TOPCODE: DISENTANGLING VISUAL EVIDENCE AND PHYSICAL DEPENDENCIES FOR EXECUTABLE 3D SCENE SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating executable 3D indoor scenes from top-down references supports interior design and embodied intelligence. However, existing methods struggle to reconcile visual fidelity with physical completeness while preserving unrelated content during revision. We propose **TopCode**, an agent harness that translates **Top**-down images into executable scene **Code** by disentangling physical dependencies from visual evidence. Scene Memory preserves accepted attributes and their evidence in DeltaTree, while a separate Coupling Graph captures physical dependencies. Incremental scene compilation in Blender completes unobserved geometry and articulation, accepting updates only after visual and execution checks pass. When checks fail, counterexample-guided revision uses dependencies and hierarchical pose composition to localize changes while preserving accepted values with valid visual support. Across 100 top-down references and 120 local editing tasks, TopCode improves reference consistency, scene validity, and preservation of unrelated content over evaluated baselines. Generated scenes also improve navigation through post-training and support simulated manipulation, demonstrating their utility for embodied tasks.

1 INTRODUCTION

The automated synthesis of executable 3D indoor scenes supports applications spanning interior design, virtual reality (Feng et al., 2025; Yu et al., 2025), and embodied intelligence (Nasiriany et al., 2024; Li et al., 2026). Across these applications, useful generated scenes must combine physical validity and interactivity (Li et al., 2021; Chen et al., 2024b) with fidelity to a target spatial configuration (Cao et al., 2026). In this setting, top-down views, whether captured, reconstructed, or generated, provide dense visual specifications that encode both the overall layout and individual object details. As shown in Fig. 1(a), these specifications make faithful 3D realization challenging, as constructing an executable scene requires completing unobserved structure while preserving the specified spatial intent (Zhao et al., 2025; Liu et al., 2025).

Recent approaches to 3D scene synthesis primarily use textual descriptions or reference images to specify scene content and layout, as shown in Fig. 1(b). Although text-driven methods provide flexible semantic control, textual descriptions alone may not fully specify the detailed object arrangements depicted in a reference image (Höllein et al., 2023; Lin & Mu, 2024; Yang et al., 2024). Image-to-3D methods recover detailed geometry and appearance from visual observations, but their outputs often lack the part geometry and joint definitions needed for articulated interaction (Huang et al., 2025; Wu et al., 2025). Recent advances in the coding capabilities of large pretrained foundation models have enabled scene generation through executable Blender programs, offering explicit control over scene structure (Hu et al., 2024; Huang et al., 2024). However, this programmatic control places the burden of maintaining both visual and physical consistency throughout scene construction on foundation models (Yang et al., 2026).

Our key observation is that scene completion should disentangle the roles of visual evidence and physical dependencies: the former determines which attributes must be constrained by the reference, while the latter guides joint inference over coupled attributes to complete unobserved structure, as shown in Fig. 1(c). Foundation models face three challenges in completing executable scenes from

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.

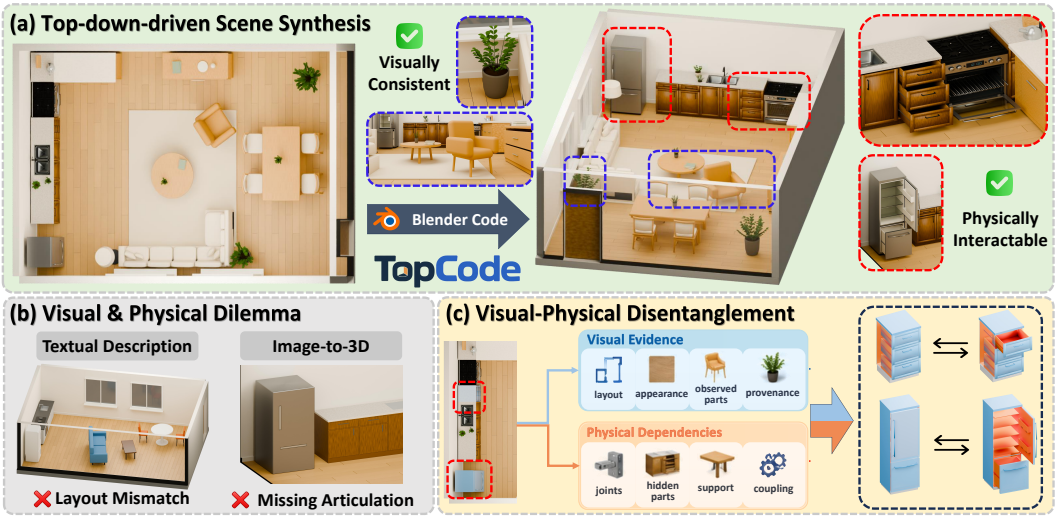

Figure 1: (a) Illustration of top-down-driven scene synthesis. (b) Layout mismatch in text-driven generation and missing articulation in image-to-3D generation. (c) TopCode Disentangles visual evidence and physical dependencies.

top-down references under these constraints. First, foundation models can omit recorded attributes or their supporting evidence when summarizing context during construction and revision, obscuring which attributes are supported by the reference (Wu et al., 2024). Second, an emphasis on matching visible geometry and appearance can lead foundation models to omit internal parts and joints needed for interaction (Qiu et al., 2025). Third, repairing execution failures may require coordinated changes across related entities, but foundation models can make broader changes that disrupt unrelated scene content (Noh et al., 2026). These challenges motivate an agent harness that translates consistency requirements into explicit controls on the scene generation process, reducing reliance on foundation models' reasoning alone.

We propose **TopCode**, an agent harness that turns **Top**-down references into executable 3D scene **Code** through incremental construction and local revision. Scene Memory disentangles physical dependencies from visual evidence while preserving accepted attributes, their provenance, and supporting evidence across updates. DeltaTree organizes these records hierarchically for local updates, while the Coupling Graph separately records physical dependencies among attributes to guide completion. Using this memory, scene compilation incrementally completes unobserved geometry and articulation in Blender, accepting updates only after visual and execution checks pass. When a check fails, a counterexample-guided revision algorithm combines attribute dependencies with hierarchical pose composition, excluding shared rigid motions for checks that depend only on relative pose. Provenance and evidence review further restrict the write scope, keeping accepted values with valid visual support fixed. Through this closed loop, TopCode turns foundation models' coding capabilities into a controllable process for constructing consistent environments that support embodied learning and interaction.

To assess the contribution of the harness, we evaluate TopCode's ability to construct scenes that match visual references, satisfy physical constraints, and support local editing. Our evaluation covers 100 shared top-down references and 120 local editing tasks, alongside navigation post-training and simulated manipulation using the generated scenes. Through extensive experiments, we show that TopCode improves reference consistency and scene validity while completing local edits with better preservation of unrelated content, and that the generated scenes improve navigation through post-training and support physical interaction.

Contributions: (1) To the best of our knowledge, TopCode is the first agent harness to disentangle physical dependencies from visual evidence for image-conditioned room-level scene synthesis, completing unobserved physical structure under visual constraints. (2) We propose DeltaTree and the Coupling Graph to represent evidence-linked scene state and physical dependencies separately, supporting a revision algorithm that combines coupling closure with hierarchical localization to identify candidate changes. (3) We assess the harness through scene generation and local editing

comparisons, including controls with the same backbone and component ablations. Navigation post-training and physical interaction further evaluate the utility of the generated scenes for embodied tasks.

2 Related Work

3D Indoor Scene Synthesis methods are categorized into three groups based on their primary construction mechanism: procedural generation, learned layout prediction, and planning with foundation models (Wang et al., 2021; Yang et al., 2026). Procedural systems arrange objects according to explicit rules (Raistrick et al., 2024), while learned generators capture spatial regularities from 3D scene datasets (Paschalidou et al., 2021; Tang et al., 2024). Foundation models translate language instructions into object layouts (Feng et al., 2023; Yang et al., 2024). Since semantic plausibility alone does not ensure feasible placements, recent systems combine model reasoning with geometric optimization and action planning (Sun et al., 2025; Jiang et al., 2026; Kuang et al., 2026; Noh et al., 2026). Agentic frameworks such as SceneSmith (Pfaff et al., 2026) and SAGE (Xia et al., 2026) use visual and physical feedback to iteratively construct simulation-ready environments. Code-generating agents specify scene assets and their spatial arrangement in executable Blender programs (Hu et al., 2024). TopCode disentangles visual evidence from physical dependencies to control attribute updates as agents iteratively generate and revise scene programs from a reference image.

Image-Conditioned 3D Scene Generation uses geometric prediction, asset retrieval, and generative completion to infer 3D structure from partial visual observations (Huang et al., 2018; Izadinia et al., 2017; Yu et al., 2025). Geometric prediction recovers room layout and object geometry (Nie et al., 2020; Liu et al., 2022; Chen et al., 2024a), whereas retrieval matches observed objects to existing 3D assets (Wu et al., 2025). Generative approaches use learned object priors to complete occluded geometry, either per object (Ardelean et al., 2025; Shi et al., 2026; Qiu & Wu, 2026) or jointly across instances (Lin et al., 2026). To maintain spatial coherence, MIDI (Huang et al., 2025) models inter-instance relations, while 3D-Fixer (Yin et al., 2026b) anchors completed geometry to observed fragments. Beyond geometric reconstruction, image-based simulation pipelines recover movable parts and joints (Chen et al., 2024b; Zhao et al., 2025) or retrieve assets with compatible affordances for articulated interaction (Dai et al., 2024). During physical completion and revision, TopCode preserves accepted attribute values whose visual support remains valid.

3 Problem Formulation

Given a top-down reference image I and optional non-visual editing conditions c, the goal is to generate an executable scene program $P \in \mathcal{P}$, where $\mathcal{P}$ denotes the Blender program space. When no additional conditions are provided, $c = \varnothing$. Execution yields the scene $X(P) = \mathrm{Execute}(P) = (G_P, \theta_P)$, where G_P describes objects, parts, and their structural relations while θ_P specifies their geometry, poses, appearance, joints, and physical attributes.

The image provides local evidence $\mathcal{E}(I) = \{e_k\}_{k=1}^{K}$, each item constraining a subset of scene variables. Let $h_k(X, e_k, c) \in \{0, 1\}$ denote a visual consistency judgment, with 1 indicating agreement with the constraints derived from e_k that remain active under c. An explicit edit in c supersedes source constraints only for the requested changes, while the remaining visual constraints stay active. Predicates $\{g_m\}_{m=1}^{M}$ encode geometric, physical, and kinematic consistency conditions, supplemented by task-specific constraints from c. They include support under gravity and joint-constrained motion, with $g_m(X, c) = 1$ indicating that all applicable instances in X satisfy the corresponding check. Together, the visual and execution predicates define the feasible program set:

$$\mathcal{F}(I, c) = \{P \in \mathcal{P} \mid h_k(X(P), e_k, c) = 1 \; \forall k, \quad g_m(X(P), c) = 1 \; \forall m\}. \tag{1}$$

The task is constrained scene completion with multiple valid solutions, since a single view underdetermines hidden geometry and interactive properties.

Starting from P_0, each operation a_t completes or revises scene entities and their attributes:

$$P_{t+1} = \mathrm{Update}(P_t, a_t), \quad t = 0, \ldots, T-1, \qquad P_T \in \mathcal{F}(I, c). \tag{2}$$

Intermediate programs P_t represent partial scenes $X(P_t)$, while P_T must satisfy the preceding visual and execution conditions.

