# SCENEPILOT: PLUG-AND-PLAY 3D SCENE REPAIR ACROSS GENERATORS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Text-driven 3D scene generators have made substantial progress, yet their outputs often retain collisions, boundary violations, orientation errors, and structural inconsistencies. Existing correction mechanisms are typically coupled to generator-specific scene representations and execution pipelines, making repair difficult to reuse across heterogeneous systems. We introduce **ScenePilot**, a plug-and-play 3D scene repair policy designed to make corrective behavior transferable across generators. ScenePilot decouples repair reasoning from generator-specific execution through a canonical scene-editing interface, allowing one shared multimodal policy to operate across heterogeneous generators without modifying or retraining their underlying generation models. At inference time, ScenePilot performs conservative closed-loop repair: each round proposes at most one atomic edit, executes it tentatively, and retains the change only when a shared verifier confirms improvement. We further construct **SceneReverse**, which transforms clean 3D scenes into executable recovery trajectories and provides generator-independent supervision for learning atomic repair decisions. Combined with supervised learning and reward-guided policy optimization based on realized improvements, SceneReverse enables repair learning without relying on failure traces from any target generator. On SceneEval, the same frozen ScenePilot policy consistently improves scenes from five heterogeneous generators unseen during policy training, without backend-specific policy fine-tuning, using only lightweight backend adapters. These results support transferable 3D scene repair as a reusable capability that can be attached to existing generators through a canonical interface.

## 1 INTRODUCTION

Text-driven 3D indoor scene generation turns a request such as "a bedroom with a bed, two nightstands, and a desk" into an editable 3D environment. This capability supports interior design, embodied AI, robotics simulation, and virtual-world authoring (Deitke et al., 2022; Xia et al., 2026). Unlike image generation, however, the result must not merely look plausible: objects cannot intersect walls or one another, related furniture should be arranged coherently, and enough free space must remain for navigation and everyday use (Wang et al., 2019; Lin & Mu, 2024). A scene may therefore appear reasonable while still containing serious geometric or functional errors.

Early data-driven approaches learn indoor scene distributions through autoregressive modeling, transformers, or diffusion (Wang et al., 2018; Ritchie et al., 2019; Wang et al., 2021; Paschalidou et al., 2021; Tang et al., 2024). Complementary approaches address scene geometry and appearance through radiance fields, room synthesis, and panoramic reconstruction (Bautista et al., 2022; Song et al., 2023; Zhou et al., 2025a; Zhang et al., 2024). Language-guided methods improve semantic control through layout planning, structured representations, and procedural generation (Feng et al., 2023; Berdoz et al., 2025; Çelen et al., 2024; Öcal et al., 2024; Zhou et al., 2025b). Meanwhile, spatial search and zone-based organization improve scene planning (Deng et al., 2025; Zhang et al., 2026), while agentic methods incorporate visual feedback and iterative correction (Yang et al., 2025a; Ling et al., 2025; Zhao et al., 2026). As illustrated in Fig. 1, existing paradigms either leave residual errors after one-pass generation or couple iterative refinement to a specific pipeline. In contrast, ScenePilot treats repair as a shared capability that operates across heterogeneous generators through a canonical adapter and conservative closed-loop verification. This motivates a distinct question: can corrective behavior be learned once and reused across heterogeneous generators?



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.

