ROOMER: REFLECTIVE OBJECT-GROUNDED MODEL EDITING AND REPAIR FOR 3D INDOOR LAYOUT SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing indoor layout generators produce globally plausible layouts yet may retain local violations such as collisions, out-of-bounds placements, obstructed openings, and blocked circulation. Most prior work focuses on full-scene synthesis or scene-level optimization, with limited support for identifying responsible objects and locally repairing affected regions. We present Roomer, a reflective repair framework that casts these violations as sparse, object-grounded repair problems. Roomer encodes layouts as "RoState" and uses "RoReview" to bind measured violations to implicated objects. A geometry-conditioned vision-language model planner proposes a structured local edit, while a deterministic solver validates it and generates a finite set of candidate edits when needed. Each candidate is committed only if full-scene floor-plane verification confirms that it resolves the target violation without new hard-rule violations or broken protected constraints. We train the planner on Roomer-CC, a controlled-corruption dataset that pairs faulty layouts with object-grounded violation evidence and known-feasible inverse StatePatches. Since existing benchmarks rarely assess whether physically valid layouts are usable, we introduce Roomer-Eval to assess distributional quality, physical validity, and practical usability. Experiments show that Roomer repairs residual violations while preserving valid regions, improves physical validity and usability, and transfers across external generators.

1 INTRODUCTION

Indoor layout synthesis aims to arrange furniture within a given architectural space while satisfying semantic, geometric, and functional requirements. Despite recent progress, existing generators do not always satisfy these requirements. A generated layout may appear visually plausible and semantically coherent while still containing local violations, including collisions, out-of-bounds placements, obstructed architectural openings, and blocked circulation. Correcting these violations through full-scene regeneration may disrupt already-valid furniture configurations and spatial relations. This motivates us to study local repair for indoor layouts: given a completed layout produced by an arbitrary layout generator, the system must identify residual violations, localize the implicated objects, and resolve the violations through a limited set of local edits. The central challenge is to translate measured violation evidence into precise object-level edits and verify that the resulting modifications improve layout quality. Improving generation may reduce these violations, yet correcting residual errors in a particular output while preserving already-valid content remains a distinct task.

Existing approaches mainly follow two paradigms. Learning-based methods learn furniture composition and spatial distributions from indoor-scene datasets using autoregressive, graph-based, diffusion, and constraint-guided models (Paschalidou et al., 2021; Tang et al., 2024; Lin & Mu, 2024; Yang et al., 2024a). Although they generate globally plausible layouts, their scene-level objectives do not explicitly attribute residual violations to responsible objects. LLM-based layout planners instead leverage open-domain knowledge for semantic reasoning and spatial planning, with recent systems incorporating structured scene representations, visual feedback, numerical optimization, and iterative refinement (Feng et al., 2023; Yang et al., 2024b; Ling et al., 2026; Xia et al., 2026). However, these revisions primarily serve prompt satisfaction or scene-level improvement rather than

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.

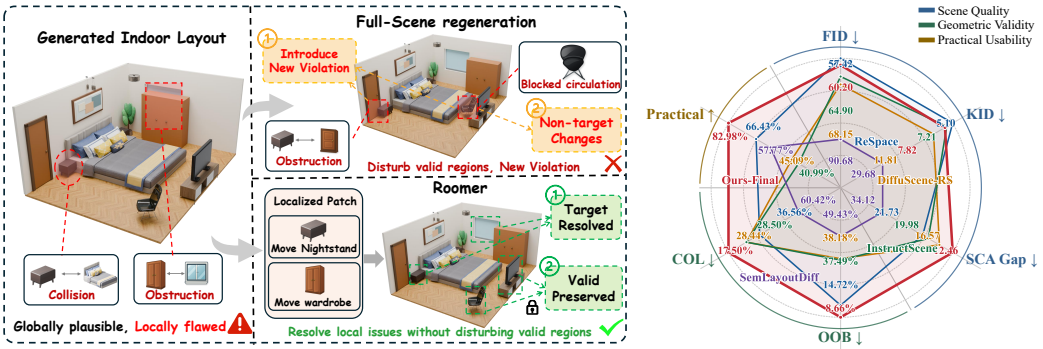

Figure 1: Motivation for Roomer. Generated layouts can be globally plausible while containing sparse object-level violations. Full-scene regeneration may introduce new violations and alter valid regions, whereas Roomer performs localized object-grounded repair to resolve issues while preserving valid layouts. The radar chart summarizes comparisons across scene quality, geometric validity, and practical usability.

being driven by geometric measurements grounded in specific objects. Both paradigms therefore offer limited support for measurement-grounded object attribution and verifiable repair of residual violations.

Our key insight is that residual violations in generated layouts are typically sparse, localized, and attributable to a small number of objects and their local geometric relations. Such correction should therefore be formulated as verification-gated local state repair rather than full-scene regeneration. Under this formulation, measured evidence identifies the responsible objects and guides local edits, while full-scene verification checks target resolution and non-regression under the floor-plane rule evaluator before commitment.

Based on this insight, we propose Roomer, a reflective repair framework for indoor layouts. Roomer converts a completed layout into RoState, an object-addressable canonical representation, and constructs RoReview to associate each residual violation with geometric measurements and implicated objects. Conditioned on the current layout, RoState, and RoReview, a geometry-conditioned vision-language model planner predicts a schema-constrained StatePatch specifying the repair target, action, and initial parameters. A deterministic solver first validates this proposal and, when necessary, instantiates a finite set of reproducible alternatives. The first candidate that passes full-scene verification is committed; otherwise, the committed layout remains unchanged. Repeating this process yields a controlled and verifiable local repair loop.

Existing indoor-scene datasets do not provide the supervision needed to train an object-grounded repair planner: they contain complete layouts but lack paired faulty inputs, attributed violation evidence, and corrective actions. We therefore construct Roomer-CC by applying six parameterized object-level corruptions to valid 3D-FRONT layouts, yielding 67,550 paired repair examples with object-grounded violation evidence and known-feasible inverse StatePatch targets. These examples jointly supervise violation attribution and localized repair planning.

Conventional layout benchmarks emphasize distributional similarity and basic physical validity, but rarely assess whether a physically valid layout remains usable. We therefore introduce Roomer-Eval, which combines standard metrics for distributional quality and 3D physical validity with five reproducible rule families for assessing practical spatial usability. Experiments on 3D-FRONT layouts and outputs from external generators show that Roomer repairs residual violations while preserving valid regions, improving physical validity and practical usability across generators.

Our main contributions are as follows:

- We formulate layout correction as **verification-gated local state repair** and introduce Roomer. Roomer attributes measured residual violations to implicated objects, generates structured local edits, and commits only candidates that pass full-scene verification.

- We construct **Roomer-CC**, a controlled-corruption dataset of 67,550 paired repair examples derived from valid 3D-FRONT layouts, enabling joint supervision of object-level violation attribution and localized repair planning.
- We introduce **Roomer-Eval**, a unified evaluation protocol that combines standard metrics for distributional quality and 3D physical validity with five reproducible rule families for assessing practical spatial usability.

2 Related Work

Indoor Layout Synthesis. Early indoor-layout synthesis methods use design rules, scene priors, and search or optimization to construct feasible furniture arrangements (Yu et al., 2011; Merrell et al., 2011; Fisher et al., 2012; 2015; Qi et al., 2018). Recent learning-based approaches instead model layout distributions with autoregressive, graph-based, and diffusion models (Wang et al., 2021; Paschalidou et al., 2021; Wang et al., 2019; Li et al., 2019; Dhamo et al., 2021; Gao et al., 2023; Tang et al., 2024; Lin & Mu, 2024), while constraint-aware methods incorporate semantic and physical guidance to improve geometric plausibility (Yang et al., 2024a; Sun et al., 2026).

More recently, language- and vision-driven systems use LLMs and VLMs for scene planning, structured generation, and geometric optimization (Feng et al., 2023; Yang et al., 2024b; Fu et al., 2024; Sun et al., 2025b;a; Feng et al., 2025; Xiang et al., 2026; Berdoz et al., 2026). Scenethesis further combines LLM planning, vision-guided refinement, physics-aware optimization, and verification for 3D scene generation (Ling et al., 2026). Despite these advances, existing methods mainly construct complete scenes or improve generation-time feasibility. Roomer instead targets residual violations after generation, attributing them to specific object instances and performing verification-gated local repair.

Reflective Closed-Loop Scene Editing. Feedback-driven methods iteratively evaluate intermediate outputs and use the resulting feedback to guide subsequent updates. Language-model-based and agentic frameworks rely on self-generated critiques, environmental feedback, or external tools to refine predictions and actions (Madaan et al., 2023; Shinn et al., 2023; Yao et al., 2023; Gou et al., 2024). Related closed-loop co-refinement has also been explored in multimodal human–object interaction synthesis (Dang et al., 2025; 2026). For explicit 3D scenes, prior methods broadly follow two directions. Instruction-driven editors translate user-specified goals into compositional object operations or physically valid action sequences (Zheng et al., 2025; Bucher & Armeni, 2026; Noh et al., 2026), whereas scene-feedback-driven systems revise generated scenes over multiple rounds using semantic, visual, geometric, physical, or functional assessments (Yang et al., 2025; Ling et al., 2026; Xia et al., 2026; Zhao et al., 2026; Wang et al., 2026). These studies demonstrate the value of iterative correction, but their objectives are typically specified by user instructions, model-generated critiques, or scene-level quality signals rather than deterministic measurements attributed to specific object instances. Roomer instead derives repair targets from measured residual violations, attributes them to implicated objects, and commits only localized edits that pass full-scene verification without regressions.

Physical Validity and Practical Spatial Usability. Indoor layout evaluation has primarily focused on distributional quality and basic physical validity. FID, KID, generative precision and recall, and category statistics measure similarity to real-scene distributions and furniture compositions, whereas collision and out-of-bounds rates assess geometric feasibility (Heusel et al., 2017; Binkowski et al., 2018; Kynkäänniemi et al., 2019; Paschalidou et al., 2021; Tang et al., 2024; Yang et al., 2024a). SceneEval (Tam et al., 2026) further assesses compliance with explicit object and relation requirements and evaluates support, collision, and navigability. However, human-centered and ergonomics-oriented studies show that layouts may remain unusable even when collision-free and within room boundaries if they lack sufficient clearance for approach, interaction, or circulation (Fisher et al., 2015; Qi et al., 2018; Leimer et al., 2022). Practical spatial usability should therefore be evaluated as a distinct dimension rather than inferred from distributional quality or physical validity. Roomer-Eval complements existing metrics with deterministic and reproducible rules for functional organization, operational clearance, and circulation in residential layouts.

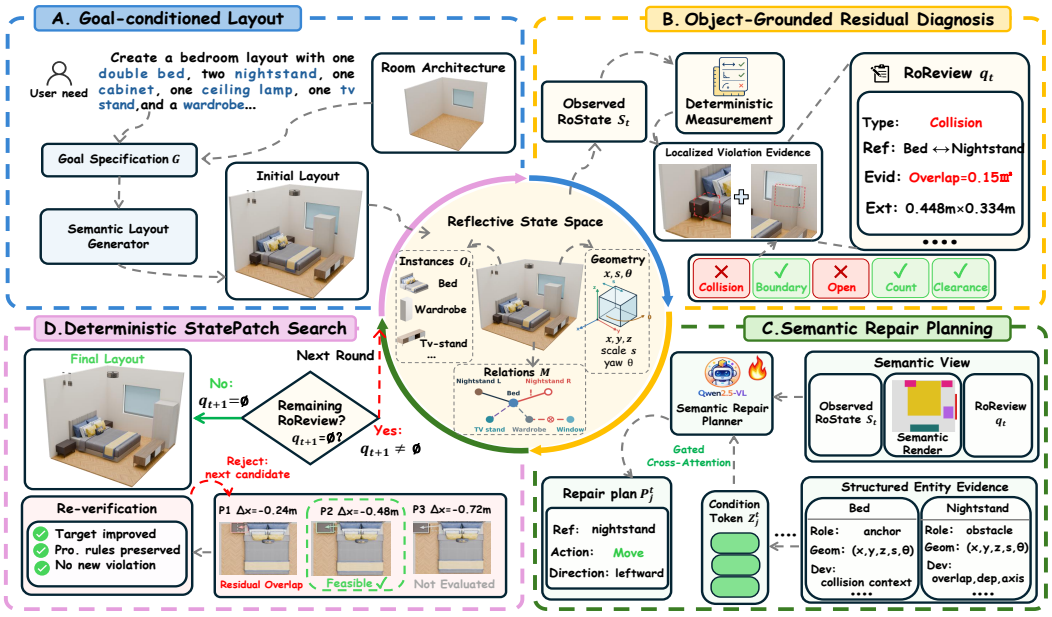

Figure 2: Overview of Roomer. A generated layout is converted into an object-addressable RoState and evaluated to construct an instance-grounded RoReview. The geometry-conditioned vision-language model planner then predicts an action-specific StatePatch seed; the deterministic solver evaluates the seed first and traverses an ordered fallback sequence only after rejection. The first candidate that passes full-scene verification is committed; otherwise, the transaction is rolled back.

3 METHOD

3.1 PROBLEM FORMULATION AND OVERVIEW

Roomer repairs residual violations in a complete indoor layout through verified local edits. Given an architectural environment $\mathcal{A}$, a target room specification $\mathcal{G}$ containing textual user requirements and structured room constraints, and an initial layout $\mathcal{Y}^0$ produced by an arbitrary upstream generator, Roomer repeatedly (i) attributes one detected violation to the implicated scene entities, (ii) predicts a local StatePatch, and (iii) commits an instantiated candidate only after full-scene verification.

At each planning attempt $t \in \{0, \ldots, K-1\}$, the committed layout is updated to $\mathcal{Y}^{t+1} = \mathcal{Y}^t \oplus p^{t,\star}$ only when a candidate $p^{t,\star}$ passes verification; otherwise $\mathcal{Y}^{t+1} = \mathcal{Y}^t$, and the rejected attempt still consumes the planning budget. Here $\oplus$ denotes an object-level state update. Roomer accepts outputs from any upstream generator provided they can be converted to its canonical representation (Fig. 2).

Initial layout generation. We use a Qwen-Image-based upstream generator, fine-tuned to map an architectural condition map and structured room prompt to a fixed-palette top-down semantic layout. A parser converts this output into the structured initial layout $\mathcal{Y}^0$ by mapping RGB values to semantic labels, filtering artifacts and room-incompatible categories, extracting connected components as furniture instances, and augmenting their planar geometry with attributes predicted by a pretrained model.

3.2 INSTANCE-GROUNDED REPAIR CONTEXT

At attempt t, the object-addressable RoState is $\mathcal{S}^t = (\mathcal{A}, \mathcal{Y}^t, \mathcal{F}^t, \Pi^t)$, where $\mathcal{F}^t$ contains geometry-derived functional regions and Π^t maps stable references to scene entities. From the detected rules, the scheduler selects $v^t = (r^t, \rho^t)$, containing its family and participating references.

The active instance becomes a RoReview tuple $q^t = (\kappa(v^t), \tau^t, \mathcal{E}^t, \mathcal{W}^t, \boldsymbol{\delta}^t)$, encoding its stable key, type, entities, roles, and measurements. For example, a bedside-clearance review identifies the

bed, protected clearance region, obstructors, and measured deficit. RoReview provides attributable evidence without prescribing an action; details are in Appendix A.

3.3 Geometry-Conditioned StatePatch Planning

For active review q^t, the planner receives the semantic rendering I^t, room specification $\mathcal{G}$, and serialized RoState and RoReview. Stable references align this semantic context with geometric evidence needed for local feasibility.

