigid code or retrieval	Image-to-3D
Movable parts	Articulated code; compatible retrieval or explicit composite construction	Image-to-3D, subject to articulation checks
Thin or sheet-like object	Thin-surface construction	Image-to-3D
Composite apparatus	Specified articulated or rigid construction	Retrieval, then image-to-3D

proxies; other rigid objects use convex decomposition, hulls, or a documented fallback. Articulated assets require valid links, joints, axes, and limits. Table 8 lists separate budgets for construction and repair.

Table 8: **Construction and refinement budgets.**

Operation	Maximum attempts
Asset routes per normal chain	3
Code regeneration / checker-directed repair	3 / 3
Scene-code repair / selective object repair	3 / 3 rounds
Layout optimization	3 iterations

A.3 Verification Criteria

All required checks in Table 9 must pass. Stage 3 uses bounding-box collision proxies; final verification uses prepared asset collision geometry. Missing evidence leaves a check unresolved. Revisiting a stage invalidates its downstream checks; programs, bindings, measurements, and repair history are retained for replay.

Table 9: **Stage-specific verification criteria.**

Stage	Required checks
Object discovery	Valid confirmed objects and relations; unresolved candidates excluded.
Asset construction	Valid geometry and canonical poses; required physical and articulated representations available.
Scene assembly	Executable program, valid bindings and dependencies, support and layout checks, sampled articulation clearance, and local physical checks.
Final verification	Object completeness, support contact, spatial relations, articulation clearance, settling and perturbation tests, and consistency of the resulting scene and verification records.

Reference implementation settings. Drake uses a 1 ms timestep and a 0.5 s Stage 3 rollout. Final verification runs 2 s of settling, then independently repeats settling followed by a horizontal velocity reset of 0.05 m/s along $+x$ or $+y$ by object index and another 1 s of simulation. Settling speeds are medians of 11 samples over the last 0.5 s; all pose changes are relative to the initial pose. Table 10 lists motion rejection thresholds, separate from the Isaac Sim benchmark in Appendix D.

The angular-speed rejection additionally requires rotation activity of at least 2° in Stage 3 or 1° in final verification, using the larger settling or perturbation rotation for the latter. Independent pose-change limits, support violations, and falling remain blocking.

Articulation clearance uses seven evenly spaced joint-range fractions, including closed, half-open, and fully-open states; all movable joints share the sampled fraction, and continuous joints use $[0, \pi]$. Link/environment AABBs use a 2 mm overlap tolerance, excluding adjacent-link self-contact and

Table 10: **Default generation-verifier motion thresholds.** Reaching a threshold rejects the candidate, subject to the angular-activity rule below.

Measurement	Stage 3 proxy	Final verification
Settling linear speed	0.05 m/s	0.01 m/s
Settling angular speed	0.50 rad/s	0.10 rad/s
Planar displacement	3 cm	2 cm
Rotation change	15°	10°
Post-perturbation planar displacement	–	5 cm
Post-perturbation rotation change	–	20°

the declared support from environment checks. Base support requires 25% footprint coverage and a gap of at most 2 cm in magnitude. This sampled kinematic test does not cover all multi-joint combinations or unlocked articulated dynamics.

A.4 Simulation-Guided Refinement

At refinement step t, the accepted state is (P_t, S_t, E_t) with bindings B_t and dependencies D_t; $(\widehat{P}, \widehat{S}, \widehat{E})$ is a proposal. Requirements G, capabilities $\mathcal{C}$, and settings Π remain fixed. Algorithm 1 handles deterministic and model-proposed edits; candidate improvement alone does not establish final verification.

Algorithm 1 Simulation-guided refinement and final verification

Input: program P_0, requirements G, dependencies D_0, required capabilities $\mathcal{C}$, settings Π, budget K.
Output: verified $(P^\star, S^\star, E^\star)$ or an unresolved result; both retain refinement history.

Compile (P_0, D_0); return unresolved if compilation fails.
Execute and verify under $(G, \mathcal{C}, \Pi)$ to initialize (P_0, S_0, E_0).
$t \leftarrow 0$
for $k = 1, \ldots, K$ **do**
  Attribute violations I_t to assets or scene instructions; break if all checks pass.
  Propose an edit and compile candidate $(\widehat{P}, \widehat{D})$.
  if compilation fails **then** record rejection; **continue**.
  Re-execute affected stages and renew downstream checks to obtain $(\widehat{S}, \widehat{E})$.
  Record the edit, verification results, and candidate acceptance decision.
  if accepted **then** set $(P_{t+1}, D_{t+1}, S_{t+1}, E_{t+1})$ to the candidate; $t \leftarrow t + 1$.
  else restore the last accepted program and scene.
end for
Renew prerequisite checks; return unresolved if required checks remain unsatisfied.
Assemble from the accepted program and rerun final scene verification.
if all stage criteria pass and scene and evidence are consistent
  then return $(P^\star, S^\star, E^\star)$; **else return** unresolved result.

For layout evidence E, let $q(E)$ be the visual critic score, $\mathcal{B}(E)$ the blocking issues, and $\mathcal{O}(E)$ the objects requiring repair. Candidate acceptance requires

$$\begin{aligned}\mathrm{Accept}(E, \widehat{E}, I, \widehat{c}) = [\widehat{c} = \textsc{complete}] \wedge \mathrm{Improve}(E, \widehat{E}, I) \\ \wedge \neg\,\mathrm{ScoreRegress}(E, \widehat{E}) \wedge \neg\,\mathrm{BlockRegress}(E, \widehat{E}) \\ \wedge \neg\,\mathrm{ObjectRegress}(E, \widehat{E}) \wedge \neg\,\mathrm{SimRegress}(E, \widehat{E}).\end{aligned} \tag{4}$$

Here $\widehat{c} = \textsc{complete}$ denotes completed execution. Improvement means new convergence, a higher critic score, fewer blocking issues or affected objects, better simulation evidence, or resolution of a targeted issue I. Regression checks forbid a score drop beyond numerical tolerance, increased issue/object counts, or a newly failing simulation decision or blocking object. Rejected edits are recorded and rolled back. Deterministic final-assembly physics repairs instead require successful execution, layout validation, and scene stability; articulation clearance and all stage gates are rechecked before final acceptance.