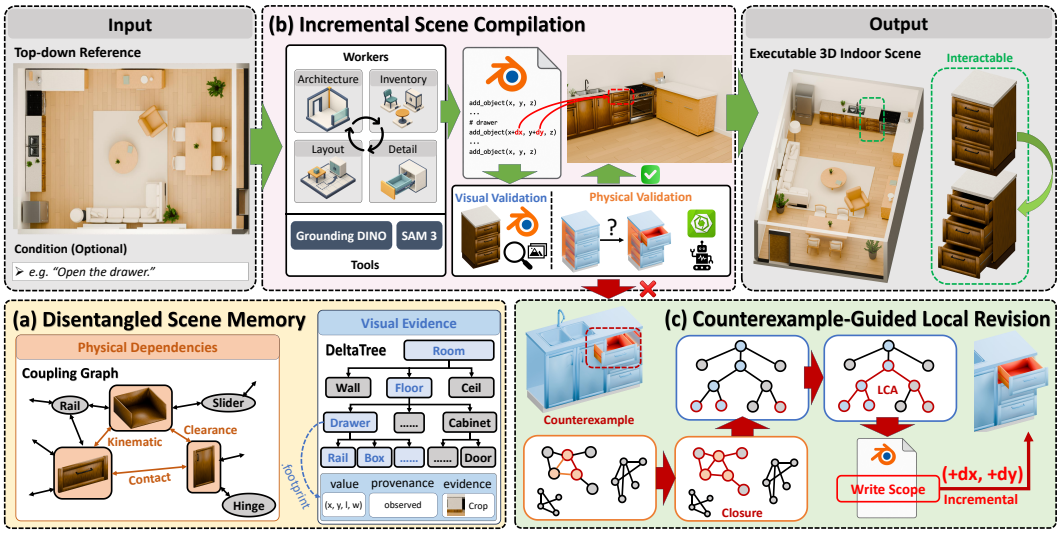

Figure 2: Overview of TopCode. (a) DeltaTree and the Coupling Graph disentangle visual evidence and physical dependencies. (b) Scene Compilation incrementally builds and validates scenes. (c) Coupling closure and LCA localization guide local revision.

4 METHOD

TopCode generates a scene program P through incremental construction and local revision, both guided and constrained by a shared scene memory. Disentangled Scene Memory comprises two complementary structures: DeltaTree binds visual evidence e_k to hierarchical scene records, while the Coupling Graph encodes physical dependencies through local instances of g_m. Scene Compilation translates these records into executable code and updates code affected by their changes. Local Revision uses the same memory to localize changes prompted by failed visual or execution conditions and passes the proposed deltas to Scene Compilation.

4.1 DISENTANGLED SCENE MEMORY

Scene Memory retains the scene structure and entity attributes needed to extend or revise P_t without losing accepted content. Within the room–object–part hierarchy, local additions and replacements must preserve unchanged attributes and their supporting evidence. Visual evidence e_k constrains projected positions and footprints but can admit different object heights, hidden geometries, and joint parameters. These choices are further coupled by g_m: support and contact constrain relative placement, while motion clearance requires articulated parts to move without interference with surrounding geometry. To disentangle these roles, DeltaTree maintains incremental scene records with their evidence bindings, while the Coupling Graph separately represents the conditions coupling their attributes within or across tree branches.

DeltaTree. At step t, DeltaTree organizes scene records and their supporting visual evidence in a parent–child hierarchy $\mathcal{T}_t$ of rooms, objects, parts, and joints with node set N_t. Nodes store geometry, pose, appearance, and applicable physical and joint parameters as attribute entries. For each entry v, the node retains its value, provenance, and references $\mathcal{B}_t(v) \subseteq \mathcal{E}(I)$ to any supporting image evidence. A non-root node n stores its pose $F_{p\to n} \in \mathrm{SE}(3)$ relative to its parent p, so their world poses satisfy $F_n = F_p F_{p\to n}$. Starting from the root's world pose, this recurrence composes the transforms along the path to each node. Entities with a common ancestor share the transform accumulated up to that ancestor, while their branch transforms determine their poses in its local frame. Local deltas record newly attached entities, rigid-pose increments for existing nodes, and assignments to other attributes, with provenance and evidence links for each change. A node's current pose incorporates its accepted increments through transform composition, while other attributes take their latest accepted assignments. Entries untouched by a delta retain their values, provenance, and evidence references.

Coupling Graph. The graph $G_t^{\mathrm{cp}} = (N_t, E_t^{\mathrm{cp}})$ shares DeltaTree's entity set and explicitly records physical dependencies among their attributes. Each edge r instantiates $g_{m(r)}$ as a support, contact,

kinematic, or motion-clearance check for its participating entities. Each relation directly references attributes of the participating entities through $\mathrm{vars}(r)$. A motion-clearance check combines part geometry, relative poses, and joint parameters to test for interference during prescribed motion. Links encode symmetric constraint coupling while retaining relation roles, such as the supporting and supported entities. Each instance retains its provenance and references $\mathcal{B}_t(r) \subseteq \mathcal{E}(I)$ to any supporting image evidence.

Both construction and revision use this shared memory, so accepted changes and their supporting evidence remain available to subsequent actions.

4.2 Incremental Scene Compilation

Starting from the source image I and initial program P_0, a programmatic controller coordinates four workers in dependency order to build the shared memory. *Architecture* establishes the room structure, *Inventory* identifies and records scene objects, *Layout* determines their placements and support relations, and *Detail* completes their parts and joints. Visual models identify entities and inspect local boundaries and visible structure through zoomed-in crops, with Grounding DINO (Liu et al., 2024) and SAM3 (Carion et al., 2026) providing localization candidates and masks as needed. The resulting masks, image crops, and footprint estimates guide construction. A rule-based relation builder instantiates checks corresponding to g_m from geometry and support or joint declarations, binding each check to its DeltaTree inputs. For each attribute assignment, a VLM determines whether its value is directly supported by visible source cues or inferred to complete unobserved structure, with this distinction recorded in its provenance. Accepted scene records enter DeltaTree and relation instances enter the Coupling Graph, each with provenance and any supporting image evidence.

Incremental Compilation. Before each update a_t, the controller assigns construction targets to workers and determines permitted record changes from each assignment's provenance. Accepted values with direct visual support remain fixed unless c explicitly authorizes changing them, while inferred values remain editable. Within this scope, the worker combines new observations with DeltaTree records, drawing on $\mathcal{B}_t(v)$ for attribute evidence and $\mathcal{B}_t(r)$ for relation evidence. Checks stored in the Coupling Graph constrain completion through support, contact, and motion-clearance conditions. For an existing node n retaining parent p, the worker proposes a rigid increment $\Delta F_{n,t} \in \mathrm{SE}(3)$ expressed in the node's current local frame, yielding the candidate pose

$$\widetilde{F}^{\,t+1}_{p\to n} = F^{\,t}_{p\to n} \Delta F_{n,t}. \tag{3}$$

Updating an ancestor propagates motion to its descendants through the hierarchy, while descendants receiving no pose update retain their parent-local transforms. After checking the proposed record changes against the permitted write scope, the controller updates the candidate tree and relation records, instantiating checks for added or revised structure. Scene Compilation updates the scene program to reflect the candidate records, yielding $\widetilde{P}_{t+1}$.

Validation and Acceptance. Blender executes $\widetilde{P}_{t+1}$ and renders candidate views, which a VLM verifier compares with the source evidence referenced by affected attributes and relations to evaluate h_k. Geometry tools perform geometric checks, while Isaac Sim runs the prescribed physical and articulated-motion protocols once the required scene structure is present. Applicability is determined by the task and candidate scene, including interactions discovered during execution. Geometry-based conflict discovery also examines entity pairs with no stored graph edge. The controller accepts records, relations, and P_{t+1} together only after applicable checks pass, otherwise retaining the previous state. Local Revision reuses this process, and final validation covers all h_k and g_m, including checks for required structure missing from the scene.

4.3 Counterexample-Guided Local Revision

Attribute-Coupling Closure. When validation detects a failure, the controller forms a counterexample $q = (f_q, o_q)$ from the failed visual or execution check f_q and the rendered view or tool output o_q that reveals the failure. Guided by o_q, an LLM traces f_q through evidence bindings $\mathcal{B}_t(v), \mathcal{B}_t(r)$ and relation references $\mathrm{vars}(r)$ to form S_q from implicated entity records and relevant records of existing parents that anchor missing structure. For assignments in S_q with recorded visual support, the VLM rechecks the cited image regions for incorrect evidence associations or misread visual cues.

The controller then forms $\mathcal{A}_q$ from currently editable records in Scene Memory, entries explicitly authorized for editing by c, and assignments whose recorded visual support is invalidated by this review. When structure is missing, proposed entities are provisionally attached to their specified parents before closure, with incident checks added to the Coupling Graph and new records included in S_q and $\mathcal{A}_q$. Starting from $U_0 = S_q \cap \mathcal{A}_q$, the controller follows physical dependencies in the Coupling Graph, selecting checks $\mathcal{R}_i$ that reference entries in U_i and adding their eligible inputs:

$$\mathcal{R}_i = \{r \in E_t^{\mathrm{cp}} \mid \mathrm{vars}(r) \cap U_i \neq \varnothing\}, \qquad U_{i+1} = U_i \cup \bigcup_{r \in \mathcal{R}_i} (\mathrm{vars}(r) \cap \mathcal{A}_q)\,. \tag{4}$$

With Scene Memory and $\mathcal{A}_q$ fixed at step t, the controller iterates over i until $U_{i+1} = U_i$, yielding the smallest closed set $U_\star$ containing U_0. Entries outside $\mathcal{A}_q$ remain fixed as boundary context. The entity set C_q collects the entities storing entries in S_q or inputs to the checks reached during closure, including those contributing fixed boundary values.

LCA Localization. To localize pose increments, we distinguish transforms shared by the entities in C_q from those acting along their individual branches. In DeltaTree, the controller locates the lowest common ancestor n_q and the paths connecting it to C_q:

$$n_q = \mathrm{LCA}_{\mathcal{T}_t}(C_q), \qquad \Pi_q = \bigcup_{n \in C_q} \mathrm{Path}_{\mathcal{T}_t}(n_q, n). \tag{5}$$

For nonempty C_q, these paths, including their endpoints, form the unique smallest connected subtree spanning C_q, since every such subtree must contain the entities' connecting paths. These paths locate the parent-local transforms through which pose increments affect the implicated entities, while other branches provide context. Algorithm 1 in Appendix A.3 combines queue-based closure with binary-lifting LCA queries and path construction.

Relative-Pose Invariance. For $n \in C_q$, let $F_{n_q \to n}$ denote the product of parent-local transforms along the path from n_q to n. The parent-local composition gives $F_n = F_{n_q} F_{n_q \to n}$, so the relative pose of $u, w \in C_q$ is

$$F_u^{-1} F_w = F_{n_q \to u}^{-1} F_{n_q \to w}. \tag{6}$$

With local geometry fixed, conditions depending solely on relative rigid pose are invariant to the common ancestor's rigid motion. The candidate set $\mathcal{V}_q^{\mathrm{cand}}$ includes $U_\star$ and local pose parameters along Π_q, excluding parameters of shared rigid motion for conditions satisfying this invariance. Geometry, joint, and structural parameters referenced by the target or incident checks remain candidates, including those at shared ancestors. For conditions involving image projection, world direction, or gravity, the candidate set also includes transforms along the affected entities' paths to the world frame.