Each participating entity is encoded by its category, role, editability, normalized floor-plane geometry, and type-specific measurements. Learnable Repair Queries aggregate the masked evidence as $\mathbf{Z}^t = E_\phi(\mathbf{Q}, \mathbf{X}^t; \mathbf{M}^t)$, where $\mathbf{M}^t$ masks padded entities. Let $\bar{\mathbf{H}}^t_{\ell,\mathrm{sa}} = \mathrm{Norm}(\mathbf{H}^t_{\ell,\mathrm{sa}})$ and $\bar{\mathbf{Z}}^t = \mathrm{Norm}(\mathbf{Z}^t)$ denote the normalized language and geometry states, respectively. At selected decoder layers, the geometry tokens condition the language states through a residual cross-attention adapter:

$$\widetilde{\mathbf{H}}^t_\ell = \mathbf{H}^t_{\ell,\mathrm{sa}} + \gamma_\ell \mathbf{W}^o_\ell \,\mathrm{CrossAttn}_\ell\big(\mathbf{W}^q_\ell \bar{\mathbf{H}}^t_{\ell,\mathrm{sa}}, \bar{\mathbf{Z}}^t, \bar{\mathbf{Z}}^t\big)\,. \tag{1}$$

The planner predicts the action, target, and parameter seed $\widehat{p}^t = (\widehat{a}^t, \widehat{\boldsymbol{\xi}}^t, \widehat{\boldsymbol{\alpha}}^t_0)$. $\widehat{a}^t$ selects `MOVE`, `ROTATE`, `SCALE`, `INSERT`, `DELETE`, or `REPLACE`; the solver realizes precise candidates rather than committing the seed directly. Details and examples are in Appendix D and Fig. A1.

3.4 Deterministic Instantiation and Verification-Gated Commit

After validating the schema, references, and action, the solver tests the normalized seed, then an action-specific neighborhood ordered by seed distance and edit magnitude. It returns the first verified candidate or $\varnothing$; domains and fallback orders are in Appendix G.

The edits change floor-plane position, yaw, and footprint size (Sun et al., 2026); because RoState footprints are not certified envelopes of the assembled meshes, the floor-plane gate does not guarantee mesh-level collision or boundary validity (Feng et al., 2025).

Each candidate p is applied only to a temporary layout $\mathcal{Y}^t_p \triangleq \mathcal{Y}^t \oplus p$, after which all functional regions and rule instances are recomputed. Let $\mathcal{H}$ be the stable hard-violation key set emitted by the floor-plane rule evaluator, $\mathcal{Z}^t_{\mathrm{keep}}$ the protected satisfied relations, $\mathcal{C}$ the structural-validity predicate, and $\mathcal{V}$ the family-balanced residual over hard, content, relational, and practical repair rules. For compactness, write $\mathcal{F}^t_p = \mathcal{F}(\mathcal{Y}^t_p; \mathcal{A}, \mathcal{G})$ and $\mathcal{F}^t = \mathcal{F}(\mathcal{Y}^t; \mathcal{A}, \mathcal{G})$ for $\mathcal{F} \in \{\mathcal{H}, \mathcal{Z}, \mathcal{C}, \mathcal{V}\}$. The candidate is accepted if and only if

$$\mathsf{Acc}_t(p) = 1 \Longleftrightarrow \begin{cases} \mathsf{Target}_t(p) = 1, \\ \mathcal{H}^t_p \setminus \mathcal{H}^t = \varnothing, \\ \mathcal{Z}^t_{\mathrm{keep}} \subseteq \mathcal{Z}^t_p, \\ \mathcal{C}^t_p = 1, \\ \mathcal{V}^t_p \le \mathcal{V}^t - \varepsilon_V. \end{cases} \tag{2}$$

The gate allows unresolved pre-existing hard issues but does not certify mesh COL or OOB. Unresolved issues are blocked until a successful commit changes the geometry; the loop ends when no issue remains, all are blocked, or budget K is exhausted (Appendix F).

3.5 Controlled-Corruption Supervision

Roomer-CC derives object-grounded repair supervision from valid 3D-FRONT layouts that satisfy their target room specifications (Fu et al., 2021a). For each example, we apply one of six object-level corruptions and recompute the observation, RoState, RoReview, and geometric evidence. The corrupted layout is the input and its known inverse is a feasible StatePatch target.

We retain a pair only when the intended violation is triggered, attributed to the expected object or role, and removed by the inverse StatePatch without introducing a new hard violation or breaking a protected relation. Construction, splits, and action–issue coverage are provided in Appendix B and G.

Table 1: Density-controlled comparison on *common-1100*. FID, KID, and SCA Gap are unweighted Macro-3 averages; OOB and COL are object-level micro-averages, and Practical is a rule-instance micro-average over all 1,100 scenes. LEGO-Net and Ours-Final refine the same Ours-Initial layouts. Best results are bolded.

Method	FID↓	KID×10^3↓	SCA Gap↓	OOB↓	COL↓	Practical↑
Generation						
DiffuScene-RS [CVPR 2024]	68.15	11.81	16.57	38.18%	28.44%	45.09%
InstructScene [ICLR 2024]	64.90	7.21	19.98	37.49%	28.50%	40.99%
SemLayoutDiff-RS [3DV 2026]	90.68	29.68	34.12	49.43%	60.42%	57.77%
ReSpace [ES-Reasoning @ ICLR 2026]	**57.42**	**5.10**	21.73	14.72%	36.56%	66.43%
Ours-Initial	64.50	10.80	19.20	24.40%	29.52%	72.50%
Refinement / Repair						
LEGO-Net [CVPR 2023]	61.80	8.46	16.25	13.90%	21.35%	78.10%
Ours-Final	60.20	7.82	**12.46**	**8.66%**	**17.50%**	**82.98%**

Continuous parameters are serialized in the StatePatch and learned with a field-weighted next-token objective emphasizing action and target tokens (Appendix H).

4 Roomer-Eval: Unified Evaluation Protocol

Roomer-Eval measures distributional quality with FID, KID, and SCA Gap on shared semantic renderings, and reports assembled-mesh OOB and COL as independent object-level physical-validity diagnostics. These differ from scene-level Hard Validity under the floor-plane repair evaluator; passing the verification gate does not imply zero mesh OOB or COL. FID/KID measure similarity to the reference distribution, but valid repairs may differ from it because a room admits multiple functional layouts. We therefore treat them as complementary metrics rather than direct measures of repair effectiveness. Practical captures functional failures even in collision-free, in-bounds layouts. Although aligned with Roomer's objective, it is a fixed rule-based metric applied equally to all methods.

For rule instance j in scene i, let $a_{ij}, s_{ij} \in \{0, 1\}$ denote applicability and satisfaction. Practical is the micro-average over applicable instances:

$$\text{Practical} = \frac{\sum_{i,j} a_{ij} s_{ij}}{\sum_{i,j} a_{ij}}. \tag{3}$$

The five rule families cover living-room functional organization, dining-table clearance, door swing-proxy avoidance, walkable connectivity, and bedside clearance (full definitions in Appendix J). Final scores are recomputed from each method's complete output under the same frozen rules, without reusing repair-time detections or decisions.

5 Experiments

Unless otherwise stated, all quantitative experiments use the frozen *common-1100* cohort with a shared evaluator and at most ten repair rounds; professional validation uses a separate frozen sample.

5.1 Experimental Setup

Data and baselines. The frozen *common-1100* cohort contains Qwen-Image outputs (Wu et al., 2025) for 1,100 held-out 3D-FRONT/3D-FUTURE rooms (Fu et al., 2021a;b): 777 bedrooms, 155 living rooms, and 168 dining rooms. It is excluded from all training and model selection. Roomer-CC contains 61,010 training, 3,270 validation, and 3,270 held-out test examples. We compare with ReSpace (Bucher & Armeni, 2026), DiffuScene (Tang et al., 2024), InstructScene (Lin & Mu, 2024), and SemLayoutDiff (Sun et al., 2026), retaining each method's native conditioning, asset-retrieval, and assembly pipeline. Density-controlled selection uses only output validity and floor-standing object count, never evaluation metrics. Complete split provenance, baseline interfaces, and selection rules are provided in the supplementary material.

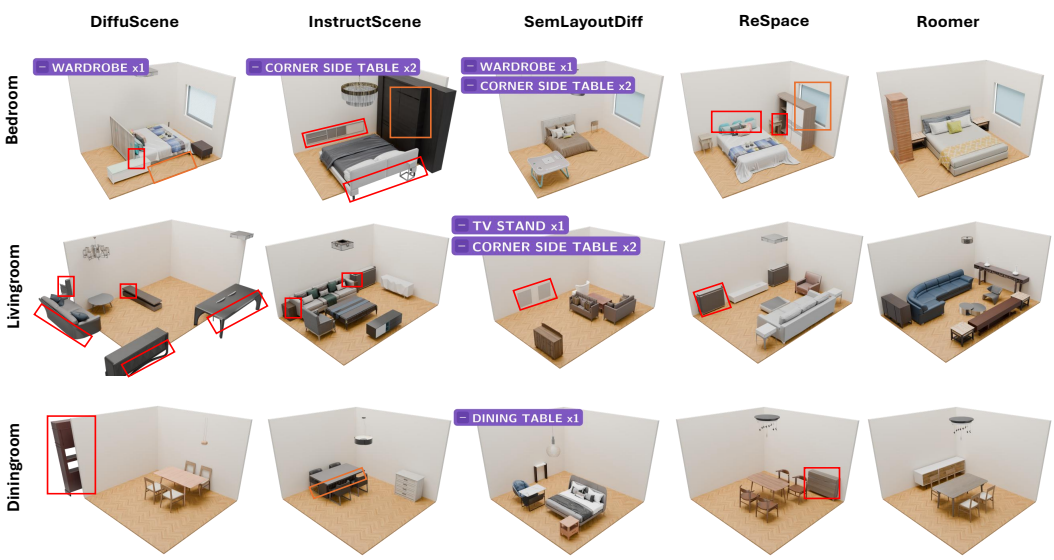

Figure 3: Representative final-layout comparisons on *common-1100*. Red boxes highlight geometric violations, orange boxes indicate practical violations, and purple tags denote unmet furniture requirements caused by missing or mismatched objects.

Implementation. Ours-Initial is generated at 512×512 resolution using Qwen-Image (Wu et al., 2025) with rank-64 DiT LoRA. The repair planner is initialized from Qwen2.5-VL-7B-Instruct (Bai et al., 2025), with the backbone frozen and rank-8 LoRA adapters and geometry-conditioning modules optimized. The geometry branch encodes up to 16 evidence entities as 512-dimensional tokens, aggregates them with eight Repair Queries, and injects residual cross-attention into decoder layers 22, 24, 26, and 28.

Qwen-Image is trained for four epochs with AdamW on one NVIDIA A800 80GB GPU. The planner is trained for five epochs on two A800 GPUs using learning rates of $1e-5$ and $1e-4$ for the LoRA and geometry parameters, respectively, cosine scheduling with 5% warmup, and an effective batch size of 32. Full optimization details are provided in the supplementary material.

5.2 Density-Controlled Final-Layout Comparison

Table 1 shows that Ours-Final leads on SCA Gap, OOB, COL, and Practical, whereas ReSpace leads on FID and KID. Relative to Ours-Initial, Roomer reduces OOB and COL by 15.74 and 12.02 percentage points and increases Practical by 10.48 points; all three distributional metrics also improve. On the same Ours-Initial layouts, Roomer outperforms LEGO-Net (Wei et al., 2023) on all six metrics. This may partly reflect Roomer's broader edit space: LEGO-Net translates and rotates existing objects, whereas Roomer can also insert, delete, and replace furniture. Metric interpretation and the floor-plane gate's scope are discussed in Sec. 4. Figure 3 illustrates the differences behind the aggregate scores across three room types.

5.3 Repairing Outputs from External Generators

We apply the same frozen repair procedure to outputs from four external generators, retaining all 1,100 scenes per generator and rolling back rejected edits. OOB, COL, and Practical improve for every generator (Table 2). Distributional metrics improve in 10 of 12 comparisons; ReSpace KID and DiffuScene-RS SCA Gap are the exceptions (Table A9). SemLayoutDiff-RS retains the highest OOB and COL after repair, indicating that final quality still depends on the upstream layout. Figure 4 shows representative local repairs that preserve the remaining scene content.

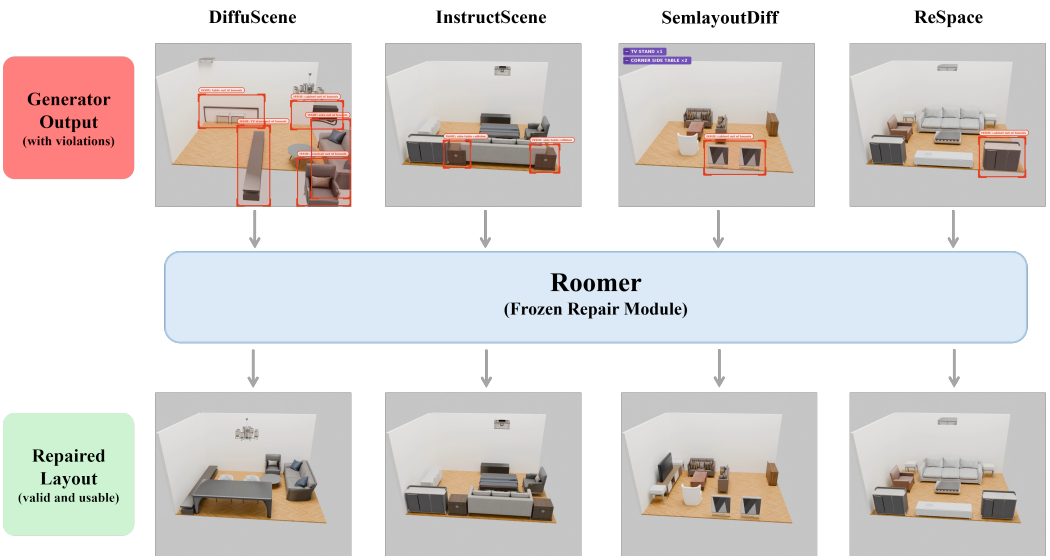

Figure 4: Qualitative plug-and-play repair across four external generators. The top row shows frozen generator outputs with detected violations, and the bottom row shows the corresponding layouts after applying the same frozen Roomer repair procedure. Roomer resolves the highlighted local failures without modifying or retraining the upstream generators.

Table 2: Paired transfer results on frozen outputs from four external generators. All 1,100 scenes are retained. Bold marks the better result within each Before–After pair.

Method	State	OOB↓	COL↓	Practical↑
ReSpace	Before	14.72	36.56	66.43
	After	**1.16**	**7.61**	**75.12**
DiffuScene-RS	Before	38.18	28.44	45.09
	After	**6.98**	**6.95**	**47.73**
InstructScene	Before	37.49	28.50	40.99
	After	**5.67**	**5.88**	**45.70**
SemLayoutDiff-RS	Before	49.43	60.42	57.77
	After	**32.83**	**43.92**	**66.86**

Table 3: Controlled ablation of Roomer components on identical common-1100 initial states. Ours-Init./Final denote $K = 0/10$; all values are percentages.

Variant	Hard↑	Prac.↑	Tgt.↑	New↓	NT↑	Safe↑
Ours-Init.	19.27	72.50	–	–	–	–
w/o RoReview	19.73	50.39	2.44	0.00	94.15	0.45
w/o Geom. Adapt.	62.82	74.90	88.40	0.00	99.81	81.64
Direct Seed	48.00	67.80	65.70	0.00	99.93	55.18
Seed-free	68.82	77.60	90.70	0.00	99.91	89.64
VLM-Free	64.18	76.34	87.42	0.00	99.89	84.09
Full JSON	16.55	49.82	32.48	32.64	48.61	1.91
w/o Verif./RB	23.55	71.72	64.91	54.27	97.63	23.18
Ours-Final	**72.64**	**82.98**	**98.96**	**0.00**	**99.94**	**96.27**

5.4 Controlled Analysis of the Roomer Repair Loop

Protocol and metrics. All variants start from byte-identical layouts and RoReview issues and share the same action space, round budget, and stopping conditions. Hard, Prac., Tgt., New, NT, and Safe denote hard validity, Practical, target resolution, new hard errors, non-target preservation, and strict safe repair, respectively; RB denotes rollback. Controlled metric definitions and round-prefix results are provided in Appendix K.