A.5 Ablation Configurations

The reference implementation defines four component variants. *Without hierarchical program* replaces scene-level program execution with direct incremental scene-graph layout, retaining asset construction. *Without heterogeneous asset routing* fixes the configured route to retrieval, disables route analysis, and limits the route chain to one attempt without an image-to-3D fallback. *Without staged verification* uses one scene-construction round, disables intermediate fast verification and the visual critic, and retains the final audit. *Without local refinement* disables scene-code and selective-object repair, pose-only repair fallback, and final-assembly scene and asset repair budgets. These switches specify construction variants; their outputs remain subject to the independent evaluation in Appendices C and D.

B TACER-11K Dataset Details

B.1 Scene Composition and Complexity

TACER-11K contains 11,399 accepted scenes spanning nine tabletop environments. Table 11 summarizes the corpus using one record per scene. An original object belongs to the input layout before interactive augmentation; a final object is counted after replacement and insertion; and an interactive object is an interactive-asset instance placed in the final scene. The free-area ratio is the unoccupied tabletop area divided by the valid placement-zone area.

Table 11: **TACER-11K scene composition and complexity.** Panel (a) reports corpus-level statistics over 11,399 scenes. Panel (b) gives the number and percentage of scenes in each tabletop environment.

Quantity	Value	Unit or scope
(a) Overall composition and complexity		
Corpus size	11,399	scenes
Environment coverage	9	environments
Original objects per scene	mean 9.51; median 9	objects
Final objects per scene	mean 12.50; median 11	range 5–29
Interactive objects per scene	mean 4.40; median 5	range 1–6
Free tabletop area ratio	median 0.749	per scene
(b) Environment composition		
Bathroom vanity	1,612	14.1%
Workshop tabletop	1,600	14.0%
Laboratory tabletop	1,500	13.2%
Coffee table	1,419	12.4%
Dressing table	1,378	12.1%
Gaming tabletop	1,200	10.5%
Kitchen counter	1,124	9.9%
Office table	820	7.2%
Dining table	746	6.5%
Total	11,399	100.0%

Each scene retains asset and scene programs, object-to-asset bindings, dependencies, asset sources, and available simulation evidence for replay and refinement.

C TabletopAudit Evaluation Protocol

TabletopAudit scores final scene submissions independently of their construction. Robot evaluation is specified in Appendix D.

C.1 BENCHMARK COMPOSITION

The 500 cases comprise 333 MesaTask-derived cases across six indoor domains and 167 new laboratory, office, and workshop cases (Table 12). The former retain reference renders and layout annotations; the latter pair written specifications with GPT-image-2 references in a matched visual style. Every pair is manually checked for inventory, attributes, relations, and placement consistency. Text and Image tracks share case identities but expose only their respective input. Image-track requirements are restricted to visible, unambiguous content.

Table 12: **TabletopAudit-500 composition.** Both input tracks use the same cases.

Domain	Cases	Domain	Cases	Domain	Cases
Bathroom vanity	56	Coffee table	56	Dining table	55
Dressing table	56	Gaming tabletop	55	Kitchen counter	55
Laboratory tabletop	56	Office table	55	Workshop tabletop	56
				Total	500

The semantic annotation contains 4,835 count, 2,000 object-relation, 4,739 placement, and 1,663 deduplicated attribute requirements, plus one access-clearance requirement per case. Simulation and robot evaluation use the Embodied-100 subset.

C.2 METRIC DEFINITIONS

For submitted objects $\mathcal{O}_c$, let $k(o)$ denote canonical category. `CNT` evaluates each category requirement k_r with its exact, lower-bound, or upper-bound quantifier Q_r:

$$N_c(k_r) = \sum_{o \in \mathcal{O}_c} \mathbf{1}[k(o) = k_r], \qquad y_{c,r}^{\mathrm{CNT}} = \texttt{pass} \iff Q_r(N_c(k_r)) = 1. \tag{5}$$

For a relation requirement $r = (k_s, \rho, k_a, Q_r)$, let $\phi_\rho(s, a)$ be the deterministic predicate for relation ρ between a subject and an anchor. The default count unit is a unique subject,

$$N_{c,r}^{\mathrm{REL}} = \sum_{s \in \mathcal{O}_c : k(s) = k_s} \mathbf{1}[\exists a \in \mathcal{O}_c \setminus \{s\} : k(a) = k_a \wedge \phi_\rho(s, a) = 1]\,. \tag{6}$$

Each subject counts once even if multiple anchors satisfy the relation; Q_r tests the resulting count. Requirements can explicitly request one-to-one matching or valid-pair counting. Predicates use yaw-rotated OBB footprints. The reference evaluator uses $-y$ as front: with zero margin, `left_of` requires $x_s^{\max} \le x_a^{\min}$ and `front_of` requires $y_s^{\max} \le y_a^{\min}$; inverse relations swap the arguments. This differs from the construction frame and tolerances in Appendix A.1.