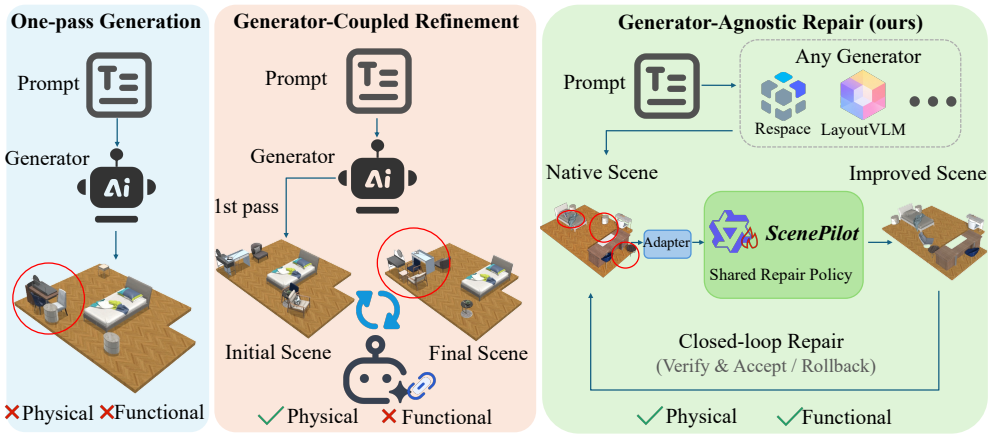


Figure 1: **Comparison of scene generation and repair paradigms.** One-pass generation may leave residual geometric or functional errors, while generator-coupled refinement remains tied to a specific pipeline. **ScenePilot** instead treats repair as a generator-agnostic capability, applying a shared policy across heterogeneous backends through lightweight adapters and conservative closed-loop verification with accept-or-rollback decisions.

This work studies the less explored problem of *transferable 3D scene repair*: how can a single policy improve scenes from heterogeneous text-driven 3D generators without being tied to their internal representations or generation procedures? The goal is not merely to optimize another generator, but to make *scene repair itself transferable*. Existing generators differ substantially in scene formats, placement mechanisms, and supported editing operations, making repair difficult to transfer across systems. Moreover, generated scenes often contain residual collisions, boundary violations, orientation errors, and structural inconsistencies that require iterative correction. Addressing this problem introduces three key challenges. 1) **Can a single repair policy generalize across heterogeneous scene generators?** Existing optimization mechanisms are typically tailored to a specific scene representation and execution pipeline, and therefore cannot be directly reused when the underlying generator changes. 2) **How can such a shared policy reliably repair imperfect scenes?** A common interface makes transfer possible, but not necessarily effective: generated scenes contain collisions, boundary violations, orientation errors, and structural inconsistencies, while each correction changes the scene and may introduce new defects. Reliable repair therefore requires closed-loop interaction—repeatedly observing the updated scene, selecting a targeted edit, assessing its realized effect, and deciding whether to continue or stop while preserving already valid regions. 3) **How can this transferable behavior be learned without collecting supervision for every generator?** Training on generator-specific failure traces would undermine the goal of plug-and-play transfer, yet standard 3D scene datasets provide mostly clean final layouts rather than imperfect states paired with executable recovery decisions. These challenges call for a plug-and-play framework that decouples repair from individual generators, iteratively refines scenes from fresh multimodal observations, and learns corrective behavior from generator-independent supervision.

**Contributions.** To address these three challenges, we propose **ScenePilot**, a plug-and-play repair policy that turns scene refinement from a generator-specific component into a transferable capability. 1) **Generator-Agnostic Repair Interface:** ScenePilot separates *what should be repaired* from *how a particular generator executes the repair*. A lightweight adapter maps each generator's native scene into a shared object-centric representation, exposes its supported editing operations, and translates canonical actions—`translate`, `rotate`, `scale`, `add`, and `delete`—back into native edits. This allows one frozen policy to operate across heterogeneous generators without generator-specific policy fine-tuning. 2) **Conservative Closed-Loop Repair:** ScenePilot performs repair as an iterative perception–action loop rather than one-shot post-processing. At each round, the multimodal policy predicts one atomic edit or `STOP`; the candidate edit is executed on a temporary scene and retained only if a shared verifier confirms improvement, otherwise it is rolled back. The retained scene is then re-rendered for the next decision, enabling traceable multi-step correction while preserving already valid regions. 3) **SceneReverse: Generator-Independent Process Supervision:** We construct **SceneReverse** from high-quality 3D-FRONT scenes (Fu et al., 2021) by applying controlled perturbations and reversing them into recovery trajectories. Intermediate states provide atomic repair

supervision, while recovered states provide explicit `STOP` targets. Combined with supervised learning and reward-guided policy optimization based on realized improvements, SceneReverse enables the policy to learn repair behavior without relying on failure traces from any particular generator.