Revision and Revalidation. A candidate delta's directly modified entries D_q must satisfy $D_q \subseteq \mathcal{W}_q = \mathcal{V}_q^{\mathrm{cand}} \cap \mathcal{A}_q$, where $\mathcal{W}_q$ is the permitted write scope. Within this scope, the revision agent proposes a local delta containing the required pose increments and other record changes for compilation and acceptance in Sec. 4.2. When visual support is invalidated, the proposed delta also corrects the affected evidence links and provenance based on the source review. Scene Compilation rechecks f_q alongside the visual and execution conditions affected by D_q or its derived geometric effects, including boundary relations. Global tests are rerun when their effects cannot be localized. A local revision is accepted when these checks pass, while pre-existing issues outside the affected scope may remain for subsequent revisions. Local revision stops when the applicable checks pass or its execution budget is exhausted.

5 Experiments

5.1 Experimental Setup

Benchmarks. We evaluate TopCode in three settings: (1) *Scene generation:* We convert the 100 room descriptions in SceneEval-100 (Tam et al., 2026) into top-down references using a fixed GPT image generator, then review and freeze the images for all image-conditioned methods. This setting evaluates reference consistency and scene validity. (2) *Local editing:* We construct 120 independent tasks on 30 scenes generated by TopCode from SceneEval-100 references, covering move/rotate, add,

Table 1: Scene generation on 100 shared top-down references. Obj. F1, IoU, and DINO denote Object F1, Layout IoU, and DINOv2 similarity. OPC evaluates static clearance at reliably annotated openings. Best and second-best values in each column are **bold** and underlined, respectively.

	Reference Alignment			Scene Validity					
Method	**Obj. F1↑**	**L. IoU↑**	**DINO↑**	**COL↓**	**SUP↑**	**NAV↑**	**ACC↑**	**OOB↓**	**OPC↓**
Qwen3.8-max	0.826	0.208	0.761	0.460	0.461	0.899	0.638	**0.008**	0.477
MIDI	0.318	0.035	0.421	0.519	N/A	N/A	N/A	N/A	N/A
Gen3DSR	0.784	0.027	0.248	0.822	0.032	0.418	0.145	0.802	0.875
3D-Fixer	0.251	0.084	0.452	**0.284**	0.059	0.964	**0.850**	0.661	0.442
VIGA	0.853	0.245	0.817	0.504	0.588	**0.996**	0.654	0.124	0.469
Ours (**TopCode**)	**0.932**	**0.461**	**0.899**	0.316	**0.800**	0.992	0.837	0.011	**0.269**

remove, and replace operations to assess editing success and preservation of unrelated content. (3) *Embodied applications:* We post-train ETPNav (An et al., 2024) on generated scenes and evaluate it on held-out HM3D (Ramakrishnan et al., 2021) episodes to assess their value as navigation training data. We also demonstrate physical interaction with a Franka gripper. Appendix D details reference preparation and evaluation protocols.

Compared Methods. For scene generation, we compare with five baselines. One-shot Qwen3.8-max produces a complete Blender program without feedback. MIDI (Huang et al., 2025) jointly generates multiple 3D instances through multi-instance diffusion. Gen3DSR (Ardelean et al., 2025) decomposes a scene into objects and reconstructs them using scene-level depth and semantic estimates. 3D-Fixer (Yin et al., 2026b) completes partial scene geometry in place through coarse-to-fine generation. VIGA (Yin et al., 2026a) iteratively revises scene programs through a code–render–inspect loop. For local editing, we compare with one-shot Qwen3.8-max and Edit-As-Act (Noh et al., 2026), a goal-regressive planning method that translates editing instructions into scene actions. Appendix C.1 further compares the complete text-to-scene pipeline with Holodeck (Yang et al., 2024), SceneSmith (Pfaff et al., 2026), and SAGE (Xia et al., 2026).

Implementation Details. We use Qwen3.8-max for all general-purpose vision-language components in the main comparisons. Each iterative method retains its own refinement and stopping rules. Within each task, all methods receive the same inputs and follow a common evaluation protocol. Evaluation renders use the same resolution with consistent camera and lighting settings within each comparison. Further implementation and evaluation details are provided in Appendix D.

5.2 Main Results

Scene Generation. Tab. 1 compares the six methods on the shared top-down references. We measure reference alignment with Object F1, Layout IoU, and DINOv2 similarity (Oquab et al., 2023), and scene validity with collisions (COL), support (SUP), navigability (NAV), accessibility (ACC), boundary violations (OOB), and opening clearance (OPC) (Tam et al., 2026). The results support three observations. (1) Compared with one-shot Qwen3.8-max, TopCode improves Object F1 and Layout IoU by 0.106 and 0.253, respectively, indicating better preservation of reference content and layout during scene completion with the same foundation model. (2) TopCode achieves the best scores on all three reference-alignment metrics, SUP, and OPC, and ranks second on the remaining scene-validity metrics. These joint results are consistent with our disentangled design: visual evidence constrains observed attributes to preserve reference alignment, while physical dependencies constrain scene completion to maintain scene validity. (3) Compared with VIGA using the same backbone, TopCode improves eight of nine metrics, with a slightly lower NAV score (0.992 vs. 0.996). These gains suggest that iterative visual feedback can benefit from explicit scene-state management and constrained revision. The controlled ablations in Section 5.3 examine how scene representation, local write scope, and execution validation contribute to these gains.

Local Editing. We evaluate instruction satisfaction, preservation of unrelated content, and geometric violations introduced by local edits. All methods start each task from the same initial scene, with each edit evaluated independently. Tab. 2 shows that TopCode achieves the highest Editing Success Rate (ESR, 91.67%) and Non-target Preservation Rate (NPR, 100.00%). TopCode also has the lowest

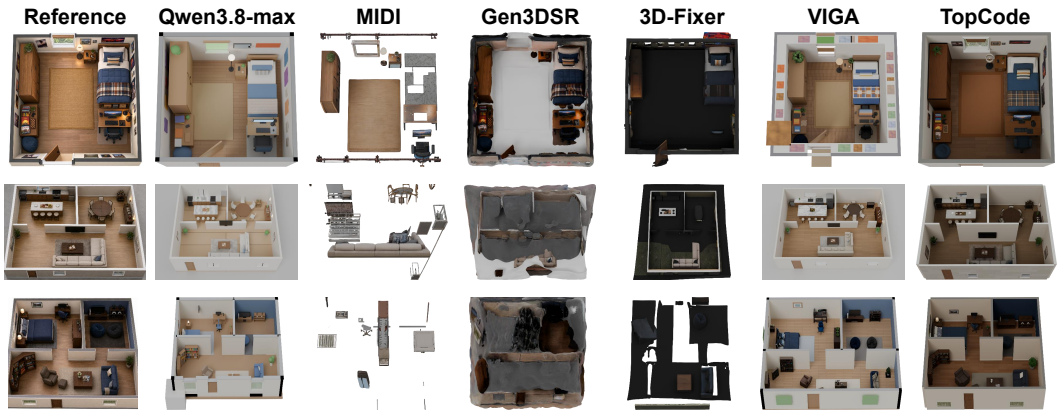

Figure 3: Qualitative scene generation on three top-down references. Each row shows a reference followed by outputs from one-shot Qwen3.8-max, MIDI, Gen3DSR, 3D-Fixer, VIGA, and TopCode. The examples compare furniture appearance and placement, room divisions, and the completeness of generated geometry.

rates of new collisions and support loss. These results support Local Revision: coupling closure coordinates related updates, LCA localization and the permitted write scope restrict changes to relevant entries, and revalidation checks affected physical constraints. Appendix D.4 provides task conditions and preservation checks, with consecutive edits evaluated in Appendix C.2.

Qualitative Comparison. Fig. 3 shows that TopCode preserves room divisions and furniture placement while producing detailed, coherent geometry. Baselines exhibit missing detail, fragmented structures, or boundary violations. Together with the scene-validity results in Tab. 1, these examples illustrate the complementary goals of visual preservation and physical completion. Human evaluation is reported in Appendix D.6.

Table 2: Local editing on 120 tasks. COL, SUP, and OOB report the percentages of tasks introducing collisions, support loss, and boundary violations, respectively.

	Editing Quality (%)		**Scene Validity** (%)		
Method	**ESR↑**	**NPR↑**	**COL↓**	**SUP↓**	**OOB↓**
Qwen3.8-max	65.83	80.83	21.67	<u>15.83</u>	5.83
Edit-As-Act	<u>79.17</u>	<u>95.00</u>	<u>10.00</u>	25.00	**1.67**
TopCode	**91.67**	**100.00**	**6.67**	**5.00**	<u>2.50</u>

5.3 Ablation Studies

Tab. 3 compares full TopCode with four variants on the same 100 references under the same backbone and token limit: (1) *w/o LWS* removes the local write scope, permitting changes to all currently editable records; (2) *w/o DT+LWS* additionally replaces DeltaTree with flat records and world-coordinate updates; (3) *w/o CG* replaces the Coupling Graph with LLM-inferred related entries; and (4) *w/o EV* removes execution-check feedback and the corresponding acceptance gates. Appendix D.5 specifies the replacements and retained functionality.

Table 3: Scene-generation ablations on 100 references. Obj. F1 and L. IoU denote Object F1 and Layout IoU.

	Reference Alignment		**Scene Validity**		
Variant	**Obj. F1↑**	**L. IoU↑**	**COL↓**	**SUP↑**	**OOB↓**
w/o LWS	0.861	0.412	0.376	0.768	0.019
w/o DT+LWS	0.848	0.339	<u>0.341</u>	0.765	0.026
w/o CG	<u>0.909</u>	<u>0.448</u>	0.452	0.729	0.081
w/o EV	0.893	0.442	0.380	<u>0.788</u>	<u>0.012</u>
Full **TopCode**	**0.932**	**0.461**	**0.316**	**0.800**	**0.011**

With evidence retained and local write scopes removed, *DeltaTree* improves Layout IoU by 0.073 over flat records. Hierarchical updates thus help preserve the layout specified by visual evidence. Replacing the *Coupling Graph* worsens all scene-validity metrics despite retaining visual evidence handling and validation, supporting explicit dependencies and coupling closure for physical completion.

Removing the *Local Write Scope* reduces reference alignment with both memory structures retained, showing the benefit of localizing dependency-guided revisions during physical completion. Disabling *Execution Validation* increases COL despite retaining visual revision. Together, these results support our disentangled design, combining evidence-linked hierarchical state for visual preservation with explicit dependencies for physical completion.

5.4 Embodied Applications

VLN Post-training. We compare ETPNav (An et al., 2024) post-training on source data alone and on matched mixtures with TopCode or Qwen scenes. Runs share the checkpoint and update budget; generated-data conditions use equal scene and episode counts. On held-out HM3D (Ramakrishnan et al., 2021) episodes (Tab. 4), TopCode improves SR by 7.55 percentage points over source-only post-training and 5.36 over the Qwen mixture, with the best SPL, nDTW, and NE. These gains support the navigation-training value of scenes generated under visual and physical constraints. Appendix D.7 provides the training and evaluation details.

Table 4: ETPNav on held-out HM3D episodes. None denotes the source checkpoint; "+" denotes a 50:50 mixture of source and generated episodes. NE is in meters. Best values are **bold**.