`PLC` replaces the anchor in Equation 6 with a declared support or container. The reference codebook accepts `on_top` for a bottom-to-top gap in $[-0.25, 1.0]$ cm, at least 20% subject-footprint coverage, and a support normal within 15° of $+z$. `Inside` requires at least 95% coverage of both the footprint and height interval by the usable inner polygon and vertical range; missing inner geometry yields unknown. These are placement predicates, distinct from mesh support and dynamic stability.

For an attribute requirement with category k_r and attribute set A_r, `ATR` passes when at least one matching candidate object satisfies every requested attribute:

$$y_{c,r}^{\mathrm{ATR}} = \texttt{pass} \iff \exists o \in \mathcal{O}_c : k(o) = k_r \ \wedge \bigwedge_{a \in A_r} J(o, a) = 1. \tag{7}$$

J is Kimi-K3 applied to front and front-right object-only renders with native materials and fixed settings. Missing renders, malformed judgments, and ambiguous attributes yield unknown; `ATR` is an auxiliary visible-attribute measure.

Table 13 defines mesh-based quality metrics. Both tracks report `COL`/`SUP`; `EDGE`/`CLR` are Text-only. Transformed meshes are required for `COL`, `SUP`, and `EDGE`; boxes cannot substitute for missing mesh evidence.

Table 13: **3D Scene Quality metrics and their aggregation units.**

Metric	Unit	Pass condition
COL	scene	No non-exempt object pair remains in FCL mesh collision after a 5 mm decontact translation of the second object's copy away from the mean contact point. Declared support contacts require structure-backed evidence to be exempted; inconclusive checks remain unknown.
SUP	object	Downward mesh rays yield at least three contact points on the selected support with signed gaps in $[-0.25, 1.0]$ cm, and the transformed mesh centroid projects inside their contact hull. Negative gaps denote penetration; this is a static support check, not a settling result.
EDGE	object	Sampled object-mesh points project within the tabletop polygon with nonnegative signed margin, and downward rays confirm the corresponding tabletop region.
CLR	access requirement	At least one candidate object matching the requested category has an obstacle-free 10 cm planar approach corridor, formed by sweeping its yaw-oriented footprint with a 2 cm lateral operating margin and 2 cm clearance.

CLR measures geometric access, not robot reachability. Image-track appearance metrics use a fixed front view without best-view selection: LPIPS-VGG, DINOv2 ViT-L/14 with four register tokens (Darcet et al., 2024), and CLIP ViT-L/14. DINOv2 and CLIP report feature cosine similarity.

C.3 Scoring and Aggregation

An applicable unit receives pass, fail, or unknown; inapplicable units are excluded. Failure requires sufficient evidence; missing geometry, unresolved categories, invalid renders, and interrupted evaluation yield unknown. With $n_{m,\text{applicable}} = n_{m,\text{pass}} + n_{m,\text{fail}} + n_{m,\text{unknown}}$, aggregation is

$$\begin{aligned} \text{PassRate}_m &= \frac{n_{m,\text{pass}}}{n_{m,\text{applicable}}}, \\ \text{Coverage}_m &= \frac{n_{m,\text{pass}} + n_{m,\text{fail}}}{n_{m,\text{applicable}}}, \\ \text{PassGivenEvidence}_m &= \frac{n_{m,\text{pass}}}{n_{m,\text{pass}} + n_{m,\text{fail}}}. \end{aligned} \tag{8}$$

Main tables report PassRate; coverage must accompany comparisons when below one. Binary metrics are micro-averaged over requirements for CNT/REL/PLC/ATR/CLR, scenes for COL, and submitted objects for SUP/EDGE. Image metrics average cases with valid front-view evidence. Input tracks, domains, and rigid/articulated capabilities remain separate; no overall score is formed. A cumulative gate passes only when all preceding gates pass; subsequent gates remain unknown after a failure or unknown. Method-level N/S denotes unsupported capability, distinct from case-level unknown or inapplicability.

C.4 Shared Evaluation Settings

Evaluation fixes cases, requirements, thresholds, cameras, evaluator model versions, and seeds. Submissions are canonicalized to centimeters, $+Z$ up, $+X$ right, and $-Y$ front. The evaluator resolves categories and renders top, front, left, and front-right RGB and instance masks; extra views do not affect scoring.

Each submission record specifies case/submission IDs, coordinate conventions, and inventory/geometry completeness. Objects carry local IDs, categories and provenance, poses and dimensions; mesh metrics additionally require asset paths, hashes, units, and world transforms. Supports provide polygons/heights and containers usable inner geometry; render instances map to object IDs. Count needs trusted inventory, relations need poses/dimensions, and attributes need fixed-view RGB evidence. Metric records retain requirement/object IDs, status, measurements, thresholds, evidence

sources, and unknown reasons. Complete inventories distinguish absent objects from missing evidence; construction histories are not required.

D Simulation and Robot Evaluation

Methods share the robot/gripper, base pose, collision configuration, candidate budgets, planner settings, seeds, and retry limits. Stage records retain candidate identity, measurements, thresholds, and outcome.

Evaluator version and budgets. The frozen run configuration is identified by its configuration fingerprint alongside the evaluator version. Planning permits at most two attempts with a 2 s timeout per candidate and motion segment, usually over at most eight candidates. Execution is capped at 60 s simulated time; planning has no uniform per-case wall-clock cap.

D.1 Task Composition and Evaluation Stages

RigidManip-60 contains 20 cases each of Pick, Place-into, and Place-onto; Articulation-40 contains 20 Open, 8 Press, 8 Slide/Pull, and 4 Rotate cases (Figure 8). The tracks are disjoint and span all nine domains (Table 14). Each case specifies its target, applicable destination/contact region/moving link, initial and goal states, robot base pose, and tolerances. Manual review checks task semantics, target identity, clutter, and visible support/containment plausibility; execution is evaluated separately.