## 2 Related Work

**3D Indoor Scene Generation and Layout Priors.** Indoor scene synthesis spans learned object arrangements (Wang et al., 2018; Ritchie et al., 2019; Wang et al., 2021; Paschalidou et al., 2021; Tang et al., 2024; Maillard et al., 2024), graph-conditioned generation (Lin & Mu, 2024; Chen et al., 2025; Yang et al., 2025b), and language-driven layout prediction (Yang et al., 2024b; Berdoz et al., 2025; Bucher & Armeni, 2025; Wan et al., 2026). Complementary methods connect layouts with appearance synthesis (Fang et al., 2025), use image intermediaries (Gu et al., 2025), or address open-set geometry and pose estimation (Shi et al., 2025). These approaches primarily construct scenes with different representations and controls. ScenePilot instead focuses on transferable repair: retrieved priors support optional incremental generation, while the shared repair policy remains independent of the planner and generator.

**Scene Optimization and Repair.** Existing methods address scene validity through human-motion conditioning (Yi et al., 2023), relational placement constraints (Yang et al., 2024b), differentiable layout optimization (Sun et al., 2025), denoising (Maillard et al., 2024), physically grounded guidance (Yang et al., 2024a), and differentiable simulation (Lin et al., 2026). Agentic approaches incorporate visual refinement and self-reflection (Ling et al., 2025; Yang et al., 2025a). These mechanisms improve scene quality, but do not by themselves establish the transfer of a shared repair policy across heterogeneous generators. ScenePilot studies this setting by separating repair decisions from backend-specific execution through a canonical scene–action interface and lightweight adapters.

**Process Supervision and Policy Learning.** More broadly, iterative refinement with self-feedback and interleaved reasoning and acting motivate closed-loop correction beyond a single generation step (Madaan et al., 2023; Yao et al., 2023). Recent methods introduce multi-turn reinforcement learning (Pan & Liu, 2025), reward-guided layout reasoning (Ran et al., 2025), and supervised learning followed by policy alignment (Bucher & Armeni, 2025; Zhang et al., 2026). Closely related, SceneReVis combines reverse-engineered construction trajectories, supervised fine-tuning, and multi-turn reinforcement learning for visual scene refinement (Zhao et al., 2026). ScenePilot shares trajectory supervision and policy optimization, but focuses on generator-agnostic repair: one policy checkpoint is evaluated on outputs from multiple generators unseen during policy training, without backend-specific policy fine-tuning. Recovery trajectories supervise executable interventions, and the same policy supports both completed and intermediate scene states.

## 3 ScenePilot Method

ScenePilot is a plug-and-play intervention policy for heterogeneous 3D scene generators. Its central design principle is to decouple *intervention reasoning* from *generator-specific execution*: one shared multimodal policy operates on a canonical scene representation, while lightweight adapters normalize native states and translate canonical actions into executable edits. The same frozen policy can therefore be applied either to completed scenes for post-hoc refinement or to intermediate states during incremental generation, without generator-specific policy fine-tuning.

### 3.1 Generator-Agnostic Repair Interface

Different 3D scene generators represent the same physical scene using different coordinate systems, object formats, and editing APIs. ScenePilot therefore does not require generators to share their internal representations. Instead, it defines a common object-centric repair interface that standardizes the *semantics* of scene states and editing operations, while leaving generator-specific execution to lightweight backend adapters.