Post-training	SR↑	SPL↑	nDTW↑	NE↓
None	32.36	25.06	45.98	8.341
Source	31.14	26.52	47.45	8.695
+ Qwen	33.33	28.22	51.82	7.297
+ TopCode	**38.69**	**31.14**	**56.20**	**5.218**

Franka Interaction. We examine whether the generated scene structures support contact-driven manipulation in Isaac Sim. Fig. 4 shows three examples from the SceneEval outputs: pulling a drawer, opening a cabinet door, and grasping and lifting a small object. For each task, we position a fixed-base Franka Panda near the target and specify the grasp pose and end-effector waypoints from the known object pose. Inverse kinematics (IK) converts these waypoints into joint configurations, which are interpolated and tracked with joint-position control. After the gripper closes, physical contact drives drawer translation, door rotation, or object lifting. The drawer and door motions rely on separable parts with sufficient clearance, illustrating how physical dependencies complement visual evidence when completing interactive scenes. Appendix D.8 provides the robot configuration and execution procedure.

6 Conclusion

We present TopCode, an agent harness that disentangles visual evidence from physical dependencies for executable 3D indoor scene synthesis from top-down references. With DeltaTree retaining scene records and their evidence and the Coupling Graph encoding physical dependencies, incremental compilation and local revision complete unobserved geometry and articulation under visual constraints. Scene generation and local editing experiments show gains in reference alignment, scene validity, and preservation of unrelated content, while navigation post-training and simulated manipulation demonstrate the utility of the generated scenes. These findings support visual–physical disentanglement as a design principle for preserving reference content while completing executable, interactive environments. Future work will explore extending this disentangled design beyond top-down references.

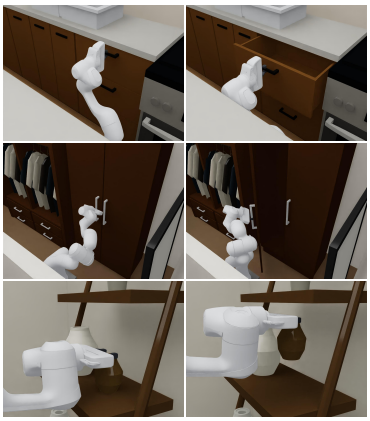

Figure 4: Franka interaction. Rows: drawer opening, door opening, and object lifting. Left: contact; right: completed motion.

Limitations. First, repeated model calls, rendering, and simulation introduce computational overhead during scene construction and revision. More selective model calls and cheaper intermediate checks could reduce this cost. Second, TopCode currently assumes a top-down reference with broad coverage of the room layout. Extending this design to a single image captured at an arbitrary viewpoint will require handling occlusion and depth ambiguity while completing the entire room as an executable, interactive scene.

AI USE STATEMENT

Generative models are used for reference-image generation, scene synthesis, and VLM-based attribute evaluation, as described in the methodology and experimental protocols. We also use LLM as an assistant to help us polish our writing. All AI-assisted revisions were reviewed, revised, and verified by the authors. The authors take full responsibility for the final content of this work. We are grateful for the convenience that LLMs has brought to research writing and call for the rational utilization.

ETHICS STATEMENT

This study relies on publicly available scene synthesis benchmarks (e.g., SceneEval) and tools (e.g., SAM3), and uses them in compliance with their respective licenses and terms of use. It includes an anonymous questionnaire evaluation with 15 participants (Appendix D.6). Participants read the study instructions and indicate their agreement before starting. Participation is voluntary, participants may stop at any time, and the questionnaire uses anonymous participant IDs without requesting names. The authors report no financial, professional, or personal conflicts of interest associated with this work. We have reviewed and complied with the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

To support reproducibility, the main text and appendix describe the methodology, algorithms, experimental procedures, and implementation configurations. In particular, Appendix A details Scene Memory records and condition dependencies, with pseudocode for coupling closure and LCA localization; Appendix B presents the worker designs and prompts; Sec. 5.1 outlines the experimental setup; and Appendix D provides further details on reference preparation, baseline and ablation configurations, evaluation metrics, editing protocols, human evaluation, and embodied applications.

REFERENCES

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7):5130–5145, 2024.

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018a.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 3674–3683. IEEE, 2018b.

Andreea Ardelean, Mert Özer, and Bernhard Egger. Gen3dsr: Generalizable 3d scene reconstruction via divide and conquer from a single view. In *2025 International Conference on 3D Vision (3DV)*, pp. 616–626. IEEE, 2025.

Jianbao Cao, Zhangrui Zhao, Bohan Feng, Zixuan Hu, Rui Li, Haiyuan Wan, Chenxi Li, Jingyuan Li, Wenzhe Cai, Lei Bai, et al. Labbuilder: Protocol-grounded 3d layout generation for interactable and safe laboratory. *arXiv preprint arXiv:2605.02288*, 2026.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Yixin Chen, Junfeng Ni, Nan Jiang, Yaowei Zhang, Yixin Zhu, and Siyuan Huang. Single-view 3d scene reconstruction with high-fidelity shape and texture. In *2024 International Conference on 3D Vision (3DV)*, pp. 1456–1467. IEEE, 2024a.

Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024b.

Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Automated creation of digital cousins for robust policy learning. *arXiv preprint arXiv:2410.07408*, 2024.

Weitao Feng, Hang Zhou, Jing Liao, Li Cheng, and Wenbo Zhou. Casagpt: cuboid arrangement and scene assembly for interior design. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29173–29182. IEEE, 2025.

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. *Advances in neural information processing systems*, 36:18225–18250, 2023.

Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7875–7886. IEEE, 2023.

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 scenes as blender code. In *ICML*, pp. 19252–19282, 2024.

Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. In *European Conference on Computer Vision*, pp. 297–314. Springer, 2024.

Siyuan Huang, Siyuan Qi, Yinxue Xiao, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23646–23657. IEEE, 2025.

Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *arXiv preprint arXiv:1907.05446*, 2019.

Hamid Izadinia, Qi Shan, and Steven M Seitz. Im2cad. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2422–2431. IEEE, 2017.

Haiyan Jiang, Deyu Zhang, Dongdong Weng, Weitao Song, and Henry Been-Lirn Duh. Hog-layout: Hierarchical 3d scene generation, optimization and editing via vision-language models. *arXiv preprint arXiv:2604.10772*, 2026.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pp. 104–120. Springer, 2020.

Zhengfei Kuang, Rui Lin, Long Zhao, Gordon Wetzstein, Saining Xie, and Sanghyun Woo. Vulcan: Tool-augmented multi agents for iterative 3d object arrangement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23763–23773, 2026.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

Rui Li, Zixuan Hu, Wenxi Qu, Jinouwen Zhang, Zhenfei Yin, Sha Zhang, Xuantuo Huang, Hanqing Wang, Tai Wang, Jiangmiao Pang, et al. Labutopia: High-fidelity simulation and hierarchical benchmark for scientific embodied agents. *Advances in Neural Information Processing Systems*, 38, 2026.

Chenguo Lin and Yadong Mu. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. In *International Conference on Learning Representations*, volume 2024, pp. 25687–25718, 2024.

Yuchen Lin, Chenguo Lin, Panwang Pan, Honglei Yan, Feng Yiqiang, Yadong Mu, and Katerina Fragkiadaki. Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers. *Advances in Neural Information Processing Systems*, 38:35387–35415, 2026.

Haolin Liu, Yujian Zheng, Guanying Chen, Shuguang Cui, and Xiaoguang Han. Towards high-fidelity single-view holistic reconstruction of indoor scenes. In *European Conference on Computer Vision*, pp. 429–446. Springer, 2022.

Jiayi Liu, Denys Iliash, Angel Chang, Manolis Savva, and Ali Mahdavi Amiri. Singapo: Single image controlled generation of articulated parts in objects. In *International Conference on Learning Representations*, volume 2025, pp. 97511–97532, 2025.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pp. 38–55. Springer, 2024.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 52–61. IEEE, 2020.

Seongrae Noh, SeungWon Seo, Gyeong-Moon Park, and HyeongYeop Kang. Edit-as-act: Goal-regressive planning for open-vocabulary 3d indoor scene editing. *arXiv preprint arXiv:2603.17583*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in neural information processing systems*, 34:12013–12026, 2021.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. Scenesmith: Agentic generation of simulation-ready indoor scenes. In *Forty-third International Conference on Machine Learning*, 2026.

Xiaowen Qiu, Jincheng Yang, Yian Wang, Zhehuan Chen, Yufei Wang, Tsun-Hsuan Wang, Zhou Xian, and Chuang Gan. Articulate anymesh: Open-vocabulary 3d articulated objects modeling. *arXiv preprint arXiv:2502.02590*, 2025.

Zidian Qiu and Ancong Wu. Pano3dcomposer: Feed-forward compositional 3d scene generation from single panoramic image. *arXiv preprint arXiv:2603.05908*, 2026.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, et al. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21783–21794. IEEE, 2024.

Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9338–9346. IEEE, 2019.

Yukai Shi, Weiyu Li, Zihao Wang, Hongyang Li, Xingyu Chen, Ping Tan, and Lei Zhang. Scenemaker: Open-set 3d scene generation with decoupled de-occlusion and pose estimation model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27146–27156, 2026.

Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12078–12088, 2025.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29469–29478. IEEE, 2025.

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 *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 7355–7365. IEEE, 2026.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20507–20518. IEEE, 2024.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020.

Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *2021 International conference on 3D vision (3DV)*, pp. 106–115. IEEE, 2021.

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. In *European Conference on Computer Vision*, pp. 96–114. Springer, 2026.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Qirui Wu, Denys Iliash, Daniel Ritchie, Manolis Savva, and Angel X Chang. Diorama: Unleashing zero-shot single-view 3d indoor scene modeling. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8896–8907. IEEE, 2025.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, et al. Sage: Scalable agentic 3d scene generation for embodied ai. *arXiv preprint arXiv:2602.10116*, 2026.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent. *Advances in neural information processing systems*, 38:140319–140351, 2026.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16277–16287. IEEE, 2024.

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. In *European Conference on Computer Vision*, pp. 557–575. Springer, 2026a.

Ze-Xin Yin, Liu Liu, Xinjie Wang, Wei Sui, Zhizhong Su, Jian Yang, and Jin Xie. 3d-fixer: Coarse-to-fine in-place completion for 3d scenes from a single image. *arXiv preprint arXiv:2604.04406*, 2026b.

Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T Freeman, and Jiajun Wu. Wonderworld: Interactive 3d scene generation from a single image. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5916–5926. IEEE, 2025.

Mandi Zhao, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. In *International Conference on Learning Representations*, volume 2025, pp. 668–686, 2025.

TopCode: Disentangling Visual Evidence and Physical Dependencies for Executable 3D Scene Synthesis

Appendix

The appendix is organized as follows:

- Appendix A details Scene Memory records, condition instantiation, and the local revision algorithm.
- Appendix B describes the workers and their prompts for scene construction, verification, diagnosis, and revision.
- Appendix C presents additional experiments on text-driven scene generation and sequential local editing.
- Appendix D provides experimental settings and evaluation protocols, including the human evaluation questionnaire and embodied applications.

A Method and Algorithm Details

A.1 Scene Memory Records

Tab. 5 and Tab. 6 specify the logical record fields and illustrate their use for a cabinet with an articulated drawer.

Table 5: Logical fields of Scene Memory records. An attribute address $v = (n, a)$ identifies entity n and field path a. Provenance and evidence are stored per entry.