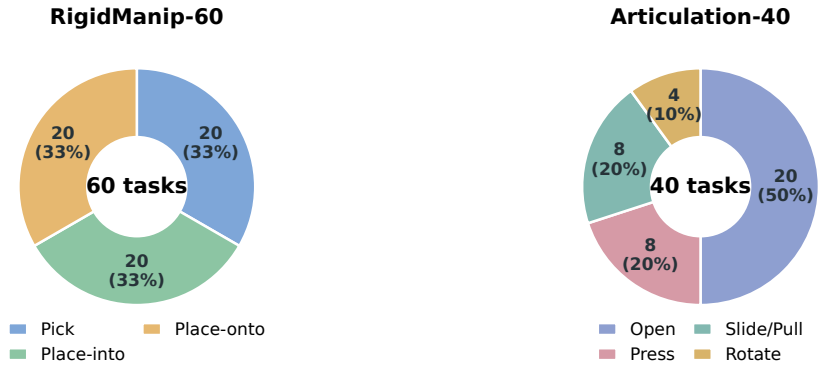

Figure 8: **Task-family composition of Embodied-100.** RigidManip-60 uses equal quotas for three rigid task families, while Articulation-40 emphasizes opening and retains three additional interaction families.

Table 14: **Domain coverage of Embodied-100.** The task subset covers all nine TabletopAudit domains; it is task-diverse rather than domain-balanced.

Domain	Cases	Domain	Cases	Domain	Cases
Bathroom vanity	7	Coffee table	9	Dining table	13
Dressing table	9	Gaming tabletop	12	Kitchen counter	12
Laboratory tabletop	13	Office table	10	Workshop tabletop	15
				Total	100

Table 15 lists the ordered gates. Each requires all predecessors to pass; cumulative scoring follows Appendix C.3.

D.2 Simulator and Target-Stability Protocol

The target is initialized at its submitted pose with zero velocity and then released under dynamics for the settling rollout. Table 16 reports the settling configuration and acceptance thresholds. Trans-

Table 15: **Embodied-100 gates.** Collision checks use the configured exemptions; track-specific success conditions are given below.

Track	Gate	Passing condition
Common	LOAD	Scene, robot, target, and required RGB-D observations instantiate.
	PHYS	At least one zero-action step completes without exception or premature termination.
	STAB	Target passes the settling thresholds.
RigidManip-60	GRASP	At least one target-OBB grasp fits the gripper.
	REACH	A candidate pregrasp passes collision-aware IK.
	PLAN	A surviving candidate has a planned pregrasp–grasp–lift trajectory.
	PICK	Physical contact, grasp, lift, and hold thresholds pass.
	TASK	Pick or placement satisfies the task-specific conditions below.
Articulation-40	POSE	A valid contact/action definition follows the joint axis and task goal.
	REACH	A candidate passes geometry precheck and collision-aware IK.
	PLAN	A surviving candidate has a planned precontact–contact–action trajectory.
	MOVE	Executed joint motion satisfies the prescribed goal/tolerance check.
	TASK	Goal attainment, measured contact/timing, and base-drift conditions pass.

lation and rotation are measured relative to the initial target pose; velocity limits are evaluated over the final 0.5 s.

Table 16: Frozen simulator configuration and target-stability thresholds.

Quantity	Setting or acceptance threshold
Physics timestep	$1/120$ s
Control decimation	2
Environment rate	60 Hz
Settling rollout	120 environment steps / 240 physics steps / 2.0 s
Settling window	final 0.5 s
Target translation	≤ 5 mm
Target rotation	$\leq 1^\circ$
Maximum settling-window linear speed	≤ 0.01 m/s
Maximum settling-window angular speed	≤ 0.1 rad/s
Maximum tabletop penetration	≤ 2 mm
Maximum footprint extension beyond tabletop	≤ 5 mm

The rollout must also complete without a simulator exception. Non-target clutter remains visually instantiated but is excluded from collision in this stability mode. Thus STAB certifies target-level settling under the frozen task profile, not scene-wide dynamic stability.

D.3 RIGIDMANIP-60

Grasp and planning. Up to eight grasps are constructed from the target OBB's principal axes/extents; the closing extent plus 4 mm clearance must fit the Franka Panda's 80 mm opening. Pregrasps are offset 12 cm along the reverse approach direction from the settled target. cuRobo solves collision-aware IK and plans pregrasp, grasp, and 12 cm lift for the same candidate. The pre-grasp collision world excludes the target; lift planning attaches a target proxy. Clutter collisions are disabled in PhysX PICK replay, so obstacle-collision evidence comes from planning.

Physical pick. PICK requires finger–target contact above 0.25 N, final gripper width strictly between 2 and 78 mm, at least 8 cm lift, and a continuous 0.5 s hold at 8–13 cm above settled height. Hold speed is at most 0.05 m/s and maximum joint-tracking error at most 0.12 rad. These lift/hold conditions define Pick TASK; placement tasks continue through transport, release, and retreat.

Placement success. Place-into requires the complete target bounding volume inside the reviewed cavity with 1 mm boundary tolerance, not a containment ratio. The slot configuration uses 3 mm position and 0.05 rad orientation tolerances. Place-onto requires the complete projected footprint within the support region with 1 mm boundary tolerance and bottom-to-support height error at most 8 mm. Both require at least 0.5 s after release with maximum linear speed at most 0.05 m/s. Geometry is checked only at the final pose, not throughout this window.

D.4 ARTICULATION-40

POSE constructs eight yaw variants with precontact, contact, and action endpoints from the task contact region and repaired URDF joint axis. A geometry precheck for non-contact links precedes collision-aware IK and cuRobo planning; candidate identity is preserved across gates. MOVE executes the trajectory in PhysX and checks measured joint motion against the task goal and tolerance.