**Shared scene state.** Let $S_t^{(g)}$ denote the native scene produced by generator $g \in \mathcal{G}$ at repair step $t$. Before being observed by the repair policy, the scene is mapped to a canonical state

$$z_t = \left( B_t, \{(\mathrm{id}_i, c_i, \mathbf{p}_i, \theta_i, \mathbf{s}_i)\}_{i=1}^{N_t} \right), \tag{1}$$

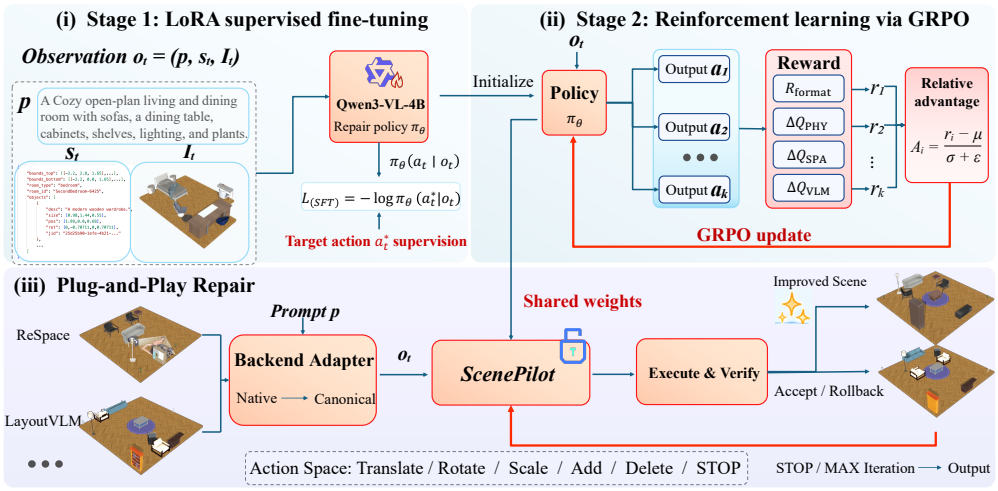


Figure 2: **Overview of ScenePilot.** (i) LoRA supervised fine-tuning adapts Qwen3-VL-4B-Instruct (Bai et al., 2025) to predict repair actions from user prompts, structured scene states, and rendered views. (ii) GRPO further optimizes the policy using format validity and physical, spatial, and visual-semantic improvement rewards. (iii) Backend adapters map native scenes into a canonical representation, enabling plug-and-play repair across heterogeneous generators with one shared checkpoint. Candidate edits are executed and verified through accept-or-rollback decisions until STOP or the iteration limit is reached.

where $B_t$ denotes the room boundary and each object is represented by a stable identifier $\mathrm{id}_i$, semantic category $c_i$, metric position $\mathbf{p}_i$, yaw angle $\theta_i$, and physical size $\mathbf{s}_i$. Backend-specific units, coordinate axes, rotation conventions, and boundary formats are normalized into this shared metric representation. The purpose of this representation is not merely to unify serialization formats, but to provide consistent object references and geometric semantics on which a single repair policy can reason across heterogeneous generators.

**Canonical edits and capabilities.** ScenePilot operates in a shared editing space $\mathcal{A} = \{\texttt{translate}, \texttt{rotate}, \texttt{scale}, \texttt{delete}, \texttt{add}\}$, with `STOP` defined separately as the terminal decision. These operations specify *what* change should be made to the scene without assuming how a particular generator implements that change. Not every backend needs to support the full action space. Each generator therefore exposes a capability descriptor $C_g$, which defines its executable subset $\mathcal{A}_g \subseteq \mathcal{A}$ together with the valid parameter ranges of the supported operations. The capability descriptor is provided to the shared policy as part of its observation, allowing the same policy to reason under different execution constraints. Unsupported operations are not silently converted into another action; they are rejected by the capability-aware validation layer described in Sec. 3.2.