Record	Field(s)	Stored content
Entity	`id`, `kind`	Entity identifier and type: room, object, part, or joint.
	`parent_id`	Parent reference with assignment metadata; empty for the root.
	`fields`	Map from field paths to attribute records.
Attribute	`value`	Scalar, vector, parent-local transform, geometry/material description, or applicable physical/joint parameter.
	`provenance`	Source observation or inference behind the assignment, and instruction authorization when applicable.
	`evidence_refs`	Supporting source-image references $\mathcal{B}_t(v)$; possibly empty for inferred entries.
Evidence binding	`reference`	Source image, crop, or mask reference with its region locator when applicable.
	`targets`	Attribute addresses or relation identifiers supported by that observation.
Relation	`id`, `type`	Relation identifier and condition type.
	`participants`	Mapping from participant roles to entity identifiers.
	`vars`	Explicit attribute addresses (n, a) consumed by this check.
	`provenance`, `evidence_refs`	Origin of the relation and any supporting observations $\mathcal{B}_t(r)$.
Local delta	`additions`	New entity records with specified parent assignments.
	`pose_increments`	Existing target entities and rigid increments $\Delta F_{n,t}$.
	`assignments`	Target attribute addresses and proposed values, with provenance and evidence for each change.

Table 6: Illustrative Scene Memory records and a local delta. Attribute entries are shown as (value, provenance basis, evidence references). The delta updates only the drawer geometry.

Record	Fields and example values
Cabinet n_c	`kind`: object; `parent_id`: room `footprint`: $(B_c,\ \text{observed},\ \{e_c\})$ `shell`: $((h_c, t_c),\ \text{inferred},\ \varnothing)$
Drawer n_d	`kind`: part; `parent_id`: n_c `geometry`: $(G_d,\ \text{inferred},\ \varnothing)$ `pose`: $(F_{n_c \to n_d},\ \text{inferred},\ \varnothing)$
Joint n_j	`kind`: joint; `parent_id`: n_c `type`: $(\text{prismatic},\ \text{inferred},\ \varnothing)$ `origin`: $(O_j,\ \text{inferred},\ \varnothing)$ `axis`: $(a_j,\ \text{inferred},\ \varnothing)$ `limits`: $([0, \ell_j],\ \text{inferred},\ \varnothing)$
Evidence e_c	`reference`: source crop of the cabinet `targets`: $\{(n_c, \texttt{footprint})\}$
Relation r	`type`: motion clearance `participants`: moving part n_d, surrounding body n_c, joint n_j `vars`: $(n_c, \texttt{footprint})$, $(n_c, \texttt{shell})$, $(n_d, \texttt{geometry})$, $(n_d, \texttt{pose})$, $(n_j, \texttt{type})$, $(n_j, \texttt{origin})$, $(n_j, \texttt{axis})$, $(n_j, \texttt{limits})$ `provenance`: instantiated from the completed structure `evidence_refs`: $\varnothing$
Local delta	`assignments`: $(n_d, \texttt{geometry}) \mapsto (G'_d,\ \text{inferred},\ \varnothing)$ `additions`: $\varnothing$; `pose_increments`: $\varnothing$

Symbolic values: B_c, cabinet footprint; (h_c, t_c), height and wall thickness; G_d, G'_d, drawer geometry before and after the update; $O_j, a_j, [0, \ell_j]$, joint origin, axis, and travel range. Omitted entries retain their accepted values and metadata.

A.2 Conditions and Relation Instantiation

For each condition g_m, a template defines its participants, attribute inputs, applicability, and test, including tolerances and simulation protocols. Let $\Omega_m(X, c)$ enumerate applicable instances and $\phi_m(X, z, c)$ test instance z:

$$g_m(X, c) = \bigwedge_{z \in \Omega_m(X,c)} \phi_m(X, z, c). \tag{7}$$

Each instance binds its inputs through $\mathrm{vars}(r)$ (Tab. 7); DeltaTree composes their geometry and poses.

Support and joint declarations, together with geometric conflict discovery, determine the instances. Structural changes refresh these instances and their bindings before closure; missing entities and incident checks are provisionally attached to existing parents. Required but absent structure remains subject to validation, and nonlocalizable conditions are checked globally.

Table 7: Condition families and their attribute dependencies. Each instance binds the records required by its check; the applicable contact and simulation protocols determine the test.

Condition	Participating inputs	Check
Support	Supported and supporting entities; geometry, poses, and physical parameters required by the protocol.	Support under the prescribed gravity and contact conditions.
Contact	Interacting entities; geometry, relative configuration, and designated contact roles.	Intended contact and absence of unintended penetration under the contact policy.
Kinematics	Connected parts and joints; geometry, poses, joint origins, axes, and limits.	Consistency of the articulated configuration with the joint definition.
Motion clearance	Moving parts and surrounding entities; geometry, relative poses, and joint parameters.	Interference during the prescribed articulated motion.

A.3 Coupling Closure and LCA Localization

In Algorithm 1, $\mathrm{owner}(v)$ denotes the entity storing entry v, and $\mathrm{up}[n, j]$ stores the precomputed 2^j-th ancestor of n in the current tree. We use $N = |N_t|$ and $J = \lfloor \log_2 N \rfloor$.

With fixed relations and $\mathcal{A}_q$, the closure grows monotonically within a finite set. Every closed superset of $S_q \cap \mathcal{A}_q$ contains each iterate, so Algorithm 1 returns the least such set. Binary lifting uses $O(N \log N)$ preprocessing time and space and $O(\log N)$ time per pairwise query. Folding queries over C_q costs $O(|C_q| \log N)$, with additional path-construction time proportional to the total traversed path length.

B Worker Designs and Prompts

The templates below accompany runtime images, records, identifiers, and output schemas for the workers in Sec. 4.2.

B.1 Controller and Shared Context

The controller runs Architecture, Inventory, Layout, and Detail in dependency order, importing validated stage-specific JSON into Scene Memory and using typed deltas for revision. It handles scheduling, coordinate conversion, permissions, compilation, and candidate promotion.

Initial requests contain upstream plans and visual evidence. Revision requests add the issue, actions, editable targets, ancestors, descendants, relation neighbors, world transforms, and recent feedback. Field metadata contain the value, responsible worker, provenance, runtime status, and evidence references. Provenance distinguishes observed and inferred values; runtime status controls current writability without overriding visual protection. The eligible set $\mathcal{A}_q$ includes editable entries, explicitly authorized changes in c, and entries whose visual support is invalidated by review. Other visually supported values remain fixed; derived fields are recomputed from their sources.

Shared Coordinates and Evidence Rules

The source is an elevated top-down room image. In the source image, orientation-preserving crops, and top-down renders, image top is room north, bottom is south, left is west, and right is east. Normalized image coordinates increase rightward in x and downward in y. Room coordinates use $+X$ east, $+Y$ north, and $+Z$ upward. Use cardinal directions for horizontal room locations and top/bottom for vertical height. Do not infer cardinal directions from screen positions in an oblique render. Follow the explicit local-axis annotations in a canonical object view.

For each proposed attribute assignment, report its value, provenance, and supporting evidence identifiers. Mark an assignment as observed only when the cited source cues directly support that value; mark completion of hidden geometry or joint parameters as inferred. Attach evidence to the specific supported entries. Preserve the evidence and provenance of unchanged entries. Record explicit edit authorization separately from source observations.

Algorithm 1 Coupling Closure and LCA Localization

Input: DeltaTree $\mathcal{T}_t$, Coupling Graph G_t^{cp}
Nonempty diagnosed entries S_q, eligible entries $\mathcal{A}_q$
Depths d and ancestor table $\mathrm{up}[\cdot, 0{:}J]$ for current $\mathcal{T}_t$
Output: Closed entries $U_\star$, entities C_q, LCA n_q, path set Π_q

$U \leftarrow S_q \cap \mathcal{A}_q$
$Q \leftarrow \mathrm{Queue}(U)$
$R \leftarrow \varnothing$
$C_q \leftarrow \{\mathrm{owner}(v) : v \in S_q\}$
while Q is not empty **do**
  $u \leftarrow Q.\mathrm{pop}()$
  for each $r \in E_t^{\mathrm{cp}}$ with $u \in \mathrm{vars}(r)$ and $r \notin R$ **do**
    $R \leftarrow R \cup \{r\}$
    $C_q \leftarrow C_q \cup \{\mathrm{owner}(v) : v \in \mathrm{vars}(r)\}$
    for each $v \in (\mathrm{vars}(r) \cap \mathcal{A}_q) \setminus U$ **do**
      $U \leftarrow U \cup \{v\}$
      $Q.\mathrm{push}(v)$
$U_\star \leftarrow U$
Choose $c_0 \in C_q$; $n_q \leftarrow c_0$
for each $n \in C_q \setminus \{c_0\}$ **do**
  $n_q \leftarrow \textsc{BinaryLCA}(n_q, n)$
$\Pi_q \leftarrow \{n_q\}$
for each $n \in C_q$ **do**
  $x \leftarrow n$
  while $x \neq n_q$ **do**
    $\Pi_q \leftarrow \Pi_q \cup \{x\}$
    $x \leftarrow \mathrm{parent}_{\mathcal{T}_t}(x)$
return $U_\star, C_q, n_q, \Pi_q$

function $\textsc{BinaryLCA}(u, v)$
  if $d[u] < d[v]$ **then** swap u, v
  for $j = J, J-1, \ldots, 0$ **do**
    if $d[u] - d[v] \geq 2^j$ **then**
      $u \leftarrow \mathrm{up}[u, j]$
  if $u = v$ **then return** u
  for $j = J, J-1, \ldots, 0$ **do**
    if $\mathrm{up}[u, j] \neq \mathrm{up}[v, j]$ **then**
      $(u, v) \leftarrow (\mathrm{up}[u, j], \mathrm{up}[v, j])$
  return $\mathrm{up}[u, 0]$

The clean source provides visual evidence; boxes, masks, descriptions, and retrieved structures support localization or interpretation under each worker's instructions.

B.2 Architecture Worker

Architecture uses the source image, wall candidates, opening masks, and scale references to produce metric room bounds, walls, and openings with evidence identifiers. Openings retain their host wall and dimensions; the compiler constructs the remaining wall geometry and specified headers.

Architecture: Wall Grounding and Opening Recognition

Inspect visible physical wall caps and architectural openings in the clean source image. Return JSON only. For walls, propose one tight normalized bounding box per maximal straight, visibly continuous wall cap, with its horizontal or vertical orientation, confidence, and visual reason. Split at turns, physical openings, visible gaps, major occlusions, and uncertain intervals. Do not identify vertical wall faces, shadows, floor bands, cabinet tops, or image borders as wall caps.

Architecture: Wall Grounding and Opening Recognition (continued)

Identify each visible door, doorway, and window. A door requires evidence of a physical assembly; a doorway requires a localized passable opening with identifiable bounds; a window requires glazing or frame evidence. Do not turn a broad open-plan connection into a doorway. Include the complete door assembly and corresponding opening when refining its crop. Mark a doorway header only when the lintel is visible. Preserve the target identifier during crop refinement. Return no region when the intended opening cannot be uniquely identified. Report uncertainty rather than inventing a box. Do not output metric planar coordinates.

Opening records contain type, identifier, normalized box, segmentation guidance, confidence, and target description. Wall review permits supplied-grounding replacement, bounded normal shifts, and endpoint snapping to supplied walls, validated before compilation.

Architecture: Metric Estimation

Given the source image and mask summary, estimate size intervals for the supplied scale references, wall height, opening height, and sill height. Match the supplied identifiers. Express familiar object dimensions as intervals with confidence and a brief rationale. Use overall room dimensions as a low-confidence fallback when no reliable reference is available. Preserve masks, wall lines, opening attachments, and planar coordinates.