Component ablation. Table 3 isolates the repair components. Without RoReview, Target Resolution falls to 2.44%, showing the importance of object-grounded diagnosis. Seed-Free Search remains effective, showing that deterministic candidate search supplies substantial coverage, while the planner seed adds further gains. With RoReview and seed-free search fixed, Seed-Free exceeds VLM-Free by 3.28 points in Target Resolution and 5.55 points in Strict Safe Repair, highlighting the VLM's contribution to target and action selection. Removing verification and rollback commits new hard errors; the full loop reaches 96.27% Strict Safe Repair with no new hard errors.

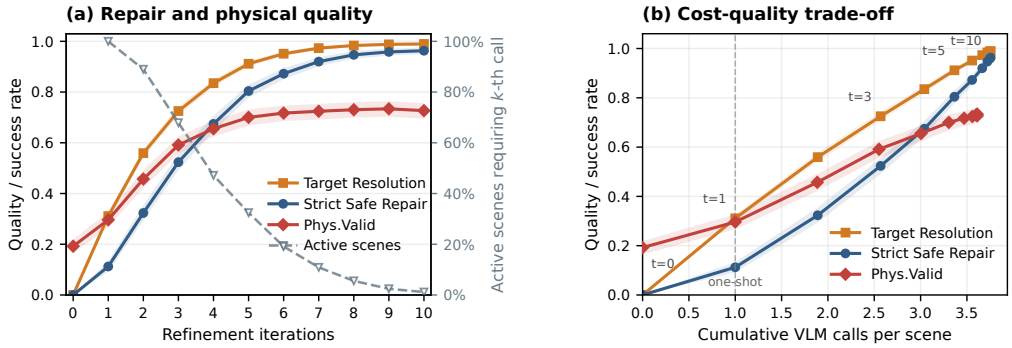

Figure 5: Round-wise convergence and cost–quality trade-off on *common-1100*. Adaptive calls concentrate on a shrinking set of unresolved scenes.

Iterative benefit and adaptive computation. We evaluate $K \in \{1, 3, 5, 7, 10\}$ as prefixes of the same frozen $K = 10$ trajectories, without resampling. Figure 5 shows that most gains occur early, while later rounds improve safe repair for a shrinking set of difficult scenes. Adaptive stopping averages 3.756 planner calls per scene, 62.44% fewer than fixed ten-round execution, while retaining the later-round benefit.

5.5 Professional Validation of Practical Usability

We assess Practical against professional judgment using 90 independent baseline layouts, balanced across room types and Practical strata. Ten evaluators with interior-design experience assess anonymized layouts, blinded to generator identity and Practical scores. Expert approval rises from Low to High Practical ($Z = 4.67$, $p = 3.07 \times 10^{-6}$), supporting the metric's alignment with professional judgment. A separate blinded study compares 30 matched external-generator scenes before and after repair. Approval improves by five scenes (16.7 percentage points), supporting improved usability in this sample. Appendix M details the evaluation protocol.

Table 4: Professional approval in two blinded studies. Low/Medium/High refer to 90 independent baseline layouts; Before/After refer to 30 matched external-generator scenes. Approval requires at least seven of ten evaluators; intervals are Wilson 95% confidence intervals.

Practical Level	Approved	Rate	95% CI
Low	7/30	23.3%	[11.8%, 40.9%]
Medium	17/30	56.7%	[39.2%, 72.6%]
High	25/30	83.3%	[66.4%, 92.7%]
Before	14/30	46.7%	[30.2%, 63.9%]
After	19/30	63.3%	[45.5%, 78.1%]

6 Conclusions

We presented Roomer, a reflective framework for repairing residual violations in 3D indoor layouts. It combines object-grounded RoReview diagnosis, geometry-conditioned StatePatch planning, deterministic candidate search, and verification-gated commitment. Roomer-CC provides paired repair supervision, while Roomer-Eval assesses distributional quality, physical validity, and practical spatial usability. Experiments on frozen layouts and outputs from four external generators show gains in physical validity and Practical, with expert evaluation supporting improved usability in the sampled repaired scenes. These results support object-grounded diagnosis and verified local edits for correcting residual violations while preserving valid scene content.

The gate's non-regression guarantee applies to its floor-plane rules; the independently measured mesh COL and OOB can remain nonzero. Roomer is also limited to violations covered by its predefined residential rule set and to repairs reachable within its finite action-specific candidate spaces; issues outside this coverage or without an acceptable candidate remain unresolved. Future work will extend the rule library and candidate generation to support broader room types and a wider range of functional requirements.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation and to clean and reformat datasets. We have not used generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, or interpret results; assisting in writing proofs and supporting qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to edit the manuscript to improve readability, including refining the presentation of the conceptual framework and assisting with language polishing and grammar checking. We have reviewed all AI-assisted work. Specifically, the authors reviewed AI-assisted translations and dataset transformations against the source materials and manually reviewed all AI-assisted manuscript edits for accuracy and consistency. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The professional validation study involved ten adult evaluators with interior-design experience. All evaluators voluntarily participated and provided informed consent. Evaluators assessed only anonymized indoor-layout images, and the analysis reports aggregate binary judgments without disclosing participant identities. The study involved no more than minimal foreseeable risk to participants.

REPRODUCIBILITY STATEMENT

To support reproducibility, the main paper specifies the Roomer state transition in text and the geometry-conditioning mechanism, verification gate, and Practical aggregation in Eqs. (1), (2), and (3). Appendices B–H provide the Roomer-CC construction procedure and room-level data partitions, complete training configurations for the upstream generator and repair planner, geometry-conditioning details, StatePatch serialization, verification definitions, deterministic candidate domains and ordering, and repair scheduling rules. Appendices I–M document the evaluation rendering and metric implementations, aggregation units, software versions and baseline commit identifiers, candidate budgets and available randomness records, the complete Practical rule definitions, controlled ablation protocols, and the professional-validation procedure. All evaluation cohorts, method configurations, and controlled comparisons are frozen before metric computation, and round-prefix results are derived from the same stored repair trajectories. Assembled-mesh OOB surface sampling did not use a fixed random seed; the reported OOB rates refer to the retained evaluation run (Appendix I).

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report. *CoRR*, abs/2502.13923, 2025. doi: 10.48550/ARXIV.2502.13923. URL https://doi.org/10.48550/arXiv.2502.13923.

Frédéric Berdoz, Luca A. Lanzendörfer, Nick Tuninga, and Roger Wattenhofer. Text-to-Scene with Large Reasoning Models. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *AAAI 2026*, pp. 2435–2443. AAAI Press, 2026. doi: 10.1609/AAAI.V40I4.37229. URL https://doi.org/10.1609/aaai.v40i4.37229.

Mikolaj Binkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=r1lUOzWCW.

Martin JJ. Bucher and Iro Armeni. ReSpace: Text-Driven Autoregressive 3D Indoor Scene Synthesis and Editing. In *ES-Reasoning Workshop at ICLR 2026*, 2026. URL https://openreview.net/forum?id=WPP6ThdEzk.

Lingwei Dang, Ruizhi Shao, Hongwen Zhang, Wei Min, Yebin Liu, and Qingyao Wu. SViMo: Synchronized Diffusion for Video and Motion Generation in Hand-object Interaction Scenarios. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=huZzy5w2Js.

Lingwei Dang, Juntong Li, Zonghan Li, Hongwen Zhang, Liang An, Wei Min, Yebin Liu, and Qingyao Wu. HarmoHOI: Harmonizing Appearance and 3D Motion for Multi-view Hand-Object Interaction Synthesis, 2026. URL https://arxiv.org/abs/2607.17097.

Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3D: End-to-End Generation and Manipulation of 3D Scenes Using Scene Graphs. In *ICCV 2021*, pp. 16352–16361, 2021.

Weitao Feng, Hang Zhou, Jing Liao, Li Cheng, and Wenbo Zhou. CASAGPT: Cuboid Arrangement and Scene Assembly for Interior Design. In *CVPR 2025*, pp. 29173–29182. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.02716. URL https://openaccess.thecvf.com/content/CVPR2025/html/Feng_CASAGPT_Cuboid_Arrangement_and_Scene_Assembly_for_Interior_Design_CVPR_2025_paper.html.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun R. Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional Visual Planning and Generation with Large Language Models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/3a7f9e485845dac27423375c934cb4db-Abstract-Conference.html.

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas A. Funkhouser, and Pat Hanrahan. Example-based synthesis of 3D object arrangements. *ACM Trans. Graph.*, 31(6):135:1–135:11, 2012. doi: 10.1145/2366145.2366154. URL https://doi.org/10.1145/2366145.2366154.

Matthew Fisher, Manolis Savva, Yangyan Li, Pat Hanrahan, and Matthias Nießner. Activity-centric scene synthesis for functional 3D scene modeling. *ACM Trans. Graph.*, 34(6):179:1–179:13, 2015. doi: 10.1145/2816795.2818057. URL https://doi.org/10.1145/2816795.2818057.

Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics. In *ICCV 2021*, pp. 10913–10922. IEEE, 2021a. doi: 10.1109/ICCV48922.2021.01075. URL https://doi.org/10.1109/ICCV48922.2021.01075.

Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Stephen J. Maybank, and Dacheng Tao. 3D-FUTURE: 3D Furniture Shape with TextURE. *Int. J. Comput. Vis.*, 129(12):3313–3337, 2021b. doi: 10.1007/S11263-021-01534-Z. URL https://doi.org/10.1007/s11263-021-01534-z.

Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. AnyHome: Open-Vocabulary Generation of Structured and Textured 3D Homes. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XXXIX*, volume 15097 of *Lecture Notes in Computer Science*, pp. 52–70. Springer, 2024. doi: 10.1007/978-3-031-72933-1_4. URL https://doi.org/10.1007/978-3-031-72933-1_4.

Lin Gao, Jia-Mu Sun, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas, and Jie Yang. SceneHGN: Hierarchical Graph Networks for 3D Indoor Scene Generation With Fine-Grained Geometry. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(7):8902–8919, 2023. doi: 10.1109/TPAMI.2023.3237577. URL https://doi.org/10.1109/TPAMI.2023.3237577.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6626–6637, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved Precision and Recall Metric for Assessing Generative Models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3929–3938, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html.

Kurt Leimer, Paul Guerrero, Tomer Weiss, and Przemyslaw Musialski. LayoutEnhancer: Generating Good Indoor Layouts from Imperfect Data. In Soon Ki Jung, Jehee Lee, and Adam W. Bargteil (eds.), *SIGGRAPH Asia 2022 Conference Papers, SA 2022, Daegu, Republic of Korea, December 6-9, 2022*, pp. 27:1–27:8. ACM, 2022. doi: 10.1145/3550469.3555425. URL https://doi.org/10.1145/3550469.3555425.

Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao (Richard) Zhang. GRAINS: Generative Recursive Autoencoders for INdoor Scenes. *ACM Trans. Graph.*, 38(2):12:1–12:16, 2019. doi: 10.1145/3303766. URL https://doi.org/10.1145/3303766.

Chenguo Lin and Yadong Mu. InstructScene: Instruction-Driven 3D Indoor Scene Synthesis with Semantic Graph Prior. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=LtuRgL03pI.

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Zhaoshuo Li. Scenethesis: A Language and Vision Agentic Framework for 3D Scene Generation. In *ICLR 2026*, 2026. URL https://iclr.cc/virtual/2026/poster/10009362.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87, 2011. doi: 10.1145/2010324.1964982. URL https://doi.org/10.1145/2010324.1964982.

Seongrae Noh, SeungWon Seo, Gyeong-Moon Park, and HyeongYeop Kang. Edit-As-Act: Goal-Regressive Planning for Open-Vocabulary 3D Indoor Scene Editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19864–19873, 2026.

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.

Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-Centric Indoor Scene Synthesis Using Stochastic Grammar. In *CVPR 2018*, pp. 5899–5908. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00618. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Qi_Human-Centric_Indoor_Scene_CVPR_2018_paper.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. LayoutVLM: Differentiable Optimization of 3D Layout via Vision-Language Models. In *CVPR 2025*, pp. 29469–29478. Computer Vision Foundation / IEEE, 2025a. doi: 10.1109/CVPR52734.2025.02744. URL https://openaccess.thecvf.com/content/CVPR2025/html/Sun_LayoutVLM_Differentiable_Optimization_of_3D_Layout_via_Vision-Language_Models_CVPR_2025_paper.html.

Weilin Sun, Xinran Li, Manyi Li, Kai Xu, Xiangxu Meng, and Lei Meng. Hierarchically-Structured Open-Vocabulary Indoor Scene Synthesis with Pre-trained Large Language Model. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI 2025*, pp. 7122–7130. AAAI Press, 2025b. doi: 10.1609/AAAI.V39I7.32765. URL https://doi.org/10.1609/aaai.v39i7.32765.

Xiaohao Sun, Divyam Goel, and Angel X. Chang. SemLayoutDiff: Semantic Layout Generation with Diffusion Model for Indoor Scene Synthesis. In *3DV 2026*, pp. 1545–1555. IEEE, 2026. doi: 10.1109/3DV69130.2026.00147. URL https://doi.org/10.1109/3DV69130.2026.00147.

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 *WACV 2026*, pp. 7355–7365. IEEE, 2026. doi: 10.1109/WACV61042.2026.00710. URL https://doi.org/10.1109/WACV61042.2026.00710.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. DiffuScene: Denoising Diffusion Models for Generative Indoor Scene Synthesis. In *CVPR 2024*, pp. 20507–20518. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01938. URL https://doi.org/10.1109/CVPR52733.2024.01938.

Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. PlanIT: planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Trans. Graph.*, 38(4):132:1–132:15, 2019. doi: 10.1145/3306346.3322941. URL https://doi.org/10.1145/3306346.3322941.

Ruiqi Wang, Qimin Chen, Daniel Ritchie, Angel X. Chang, Manolis Savva, Kai Wang, and Hao Zhang. Function2Scene: 3D Indoor Scene Layout from Functional Specifications. *CoRR*, abs/2605.30819, 2026. doi: 10.48550/ARXIV.2605.30819. URL https://doi.org/10.48550/arXiv.2605.30819.

Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. SceneFormer: Indoor Scene Generation with Transformers. In *3DV 2021*, pp. 106–115. IEEE, 2021. doi: 10.1109/3DV53792.2021.00021. URL https://doi.org/10.1109/3DV53792.2021.00021.

Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. LEGO-Net: Learning regular rearrangements of objects in rooms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19037–19047, 2023.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Shengming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-Image Technical Report. *CoRR*, abs/2508.02324, 2025. doi: 10.48550/ARXIV.2508.02324. URL https://doi.org/10.48550/arXiv.2508.02324.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. SAGE: Scalable Agentic 3D Scene Generation for Embodied AI. In *CVPR 2026*, pp. 22358–22368, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Xia_SAGE_Scalable_Agentic_3D_Scene_Generation_for_Embodied_AI_CVPR_2026_paper.html.

Chucheng Xiang, Ruchao Bao, Biyin Feng, Wenzheng Wu, Zhongyuan Liu, Yirui Guan, and Ligang Liu. Co-Layout: LLM-driven Co-optimization for Interior Layout. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *AAAI 2026*, pp. 14371–14379. AAAI Press, 2026. doi: 10.1609/AAAI.V40I17.38452. URL https://doi.org/10.1609/aaai.v40i17.38452.

Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. PhyScene: Physically Interactable 3D Scene Synthesis for Embodied AI. In *CVPR 2024*, pp. 16262–16272, 2024a. doi: 10.1109/CVPR52733.2024.01539. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yang_PhyScene_Physically_Interactable_3D_Scene_Synthesis_for_Embodied_AI_CVPR_2024_paper.html.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. SceneWeaver: All-in-One 3D Scene Synthesis with an Extensible and Self-Reflective Agent. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/cd2b3c429c8a2ca57656970e010b4b60-Abstract-Conference.html.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language Guided Generation of 3D Embodied AI Environments. In *CVPR 2024*, pp. 16277–16287. IEEE, 2024b. doi: 10.1109/CVPR52733.2024.01536. URL https://doi.org/10.1109/CVPR52733.2024.01536.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Lap-Fai Yu, Sai Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30 (4):86, 2011. doi: 10.1145/2010324.1964981. URL https://doi.org/10.1145/2010324.1964981.