**Adapter-based execution.** For each generator, we implement a lightweight interface

$$\mathcal{I}_g = (\phi_g, \psi_g, C_g), \tag{2}$$

where $\phi_g$ maps the native scene $S_t^{(g)}$ to the canonical state $z_t$, $\psi_g$ grounds a validated canonical action into an executable native edit, and $C_g$ specifies the capabilities exposed by the backend. The adapter is restricted to representation conversion and action grounding:it does not decide which object should be repaired and contains no generator-specific repair policy.

Consequently, changing the underlying scene generator changes only $\phi_g$, $\psi_g$, and $C_g$, while the repair-policy parameters $\theta$ remain shared. This separation allows ScenePilot to determine *what should be changed* independently of *how that change is executed* by a particular generator, forming the basis for plug-and-play transfer across heterogeneous scene-generation systems.

### 3.2 Conservative Closed-Loop Repair

A generator-agnostic interface makes it possible to execute a shared repair policy across heterogeneous backends, but an executable edit is not necessarily beneficial. A modification that resolves

one local defect may introduce another, while an overly aggressive edit may unnecessarily alter an already plausible scene. ScenePilot therefore performs repair as a conservative closed loop in which each atomic proposal is executed on a temporary scene and its realized effect is assessed before the modification is committed.

**Multimodal atomic decisions.** At repair step $t$, ScenePilot observes the current canonical scene together with two complementary rendered views and the capability descriptor of the current backend:

$$o_t = (I_t, z_t, C_g), \qquad I_t = \left\{ I_t^{\text{diag}}, I_t^{\text{ann-top}} \right\}, \tag{3}$$

where the diagonal view captures object geometry and local spatial interactions, the annotated top-down view exposes the global layout and stable object references, and $z_t$ provides precise object poses and physical sizes. The capability descriptor $C_g$ specifies which canonical operations are available for the current backend.

Given the current observation, the shared policy predicts exactly one atomic decision:

$$\hat{a}_t \sim \pi_\theta(\cdot \mid o_t), \qquad \hat{a}_t \in \mathcal{A} \cup \{\texttt{STOP}\}, \tag{4}$$

where $\mathcal{A}$ is the canonical edit space defined in Sec. 3.1. Each non-terminal decision targets one object with one canonical operation. Restricting each round to a single edit makes the resulting state transition attributable to a specific intervention and directly matches the atomic supervision used for policy learning in Sec. 3.3. If the policy predicts `STOP`, the repair process terminates without modifying the current scene.

**Execution and conservative acceptance.** Before execution, each non-terminal proposal is passed through a capability-aware validation layer:

$$a_t = \text{ValidateAndProject}(\hat{a}_t; C_g, S_t^{(g)}), \tag{5}$$

which checks the output schema, object references, backend capabilities, and valid parameter ranges. Malformed, unsupported, or ungroundable proposals are rejected rather than silently converted into another operation.

A valid action is grounded through the backend adapter and first applied to a temporary copy of the native scene:

$$\widetilde{S}_{t+1}^{(g)} = F_g\left(S_t^{(g)}, \psi_g(a_t; S_t^{(g)})\right), \tag{6}$$

where $F_g$ denotes the native scene-editing operation of generator $g$. ScenePilot then measures the realized change using a shared scene-quality score

$$V(S) = \lambda_{\text{phy}} Q_{\text{PHY}}(S) + \lambda_{\text{spa}} Q_{\text{SPA}}(S) + \lambda_{\text{vlm}} Q_{\text{VLM}}(S), \tag{7}$$

where $Q_{\text{PHY}}$, $Q_{\text{SPA}}$, and $Q_{\text{VLM}}$ measure physical validity, spatial consistency, and higher-level visual-semantic quality, respectively. The realized improvement is

$$\Delta V_t = V(\widetilde{S}_{t+1}^{(g)}) - V(S_t^{(g)}). \tag{8}$$

The tentative edit is committed only when the resulting scene improves the shared score:

$$S_{t+1}^{(g)} = \begin{cases} \widetilde{S}_{t+1}^{(g)}, & \Delta V_t > \epsilon, \\ S_t^{(g)}, & \text{otherwise.} \end{cases} \tag{9}$$

Thus, capability-aware validation determines whether a proposed edit can be executed, while the acceptance rule determines whether its realized consequence should be retained. Because this decision is made from the resulting scene rather than from the textual action itself, the same closed-loop procedure can be applied across heterogeneous backends.