B.3 Inventory Worker

Inventory records contain `id`, `name`, `category` (`major` or `minor`), `placement_type` (`floor`, `surface`, `contained`, `wall`, or `ceiling`), `parent_object`, and `visual_locator`; `name` carries the open-vocabulary identity.

Inventory: Object Identification

List visible non-architectural objects at the object level. Return one JSON object containing an `objects` array. Use a concrete physical-object name and assign an identifier to each emitted record. Represent the largest visually confirmed continuous or explicitly joined assembly. Fixed shelves, doors, drawers, and handles belong to that assembly. Contact or matching appearance alone does not merge independent objects.
Represent distinguishable complete chairs, pillows, lamps, frames, and other objects as separate instances. Dense contents or indistinguishable clutter may be recorded as a descriptive group per contiguous visible support area. Inspect the room center, wall-side strips, corners, openings, and visible support surfaces for omissions. Exclude architecture, empty regions, and whole–part duplicates.
Use `target: ...; extent: ...; exclude: ...` for each visual locator. Describe the visible body or connected path of the target and exclude adjacent objects and architecture. Do not infer hidden continuation in the inventory locator. Surface and contained objects must reference their supporting or enclosing object in this inventory. Floor objects have no semantic `parent_object`; the controller assigns their structural room parent when importing them into DeltaTree. Wall attachment is resolved later; ceiling objects refer to `ceiling:room`. Return no boxes, masks, dimensions, or metric coordinates.

An independent scope review labels entries `correct`, `too_narrow`, `too_broad`, `wrong_target`, `duplicate_part`, or `not_visible`. Retained entries receive a name, locator, and `identity: ...; boundaries: ...; distinction: ...` evidence. Duplicates reference their canonical assembly; review preserves identifier identity and adds no objects.

B.4 Layout Worker

Layout converts reviewed inventory, grounding, and segmentation into centers, dimensions, orientations, footprints, support, optional wall adjacency, and evidence. Rectification and scale map image coordinates to the metric room frame.

Layout: Footprint and Height Estimation

Estimate the physical height and complete top-down projected footprint of every supplied object. Image 1 is the clean source image; Image 2 contains coarse grounding and stable identifiers. Grounding components

Layout: Footprint and Height Estimation (continued)

locate the target but are not exact boundaries. Return exactly one record per supplied identifier, including its visible construction, appearance, materials, colors, height, occlusion status, footprint, and confidence. Represent the footprint with one to three rectangles or ellipses. Use multiple rectangles for an L-shaped or U-shaped object. Exclude walls, nearby objects, loose contents, and empty room area. Report partial occlusion only when another supplied object hides the target, and name that occluder. An irregular visible outline alone is not occlusion. Use normalized image coordinates and an undirected footprint-axis angle in $[0, 180)$ degrees, increasing clockwise from image east. This angle does not encode the object's semantic front.

Crop-based completion may add footprint primitives only beneath declared occluders, preserving visible masks; uncertain completion returns no additions. Wall objects receive depth, height, and bottom height; ceiling objects receive hanging height, with upstream support and planar placement fixed. During revision, `pose.position_image` is converted to room coordinates, and `pose.rotation_deg` specifies parent-frame XYZ Euler angles. Top-level positive yaw turns east toward north, unlike the initial undirected image-footprint angle. Target poses are converted to the increments in Equation 3.

B.5 Detail Worker

Detail receives the marked scene, clean target crop, local-axis/footprint overlay, and support context; optional articulation references guide occluded completion.

Detail: Part Geometry

Reconstruct the target as geometric Parts. Use the clean source crop for visible shape and internal structure. Keep the host center, dimensions, footprint, and rotation fixed. Part locations are measured in metres from the host geometric center, with local Z upward. The complete host pose is applied afterwards; do not encode it again in the Parts.
Reproduce visible openings, cavities, shelves, separated components, curved profiles, folds, and seams. Keep large panels within the union of the supplied footprint components. Do not bridge empty regions of an L-shaped assembly with a solid box. Generate only the target assembly and preserve independently represented objects. Use optional structural references only for occluded completion consistent with the source.
Choose supported operators per Part. Use mesh directly when free outlines, holes, or local folds require it. Primitive type and Part count are not quality scores. Distinguish filled soft volumes from fabric surfaces. For identifiable garments, complete basic category-consistent structure while preserving the visible silhouette, support, and fixed dimensions. For an ambiguous garment, use a plausible draped or folded surface. Keep intended apertures open. Supply folds explicitly through geometry; Blender constructs the supplied surfaces without adding cloth dynamics.
Return JSON containing the Part records. Use `rotation_deg` for local XYZ Euler angles, positive dimensions, and the required operator parameters. Keep all Parts within the host bounding box in the reference configuration. Articulated motion is checked separately over the declared travel range. During repair, preserve source-supported components and modify only the permitted local patch.

Part records specify name, operator, relative location, dimensions, rotation, and operator parameters. Operators include box variants, cylinders, cones, spheres, capsules, rods, tori, sheets, profile extrusions, sweeps, revolutions, and meshes. Soft volumes use an empty, `pillow`, `cushion`, or `mattress` profile, with `bulge`, `edge_tension`, and `center_dent` in $[0, 1]$. Profile/path coordinates lie in $[-0.5, 0.5]$ and radii in $(0, 0.5]$. Mesh vertices lie in $[-0.5, 0.5]^3$, scaled by Part dimensions; faces have valid zero-based indices and nonzero area. Open boundaries are allowed; thickness defaults to zero, with positive values in metres.

Joint records and evidence enter DeltaTree; the relation builder binds their geometry, poses, and parameters to Coupling Graph checks.

Detail: Articulation Completion

Complete the required movable structure within the supplied host and permitted targets. Identify the stationary body and each moving assembly. Represent their geometry as separate Parts and group rigidly

Detail: Articulation Completion (continued)

connected Parts into links. Use the supplied structural references to complete occluded components consistently with the source evidence.
Return link membership and joint records using stable identifiers. For each joint, specify the type, connected links, origin, axis, limits, and reference configuration. Express the origin and axis in the declared parent-link frame and use the units required by the schema. The axis must be nonzero, limits must be ordered, and the reference configuration must lie within them. Preserve accepted exterior attributes with valid visual support unless the edit instruction explicitly authorizes a change.
Mark unobserved joint and internal-structure parameters as inferred and retain evidence for directly supported entries. Supply the geometry needed to evaluate motion clearance. Report unresolved structural requirements when the requested articulation cannot be specified. Parameter validity alone does not establish collision-free motion; the controller must run the applicable execution checks.

Surface and contained objects are placed after usable parent geometry is available.

Detail: Surface and Contained Object Placement

Return one decision for each required inventory identifier. Place a surface object using an available plane and local UV coordinates. Place a contained object using an existing internal Part as its support or hanging anchor. Do not substitute an external cabinet top for a missing internal anchor. Return `unresolved` when no valid internal anchor exists. Use `already_represented` only when exact existing Part names express the item, and bind those Parts to its inventory identifier. Avoid duplicating existing items.

Material assignment receives object/Part names, source appearance descriptions, and the scene palette.

Detail: Material Assignment

Assign one material record to every listed Part. Distinguish structural roles when their visible materials differ. Preserve source-supported colors and finishes and use the supplied material-type vocabulary. Return `part_materials` indexed by object and Part name, with linear-RGB `base_color`, `roughness`, `metallic`, `specular`, and `material_type`. Do not merge several required Part entries into a single record.

B.6 Verification, Diagnosis, and Revision

Visual and execution failures use a shared issue interface identifying S_q, observations, evidence, and existing parents for missing structure. After evidence review and provisional topology attachment, the controller computes closure, C_q, n_q, and Π_q (Algorithm 1) using ancestor tables for the provisional tree. Relative-pose invariance and required world-frame paths determine $\mathcal{V}_q^{\text{cand}}$ (Sec. 4.3); intersecting it with $\mathcal{A}_q$ gives the `allowed_targets`. Workers act within this scope; retrieved context also includes fixed boundaries. A changed topology proposal requires recomputing the scope.

Revision: Issue Diagnosis

Inspect the source and current renders under the supplied edit conditions. Preserve source constraints outside the explicitly requested changes. Bind object identifiers using the supplied identity metadata and grounding labels. Report one falsifiable, independently repairable condition per issue, with its predicate, target nodes and fields, evidence region, observation, and evidence references. Use `error` for an unambiguous defect with an identifiable target; use `warning` when the visual evidence, identity binding, or repairability is uncertain. Report contradictions even when the target is currently protected; the controller determines eligibility. For missing structure, identify the existing parent and required entity kinds and relations. For geometry, specify concrete source-visible structural targets. Derived nodes are read-only: identify their writable sources or relations. Do not output a patch, choose an owner or action, or assign an overall score.

Revision: Evidence Review

Review every supplied issue independently. Identify supplied protected fields or relations whose recorded visual support is invalid because of an incorrect evidence association or a misread source cue. Preserve values whose visual support remains valid. Explicit edits in c are separately authorized changes; do not label their source evidence as invalid merely because the instruction requests a different value.

Revision: Evidence Review (continued)

Returning no invalidated targets is valid. For missing structure, identify the existing parent and the required child topology without discarding valid evidence for the parent. Return the issue identifier, affected fields or relations, reason, and evidence references. Do not propose replacement values or expand the supplied target set.

Revision: Typed-Delta Repair

Resolve the supplied semantic issue using one ordered, non-empty typed-delta step for every supplied action, in the supplied order. Treat the issue observation as the repair brief. Write only the allowed field, topology, and relation targets. Preserve other degrees of freedom. Each field replacement contains its complete new value. New nodes must match the supplied provisional topology, retaining their assigned `new_*` identifiers, parents, and permitted kinds. Later steps may reference these identifiers. Request a revised topology scope if an additional entity or relation is needed.
A topology target identifies an existing parent with missing geometry. Create children only of the permitted kinds; do not create another semantic object to replace that parent. For visual layout, return `pose.position_image` as normalized $[x, y]$ and `pose.rotation_deg` for parent-frame XYZ Euler rotations in degrees. These target poses encode the corresponding node-local increments; do not add them to the current position or apply the same motion twice. For reopened entries, update provenance and evidence links from the supplied review. Return typed deltas as JSON rather than editing Blender Python. If a previous candidate failed, use its counterexample within the same authorized scope. If no permitted repair can be proposed, return an unresolved reason with no executable steps.

Candidates are compiled and executed separately from accepted state. Geometry checks cover finite transforms, required geometry, bounds, placement/contact, penetration, and joint parameters; applicable physical and articulated-motion checks invoke Isaac Sim. Each check reports participants, input entries, protocol, outcome, and any counterexample. Revalidation covers affected boundaries, newly discovered conflicts, and nonlocalizable global conditions. Missing structure or unavailable required tests cannot pass validation. The visual verifier receives source, accepted, and candidate renders with identities and scope; the repairer's reasoning is withheld.