Yang Zhao, Shizhao Sun, Meisheng Zhang, Yingdong Shi, Xubo Yang, and Jiang Bian. SceneReVis: A Self-Reflective Vision-Grounded Framework for 3D Indoor Scene Synthesis via Multi-turn RL. *CoRR*, abs/2602.09432, 2026. doi: 10.48550/ARXIV.2602.09432. URL https://doi.org/10.48550/arXiv.2602.09432.

Kaizhi Zheng, Xiaotong Chen, Xuehai He, Jing Gu, Linjie Li, Zhengyuan Yang, Kevin Lin, Jianfeng Wang, Lijuan Wang, and Xin Eric Wang. EditRoom: LLM-parameterized Graph Diffusion for Composable 3D Room Layout Editing. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=Y2Dh8rWwlb.

APPENDIX

Appendix organization. Figure A1 provides an early visual overview of Roomer's action space, while main-paper Figure 4 demonstrates plug-and-play transfer across external generators. For more vivid demonstrations of the dynamic behavior of our method, please refer to the **supplementary videos** provided alongside this document. Appendices A–H document the method and reproducibility chain, from instance-grounded context and Roomer-CC supervision to geometry conditioning, verification, deterministic search, and StatePatch learning; Appendices I–M contain the main empirical support, including the complete evaluation protocol, Practical rules, controlled analyses, additional ablations, and professional validation.

A INSTANCE-GROUNDED REPAIR CONTEXT

Canonical RoState representation. At planning attempt t, Roomer encodes the current scene as the canonical RoState $\mathcal{S}^t = (\mathcal{A}, \mathcal{Y}^t, \mathcal{F}^t, \Pi^t)$, where $\mathcal{A}$ is immutable architecture, $\mathcal{Y}^t$ is the committed furniture layout, $\mathcal{F}^t$ contains geometry-derived functional regions, and Π^t maps stable object references to concrete scene entities. These references preserve object grounding across the semantic rendering, numerical state, structured geometric evidence, planner output, and StatePatch execution, preventing a predicted edit from being applied to a different object instance.

Functional regions and temporary state updates. The current implementation constructs door-protection regions, bedside-clearance regions, dining-table operation regions, and primary circulation areas. These regions are derived from the current object geometry rather than stored as immutable annotations. Whenever an associated object is translated, rotated, or scaled, Roomer applies the candidate only to a temporary layout and reconstructs the affected functional regions before rerunning the rule evaluator. INSERT, DELETE, and REPLACE similarly update the temporary entity registry and every functional region whose applicability or participating references change.

Instance-grounded RoReview construction. The evaluator converts every detected rule instance $v^t = (r^t, \rho^t)$ into the instance-grounded RoReview entry defined in the main paper. Its stable key $\kappa(v^t)$ links the serialized entry to the underlying rule instance; τ^t records the violation type, $\mathcal{E}^t$ contains implicated furniture, architectural elements, and constructed functional regions, $\mathcal{W}^t$ assigns roles such as functional anchor, protected region, obstructing object, or expected category, and $\boldsymbol{\delta}^t$ records continuous geometric evidence. Depending on the rule, this evidence includes clearance deficit, intrusion or overlap ratio, overlap depth or area, out-of-bounds distance or area, opening-region intrusion, angular deficit, or passage-width deficit.

For a bedside-clearance violation, the bed is the functional anchor, the generated side-clearance region is protected, and furniture intersecting that region is identified as an obstructing object. For a door-blocking violation, the architectural opening and its swing proxy are protected, while intersecting furniture is implicated as the obstruction. RoReview therefore binds each measured violation to the concrete entities and relational roles needed for local repair, but deliberately does not prescribe the StatePatch target or action; that semantic decision remains the planner's responsibility. Stable issue identities, severity normalization, scheduling, and blocking behavior are specified in Appendices F and G.

B ROOMER-CC: CONTROLLED-CORRUPTION SUPERVISION AND DATA PROVENANCE

Controlled-corruption generation. Roomer-CC starts from valid 3D-FRONT layouts that satisfy their target room specifications. Each source layout generates multiple independent outer corruption proposals. For every proposal, the pipeline selects one operator from `MOVE`, `ROTATE`, `SCALE`, `INSERT`, `DELETE`, or `REPLACE` and modifies position, orientation, footprint scale, object presence, or object category. The corrupted layout is rendered and converted into a fresh RoState, RoReview, and structured evidence set; no evidence is copied from the valid source state. Construction-time verification determines which proposals become retained Roomer-CC instances, yielding the split-specific multiplicities reported in Table A5.

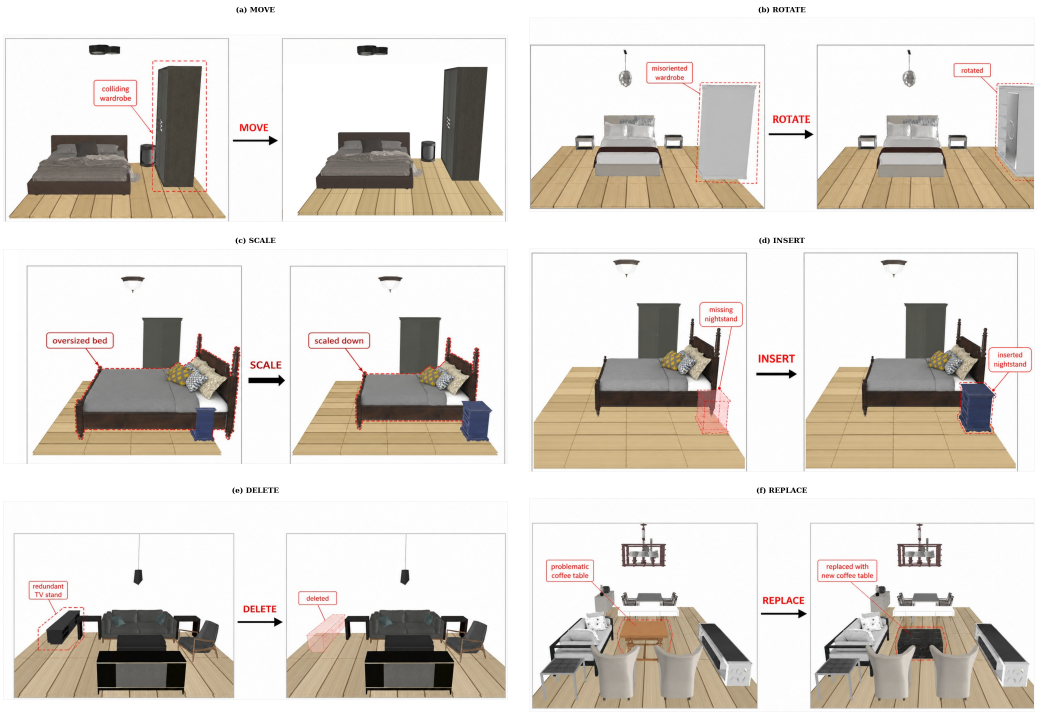

Figure A1: Representative before-and-after examples of the six StatePatch actions: `MOVE`, `ROTATE`, `SCALE`, `INSERT`, `DELETE`, and `REPLACE`. Red annotations indicate the target object or affected region.

Verified inverse StatePatch supervision. Each retained instance pairs the corrupted observation and RoState with instance-grounded violation evidence and a known-feasible inverse StatePatch $a = c^{-1}$. INSERT and DELETE corruptions are supervised by DELETE and INSERT StatePatches, respectively, while MOVE, ROTATE, SCALE, and REPLACE use inverse parameters or the original source attributes. For a DELETE corruption, the INSERT target is represented through action-specific `new_instance` fields rather than a reference to an observed object. Construction-time verification retains an instance only when (i) the intended violation is triggered, (ii) the implicated object or missing role is correctly identified, and (iii) the inverse StatePatch resolves the target violation without introducing a new hard violation or breaking a protected satisfied relation. The executable target is serialized under the same StatePatch schema used during inference.

Construction attempts and retention. A construction *attempt* is one outer corruption proposal. For examples whose inverse repair action is `DELETE`, construction applies an `INSERT` corruption and evaluates up to 120 internal placement samples for the inserted object within the same outer proposal. These internal samples therefore count as one attempt rather than 120 attempts. Table A1 reports the construction flow for the 61,010-instance Roomer-CC training split. *Retained/Attempts* is the end-to-end retention rate.

The corresponding stage-wise rates are

$$\begin{aligned} \frac{70{,}418}{81{,}867} &= 86.02\%, \\ \frac{68{,}789}{70{,}418} &= 97.69\%, \\ \frac{61{,}010}{68{,}789} &= 88.69\%. \end{aligned} \tag{A1}$$

The end-to-end retention rate is

$$\text{Retention} = \frac{\text{Retained}}{\text{Attempts}}. \tag{A2}$$

Table A1: Roomer-CC construction statistics for the training split. Rows are indexed by the inverse repair action; for example, `DELETE` supervision is generated by `INSERT` corruptions. An attempt is one outer corruption proposal, and internal placement sampling does not increase the attempt count.

Repair action	Attempts	Triggered	Correctly attributed	Retained	Retained/Attempts
`INSERT`	4,295	4,188	4,167	3,951	92.0%
`DELETE`	18,543	17,060	16,719	13,722	74.0%
`MOVE`	25,103	19,580	18,797	16,317	65.0%
`REPLACE`	15,245	13,873	13,734	12,958	85.0%
`ROTATE`	16,675	14,674	14,381	13,340	80.0%
`SCALE`	2,006	1,043	991	722	36.0%
Total	**81,867**	**70,418**	**68,789**	**61,010**	**74.5%**

Table A2: Normalized severity distribution of retained Roomer-CC corruptions in the training split.

Issue family	Mean	Median	Q25–Q75	P90
Collision	0.19	0.14	0.07–0.25	0.43
Out of bounds	0.25	0.19	0.09–0.35	0.58
Door-access violation	0.29	0.22	0.10–0.40	0.64
Window blocking	0.24	0.18	0.08–0.34	0.57
Wrong orientation	0.14	0.06	0.03–0.10	0.49
Bedside clearance (Roomer-CC)	0.35	0.29	0.15–0.49	0.72
Dining clearance	0.31	0.26	0.13–0.46	0.67
Living functional angle	0.24	0.19	0.09–0.34	0.52
Walkable disconnected	0.40	0.35	0.18–0.59	0.80
Relation distance	0.29	0.23	0.11–0.42	0.66

Table A3: Severity assigned to binary Roomer-CC content rules.

Issue	Severity
Missing required object	1.0
Extra object	1.0
Wrong object category/type	1.0

Severity distribution of retained corruptions. Table A2 summarizes the normalized severity of retained Roomer-CC training corruptions. *Door-access violation* denotes the training-time door-zone usability failure measured by actual blocking severity; the final Roomer-Eval metric instead uses the frozen door swing proxy defined in Appendix J. The wrong-orientation distribution is right-skewed: its mean exceeds its upper quartile because of a long high-severity tail.

Binary content-rule severity. Binary content failures use unit severity. Before RoRe-view construction, every object-category or object-type mismatch is canonicalized as `wrong_object_category_type`. A single object instance therefore contributes at most one category/type issue key.

Roomer-CC and Roomer-Eval bedside definitions. Roomer-CC corruption construction uses the fixed bedside-clearance threshold

$$c_{\text{usable}} \geq 0.45\,\text{m}, \tag{A3}$$

with severity

$$s_{\text{CC}} = \text{clip}\left(\frac{0.45 - c_{\text{usable}}}{0.45}, 0, 1\right). \tag{A4}$$

Roomer-Eval instead scales the requirement by bed width:

$$\frac{c_{\text{usable}}}{w_{\text{bed}}} \geq 0.30, \tag{A5}$$

Table A4: Qwen-Image room-level partitions.

Split	Bedroom	Living room	Dining room	Total
Train	2,822	543	587	3,952
Validation	409	77	85	571
Test	777	155	168	1,100

Table A5: Roomer-CC source-room partitions and verified corruption instances.

Split	Bedroom	Living room	Dining room	Source rooms	Instances	Inst./room
Train	2,539	489	528	3,556	61,010	17.16
Validation	142	27	29	198	3,270	16.52
Test	141	27	30	198	3,270	16.52
Total	2,822	543	587	3,952	67,550	17.09

with severity

$$s_{\text{Eval}} = \text{clip}\left(\frac{0.30 - c_{\text{usable}}/w_{\text{bed}}}{0.30}, 0, 1\right). \tag{A6}$$

The two severities are computed and reported separately. The final Practical metric consists only of bedside ratio, dining clearance, living functional angle, door proxy, and walkable connectivity. Collision, out-of-bounds placement, window blocking, and binary content rules are reported separately and do not enter the five Practical families.

Room-level data partitions. The data pipeline uses hierarchical room-level partitions. The Qwen-Image train, validation, and test splits are disjoint. Roomer-CC is constructed exclusively from the 3,952 Qwen-Image training rooms, which are further partitioned before corruption generation so that a source room and all derived corruptions remain in one planner split. Its test split is therefore held out from vision-language model planner training and configuration selection, whereas the separate Qwen-Image test split forms *common-1100* and is excluded from all training and tuning.

The Qwen-Image training split is used for model training and as the Roomer-CC source-room pool. Its validation split is used for Qwen-Image checkpoint selection and system hyperparameter development. The test split is the frozen *common-1100* cohort and is not used for training or tuning.

The Roomer-CC train/validation/test source-room proportions are 89.98%/5.01%/5.01%. The training split is used for vision-language model planner parameter learning, the validation split selects checkpoints and planner configurations, and the test split remains fully held out from training and configuration selection. Corruption multiplicity varies slightly because construction-time verification removes instances that fail to trigger the intended violation or whose paired repair fails verification.

C Qwen-Image Training and Optimization Details

Qwen-Image is fine-tuned to map textual furniture requirements and architectural condition images to 512×512 semantic furniture layouts. The fine-tuning corpus is exactly the 3,952-room Qwen-Image training split in Table A4, comprising 2,822 bedrooms, 543 living rooms, and 587 dining rooms. The 571-room validation split is used only for checkpoint selection, and no room from the frozen *common-1100* test split contributes a fine-tuning example. Only LoRA adapters attached to the diffusion transformer (DiT) are trainable; the base-model weights remain frozen.

Qwen Parsing and 3D Assembly

The Qwen-Image output is decoded into structured furniture instances and assembled with category-compatible 3D assets. The attribute-prediction module (APM) directly resolves yaw for orientation-sensitive objects whenever possible; unresolved cases invoke the deterministic orientation fallback.

Table A6: Complete Qwen-Image fine-tuning configuration. The two stages are executed sequentially: constant-rate adaptation followed by lower-rate cosine refinement.

Configuration	Value	Configuration	Value
Base model	Qwen-Image	Training layouts	3,952
Room-type distribution	2,822 bedroom / 543 living room / 587 dining room	Input	Textual furniture requirements + architectural condition image
Output	512×512 semantic furniture layout	Trainable modules	DiT LoRA only
LoRA rank / alpha / dropout	64 / 64 / 0	Trainable parameters	471,859,200
Optimizer	AdamW	Weight decay	0.01
Batch size / accumulation	1 / 1	Gradient clipping / warmup	none / none
Precision	BF16 computation; frozen base weights in FP8	Hardware	$1\times$ NVIDIA A800 80GB
Stage 1 schedule	Constant learning rate of 5×10^{-5} for 1 epoch	Stage 1 updates	3,952
Stage 2 schedule	Cosine decay from 2×10^{-5} to 2×10^{-6} for 3 epochs	Stage 2 updates	11,856
Total training	4 epochs / 15,808 optimizer updates	Checkpoint selection	best validation performance on the 571-room validation split

Table A7: Qwen parsing, same-category asset retrieval, and 3D assembly statistics on *common-1100*.