**Closed-loop continuation and termination.** After either acceptance or rollback, ScenePilot reconstructs the canonical state and re-renders the retained scene before invoking the policy again. Consequently, each decision is conditioned on the scene that actually results from previous repair rounds rather than on a predicted or assumed next state. The loop continues until the policy predicts `STOP` or the maximum repair budget $T_{\max}$ is reached.

The three components therefore have distinct responsibilities: the shared policy determines *what should be changed*, the backend interface determines whether and how the proposal *can be executed*, and the acceptance mechanism determines whether the realized modification *should be retained*. The complete inference algorithm, including capability validation, tentative execution, accept-or-rollback decisions, and termination, is provided in Appendix E.2.

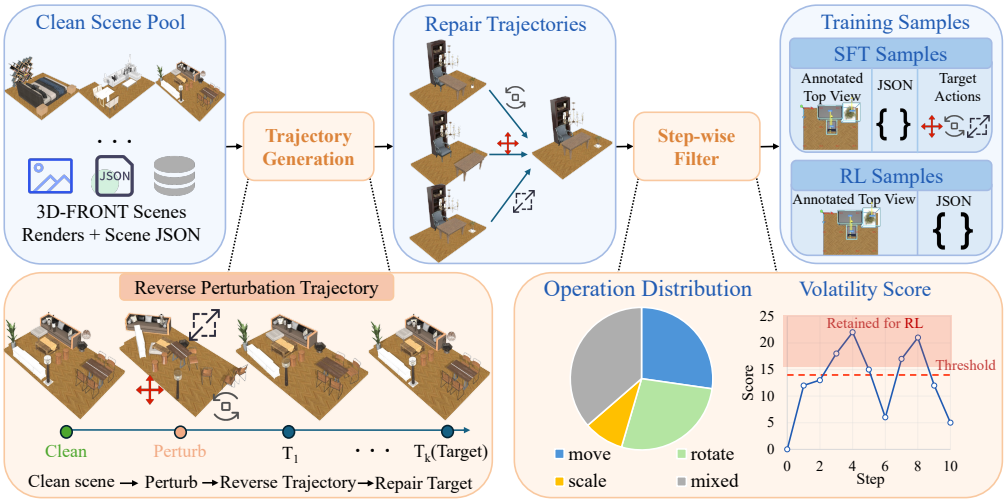


Figure 3: **Construction of SceneReverse.** Starting from a clean 3D-FRONT scene, we apply executable structured perturbations in the canonical edit space to obtain progressively degraded states. Reversing these perturbations produces atomic recovery decisions, while recovered terminal states provide explicit `STOP` supervision.

### 3.3 Learning a Shared Repair Policy from SceneReverse

Training a transferable repair policy requires supervision that describes how an imperfect scene can be improved, rather than only how a final scene should look. However, existing 3D scene datasets mainly provide clean layouts and do not contain intermediate degraded states paired with executable corrective decisions. Collecting failure trajectories from each target generator would also compromise generator-agnostic transfer. We therefore construct **SceneReverse**, a generator-independent repair trajectory dataset derived from high-quality 3D-FRONT scenes.

**Executable recovery trajectories.** Given a clean scene $S^\star$, SceneReverse generates degraded intermediate states through reversible object-level perturbations defined in the same canonical edit space as the repair policy. By reversing these perturbations, we obtain executable recovery trajectories consisting of atomic state–action transitions. Each transition records a degraded scene state and the corresponding recovery action, while the recovered terminal state is paired with a `STOP` decision.