TASK requires at least one frame within the case-specific goal tolerance, measured contact above 0.25 N no later than goal attainment, and base drift at most 1 cm. No sustained or terminal goal holding is required: reaching the goal for one frame and returning to the initial position can pass if the other conditions hold.

E DOWNSTREAM APPLICATIONS

The screwdriver-to-tray task requires full placement inside a usable tray region.

E.1 TASK AND SCENE POOL

Table 17 lists the candidate pool. Of 240 screwdriver/tray pairs, 214 fit at least one usable tray region with 20 mm clearance. This pre-execution eligibility check is separate from task success.

Table 17: Candidate scene and asset pool for the screwdriver-to-tray task.

Pool item	Count	Template difficulty	Count
Workbenches	154	Sparse	37
Screwdrivers	20	Standard	35
Trays	12	Busy	42
Asset pairs	240	Crowded	40
Eligible pairs (20 mm margin)	214	Total	154

The collection campaign draws from sparse, busy, and crowded templates.

E.2 DEMONSTRATION GENERATION

Scenes are instantiated in the simulator with compound collision geometry. M2T2 proposes grasps using the default, top-down, left-oblique, and right-oblique views. When no usable proposal is available, nine top-down candidates are constructed from the screwdriver collision geometry. cuRobo plans the manipulation motion and Isaac Sim 6.0.1 executes nine stages:

$$\begin{aligned}\text{pregrasp} \rightarrow \text{grasp} \rightarrow \text{close} \rightarrow \text{lift} \rightarrow \text{transport} \rightarrow \text{lower} \rightarrow \\ \text{release} \rightarrow \text{retreat} \rightarrow \text{settle}.\end{aligned} \tag{9}$$

An executed trajectory is accepted only if all stages complete and the following predicates hold. Let h_{lift} be the maximum rise from the initial target height, F_o the final target collision footprint, R_t the

selected usable tray region, δ_z the target-bottom offset from the prescribed region bottom, and v_T the terminal linear velocity. Then

$$\begin{aligned} h_{\text{lift}} &\geq 0.10\,\text{m}, & F_o &\subseteq R_t \oplus 2\,\text{mm}, \\ -4\,\text{mm} \leq \delta_z &\leq 15\,\text{mm}, & \|v_T\|_2 &\leq 0.10\,\text{m/s}. \end{aligned} \tag{10}$$

The tray is fixed. Acceptance checks final containment, bottom height, and terminal speed; it does not measure tray displacement or continuous post-release stability. Figure 9 shows an example trajectory and Figure 10 shows scene diversity.

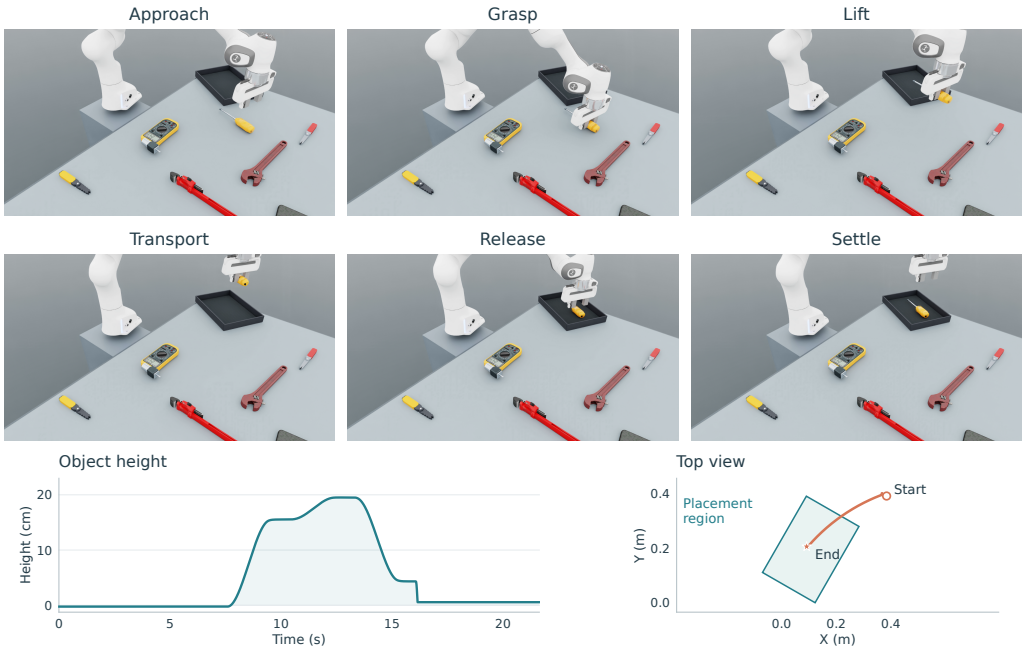

Figure 9: **A screwdriver-to-tray manipulation sequence.** Top: six stages from approach to settling. Bottom: object height over time and the top-view path relative to the placement region.