Stage	Value	Statistical unit
Palette decode rate	100.00%	Scene level
Structural parse rate	99.82%	Scene level: 1,098/1,100
Component retention rate	98.60%	Component level
APM direct-yaw resolution rate	98.40%	Orientation-required objects
Orientation fallback invocation rate	1.60%	Orientation-required objects
Same-category asset retrieval success	99.20%	Asset queries
Distortion-blocked object rate	0.55%	Asset queries
Empty-category candidate rate	0.20%	Asset queries
Other retrieval failure rate	0.05%	Asset queries
Raw scene assembly valid	**99.09%**	**1,090/1,100 scenes**
Primary frozen-output valid	**99.82%**	**1,098/1,100 scenes**
Final assembly valid after fallback	**100.00%**	**1,100/1,100 scenes**

Table A7 reports the complete parsing, retrieval, and assembly statistics on *common-1100*. Statistical units are stated explicitly because scene-, component-, object-, and query-level rates are not interchangeable.

The raw scene-assembly validity is

$$\frac{1{,}090}{1{,}100} = 99.0909\% \approx 99.09\%. \tag{A7}$$

The deterministic recovery path is

$$1{,}090 \xrightarrow[\text{fallback}]{\text{object-level}} 1{,}098 \xrightarrow{\text{scene-level frozen-candidate fallback}} 1{,}100. \tag{A8}$$

Thus, raw assembly directly succeeds for 1,090 scenes, deterministic object-level fallback recovers eight scenes, and traversal of the frozen scene-level candidate pool recovers the remaining two scenes.

The asset-query outcomes are mutually exclusive and exhaustive:

$$\begin{aligned} 100\% - 99.20\% &= 0.80\%, \\ 0.80\% &= 0.55\% + 0.20\% + 0.05\%. \end{aligned} \tag{A9}$$

Table A8: Geometry-conditioning configuration.

Component	Configuration
Maximum evidence entities	16
Entity-token dimension	512
Repair Queries K_c	8
Aggregator layers	2
Attention heads	8
Per-head dimension	64
FFN dimension	2048
Attention dropout	0
FFN dropout	0

or equivalently

$$99.20\% + 0.55\% + 0.20\% + 0.05\% = 100.00\%. \tag{A10}$$

D GEOMETRY-CONDITIONING IMPLEMENTATION

Aligned semantic and geometric repair context. For each active violation, the semantic stream contains the current top-down semantic rendering I^t of $\mathcal{Y}^t$, the target room specification $\mathcal{G}$ with its textual user requirements, the complete human-readable RoState, and the selected instance-grounded RoReview entry. Structured geometric evidence is encoded separately but uses the same stable object references as the textual state and StatePatch schema. Consequently, object grounding remains aligned across rendered regions, human-readable descriptions, evidence tokens, and executable edit targets.

Evidence entities. Each RoReview retains at most $N_e = 16$ evidence entities. If more are available, we retain (1) functional anchors, (2) obstructing objects, (3) protected architectural elements or functional regions, and (4) neighboring objects ordered by violation relevance and geometric distance. Remaining positions are padded and masked in every attention operation.

For room width W, depth D, and lower floor-plane bounds $(x_{\min}, z_{\min})$, numerical geometry is normalized as

$$\widetilde{x} = \frac{x - x_{\min}}{W}, \qquad \widetilde{z} = \frac{z - z_{\min}}{D}. \tag{A11}$$

$$\widetilde{w} = \frac{w}{W}, \qquad \widetilde{d} = \frac{d}{D}. \tag{A12}$$

$$\widetilde{d}_{ij} = \frac{d_{ij}}{\sqrt{W^2 + D^2}}. \tag{A13}$$

Yaw is represented by $(\sin\theta, \cos\theta)$ to avoid the discontinuity between -180° and 180°. Clearance deficit, intrusion ratio, and overlap ratio use their rule-specific values already normalized to $[0, 1]$. Each entity token concatenates learned embeddings for category, entity type, relational role, violation type, and editability with normalized center, footprint size, yaw sine/cosine, pairwise distance, and type-specific violation measurements; the result is projected to $d_e = 512$ and normalized before aggregation.

Decoder injection. Qwen2.5-VL-7B-Instruct has 28 language-decoder layers with hidden size 3,584. Geometry-conditioning adapters are inserted at 1-based layers $\{22, 24, 26, 28\}$, corresponding to zero-based implementation indices $[21, 23, 25, 27]$. Each geometry-conditioning adapter uses eight-head, 512-dimensional cross-attention. Language states are projected from 3,584 to 512 dimensions before cross-attention and projected back to 3,584 dimensions before residual addition.

Initialization. Learned evidence embeddings and Repair Queries are sampled from $\mathcal{N}(0, 0.02^2)$. Numerical and evidence-output projections, aggregator attention and FFN matrices, and each geometry-conditioning adapter's query and cross-attention matrices use Xavier-uniform initialization; all associated biases are zero, and LayerNorm weights and biases are initialized to one and zero. For every injected layer, $\mathbf{W}_\ell^o = \mathbf{0}$ and $\gamma_\ell = 1$. Thus, the geometry branch initially contributes

Table A9: Paired distributional-quality results before and after applying Roomer to external generators on *common-1100*. Ten of the twelve paired comparisons improve. Light shading denotes the refined state, bold indicates the better value within each Before–After pair, KID is reported as $\times 10^3$, and all 1,100 scenes are retained.

Method	State	FID↓	KID $\times 10^3$↓	SCA Gap↓
ReSpace	Before	57.42	**5.10**	21.73
	After	**56.99**	6.13	**18.49**
DiffuScene-RS	Before	68.15	11.81	**16.57**
	After	**63.05**	**8.48**	22.08
InstructScene	Before	64.90	7.21	19.98
	After	**56.06**	**2.64**	**17.88**
SemLayoutDiff-RS	Before	90.68	29.68	34.12
	After	**80.57**	**18.75**	**26.49**

zero and the modified network matches the pretrained backbone, while the nonzero gate preserves gradient flow to the output projection.

E Training and Optimization Details

The base language and vision weights remain frozen throughout training. AdamW uses four parameter groups: LoRA parameters use learning rate 10^{-5} and zero weight decay; geometry matrix weights use learning rate 10^{-4} and weight decay 0.01; geometry biases, normalization parameters, and Repair Queries use learning rate 10^{-4} and zero weight decay; residual gates use the same learning rate and zero weight decay. Geometry matrix weights include the evidence-output projection, aggregator attention and FFN projections, and the geometry-conditioning adapter query, cross-attention, and output matrices.

F Verification-Gated Commit: Formal Definitions

Rule instances and normalized severity. Let $\mathcal{R}$ contain all active repair-rule families, including hard, content, relational, and practical violations. For each rule family $r \in \mathcal{R}$, let $\mathcal{I}_r(\mathcal{Y}; \mathcal{A}, \mathcal{G})$ be its applicable instance set. Each instance is $v = (r, \rho)$, where ρ contains stable entity references and $\mathrm{refs}(v)$ denotes the participating reference set. Its normalized severity satisfies $\bar{e}_v(\mathcal{Y}; \mathcal{A}, \mathcal{G}) \in [0, 1]$, with zero denoting satisfaction. Representative continuous definitions are

$$\bar{e}_v^{\mathrm{clear}}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \mathrm{clip}\left(\frac{c_{\mathrm{req}} - c_{\mathrm{obs}}}{c_{\mathrm{req}}}, 0, 1\right). \tag{A14}$$

$$\bar{e}_v^{\mathrm{OOB}}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \mathrm{clip}\left(\frac{A_{\mathrm{outside}}}{A_{\mathrm{footprint}}}, 0, 1\right). \tag{A15}$$

$$\bar{e}_v^{\mathrm{angle}}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \mathrm{clip}\left(\frac{\theta_{\mathrm{req}} - \theta_{\mathrm{obs}}}{\theta_{\mathrm{req}}}, 0, 1\right). \tag{A16}$$

Binary composition or category violations use $\bar{e}_v \in \{0, 1\}$. The stable issue key is $\kappa(v) = (r, \mathrm{sort}(\rho))$; the implementation serializes it as `issue_type|sorted_refs`.

Stable hard-violation keys. Let $\mathcal{R}_{\mathrm{hard}}$ contain collision, out-of-floor-boundary, and architectural-opening blockage families. The hard-violation key set is

$$\mathcal{H}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \left\{ \kappa(v) \;\middle|\; \begin{array}{c} r \in \mathcal{R}_{\mathrm{hard}}, \\ v \in \mathcal{I}_r(\mathcal{Y}; \mathcal{A}, \mathcal{G}), \\ \bar{e}_v(\mathcal{Y}; \mathcal{A}, \mathcal{G}) > 0 \end{array} \right\}. \tag{A17}$$

Using stable keys rather than only a hard-error count ensures that a candidate cannot remove one hard violation while introducing a different one.

Table A10: Complete vision-language model planner training configuration. Source rooms are partitioned before corruption generation, preventing a room and its derived corruptions from crossing splits.

Configuration	Value	Configuration	Value
Base model	Qwen2.5-VL-7B-Instruct	Roomer-CC instances	61,010 / 3,270 / 3,270
LoRA rank / alpha / dropout	8 / 16 / 0.05	LoRA targets	`q_proj`, `k_proj`, `v_proj`, `o_proj`
Trainable LoRA parameters	5.046M	Trainable geometry parameters	approximately 27.446M
Total trainable parameters	approximately 32.492M	Frozen modules	vision encoder, visual merger, original LM weights
Trainable geometry modules	evidence encoder, Repair Queries, query aggregator, geometry-conditioning adapters	Optimizer	AdamW (`adamw_torch`)
LoRA learning rate / decay	10^{-5} / 0	Geometry learning rate	10^{-4}
Geometry matrix decay	0.01	Bias/norm/query/ gate decay	0
AdamW betas / epsilon	$(0.9, 0.999)$ / 10^{-8}	Gradient clipping	1.0
Scheduler / warmup	cosine / 5% (477 steps)	Epochs / optimizer steps	5 / 9,535
Per-GPU batch / GPUs	4 / 2	Accumulation / effective batch	4 / 32
Maximum sequence length	8,192	Precision	BF16; FP16 disabled
Gradient checkpointing	enabled	DeepSpeed	ZeRO-3; no offload
Hardware	2× NVIDIA A800-SXM4-80GB	Random seed	42
Checkpoint selection	best validation repair performance	Selected checkpoint	step 9,535

Verifier-derived protected satisfied relations. Protection metadata is derived deterministically by the verifier and is not predicted by the planner. Let $\mathcal{O}^t$ be the set of stable references of mutable furniture entities in the committed layout. For a candidate transaction p, define

$$\begin{aligned} T_t(p) &= \mathrm{ChangedRefs}(p) \cap \mathcal{O}^t, \\ P_t(p) &= \mathcal{O}^t \setminus T_t(p). \end{aligned} \tag{A18}$$

where $T_t(p)$ contains every existing furniture reference whose state is modified or deleted. A standard `MOVE`, `ROTATE`, `SCALE`, `REPLACE`, or `DELETE` StatePatch therefore has one changed existing reference, whereas `INSERT` has $T_t(p) = \varnothing$ because it does not alter an existing entity. The action and target are fixed across the seed and its deterministic fallback neighborhood, so T_t and P_t are constant within a standard repair transaction and the candidate argument is omitted below.

For a rule instance v, let $\mathrm{deps}(v) \subseteq \mathcal{O}^t$ denote the mutable furniture references on which its geometry or semantics depend. This dependency set includes the anchor objects of any derived functional region, so a clearance or circulation relation is not treated as independent merely because the region itself has a synthetic reference. Define

$$\mathcal{Z}^t_{\mathrm{keep}} = \left\{ \kappa(v) \;\middle|\; \begin{array}{c} v \in \bigcup_{r \in \mathcal{R}} \mathcal{I}_r(\mathcal{Y}^t; \mathcal{A}, \mathcal{G}), \\ \bar{e}_v(\mathcal{Y}^t; \mathcal{A}, \mathcal{G}) = 0, \\ \mathrm{deps}(v) \subseteq P_t \end{array} \right\}. \tag{A19}$$

and

$$\mathcal{Z}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \left\{ \kappa(v) \;\middle|\; \begin{array}{c} v \in \bigcup_{r \in \mathcal{R}} \mathcal{I}_r(\mathcal{Y}; \mathcal{A}, \mathcal{G}), \\ \bar{e}_v(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = 0 \end{array} \right\}. \tag{A20}$$

Thus, every currently satisfied relation whose mutable dependencies are disjoint from the edited target must remain satisfied after the candidate is applied. Relations that necessarily depend on a moved, scaled, rotated, deleted, or replaced target are not inserted into $\mathcal{Z}^t_{\text{keep}}$; their regressions are still controlled by the aggregate-residual condition. Target room-composition constraints are represented as content-rule instances and remain subject to the active-target, protected-relation, and aggregate-residual conditions rather than being folded into $\mathcal{C}$. The StatePatch schema remains the three-part action–target–parameter-seed representation defined in the main paper and contains no planner-supplied `protected_refs` field.

Structural validity and aggregate residual. The indicator $\mathcal{C}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = 1$ iff all object attributes are finite, sizes are positive, references are unique, asset identifiers are category-compatible, object schemas are valid under $\mathcal{G}$, and the architecture matches the fixed environment $\mathcal{A}$. Action legality is enforced separately during schema, reference, and action validation before candidate instantiation. Structural validity is distinct from physical all-pass: $\mathcal{C}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = 1$ does not imply $\mathcal{H}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \varnothing$. Accordingly, reducing the active violation is insufficient unless the candidate passes the complete five-clause full-scene verification gate in main-paper Eq. (2); this gate does not require every hard-violation key already present before the current transaction to be absent.

To prevent a family with many instances from dominating the residual, each family is first averaged over its applicable instances and all families receive equal weight. In the following display, $\mathcal{I}_r$ abbreviates $\mathcal{I}_r(\mathcal{Y}; \mathcal{A}, \mathcal{G})$:

$$\mathcal{V}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \frac{\sum_{v \in \mathcal{I}_r} \bar{e}_v(\mathcal{Y}; \mathcal{A}, \mathcal{G})}{\max(1, |\mathcal{I}_r|)}. \tag{A21}$$

A family with no applicable instance contributes zero. This yields $\mathcal{V} \in [0, 1]$ under uniform family weights $\lambda_r = 1/|\mathcal{R}|$.

Exact target resolution and global descent. Let the recomputed unsatisfied-issue key set be

$$\mathcal{U}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \left\{ \kappa(v) \;\middle|\; \begin{array}{c} r \in \mathcal{R}, \\ v \in \mathcal{I}_r(\mathcal{Y}; \mathcal{A}, \mathcal{G}), \\ \bar{e}_v(\mathcal{Y}; \mathcal{A}, \mathcal{G}) > 0 \end{array} \right\}. \tag{A22}$$

For the active rule instance v^t represented by q^t, the target predicate used in main-paper Eq. (2) is

$$\mathsf{Target}_t(p) = \mathbf{1}\left[\kappa(v^t) \notin \mathcal{U}(\mathcal{Y}^t_p; \mathcal{A}, \mathcal{G})\right]. \tag{A23}$$

Consequently, both continuously measured and binary targets must be fully resolved under the frozen rule evaluator before a candidate can be committed; a partial severity reduction is insufficient. Each detector applies its rule-specific geometric tolerance before emitting an issue, so Eq. (A23) tests absence of the stable issue key after full recomputation rather than exact equality of unthresholded floating-point geometry. The aggregate residual must additionally satisfy the main-paper condition with $\varepsilon_V = 10^{-6}$, which enforces strict global descent while avoiding rejection caused only by floating-point noise after family-wise averaging.