Revision: Visual Verification

Decide whether the candidate resolves the supplied issue while preserving source-supported structure inside the target and at its local boundary. Compare source, accepted, and candidate images in that order, applying c only to explicitly requested changes. Check the new requested values and preserve the remaining source constraints. Use top-down and oblique evidence for their appropriate geometric questions. A matching silhouette or executable mesh alone is insufficient.
Return `accept`, `reject`, or `insufficient_evidence`, with a concrete reason and evidence references. For geometry review, report coverage, preservation of internal structure, visible-feature matches, structural-target checks, changed-Part checks, and regressions. Inspect every supplied structural target and replaced Part exactly once. Distinguish inferred occluded structure from directly visible features. Missing or ambiguous visual comparisons require `insufficient_evidence`.
Use `coverage_only=true` only for a designated intermediate construction stage. Correct coverage and placement may pass that stage while finer requirements remain recorded as unresolved. This mode must not relax an active repair target, an explicit edit, or preservation of source-supported structure. It cannot establish final scene acceptance; report every unresolved requirement for later verification.

Passing affected visual and execution checks commits the state, relations, program, observations, and revision record together; rejection preserves accepted state and returns feedback. Unaffected prior issues may remain, but successful completion requires all applicable h_k and g_m. On budget exhaustion, the controller returns the best recorded result with an unvalidated status and a list of unresolved checks. Usable upstream artifacts survive nonfatal stage failures; missing required inputs block dependent execution.

C Additional Experiments

C.1 Text-driven Scene Generation

We compare complete text-to-scene pipelines on the same 100 original SceneEval-100 descriptions (Tam et al., 2026). Holodeck (Yang et al., 2024), SceneSmith (Pfaff et al., 2026), and SAGE (Xia

et al., 2026) retain their native models, assets, refinement, and stopping rules. TopCode refines each description into an image-generation prompt, generates a top-down image, and uses it as input to our image-driven scene synthesis pipeline. The entire process is automated, without manual image selection or modification.

Table 8: Text-driven scene generation on 100 SceneEval-100 descriptions. CNT, ATR, OOR, and OAR evaluate object counts, attributes, object–object relations, and object–architecture relations, respectively.

	Text Alignment				Scene Validity					
Method	**CNT↑**	**ATR↑**	**OOR↑**	**OAR↑**	**COL↓**	**SUP↑**	**NAV↑**	**ACC↑**	**OOB↓**	**OPC↓**
Holodeck	0.402	0.485	0.159	0.393	0.361	0.499	**0.996**	0.772	0.020	0.430
SAGE	0.790	0.655	0.658	**0.815**	0.351	0.743	0.971	0.750	0.022	0.344
SceneSmith	**0.803**	0.696	**0.736**	0.779	**0.295**	0.749	0.962	0.683	**0.009**	0.320
Ours (**TopCode**)	0.725	**0.717**	0.693	0.784	0.338	**0.762**	0.995	**0.819**	0.015	**0.288**

Text alignment is evaluated against the original descriptions, and architectural checks use each generated room. Both groups of metrics follow Appendix D.3, with scene scores averaged equally.

TopCode achieves the highest ATR and leads in SUP, ACC, and OPC, with second-best scores on the remaining scene-validity metrics (Tab. 8). Text alignment is less uniform: OOR and OAR rank second, while CNT trails SceneSmith and SAGE. This combination of attribute alignment and scene validity is consistent with DeltaTree retaining generated visual evidence and the Coupling Graph guiding physical completion.

C.2 Sequential Local Editing

All 30 scenes from the independent-editing experiment receive four compatible instructions covering move/rotate, add, remove, and replace, yielding 120 sequential editing steps per method. TopCode and Edit-As-Act (Noh et al., 2026) follow identical sequences, continuing from their own outputs with the independent-editing settings in Appendix D.4.

At round k, CGA requires all instruction conditions still active from rounds $1{:}k$; CNP requires unchanged content outside their combined authorized changes, relative to the initial scene. Both use Appendix D.4's checks and tolerances and report the percentage of the 30 scenes passing each criterion.

Table 9: Sequential editing on 30 scenes over four rounds. CGA and CNP denote cumulative goal attainment and cumulative non-target preservation, respectively; both are percentages.

Method	**Round**	**CGA↑**	**CNP↑**
Edit-As-Act	1	83.33	96.67
	2	73.33	93.33
	3	60.00	93.33
	4	53.33	86.67
TopCode	1	93.33	100.00
	2	80.00	100.00
	3	76.67	100.00
	4	70.00	100.00

TopCode maintains 100.00% CNP throughout the four-round sequences (Tab. 9). CGA decreases for both methods, but TopCode leads in every round, with 70.00% at round 4 versus 53.33% for Edit-As-Act. These results are consistent with DeltaTree retaining accepted records and visual evidence, while coupling closure and LCA-based write scopes coordinate and localize physically dependent updates.

D Experimental Details

D.1 Data and Reference Preparation

For the image-conditioned comparison, all 100 SceneEval-100 descriptions (Tam et al., 2026) are converted to top-down references using a fixed GPT-Image2 model and prompt. Review checks

visible categories, counts, and spatial relations; failed references are regenerated before freezing the references. All image-conditioned methods receive the same image and task instruction. Original descriptions support the text-driven comparison (Appendix C.1).

Reviewed categories, visible instance masks, room boundaries, and openings are frozen evaluation annotations; methods obtain required masks/depth through their own input-image preprocessing. Within each comparison, renders share cameras, resolution, background, and lighting, retaining generated materials. Alignment permits only a global rigid transform and uniform scale.

D.2 Baseline and Implementation Details

Main comparisons use Qwen3.8-max for replaceable language and vision-language components; dedicated reconstruction models retain released weights and preprocessing. Iterative methods retain native refinement and stopping rules; matched token limits apply only to TopCode ablations. TopCode prompts appear in Appendix B.

One-shot Qwen generates one Blender program without execution feedback. MIDI (Huang et al., 2025) and 3D-Fixer (Yin et al., 2026b) retain their native pipelines; Gen3DSR (Ardelean et al., 2025) estimates depth and camera information from RGB; VIGA (Yin et al., 2026a) uses Qwen for its code–render–inspect loop. For editing, Qwen receives the initial program, renders, and instruction and returns one edit. Edit-As-Act (Noh et al., 2026) retains planning, validation, scene actions, and asset generation. Input adapters preserve initial geometry, identities, and poses; all returned scenes use the same offline evaluation.

D.3 Evaluation Metrics

Reference alignment. Object F1 compares reference and generated semantic counts n_c^R, n_c^G for each category c:

$$\begin{aligned} \text{ObjectF1} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}}, \qquad \text{TP} = \sum_c \min(n_c^R, n_c^G), \\ \text{FP} = \sum_c \max(n_c^G - n_c^R, 0), \quad \text{FN} = \sum_c \max(n_c^R - n_c^G, 0). \end{aligned} \tag{8}$$

Counts include furniture and identifiable small objects, excluding component parts; dense groups follow frozen annotations. No overlap threshold is used.

Layout IoU uses visible top-down instance masks. Category-wise Hungarian matching $\mathcal{M}$ maximizes total intersection area:

$$\begin{aligned} I_\cap = \sum_{(i,j)\in\mathcal{M}} |R_i \cap G_j|, \qquad A_R = \sum_i |R_i|, \qquad A_G = \sum_j |G_j|, \\ \text{LayoutIoU} = \frac{I_\cap}{A_R + A_G - I_\cap}. \end{aligned} \tag{9}$$

Unmatched masks remain in A_R, A_G.

DINOv2 (Oquab et al., 2023) uses `facebook/dinov2-giant`: aspect-preserving bicubic RGB resizing, white padding to 518×518, and standard model normalization. The 1536-dimensional CLS features f_R, f_G give

$$\text{DINO} = \frac{f_R^\top f_G}{\|f_R\|_2 \|f_G\|_2}. \tag{10}$$

DINO is reported in $[-1, 1]$ without percentage scaling.

Text alignment. For Tab. 8, we use SceneEval's annotations of the original descriptions (Tam et al., 2026). **CNT** checks category-level instance counts against the specified equality or inequality constraints. **ATR** checks whether the required numbers of objects have the specified attributes, assessed by a VLM from object renders. **OOR** checks spatial relations among objects; **OAR** checks relations between objects and the generated architecture. For both relation metrics, language descriptions are mapped to geometric predicates, and the numbers of satisfying instances or groups

are tested against the annotated quantifiers. Missing required objects make the corresponding attribute or relation requirement unsatisfied. Each per-scene metric is the fraction of its annotated requirements that are satisfied, reported in $[0, 1]$ with higher values better.

Scene validity. SceneEval's six criteria (Tam et al., 2026) use reviewed instance groups, support types, functional sides, and openings. They evaluate returned geometry independently of internal validation. Let N be the number of evaluated objects.

COL: $N_{\text{collision}}/N$, counting each object once. A mesh collision is recorded only if it persists after a 5 mm displacement along the contact separation direction.

SUP: $N_{\text{supported}}/N$. Ground/object/wall/ceiling support types determine probe direction, with contact distance strictly below 1 cm; ground/object support also requires contact-point convex-hull and center-of-mass ray checks. This measures geometric support.

NAV: largest connected free-space area divided by total free area, using a 256×256 grid, eight-connectivity, and 0.2 m robot width for floor erosion and oriented-box obstacle expansion; zero if no free space.

ACC: $\frac{1}{N_F}\sum_{i=1}^{N_F} \max_{s\in F_i} a_{i,s}$, where N_F counts objects with functional sides F_i. For each side, $a_{i,s}$ is free floor pixels in a 0.2 m access strip offset 0.05 m from the surface, divided by its rasterized area before clipping.

OOB: N_{out}/N. An object is out of bounds when fewer than 99% of approximately $\max(5000V_{\text{OBB}}, 1000)$ surface samples project downward onto the generated floor; V_{OBB} is its oriented-box volume in cubic meters.

OPC: occluded/checked interior opening sides (zero to two per opening). Clearance depth equals door width for doors and 0.5 m for windows. The official `opening_occluded` flag requires a nonempty candidate-obstruction record, with no additional area threshold.

Reference-alignment and scene-validity scores average all 100 scene values equally. MIDI's SUP, NAV, ACC, OOB, and OPC are N/A. Editing metrics and support-loss rates are defined separately below.

D.4 Local Editing Protocol

Thirty frozen TopCode scenes each receive four independent instructions (move/rotate, add, remove, replace), giving $T = 120$ tasks. Every method returns an evaluable scene for all tasks. Instructions and protected content are fixed beforehand, independently of TopCode's write scope. Each task restarts from its initial scene.

Instruction satisfaction and preservation. For task t, let X_t^0, X_t^1 denote initial and final scenes, C_t the requested conditions, and P_t the protected content:

$$\begin{aligned} \text{ESR} &= \frac{100}{T}\sum_{t=1}^{T} \mathbf{1}\left[\bigwedge_{c\in C_t} c(X_t^1)\right], \\ \text{NPR} &= \frac{100}{T}\sum_{t=1}^{T} \mathbf{1}\left[\text{Preserve}(X_t^1, X_t^0; P_t)\right]. \end{aligned} \tag{11}$$

Conditions require the requested transform with shape/appearance preserved (move/rotate), specified object and placement (add/replace), or absence of the entity and its parts (remove), including explicit content, dimension, material, and support requirements. Tolerances are 2 cm for position, 2° for yaw, and 10% for general dimensions; explicit maxima use their stated bounds plus numerical measurement allowance.

NPR checks protected world geometry/pose, identity, visibility, materials, textures, and UVs, using 0.1 mm geometric tolerance and ignoring harmless triangle reordering or unused material nodes. World-space checks include parent-propagated changes.