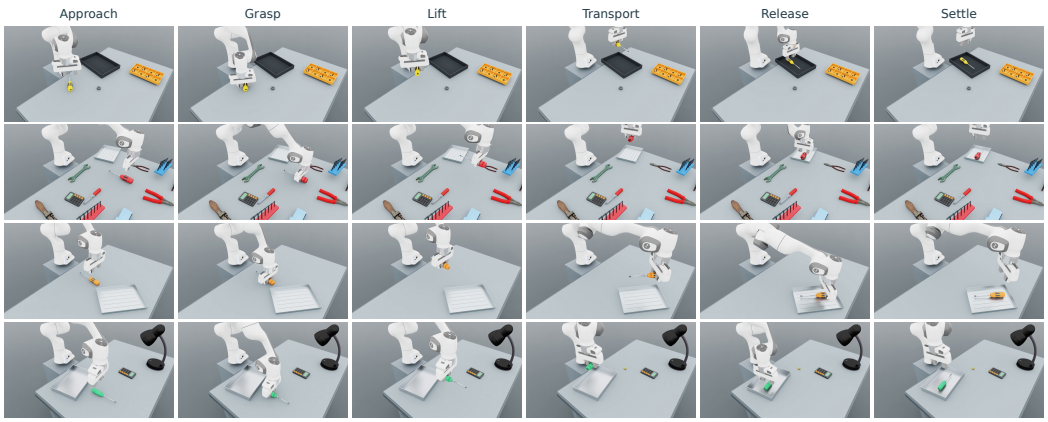

Figure 10: **Screwdriver-to-tray manipulation across generated workbench scenes.** Rows show different scenes; columns follow approach, grasp, lift, transport, release, and settling.

E.3 Training Data and Policy Setup

Table 18 summarizes 303 successful demonstrations. The clean subset supplies $\pi_{0.5}$ with 232 training episodes and 309,166 frames at 20 Hz. The 51 recovery trajectories are used only in the BC-RNN recovery ablation.

Table 18: **Demonstration data splits.** Recovery data are excluded from the clean subset.

Dataset	Train	Validation	Test	Total	Frames
Full dataset	283	10	10	303	408,986
Clean training	232	–	–	232	–
Recovery training	51	–	–	51	–
Clean subset	232	10	10	252	335,958

BC-RNN. We use robomimic (Mandlekar et al., 2022) for phase-conditioned behavior cloning: a single-layer GRU (hidden size 256) followed by an MLP with two 256-unit hidden layers. The 41-dimensional state comprises joint positions, end-effector pose, gripper position, object/tray poses, and a 9-dimensional phase code, without RGB or depth. Actions contain seven joint-target residuals and one gripper command. Training uses 10-step sequences, Adam at 10^{-4}, MSE loss, and batch 128; execution outputs one action per step. Both Table 5(a) variants start from the same base checkpoint: "Fine-tuned" uses clean data and "FT + recovery" adds the 51 recovery trajectories.

$\pi_{0.5}$. Demonstrations record three 128×128 RGB-D views; the policy uses exterior and wrist RGB resized to 224×224, with the remaining image slot masked. State is seven joint positions and gripper closure $\mathrm{clip}(1 - \bar{q}_g/0.04, 0, 1)$, where $\bar{q}_g$ is mean finger position in meters. Actions contain seven joint deltas from the current state and one binary gripper command (0=open, 1=closed). State/actions use quantile normalization; actions are padded to 32 dimensions internally, converted back to absolute joint targets on output, and truncated to eight robot dimensions.

We fine-tune OpenPI's pretrained $\pi_{0.5}$ base model on a single current observation with a 16-step action horizon and prompt "Pick up the screwdriver and place it in the tray." Training uses 20,000 AdamW steps, batch 32, gradient-norm clipping 1.0, and EMA 0.999. The learning rate warms up for 1,000 steps to 5×10^{-5} and stays constant. Only clean training episodes are used.

E.4 Evaluation Protocol

Evaluation uses ten held-out augmented instances (four sparse, three busy, three crowded); their base templates may occur in training. In Table 5, Reach denotes entry into the grasp neighborhood, Closure denotes gripper-closing behavior, and Lift denotes a rise of at least 0.10 m. These outcomes are independent, not cumulative gates. Success additionally requires finite execution, full-footprint containment, the prescribed bottom-height tolerance, and low terminal velocity. The $\pi_{0.5}$ evaluator also requires release completion, whereas BC-RNN has no separate release predicate. The panels therefore measure adaptation within each evaluator and are not pooled across policies.

E.5 Multi-Support-Surface Extension

Figure 11 illustrates four programmatic cases: a four-tier bookshelf, a multilevel computer desk, a parts-storage workbench, and a laboratory rack. Each case specifies horizontal support surfaces at distinct heights and assigns objects to a particular surface or container compartment. A supported structure can itself provide additional surfaces, as in a desk supporting a monitor riser that supports a monitor, or a laboratory table supporting a two-level reagent rack.



Figure 11: **Multiple support surfaces.** Bookshelf, office, workbench, and laboratory.

Construction combines procedural support geometry with reused assets. Asset orientation and scale are applied before placement; the transformed lower bound is aligned with the selected support height. Trays and organizers define nested regions for grouping smaller objects. Surface assignments, object identities, and asset provenance are retained alongside the assembled geometry. These qualitative cases illustrate composition across multiple support levels; dynamic stability and robot execution require separate evaluation under Appendix D.

F Additional Qualitative Results

Additional text/image comparisons and interaction sequences complement Section 5.