Correspondence to main-paper schematic labels. Main-paper Fig. 2 uses compact visual shorthand. Its label "Target improved" denotes the detector-level target-resolution test in Eq. (A23); "Pro. rules preserved" abbreviates the protected satisfied relations $\mathcal{Z}^t_{\text{keep}}$ rather than all Practical rule instances; and "No new violation" denotes the no-new-hard-key condition $\mathcal{H}^t_p \setminus \mathcal{H}^t = \varnothing$. The figure's "Repair plan P^t_j" and "Condition Token Z^t_j" correspond, respectively, to the schema-constrained StatePatch proposal $\widehat{p}^t$ and the repair-query-conditioned geometry evidence Z^t. These schematic labels do not replace any clause of main-paper Eq. (2).

G Deterministic Candidate Instantiation and Repair Scheduling

Planner-seed-first candidate ordering. After schema, reference, and action validation, the deterministic solver parses the raw autoregressive output and instantiates it as the normalized executable

seed candidate $\widehat{p}^t$. The raw planner serialization is never committed directly. The normalized executable candidate is evaluated first; if it is rejected, the solver constructs the deterministic action-specific neighborhood

$$\mathcal{N}^t = \mathcal{N}_{\widehat{a}^t}\left(\widehat{\boldsymbol{\xi}}^t, \widehat{\boldsymbol{\alpha}}_0^t; q^t, \mathcal{S}^t\right) \tag{A24}$$

and orders its instantiated candidates as

$$\mathbf{B}^t = \left(b_1^t, \ldots, b_{M_t}^t\right) = \mathrm{Order}\left[\mathrm{Inst}_{\widehat{a}^t}(\mathcal{N}^t)\right]. \tag{A25}$$

If the seed is rejected and at least one fallback passes verification, let $j^\star = \min\{j : \mathsf{Acc}_t(b_j^t) = 1\}$. The selected candidate is

$$p^{t,\star} = \begin{cases} \widehat{p}^t, & \mathsf{Acc}_t(\widehat{p}^t) = 1, \\ b_{j^\star}^t, & j^\star \text{ exists}, \\ \varnothing, & \text{otherwise}. \end{cases} \tag{A26}$$

Thus, when the normalized executable candidate instantiated from the planner seed passes verification, its resulting StatePatch is used without a fallback search. Deterministic numerical search is invoked only after that candidate is rejected and terminates at the first verified fallback. Fallback order is primarily determined by action-specific parameter distance from the seed and then by edit magnitude; the concrete sequences are listed below.

Action execution. All controlled variants use the same primitive action vocabulary $\mathbb{A} = \{\texttt{MOVE}, \texttt{ROTATE}, \texttt{SCALE}, \texttt{INSERT}, \texttt{DELETE}, \texttt{REPLACE}\}$. Standard Roomer emits one target-grounded StatePatch per round. The Full-Scene JSON Rewrite ablation may propose multiple object changes in one transaction, but every difference from the committed state must be decomposable into these same six primitives; unsupported attribute changes and architecture edits are rejected. The protocol does not freeze object identity, category, count, or size; `INSERT`, `DELETE`, `REPLACE`, and `SCALE` may therefore alter these attributes, while room architecture is immutable.

Table A11 records the exact StatePatch scope, seed-first fallback sequence, and principal validity checks. The ordered search always evaluates the normalized executable candidate instantiated from the planner-predicted StatePatch seed before the listed deterministic fallbacks.

Action–issue coverage in Roomer-CC. All retained examples use the canonical issue taxonomy applied by RoReview and the scheduler. The 61,010-instance training split contains `INSERT` supervision for `missing_required_object`; `DELETE` for `extra_object`; `REPLACE` for `wrong_object_category_type`; `MOVE` for `collision`, `out_of_floor_bounds`, `door_blocking`, `window_blocking`, `walkable_disconnected`, `bed_side_clearance_insufficient`, `dining_table_clearance_insufficient`, `living_sofa_coffee_tv_angle`, `relation_distance`, and `clearance_insufficient`; `ROTATE` for `wrong_orientation` and `collision`; and `SCALE` for `collision`, `door_blocking`, `window_blocking`, and `bed_side_clearance_insufficient`. The inference action mask uses the same action–issue pairs.

Issue scheduling. The scheduler sorts detected issues by ascending integer priority and, within an equal-priority class, by descending implementation-level scheduler score. This tie-breaking score is distinct from the normalized rule severity $\bar{e}_v$ used by the verification gate and aggregate residual in Appendix F. It is the maximum available diagnostic value among `intersection_area_m2`, `outside_area_m2`, `blocked_ratio`, `overlap_ratio`, `blocked_area_m2`, and `outside_ratio`; remaining ties preserve detector order. Because integer priority is applied first, this raw diagnostic score is not compared across different issue families and does not enter $\mathcal{V}$.

Each scene selects one unblocked issue per round. When no acceptable candidate exists, its stable identity is added to `blocked_issue_identities`; the scene remains active and a later round re-detects the layout and attempts the next unblocked issue. If every currently detected issue is blocked, the terminal status is `all_detected_issues_blocked`. Any successful commit clears the blocked set, so a previously failed issue can be reconsidered after another edit changes the scene. Consequently, an issue cannot repeatedly consume budget without an intervening commit, but it may be retried after the state changes.

Table A11: StatePatch execution domains and deterministic fallback behavior.

Action	StatePatch scope	Ordered fallback after seed rejection	Principal constraints and implementation notes
INSERT	Add one object through new_instance.	Search a 0.1-m grid within 2.0 m of the proposed center. If no local candidate passes verification, search room-wide free space on a 0.2-m grid, capped at 1,000 positions.	Category must be nonempty; center values must be finite with per-axis magnitude at most 50 m; the footprint must lie inside the floor and overlap existing furniture by at most $0.005\ \mathrm{m}^2$. The required-category count must increase, the target missing_required_object issue must disappear, and no new hard or protected-relation violation may be introduced.
DELETE	Remove exactly one object referenced by target_ref; no state-update fields are allowed.	No numerical search; evaluate the direct deletion only.	The target reference must exist uniquely; an optional expected_category must match. This action addresses extra_object; the target issue must disappear and no new hard-violation key may be introduced.
MOVE	Modify only center_m; execution is normalized internally as TRANSLATE.	Evaluate the proposed displacement, then distances 0.05, 0.10, 0.15, 0.20, 0.30, and 0.40 m, followed by 0.60, 0.80, 1.00, and 1.20 m, with orthogonal offsets $\{-0.10, 0, +0.10\}$ m. For collision, boundary, and opening-blockage targets, a room-wide 0.2-m grid is evaluated after all local candidates fail verification.	Eight cardinal/diagonal directions are supported. Proposed displacement components are bounded by 5 m, although room-wide internal candidates may exceed that displacement. The patch must remove the active target issue under the recomputed evaluator, and no new hard-violation key may be introduced; collision repair may not move implicated objects out of bounds.
REPLACE	Modify the same object's category, asset_id, and/or model_jid.	No continuous search; directly evaluate the semantic/asset substitution.	The target must exist uniquely. This action addresses wrong_object_category_type; center, orientation, and footprint are retained, the resulting category must match the expected category, the active target issue must disappear, and 3D assembly must retrieve a compatible asset.
ROTATE	Modify orientation_deg or yaw_rad.	Evaluate the signed proposed offset, then fixed offsets -180°, -90°, -45°, $+45^\circ$, $+90^\circ$, and $+180^\circ$, followed by local offsets $\Delta\theta_0 + \{-15^\circ, -5^\circ, +5^\circ, +15^\circ\}$.	Used for wrong_orientation and for collision repair when rotation reduces overlap. Positive angles are clockwise in the top-down frame; the canonical signed seed satisfies $\lvert\Delta\theta_0\rvert \le 180^\circ$. Orientation must change, the active target issue must disappear, and no new hard-violation key may be introduced. Because yaw is normalized modulo 360°, the -180° and $+180^\circ$ offsets are state-equivalent; retaining both entries preserves the configured fallback sequence but does not enlarge the reachable candidate set. The configured rotate_snap_angles_deg is not used by the candidate generator.
SCALE	Modify the two-dimensional footprint size_m.	Evaluate scale_xz, then standard factors $\{0.70, 0.80, 0.85, 0.90, 0.95\}$. Collision/blocking fallbacks search 0.60–0.95 and remain subject to category-specific minimum footprints.	Each factor must be positive and no proposed factor may exceed 20. All candidates enforce category-specific minimum sizes; size must change, the active target issue must be removed under the recomputed evaluator, and no new hard-violation key may be introduced.

H StatePatch Parameter Serialization and Autoregressive Learning

Canonical numerical representation. The data exporter rounds every floating-point StatePatch value to at most three decimal places before JSON serialization. Position and size fields use meters (e.g., center_m and size_m), orientation fields use degrees (e.g., orientation_deg), and scale is dimensionless. Continuous values are therefore learned through ordinary next-token prediction rather than a separate regression head.

For MOVE, the canonical seed is $\boldsymbol{\alpha}_0^{\mathrm{move}} = (d, r_0)$ with $d \in \mathcal{D}_8 = \{\mathrm{N}, \mathrm{NE}, \mathrm{E}, \mathrm{SE}, \mathrm{S}, \mathrm{SW}, \mathrm{W}, \mathrm{NW}\}$ and r_0 in meters; schema validation also bounds the realized displacement components. For ROTATE, the seed is a finite signed angular offset, and the resulting yaw is normalized modulo 360° during execution. For SCALE, $\boldsymbol{\alpha}_0^{\mathrm{scale}} = (s_x, s_z)$ with $s_x, s_z > 0$. The exact schema limits and seed-first deterministic fallback sequences are specified in Appendix G.

Field-weighted autoregressive objective. For target sequence $\mathbf{y} = (y_1, \ldots, y_M)$, let the planner-conditioning context be $\mathcal{K}^t = (I^t, \mathcal{S}^t, q^t, \mathcal{G})$, with assistant-token mask μ_m and serialized field class

Table A12: Scheduler priorities; lower values are attempted first.

Issue	Priority
extra_object	9
collision	10
out_of_floor_bounds	20
door_blocking	30
window_blocking	40
walkable_disconnected	50
wrong_orientation	100
bed_side_clearance_insufficient	110
dining_table_clearance_insufficient	115
living_sofa_coffee_tv_angle...	120
relation_distance	130
clearance_insufficient	140
missing_required_object	210
wrong_object_category_type	220
unknown	999

$f(m)$. The planner is optimized with

$$\mathcal{L}_{\mathrm{AR}} = -\frac{\sum_{m=1}^{M} \mu_m w_{f(m)} \log p_\theta(y_m \mid y_{<m}, \mathcal{K}^t)}{\sum_{m=1}^{M} \mu_m w_{f(m)}}. \tag{A27}$$

Token supervision and field weights. Only assistant StatePatch target tokens contribute to Eq. (A27); image tokens, prompts, RoState, and RoReview tokens are masked. Every target token inherits the weight of its serialized field:

Field class	Weight
Action type	2.0
Target reference or role	2.0
Categorical parameter	1.5
Continuous parameter	1.0
Fixed JSON schema token	0.25

The ordering $w_{\mathrm{action}} = w_{\mathrm{target}} > w_{\mathrm{categorical}} > w_{\mathrm{numeric}} > w_{\mathrm{schema}}$ reflects the system decomposition: target and action selection are the planner's primary semantic responsibilities, whereas distance, angle, and scale initialize deterministic candidate search. Assigning numerical fields greater weight than action or target fields would instead encourage exact-coordinate fitting that the deterministic solver is explicitly designed to absorb.

I ROOMER-EVAL PROTOCOL AND COMPLETE TRANSFER RESULTS

Frozen evaluation cohort. The *common-1100* cohort is the complete Qwen-Image test split in Table A4 and is shared by the final-layout comparison, cross-generator refinement, and controlled repair analysis. It contains 777 bedrooms, 155 living rooms, and 168 dining rooms and is excluded from model training, checkpoint selection, rule development, and hyperparameter tuning. Representative plug-and-play refinements for the four evaluated external generators are shown in Fig. 4.

Density control and baseline interfaces. For density-controlled comparison, each method preferentially retains a valid candidate whose number of floor-standing furniture objects matches the reference; a difference of one is allowed only when no exact match is available. Candidate selection uses output validity and object count only, never FID, KID, SCA Gap, OOB, COL, or Practical. Each baseline retains its native conditioning, asset retrieval, and assembly procedure. Adapters perform only category alignment, unit and coordinate conversion, and field standardization; they neither repair baseline layouts nor replace successfully assembled assets. Because the systems expose different input interfaces, the protocol standardizes scenes and evaluation rather than claiming identical conditioning information.

Table A13: Native inputs and density-control protocols used in the *common-1100* comparison. RS denotes rejection sampling. Candidate budgets, generation configurations, available seed policies, category mappings, interface adapters, and traversal orders are frozen before metric computation.

Method	Native input used in comparison	Density control
ReSpace	Centered room boundary and sequential additions compiled from the target furniture list	Shared validity/count-based selection
DiffuScene-RS	Native floor-plan mask	Count-filtered rejection sampling
InstructScene	Room-specific object-list instruction and room checkpoint	Native generation with shared validity/count-based selection
SemLayoutDiff-RS	Room type and floor/door/window architecture mask	Count-filtered rejection sampling
Roomer	Target room specification and architecture condition image	No post-hoc count repair

Table A14: Valid output rate (VOR) before and after deterministic traversal of each frozen candidate pool. Candidate and same-category asset orders are frozen before evaluation, and evaluation metrics are never used for candidate selection.

Method	Primary valid	Primary VOR	Final valid	Final VOR
ReSpace	1,096/1,100	99.64%	1,100/1,100	100.00%
DiffuScene-RS	1,092/1,100	99.27%	1,100/1,100	100.00%
InstructScene	1,095/1,100	99.55%	1,100/1,100	100.00%
SemLayoutDiff-RS	1,087/1,100	98.82%	1,100/1,100	100.00%
Ours-Initial	1,098/1,100	99.82%	1,100/1,100	100.00%
Ours-Final	1,098/1,100	99.82%	1,100/1,100	100.00%

In main-paper Fig. 3, the shortened column labels "DiffuScene" and "SemLayoutDiff" denote the evaluated DiffuScene-RS and SemLayoutDiff-RS configurations listed in Table A13.

Frozen-output validity and deterministic fallback. Table A14 distinguishes the validity of each method's primary frozen output from validity after deterministic traversal of the pre-generated and frozen candidate pool. The traversal order is fixed before evaluation. When the primary candidate cannot be parsed or assembled, later candidates are evaluated in frozen candidate-index order. If an asset fails assembly, later assets are attempted in a frozen same-category order; cross-category substitution is not permitted. A scene is invalid only if every frozen candidate and every same-category asset fallback fails. All reported final-layout metrics use the final deterministically selected output.

The final outputs are therefore reported *after deterministic traversal of the pre-generated and frozen candidate pool*.

Metric aggregation and sample retention. All methods are evaluated using the frozen Roomer-Eval protocol. FID, KID, and SCA Gap are computed separately for bedrooms, living rooms, and dining rooms and then averaged with equal room-type weight. OOB and COL are evaluated from the final outputs of all 1,100 scenes and micro-aggregated over successfully evaluated furniture objects; Practical is micro-aggregated over all applicable rule instances. Thus, the main-paper Table 1 phrase "use all 1,100 scenes" means that no scene is removed before evaluation; the metric denominators remain evaluated furniture objects for OOB/COL and applicable rule instances for Practical. Cross-generator refinement retains every input scene regardless of whether the repair loop solves all issues, blocks the remaining issues, or reaches the round limit. Rejected candidates are rolled back and remain represented by the last committed state.