New geometric violations. Let E_t^0, E_t^1 be the collision-pair sets before and after editing, U_t^0, U_t^1 the unsupported-object sets, and B_t^0, B_t^1 the out-of-bounds-object sets. The newly introduced violations are

$$E_t^+ = E_t^1 \setminus E_t^0, \qquad U_t^+ = U_t^1 \setminus U_t^0, \qquad B_t^+ = B_t^1 \setminus B_t^0. \tag{12}$$

New collision partners count even for previously colliding objects; replacements are new instances, and repairs do not offset new violations. Using Appendix D.3's tests,

$$\begin{aligned} \mathrm{COL}_{\mathrm{edit}} &= \frac{100}{T}\sum_{t=1}^{T} \mathbf{1}[E_t^+ \neq \varnothing], \\ \mathrm{SUP}_{\mathrm{edit}} &= \frac{100}{T}\sum_{t=1}^{T} \mathbf{1}[U_t^+ \neq \varnothing], \\ \mathrm{OOB}_{\mathrm{edit}} &= \frac{100}{T}\sum_{t=1}^{T} \mathbf{1}[B_t^+ \neq \varnothing]. \end{aligned} \tag{13}$$

All three are lower-is-better percentages over 120 tasks, counting each task once per violation type. Unchanged outputs may pass NPR and avoid new violations while failing ESR.

D.5 Ablation Configurations

All variants share 100 references, Qwen, and a cumulative token limit; the full row reuses main-generation results. Offline scores are never returned to workers.

w/o LWS: allows writes to all eligible records, retaining evidence protection, both memory structures, and validation.

w/o DT+LWS: additionally replaces hierarchical pose composition with flat records and world-coordinate updates, retaining identities, attributes, provenance, and evidence. Comparison with *w/o LWS* isolates hierarchy under the same permissions.

w/o CG: replaces persistent dependencies and closure with LLM-selected related entities/attributes from scene, task, observations, and feedback. LCA and evidence permissions remain; checks are reinstantiated from scene state and declarations with unchanged coverage.

w/o EV: removes geometric, physical, and articulated-motion feedback and acceptance gates, retaining program execution, runtime-error handling, visual checks/revision, memory, and local write scopes.

D.6 Human Evaluation

Following TabletopGen (Wang et al., 2026), the questionnaire rates IA, VF, and PP separately on a 1–7 scale.

Thirty SceneEval-100 inputs (Tam et al., 2026) are sampled by room type before inspecting outputs. Fifteen participants each evaluate all six methods on 20 scenes. Three random groups of ten scenes and three groups of five participants use assignments $(1, 2)$, $(2, 3)$, and $(3, 1)$, giving ten ratings per scene/method.

Each item shows the reference, a generated top-down render, and two oblique renders, with camera, lighting, resolution, and display size matched within each scene. Scene and output orders are randomized per participant (Yang et al., 2024); method names are hidden and candidate IDs reassigned per scene. All methods share instructions and integer scales, allowing ties. Each reported score averages ten ratings per scene, then 30 scene means (Tab. 10); totals are 1,800 items and 5,400 scalar ratings.

Table 10: Human evaluation on 30 scenes: Image Alignment (IA), Visual Fidelity (VF), and Physical Plausibility (PP), rated from 1 to 7.

Method	**IA↑**	**VF↑**	**PP↑**
Qwen3.8-max	4.03	4.70	<u>5.37</u>
MIDI	2.13	2.50	1.03
Gen3DSR	2.37	2.90	1.57
3D-Fixer	3.00	3.17	3.73
VIGA	<u>4.83</u>	<u>5.30</u>	5.07
Ours (**TopCode**)	**5.80**	**6.60**	**5.43**

TopCode receives the highest mean IA, VF, and PP ratings (Tab. 10). It exceeds VIGA by 0.97 in IA and 1.30 in VF, while PP is close to Qwen (5.43 vs. 5.37). The IA gain supports visual evidence

preservation, while VF and PP complement the objective metrics with assessments of appearance quality and visible physical plausibility.

Indoor Scene Evaluation: Participant Instructions

Purpose. You will evaluate computer-generated indoor scenes using the images shown. For each scene, assess six anonymous outputs. Each output requires three separate ratings: Image Alignment, Visual Fidelity, and Physical Plausibility.

What you will see. The reference shows the target room from above. The other three images show one generated scene from a top-down view and two oblique views. All three generated images belong to the same output.

How to answer.

- Compare the reference and generated top-down view for Image Alignment.
- Inspect all three generated views for Visual Fidelity and Physical Plausibility.
- Use the scale descriptions to choose one score from 1 to 7 for each question. Equal scores across outputs are allowed.
- Judge visible shape and surface quality under VF, and contact, support, and penetration under PP. Hidden components, dynamic stability, and the ability to move or open objects cannot be judged from these still images.
- Work independently and take breaks as needed. If an image fails to load, contact the study organizer before scoring that item.

Participation. Participation is voluntary, and you may stop at any time. The questionnaire uses an anonymous participant ID and does not ask for your name.

Before starting: I have read the instructions and choose to participate.
☐ Yes, begin the survey. ☐ No, exit the survey.

Per-output questionnaire. Repeat for each candidate with all four images visible.

Scene ID: XXX, **Candidate ID:** XX.

Images shown: Reference, generated top-down view, and two generated oblique views.

Question 1: Image Alignment (IA)

How closely does the generated scene match the reference image?

Compare reference-visible structure, object types and counts, layout, sizes, and appearance using the top-down views. Allow for lighting differences. Judge correspondence independently of visual quality.

1: No match. The scene bears little resemblance to the reference.

2: Very weak match. A few elements match, but large differences dominate.

3: Limited match. Some objects and spatial relations match, with several major discrepancies.

4: Moderate match. The arrangement is recognizable, with clear differences in objects, layout, or appearance.

5: Good match. The main structure, objects, and layout agree, with some local differences.

6: Very close match. The scene closely matches the reference, with minor localized differences.

7: Excellent match. No noticeable discrepancy remains in reference-visible content or appearance.

Select one: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

Question 2: Visual Fidelity (VF)

How would you rate the generated scene's shape and surface quality?

Inspect shapes, surface completeness, materials, and textures across all three views. Look for distortions, holes, fragments, or smeared textures. Judge the depicted style independently of reference similarity or furniture density.

1: Very poor. Most shapes or surfaces are unrecognizable or severely corrupted.

2: Poor. Severe shape distortions or incomplete surfaces dominate.

3: Somewhat poor. Several major shape or surface artifacts remain.

4: Fair. Objects are recognizable, with noticeable shape or surface artifacts.

5: Good. Shapes and surfaces are mostly coherent, with some localized defects.

6: Very good. Shapes and surfaces are convincing across views, with minor flaws.

7: Excellent. Shapes and surfaces are well formed, with no noticeable artifacts across views.

Select one: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

Question 3: Physical Plausibility (PP)

How physically plausible are the visible object arrangements?

Across all three views, inspect penetration between objects or into room structures, floating, and implausible support contacts. Occlusion alone is not evidence of missing support. Judge independently of reference similarity and texture quality.

1: Very implausible. Severe penetration or lack of support is widespread.

2: Implausible. Many objects have clear penetration or support problems.

3: Somewhat implausible. Several clear penetration or support problems remain.

4: Mixed. Many relationships appear plausible, but some clear violations remain.

5: Mostly plausible. Contact and support appear reasonable, with a few localized issues.

6: Very plausible. Visible relationships appear reasonable, with minor local irregularities.

7: Fully plausible. No noticeable penetration or support violation appears in the shown views.

Select one: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

Optional comment (main issue): _____

Before continuing: Check that you selected one score each for IA, VF, and PP.

D.7 VLN Post-training Details

Checkpoint and training data. All conditions start from the official ETPNav R2R-CE checkpoint (An et al., 2024). *None* retains it; post-training uses R2R-CE source data (Anderson et al., 2018b; Krantz et al., 2020) alone or 50:50 episode mixtures with Qwen/TopCode data. ETPNav's configuration is retained with learning rate 10^{-6} and matched optimizer settings and updates.

We use 100 paired TopCode and one-shot Qwen scenes from the main scene-generation experiment, with both scenes in each pair supporting navigation. Each method contributes 25 instruction–trajectory samples per scene (2,500 per method), constructed through the same pipeline and checked for traversability and instruction correspondence. External evaluation scenes and episodes are excluded from data construction and training.

External evaluation data. HM3D environments (Ramakrishnan et al., 2021) use public LHPR-VLN instructions and trajectories (Song et al., 2025). Instructions remain verbatim; released subtask indices crop reference paths, whose endpoints define goals. Consecutive duplicates are removed; start/goal NavMesh snapping is limited to 0.20 m with reachability checks. We evaluate 411 of 424 subtasks extracted from LHPR-VLN across 26 HM3D houses as single-goal episodes using the ETPNav protocol.

Evaluation. All four checkpoints use the same Habitat setup (Savva et al., 2019) and $E = 411$ episodes. We report Success Rate (SR), Success weighted by Path Length (SPL), normalized Dynamic

Time Warping (nDTW), and Navigation Error (NE). Let P_e, R_e be predicted and reference paths, d_e the final geodesic goal distance, ℓ_e the shortest start–goal geodesic distance, and L_e the recorded Euclidean path length. ETPNav SS-ETP success is $S_e = \mathbf{1}[d_e \le 3\,\mathrm{m}]$, without an additional STOP requirement:

$$\begin{aligned} \mathrm{SR} &= \frac{100}{E}\sum_{e=1}^{E} S_e, & \mathrm{SPL} &= \frac{100}{E}\sum_{e=1}^{E} S_e \frac{\ell_e}{\max(\ell_e, L_e)}, \\ \mathrm{nDTW} &= \frac{100}{E}\sum_{e=1}^{E} \exp\left[-\frac{\mathrm{DTW}(P_e, R_e)}{3\,\mathrm{m}\cdot|R_e|}\right], & \mathrm{NE} &= \frac{1}{E}\sum_{e=1}^{E} d_e. \end{aligned} \tag{14}$$

SPL follows Anderson et al. (2018a); nDTW follows Ilharco et al. (2019) with `fastdtw` and Euclidean pointwise distances. Episodes receive equal weights; SR/SPL/nDTW are percentages and NE is in meters. Reference paths and goal coordinates are evaluator-only inputs.

D.8 Franka Interaction Details

Isaac Sim uses a fixed-base, seven-joint Franka Panda with a parallel-jaw gripper; base/grasp poses and opening widths follow known target geometry. SciPy's trust-region reflective least-squares IK (Virtanen et al., 2020) minimizes position, orientation, and joint-continuity errors under joint limits. Sampled configurations undergo robot–scene and self-collision checks; interpolated joint-position waypoints are tracked at 120 Hz with a 20 N effort limit per finger.

From a 16 cm pregrasp offset, the gripper closes on the target and follows its sliding axis, hinge arc, or vertical lifting direction. Drawer/door motion is contact-driven through passive prismatic/revolute joints. Commands are a 23 cm pull (12 s), 45° opening (12 s), and 12 cm lift (4 s), each followed by a 2 s contact hold. Fig. 4 shows contact establishment and the hold endpoint.

D.9 Scene Export and Simulator Setup

Habitat (Savva et al., 2019) loads static room/obstacle meshes and a NavMesh for traversability, route validation, and geodesic evaluation; generated and external scenes share agent/sensor settings. Isaac Sim retains metric scale, relative placement, fixed architecture, collision geometry, and movable components: passive drawer/door joints and a free rigid body for lifting, controlled as in Appendix D.8.