F.1 Additional Text-Conditioned Comparisons

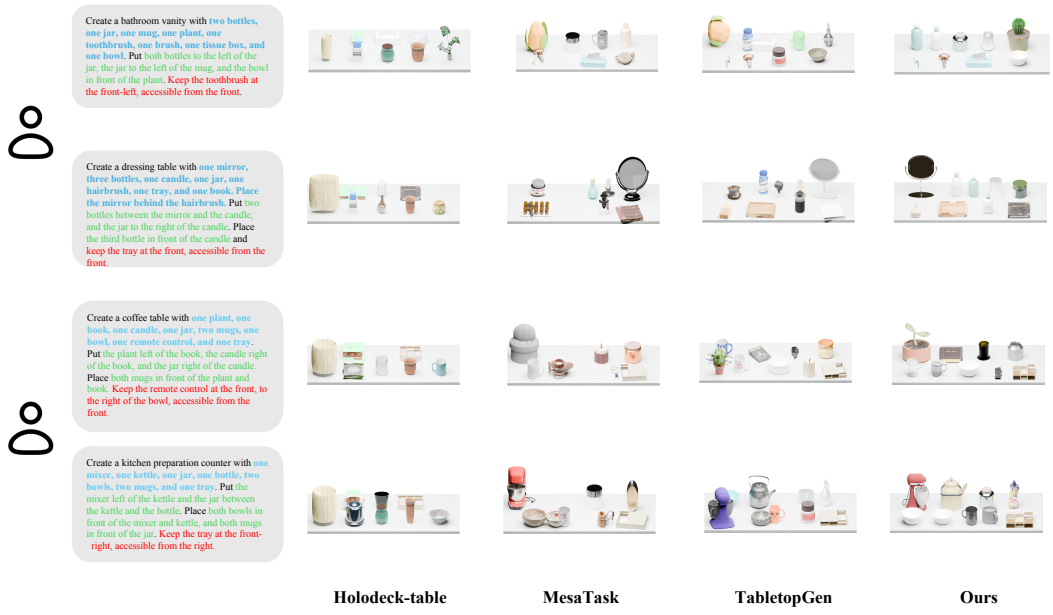

Figure 12: **Additional text-conditioned comparisons.** Additional scene specifications and their generated tabletop layouts. Colored text highlights object inventories, spatial relations, and access requirements.

F.2 Additional Image-Conditioned Comparisons

Figure 14 highlights plant retention and bottle/tray containment in dressing scenes, and small-food/utensil recovery in kitchen scenes.

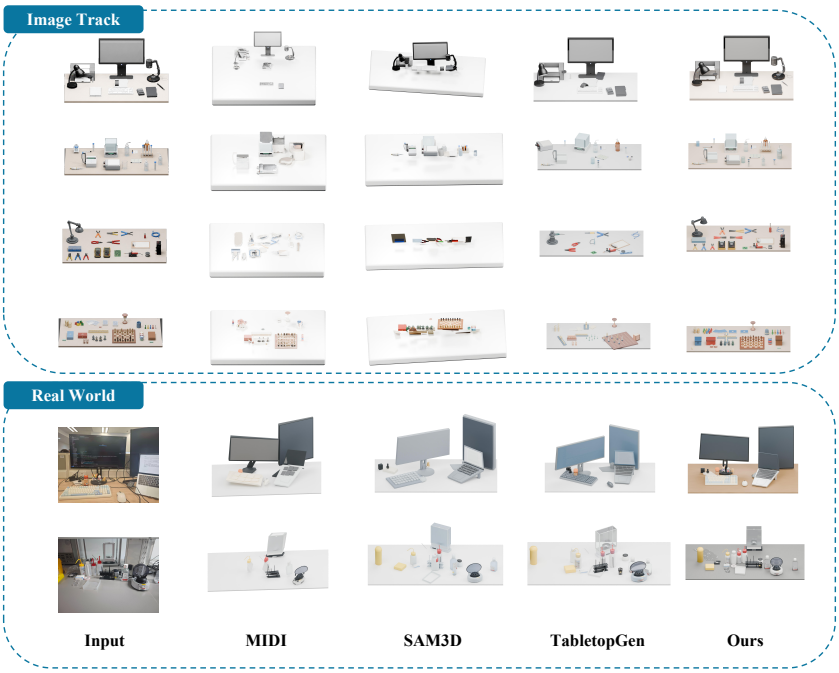

Figure 13: **Additional image-conditioned comparisons.** Image-track comparisons and reconstruction illustrations for real office and laboratory photographs.



Figure 14: **Additional image-conditioned tabletop scene reconstruction results.** Rows show bathroom, dining, dressing, and kitchen settings. Columns show the input, MIDI, SAM3D, TabletopGen, and TACER (Ours), following the order in the upper block of Figure 5.

F.3 Qualitative Interaction Sequences

Figures 15 and 16 contrast TACER successes with baseline failures in rigid and articulated tasks. The examples show successful lifting, placement, and joint motion in our scenes, while the baseline attempts fail to achieve the task goals. Articulated comparisons use TabletopGen + Articraft. Evaluation criteria follow Appendix D.