Unified semantic evaluation rendering. All methods are first parsed into complete scene layouts and then re-rendered as unified top-down semantic images. The common evaluation canvas is a

256×256 RGB orthographic rendering of a fixed $12.2\,\mathrm{m}\times12.2\,\mathrm{m}$ scene region. Pure-black pixels are replaced by white after RGB conversion. A malformed raster that is smaller than the common canvas is centered on a white background, whereas an oversized raster is center-cropped; this defensive normalization does not resize a method's native raster directly to 256×256. In particular, the Qwen-Image generator natively produces 512×512 semantic layouts, while 256×256 denotes only the shared evaluation rendering. For FID and KID, CleanFID's `clean` mode subsequently applies PIL bicubic interpolation to 299×299. Pixel values remain in $[0, 255]$ before feature extraction, and normalization is performed inside the TorchScript Inception network. The frozen real reference set contains 777 bedrooms, 155 living rooms, and 168 dining rooms.

FID implementation. FID uses CleanFID 0.1.35 with the `torchscript_inception` feature extractor in `clean` mode and its 2048-dimensional pooled features. Each room type is evaluated independently. Real features remain fixed, and each of ten outer bootstrap repetitions samples the generated features with replacement to match the real room-type count. The outer random-number generator is initialized with seed 0. Let

$$F_r \in \mathbb{R}^{N_r \times d}, \qquad F_g \in \mathbb{R}^{N_g \times d}, \qquad d = 2048, \tag{A28}$$

with means μ_r, μ_g and centered matrices X_r, X_g. The sample covariances are

$$C_r = \frac{X_r^\top X_r}{N_r - 1}, \qquad C_g = \frac{X_g^\top X_g}{N_g - 1}. \tag{A29}$$

The implementation uses float64 sample-space SVD rather than `scipy.linalg.sqrtm`, diagonal epsilon, or diagonal jitter. Defining

$$A = \frac{X_r X_g^\top}{\sqrt{(N_r - 1)(N_g - 1)}}, \tag{A30}$$

the room-type FID is computed as

$$\mathrm{FID} = \|\mu_r - \mu_g\|_2^2 + \mathrm{tr}(C_r) + \mathrm{tr}(C_g) - 2\sum_k \sigma_k(A), \tag{A31}$$

followed by $\mathrm{FID} \leftarrow \max(\mathrm{FID}, 0)$. For outer repetition b, the unweighted Macro-3 value is

$$\mathrm{FID}_{\mathrm{M3}}^{(b)} = \frac{1}{3} \sum_{r \in \{\mathrm{B,L,D}\}} \mathrm{FID}_r^{(b)}. \tag{A32}$$

The reported value is the mean of the ten outer Macro-3 results; the artifact also retains all ten values and their standard deviation.

KID implementation. KID uses the third-degree polynomial kernel

$$k(x, y) = \left(\frac{x^\top y}{2048} + 1\right)^3, \tag{A33}$$

corresponding to degree 3, $\gamma = 1/2048$, and coefficient 1. One KID evaluation is performed for each FID outer bootstrap, producing ten outer results per room type. Each evaluation requests 100 subsets with

$$m = \min(N_r, N_g, 1000). \tag{A34}$$

The resulting subset sizes are 777 for bedrooms, 155 for living rooms, and 168 for dining rooms. Sampling within a subset is without replacement. Because m equals the complete room-type sample size in the current cohort, every inner subset contains the same features. The unbiased MMD^2 estimator is permutation-invariant, so the 100 inner estimates are identical up to negligible floating-point effects; effective variation among the reported outer results comes from the generated-feature bootstrap. For subset features $\{x_i\}_{i=1}^m$ and $\{y_i\}_{i=1}^m$, KID uses the unbiased MMD^2 estimator

$$\begin{aligned} \widehat{\mathrm{MMD}}_u^2 = &\frac{1}{m(m-1)} \sum_{i \neq j} k(x_i, x_j) \\ &+ \frac{1}{m(m-1)} \sum_{i \neq j} k(y_i, y_j) \\ &- \frac{2}{m^2} \sum_{i=1}^m \sum_{j=1}^m k(x_i, y_j). \end{aligned} \tag{A35}$$

Negative estimates are retained. Each room-type result is the mean of its 100 subset estimates. The unweighted Macro-3 result is then averaged over the ten outer bootstraps and reported as $10^3 \times \overline{\mathrm{KID}}_{\mathrm{M3}}$; all ten outer Macro-3 values and their standard deviation are retained in the artifact.

SCA Gap implementation. SCA denotes Scene Classification Accuracy and operates on complete semantic scene images rather than individual furniture objects. The real-versus-generated binary classifier consists of an ImageNet-pretrained AlexNet feature trunk and a $9216 \to 1$ sigmoid head. Inputs are $[0, 1]$ RGB images without additional ImageNet mean/std normalization. Real and generated images are split independently within each room type by sorting frozen sample identifiers, dropping the final sample when the count is odd, and assigning the first half to training and the second half to testing. Equal real and generated counts preserve class balance. The resulting per-class train/test counts are 388/388 for bedrooms, 77/77 for living rooms, and 84/84 for dining rooms. Thus, each pooled split contains 549 real and 549 generated images; one real and one generated bedroom image and one real and one generated living-room image are excluded by the odd-count rule.

A single pooled classifier is trained across all three room types with Adam, learning rate 10^{-4}, and batch size 256. Training runs continuously for 100 epochs, with evaluation at epochs $10, 20, \ldots, 100$; the model is not reinitialized between checkpoints. If $\mathrm{SCA}_r^{(q)} \in [0, 1]$ is the accuracy for room type r at checkpoint q, then

$$\mathrm{SCA\ Gap}_r^{(q)} = \left| 100\,\mathrm{SCA}_r^{(q)} - 50 \right|, \tag{A36}$$

and

$$\mathrm{SCA\ Gap}_{\mathrm{M3}}^{(q)} = \frac{1}{3} \sum_{r \in \{\mathrm{B,L,D}\}} \mathrm{SCA\ Gap}_r^{(q)}. \tag{A37}$$

The final SCA Gap is the mean of the ten continuous-training checkpoints. Lower values indicate that the classifier has greater difficulty distinguishing real from generated scenes. The artifact retains the ten checkpoint values and their standard deviation. Because classification is performed on complete semantic scenes, SCA does not require a missing-furniture-category fallback.

Assembled-mesh OOB implementation. The final OOB and COL results use SceneEval commit `116881e7945dcf7dbc58f7c64590a4c0b99c5cf5`, integrated through a compatibility wrapper. These assembled-mesh metrics are independent of the two-dimensional geometric detectors used to construct RoReview and verify repair candidates. OOB is an object-level rate over furniture objects that are successfully loaded, assembled, and evaluated. For object i with oriented-bounding-box volume V_i, the evaluator samples

$$N_i = \max(\lfloor 5000 V_i \rfloor, 1000) \tag{A38}$$

points from the mesh surface and casts a downward ray from each point toward the floor mesh. Its floor-hit ratio is

$$r_i = \frac{\#\{\text{sampled points whose downward rays hit the floor}\}}{N_i}. \tag{A39}$$

The object is marked out of bounds when $r_i < 0.99$, and the reported metric is

$$\mathrm{OOB} = \frac{\#\{\text{OOB furniture objects}\}}{\#\{\text{evaluated furniture objects}\}}. \tag{A40}$$

Thus, OOB is neither a scene-level violation rate nor an outside-footprint-area ratio. The frozen evaluation implementation does not explicitly fix a random seed for surface sampling. Accordingly, the submitted values are tied to the retained per-object decisions and aggregate outputs of that execution rather than to guaranteed bitwise-identical reruns.

Assembled-mesh COL implementation. COL uses the `CollisionManager` interface from `trimesh` with `python-fcl` 0.7.0.11 and checks all unordered furniture–furniture mesh pairs. Floor, wall, door, and window meshes do not enter the pair loop. No support-pair exclusion, category whitelist, contact whitelist, or furniture-relation whitelist is applied. For a pair with an initial FCL

Table A15: Physical-validity and Practical results after applying Roomer to outputs from external generators. All values are micro-averages over their respective evaluation units.

Method after Roomer	OOB↓	COL↓	Practical↑
ReSpace	1.16%	7.61%	75.12%
DiffuScene-RS	6.98%	6.95%	47.73%
InstructScene	5.67%	5.88%	45.70%
SemLayoutDiff-RS	32.83%	43.92%	66.86%

contact, the evaluator performs a $5\,\mathrm{mm}$ separation test. Let $\mathbf{c}_{\mathrm{object}}$ be the centroid of the pair's second object and $\bar{\mathbf{c}}_{\mathrm{contact}}$ the mean contact point. The displacement direction is

$$\mathbf{d} = \frac{\mathbf{c}_{\mathrm{object}} - \bar{\mathbf{c}}_{\mathrm{contact}}}{\|\mathbf{c}_{\mathrm{object}} - \bar{\mathbf{c}}_{\mathrm{contact}}\|_2}, \tag{A41}$$

and the separation-test position is

$$\mathbf{c}'_{\mathrm{object}} = \mathbf{c}_{\mathrm{object}} + 0.005\,\mathbf{d}. \tag{A42}$$

Only a pair that remains in collision after this test is counted as a persistent collision pair. Both furniture objects in such a pair are marked as collision objects, and an object participating in multiple persistent pairs is counted once. The final metric is therefore

$$\mathrm{COL} = \frac{\#\left\{\begin{array}{c}\text{unique furniture objects in at least one}\\ \text{persistent collision pair}\end{array}\right\}}{\#\{\text{evaluated furniture objects}\}}. \tag{A43}$$

COL is a unique collision-object rate, not a collision-pair, contact-instance, or scene-level rate. Before collision testing, each asset receives its asset-specific normalization rotation, object scale, scene rotation, and scene translation as one combined 4×4 transform. Objects that cannot be loaded or assembled do not enter the OOB or COL object denominator. The artifact reports scene and object assembly coverage together with the evaluated-object denominators.

Paired distributional-quality results. Main-paper Table 2 reports the paired physical-validity and usability changes. The corresponding distributional-quality values are reported in Table A9. Ten of the twelve paired FID, KID, and SCA Gap comparisons improve; the two exceptions are ReSpace KID and DiffuScene-RS SCA Gap.

Refined physical-validity and usability values. For completeness, Table A15 lists the refined OOB, COL, and Practical values for all four external generators. OOB is micro-averaged over evaluated furniture objects marked out of bounds, COL over unique evaluated furniture objects participating in at least one persistent collision pair, and Practical over all applicable Practical rule instances. These percentages are not scene all-pass rates. The entries reproduce main-paper Table 2; the main table omits percent signs, but every entry is a percentage.

Baseline candidate budgets and seed traceability. Here, one baseline *attempt* denotes one candidate-generation or assembly attempt and is unrelated to a Roomer-CC outer corruption proposal. Candidate budgets, generation configurations, category mappings, interface conversion, candidate traversal, and the available seed policies were frozen before metric computation. Table A16 records the realized budgets.

All methods' final semantic evaluation images are re-rendered from their parsed complete scene states using the common 256×256 RGB protocol above. This common resolution is not the native output resolution of ReSpace, DiffuScene-RS, InstructScene, SemLayoutDiff-RS, or Roomer.

J Complete Roomer-Eval Practical Rule Definitions

Aggregation and applicability. For each scene i and rule instance j, applicability and satisfaction are $a_{ij}, s_{ij} \in \{0, 1\}$. Main-paper Eq. (3) micro-averages all applicable instances. N/A instances

Table A16: Realized baseline candidate-generation and assembly budgets for the frozen *common-1100* comparison. Frozen candidate records and traversal indices determine the final evaluation inputs even where a selected candidate cannot be mapped back to a separately retained generation seed.

Method	Checkpoint/version	Native output	Realized candidate budget and randomness record
ReSpace	Commit prefix `1eccb692`; `sg_llm_1p5b`	JSON; no native image resolution	1,121 attempts for 1,100 rooms (1.02 per room; maximum 3). Original generation uses seed 0, greedy decoding, and `max_new_tokens=256`; the frozen pool also contains `seed1` and `seed2` retry roots.
DiffuScene-RS	Commit prefix `d78a289`; checkpoints 30000/82000/96000 for bed-room/dining/living	Category and bounding-box JSON	11,572 attempts (10.52 per room; maximum 91). Candidate seeds follow `seed_offset + sequence_index + candidate_index` and are recorded.
InstructScene	Official code; author-hosted community checkpoints; fVQ-VAE epoch `01999`	Category/bbox/ object-feature JSON	16,341 attempts (14.86 per room; maximum 91). Candidate records, indices, and final traversal order are frozen, but the selection audit does not retain a separate generation seed for the selected candidate.
SemLayout Diff-RS	Commit prefix `6b12bc4`; official SLDN checkpoint release	120×120 label map plus APM	7,913 attempts (7.19 per room; maximum 24). The generation interface accepts a seed; candidate records, indices, and traversal order are frozen, but the selection audit does not retain a per-candidate generation seed.

have $a_{ij} = 0$ and neither enter the denominator nor count as passes. Multiple coffee tables, dining tables, or beds create multiple instances. Door swing-proxy avoidance and walkable connectivity each create at most one room-level instance.

Living functional organization. The rule applies only when a scene contains at least one sofa, coffee table, and TV; otherwise it is N/A. The sofa with the largest footprint area is the primary sofa. Each coffee table forms one instance and is paired with the TV whose center is nearest to that coffee table. With object centers $\mathbf{c}$, the measured angle is

$$\theta = \angle(\mathbf{c}_{\text{coffee}} - \mathbf{c}_{\text{sofa}}, \mathbf{c}_{\text{coffee}} - \mathbf{c}_{\text{TV}}) \,. \tag{A44}$$

The instance passes iff $\theta \geq 135°$. The current rule imposes no sofa–coffee or coffee–TV distance threshold.

Dining-table clearance. Each dining table forms one instance; scenes without a dining table are N/A. Clearances c_1 and c_2 are measured on the two sides of the table's long axis, and the instance passes iff $\min(c_1, c_2) \geq 0.60\,\text{m}$. Dining chairs, chairs, and stools associated with that table are excluded from the obstacle set. Association uses `dining_group_id` when available; otherwise, seats whose polygon distance to a dining table is at most $0.80\,\text{m}$ are assigned to the nearest table. Clearance search is capped at $2.40\,\text{m}$. Boundary-only contact between a clearance region and an obstacle is permitted.

Door swing-proxy avoidance. A scene containing at least one door forms one instance; scenes without doors are N/A. For each door, the longest edge of the opening determines the proxy radius, equal to the door width. Both opening endpoints are candidate hinges. At each endpoint, two rotation directions define candidate $90°$ sectors, and the sector with larger overlap with the room interior is retained. The retained quarter sectors form an interior door swing proxy. Furniture height is ignored. The scene passes only when no furniture footprint intersects or touches any retained proxy; either overlap or boundary contact is a violation. This rule is a geometric proxy and is not an exact door-swing simulation.

Walkable connectivity. Every scene with a valid room boundary forms one instance. The evaluator rasterizes geometry at $1\,\mathrm{cm/pixel}$, treats walls and floor-standing furniture as obstacles, and dilates obstacles by $0.30\,\mathrm{m}$, corresponding to a $0.60\,\mathrm{m}$ passage width. Ceiling lamps and pendant lamps are excluded from the ground-obstacle set. The remaining free space is analyzed with 8-connectivity; components smaller than 20 pixels are ignored. The instance passes iff the free space is nonempty and $N_{\text{components}}^{A\geq 20} \leq 1$. No entrance, bed, sofa, or other semantic anchor is required. Empty free space is a failure.

Bed-side clearance. Each bed forms one instance; scenes without beds are N/A. Let w_{bed} be the short-axis length of the bed footprint's minimum rotated bounding rectangle, and let c_1, c_2 be the clearances measured from its two short-axis sides. A side is wall-adjacent when its distance to the room boundary is at most $0.10\,\mathrm{m}$. The usable clearance is

$$c_{\text{usable}} = \begin{cases} \max(c_1, c_2), & \text{if at least one side is wall-adjacent} \\ \min(c_1, c_2), & \text{otherwise.} \end{cases} \tag{A45}$$

The instance passes iff $c_{\text{usable}}/w_{\text{bed}} \geq 0.30$. Nightstands and corner-side tables are excluded from the obstacle set. Measurement depth is capped at $2.40\,\mathrm{m}$, and boundary-only contact between a clearance region and an obstacle is permitted. The threshold scales with bed width and is not a fixed $0.45\,\mathrm{m}$ clearance.