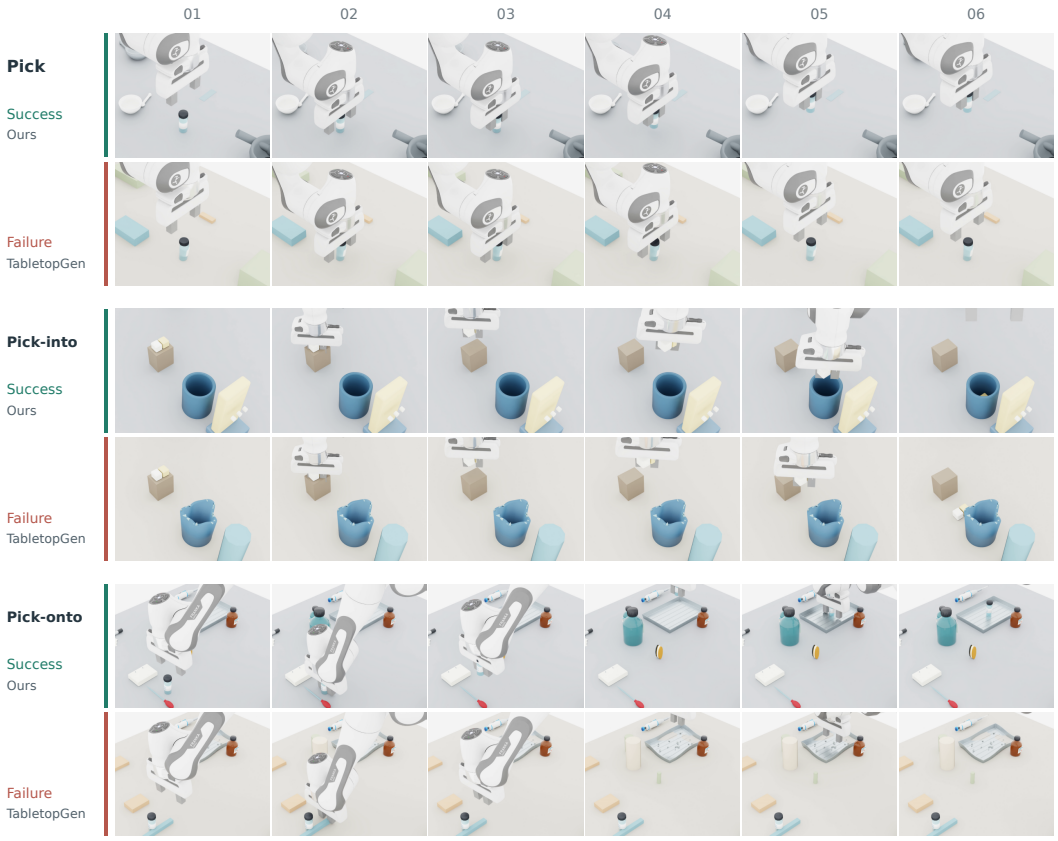

Figure 15: **Rigid-object execution.** Pick, Pick-into, and Pick-onto: TACER successes (top) and TabletopGen failures (bottom) per task. Frames progress from left to right.

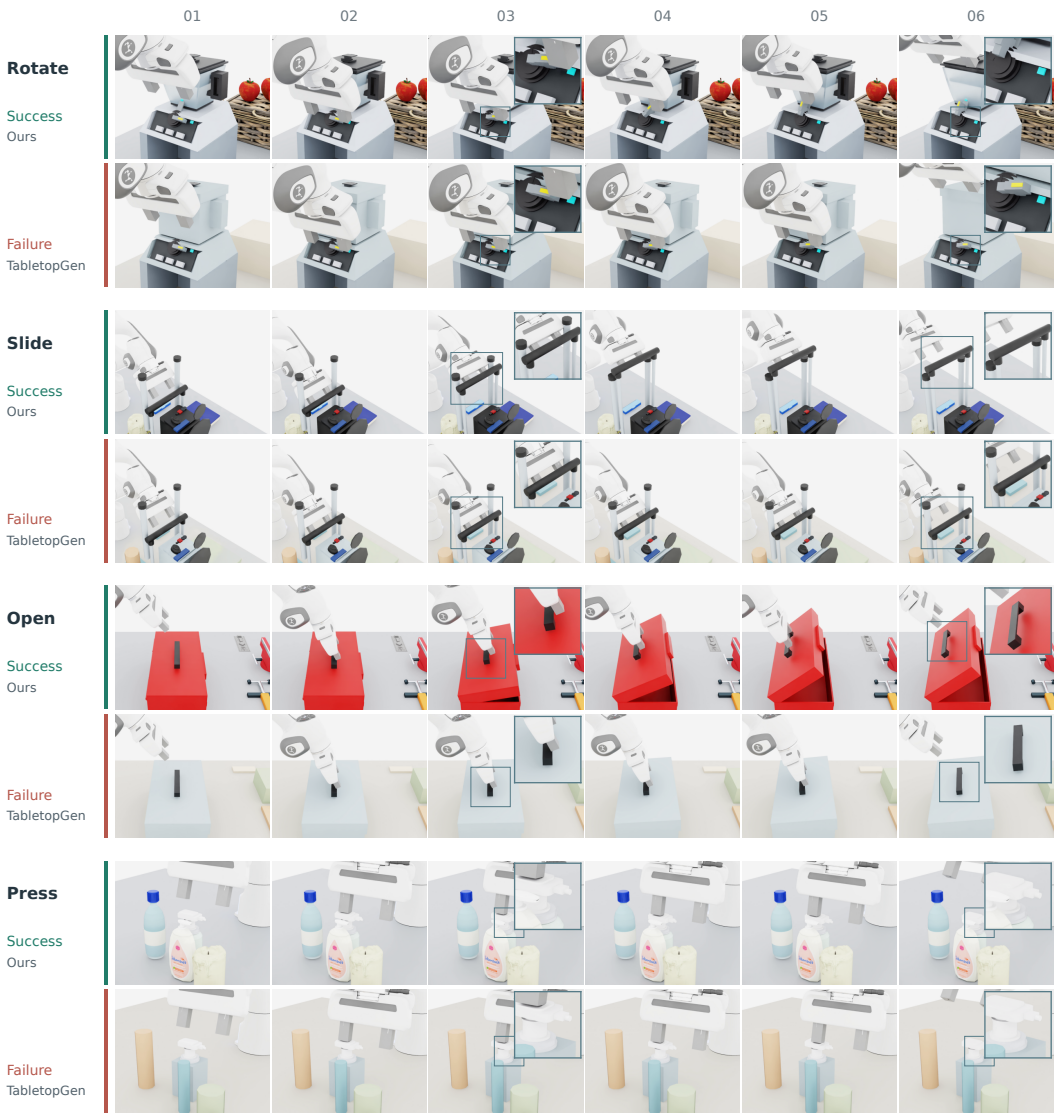

Figure 16: **Articulated execution.** Rotate, Slide, Open, and Press: TACER successes (top) and TabletopGen + Articraft failures (bottom) per task. Frames progress from left to right; insets highlight the interacting parts.