K Controlled Analysis of the Roomer Repair Loop

Shared initialization and protocol. Every controlled variant starts from byte-identical Ours-Initial layouts on *common-1100* and the same initially detected, instance-grounded RoReview entries. All variants use the shared six-primitive action vocabulary, the same maximum number of rounds, the same issue scheduler, and identical stopping conditions unless the named ablation explicitly removes a component. A standard Roomer transaction contains one target-grounded StatePatch, whereas Full-Scene JSON Rewrite changes the output representation, transaction scope, and rewrite-specific acceptance test as defined below. Ours-Initial denotes the unrepaired $K = 0$ state; Ours-Final denotes the terminal $K = 10$ state produced by the complete loop with RoReview, geometry-conditioned StatePatch planning, deterministic candidate instantiation, full-scene re-verification, and rollback.

Controlled repair metrics. The controlled analysis separates the complete hard-diagnostic set from the frozen repair-target set. Let $\mathcal{H}(\mathcal{Y})$ contain every collision, floor-boundary, and opening-blockage key emitted by the full floor-plane hard evaluator. Let $\mathcal{T}^0 = \bigcup_{i=1}^{1,100} \mathcal{T}_i^0$ contain the 4,800 initially detected RoReview keys that are object-grounded, supported by the six-action vocabulary, and admitted by the frozen scheduler, where $\mathcal{T}_i^0$ denotes the initial target subset of scene i. For compactness, let $\mathcal{U}_i^K = \mathcal{U}(\mathcal{Y}_i^K; \mathcal{A}_i, \mathcal{G}_i)$. All 1,100 scenes contain at least one admitted target. Consequently, $\mathcal{T}^0$ is the target pool used for repair analysis, whereas $\mathcal{H}$ also contains pre-existing hard conditions that are not members of $\mathcal{T}^0$.

Hard Validity (labeled "Phys.Valid" in main-paper Fig. 5) is the percentage of scenes satisfying $\mathrm{HardValid}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \mathbf{1}[\mathcal{H}(\mathcal{Y}; \mathcal{A}, \mathcal{G}) = \varnothing]$. **Target Resolution** is the scene-macro average of the resolved fraction within each scene's frozen initial target subset:

$$\mathrm{TargetResolution}(K) = \frac{1}{1{,}100} \sum_{i=1}^{1,100} \frac{\left|\mathcal{T}_i^0 \setminus \mathcal{U}_i^K\right|}{|\mathcal{T}_i^0|}. \tag{A46}$$

Each scene therefore contributes equally even though the number of initial targets varies across scenes. This analysis metric is distinct from both the transaction-level target predicate, which determines whether one candidate can be committed, and the pooled unresolved-key count reported below. **New Hard Error** is the percentage of scenes for which $\mathcal{H}(\mathcal{Y}^K) \setminus \mathcal{H}(\mathcal{Y}^0) \neq \varnothing$. **Non-target Preservation** is the fraction of protected non-target objects whose identity, category, position, size, and orientation remain unchanged.

Strict Safe Repair is a scene-level local-repair metric. A scene passes iff every key from that scene's initial target subset $\mathcal{T}_i^0$ is resolved, no new hard key is introduced, $\mathcal{C} = 1$, and every protected non-target object is preserved. Strict Safe Repair does not require complete Hard Validity: a scene can

Table A17: Complete controlled ablation from identical *common-1100* initial states. All entries are percentages. Target Resolution is the scene-macro resolved fraction over each scene's frozen initial target subset; Non-target Preservation measures the fraction of protected non-target objects that remain unchanged.

Variant	Hard Validity↑	Practical↑	Target Resolution↑	New Hard Error↓	Non-target Preservation↑	Strict Safe Repair↑
Ours-Initial ($K = 0$)	19.27	72.50	–	–	–	–
w/o RoReview	19.73	50.39	2.44	0.00	94.15	0.45
w/o Geometry-Conditioning Adapter	62.82	74.90	88.40	0.00	99.81	81.64
Direct Planner-Seed Execution	48.00	67.80	65.70	0.00	99.93	55.18
Seed-Free Search	68.82	77.60	90.70	0.00	99.91	89.64
VLM-Free	64.18	76.34	87.42	0.00	99.89	84.09
Full-Scene JSON Rewrite	16.55	49.82	32.48	32.64	48.61	1.91
w/o Verification and Rollback	23.55	71.72	64.91	54.27	97.63	23.18
Ours-Final ($K = 10$)	**72.64**	**82.98**	**98.96**	**0.00**	**99.94**	**96.27**

Table A18: Round-prefix convergence on the frozen 4,800-key initial target pool. Abbreviations: Calls, average planner calls per scene; Unres., pooled unresolved target keys per scene; TR, scene-macro Target Resolution; SSR, Strict Safe Repair; HV, Hard Validity.

K	Calls	Unres.↓	TR↑	SSR↑	HV↑
0	0.000	4.364	0.00%	0.00%	19.27%
1	1.000	3.006	31.12%	11.27%	29.64%
3	2.569	1.199	72.52%	52.36%	59.09%
5	3.365	0.386	91.16%	80.45%	70.00%
7	3.665	0.117	97.32%	92.00%	72.45%
10	**3.756**	**0.045**	**98.96%**	**96.27%**	**72.64%**

satisfy the local-repair criterion while retaining a pre-existing hard key outside $\mathcal{T}_i^0$. This distinction allows the reported 96.27% Strict Safe Repair and 72.64% Hard Validity to characterize different properties of the same final layouts.

Complete component ablation. Table A17 reports the complete six-metric version of the focused ablation shown in the main paper. Hard Validity and Practical characterize final-scene quality, while the remaining four metrics isolate target resolution, safety, and locality.

Round-prefix evaluation. For $K \in \{1, 3, 5, 7, 10\}$, results are computed from prefixes of the same frozen $K = 10$ trajectories. Thus, each smaller budget is the actual state after K rounds and cannot benefit from resampling or a separate run. In addition to the scene-macro Target Resolution in Eq. (A46), we report the pooled unresolved-target load

$$N_{\text{unres}}(K) = \sum_{i=1}^{1,100} \left| \mathcal{T}_i^0 \cap \mathcal{U}_i^K \right|, \qquad \text{TargetLoad}(K) = \frac{N_{\text{unres}}(K)}{1{,}100}. \tag{A47}$$

The frozen prefix counts are $N_{\text{unres}}(K) \in \{4{,}800, 3{,}307, 1{,}319, 425, 129, 50\}$ for $K \in \{0, 1, 3, 5, 7, 10\}$, respectively. Dividing these integer counts by 1,100 gives the displayed unresolved-target loads. Because Target Resolution first normalizes within each scene and then macro-averages across scenes, while Target Load pools issue keys before normalization, the two quantities are not algebraically interchangeable when $|\mathcal{T}_i^0|$ varies across scenes.

Table A20: Deterministic solver quality and computation. Solver labels: Direct Seed, direct execution of the planner-predicted seed without deterministic fallback; Seed-Free, deterministic search without a continuous planner seed; Seed-First, the complete seed-first deterministic solver. Abbreviations: TR, scene-macro Target Resolution; SSR, Strict Safe Repair; Calls, average planner calls per scene; Checks, candidate checks per scene; BE, scene-level budget exhaustion. TR, SSR, and BE are percentages.

Solver	TR↑	SSR↑	Calls↓	Checks↓	BE↓
Direct Seed	65.70	55.18	3.300	3.3	21.8
Seed-Free	90.70	89.64	4.000	24.6	5.1
Seed-First	**98.96**	**96.27**	**3.756**	**11.2**	**2.4**

Table A21: Loss and geometry-conditioning ablations. Condition labels: Text Only removes the geometry-conditioning adapter; Serialized Evidence exposes all normalized geometry and violation measurements only through text; RQ + Geom. Adapter uses Repair Queries and geometry-conditioning adapters. Abbreviations: HV, Hard Validity; P, Practical; TR, Target Resolution; SSR, Strict Safe Repair.

Condition	HV↑	P↑	TR↑	SSR↑
Uniform Loss	70.91	81.70	97.40	93.91
Field-Weighted Loss	**72.64**	**82.98**	**98.96**	**96.27**
Text Only	62.82	74.90	88.40	81.64
Serialized Evidence	65.73	78.90	92.20	86.55
RQ + Geom. Adapter	**72.64**	**82.98**	**98.96**	**96.27**

Inference-Time Cost

Table A19 reports the measured workload of the frozen $K = 10$ Roomer trajectories on *common-1100*. Inference uses Qwen2.5-VL-7B-Instruct with four workers assigned to GPUs 0–3. The 4,132 VLM calls correspond to 3.756 calls per scene, consistent with Table A18; 3,974 attempts commit a verified edit and 158 are rejected or rolled back. The seed-first solver's 11.2 candidate checks per scene agree with Table A20.

Table A19: Measured inference workload and executor cost on the frozen *common-1100* cohort ($K = 10$). Executor time is the sum of logged per-case computation times, not four-worker elapsed wall-clock time.

Measure	Value
Scenes	1,100
VLM calls	4,132 total; 3.756 per scene
Calls per scene	Median 3; P90 7; P95 8; maximum 10
Accepted / rolled-back attempts	3,974 / 158
Candidate checks	11.2 per scene
Summed executor compute time	6,304.82 s; 5.73 s per scene; 1.53 s per attempt
Re-tokenized generated text	294,228 tokens; 71.21 per VLM call

The executor-time sum excludes VLM inference and therefore is not an end-to-end runtime measurement. The token count is reconstructed from generated text and excludes prompt and image input tokens. We do not interpret either quantity as the total compute cost of the four-worker pipeline.

L Additional Component and Solver Ablations

Operational definitions. The *w/o Geometry-Conditioning Adapter* variant retains RoReview, stable entity references, and the human-readable RoState/RoReview context, but removes the 512-dimensional evidence tokens, Repair Queries, and decoder geometry-conditioning adapters; the deterministic solver, full-scene verification, and training data are unchanged. *Direct Planner-Seed*

Table A22: Geometry-branch design ablations. Eight Repair Queries, a two-layer aggregator, and interval injection into the final four even-numbered decoder layers provide the best overall trade-off.

Component	Setting	Hard Valid.↑	Practical↑	Target Res.↑	Strict Safe Repair↑
Repair Queries K_c	1	67.18	79.90	93.70	88.64
	4	71.36	82.10	97.80	94.64
	8	**72.64**	**82.98**	**98.96**	**96.27**
	16	72.27	82.70	98.60	95.82
Aggregator depth	1 layer	71.00	81.70	97.50	94.00
	2 layers	**72.64**	**82.98**	**98.96**	**96.27**
Injection layers	$\{28\}$	67.82	80.10	94.10	89.73
	$\{24, 28\}$	70.55	81.60	97.00	93.82
	$\{\mathbf{22}, \mathbf{24}, \mathbf{26}, \mathbf{28}\}$	**72.64**	**82.98**	**98.96**	**96.27**
	$\{18, 20, 22, 24, 26, 28\}$	72.09	82.50	98.50	95.64

Execution retains the complete planner and full-scene verification but evaluates only the normalized executable candidate instantiated from the predicted parameter seed: a passing candidate is committed and a rejected candidate is rolled back, with no deterministic fallback. *Seed-Free Search* retains the planner's target and action decisions but removes the continuous seed, so the deterministic solver starts from the first fixed fallback candidate under the same search ranges, budget, and verification gate as the complete system.

The *VLM-Free* variant starts from the same initial layouts and RoReview entries and retains the issue scheduler, six actions, candidate-search budget, ten-round limit, and verification with rollback. It makes no VLM call: a rule-based policy selects an object and action for the active RoReview issue, after which the same seed-free deterministic parameter search is used. Object tie-breaks and action priorities for issues with multiple valid choices were fixed before test evaluation. Comparing VLM-Free with Seed-Free therefore measures the added value of VLM target/action decisions, not independent error localization, because RoReview and the scheduler already provide the issue and implicated entities.

Full-Scene JSON Rewrite receives the same semantic rendering, RoState, active RoReview, and room specification, but predicts a complete candidate layout $\widetilde{\mathcal{Y}}^t$ instead of a single StatePatch. A deterministic differencer constructs

$$\mathcal{D}^t = \mathrm{Diff}_{\mathbb{A}}\left(\mathcal{Y}^t, \widetilde{\mathcal{Y}}^t\right), \tag{A48}$$

where every atomic difference must map to one of the same six primitives in $\mathbb{A}$: a changed center, orientation, footprint, or category/asset maps to `MOVE`, `ROTATE`, `SCALE`, or `REPLACE`; a removed or newly introduced reference maps to `DELETE` or `INSERT`. The rewrite is invalid if it changes architecture, uses an unsupported attribute transition, or produces an ambiguous reference mapping. The complete rewritten layout is treated as one atomic candidate and is checked for parseability, immutable architecture, supported primitive decomposition, structural validity $\mathcal{C} = 1$, and disappearance of the active target issue. The no-new-hard-key and protected-relation clauses in main-paper Eq. (2) are not imposed on this ablation; newly introduced hard errors and non-target changes are measured after commitment rather than blocked during candidate acceptance. A candidate that passes these checks is committed as a whole; otherwise the committed state remains unchanged. No seed-centered deterministic fallback is used. This retains the shared inputs, issue scheduler, primitive action vocabulary, round budget, and target check while isolating the effect of replacing a sparse single-target StatePatch with a scene-wide rewrite. In contrast, *w/o Verification and Rollback* bypasses candidate re-verification and directly commits the local planner output.

M Professional Validation of Practical Usability

Evaluation sample and Practical strata. The baseline study uses 90 frozen layouts sampled from the outputs of the evaluated baseline generators. The cohort is disjoint from the *common-1100* Roomer repair cohort and is isolated from model training, rule development, and parameter tuning.

For layout i, the scene-level Practical score is

$$P_i = \frac{\sum_j a_{ij} s_{ij}}{\sum_j a_{ij}}, \tag{A49}$$

computed from the same five frozen Practical rule families used by Roomer-Eval. Layouts with $\sum_j a_{ij} = 0$ are excluded from the professional-validation sampling pool. The strata are defined as

$$\text{Level}(i) = \begin{cases} \text{Low}, & 0 \le P_i \le 0.50, \\ \text{Medium}, & 0.50 < P_i < 0.80, \\ \text{High}, & 0.80 \le P_i \le 1.00. \end{cases} \tag{A50}$$

Within each stratum, 10 bedrooms, 10 dining rooms, and 10 living rooms are sampled without replacement, producing 30 layouts per level and preventing the comparison from being dominated by one room type.

Evaluators and blinding. In the baseline study, ten evaluators with interior-design experience independently assess the 90 anonymized layouts in randomized order (900 binary judgments), blinded to generator identity and Practical score.

Question and scene-level aggregation. Each evaluator answers the binary question: "Does this layout satisfy basic residential-use requirements without requiring further modification due to furniture organization, insufficient clearance, door-zone obstruction, or circulation problems?" A layout receives scene-level approval when at least seven of the ten evaluators answer "Yes." The baseline approval rate therefore uses 30 layouts as the denominator in each Practical group.

Paired external-generator evaluation. A separate blinded evaluation rates the before and after versions of the same 30 external-generator scenes using the question and seven-of-ten approval threshold above. Approval rises from 14/30 (46.7%) to 19/30 (63.3%), a net gain of five scenes (16.7 percentage points).

Statistical reporting. Main-paper Table 4 reports Wilson 95% confidence intervals for all five approval proportions. The Cochran–Armitage trend test applies only to the three ordered baseline Practical levels ($Z = 4.67$, $p = 3.07 \times 10^{-6}$).

Baseline inter-rater reliability is nominal Krippendorff's $\alpha = 0.65$ (bootstrap 95% CI $[0.54, 0.75]$). Pooled pairwise agreement is 84.0%, and 62/90 scenes (68.9%) receive unanimous judgments.