Importantly, the recovery action induced by the reverse process is not assumed to be the unique optimal repair. Indoor scene repair is inherently multi-modal: different edits may lead to physically and functionally valid outcomes. SceneReverse therefore provides one executable and verifiable recovery path rather than a single deterministic ground truth.

Each recovery step is converted into an atomic state–action pair $(o_t^{\mathrm{r}}, a_t^\star)$ following the same observation structure used during inference. Specifically, $o_t^{\mathrm{r}}$ contains the current rendered views, canonical scene state, and a capability descriptor indicating the available canonical operations. The recovered terminal state is paired with `STOP`, teaching the policy that a satisfactory scene should be preserved rather than modified unnecessarily. The final SceneReverse pool contains 9,582 unique rooms, 20,012 validated recovery trajectories, and 53,599 atomic recovery steps. Detailed filtering criteria, difficulty coverage, data splits, and action statistics are provided in the Appendix C.2.

**Supervised process learning.** We first use SceneReverse to teach the policy grounded atomic repair behavior through supervised fine-tuning. Given a multimodal observation $o_t$, the policy is trained to predict either the demonstrated recovery action $a_t^\star$ or the terminal `STOP` decision:

$$\mathcal{L}_{\mathrm{SFT}} = -\mathbb{E}_{(o_t, a_t^\star)\sim\mathcal{D}_{\mathrm{SR}}} \left[\log \pi_\theta(a_t^\star \mid o_t)\right], \tag{10}$$

where $\mathcal{D}_{\mathrm{SR}}$ denotes the SceneReverse training set. Since each sample contains exactly one atomic target decision, the supervision directly matches the one-action-per-round inference protocol in Sec. 3.2. This stage teaches multimodal scene understanding, object grounding, canonical action prediction, structured-output compliance, and termination on recovered states.

To prepare the shared policy for heterogeneous backend capabilities, we additionally expose it to different capability configurations during SFT. Partial-capability examples disable selected non-target operations while keeping the demonstrated target action executable, so the policy learns to condition its decision on the currently available action set rather than assuming that every backend supports the full canonical interface. Generator identity itself is never provided as an input. Consequently, the policy is trained to associate repair decisions with scene content and available operations rather than with a particular generator.

**Learning from realized improvements.** Supervised process learning provides a grounded initialization, but the demonstrated inverse action is only one valid recovery decision and need not be the only beneficial edit for a given state. We therefore further optimize the SFT policy according to the realized consequences of its sampled actions. For each repair state, the current policy samples a group of candidate atomic decisions. Each candidate is parsed and validated through the same canonical interface used during inference, and executable non-terminal actions are applied to the current scene to obtain temporary next states.

For each candidate, we define the reward as

$$R = \lambda_{\text{fmt}} R_{\text{format}} + \lambda_{\text{phy}} \Delta Q_{\text{PHY}} + \lambda_{\text{spa}} \Delta Q_{\text{SPA}} + \lambda_{\text{vlm}} \Delta Q_{\text{VLM}}, \tag{11}$$

where $R_{\text{format}}$ measures whether the predicted decision follows the canonical output schema and can be grounded into an executable command. The remaining terms measure the scene-quality changes directly caused by the sampled action:

$$\Delta Q_m = Q_m(\widetilde{S}_{t+1}) - Q_m(S_t), \qquad m \in \{\text{PHY}, \text{SPA}, \text{VLM}\}, \tag{12}$$

where $Q_{\text{PHY}}$ measures physical validity, $Q_{\text{SPA}}$ measures spatial and relational consistency, and $Q_{\text{VLM}}$ provides visual-semantic feedback from the rendered scene. Using improvement rather than absolute scene quality assigns credit to the consequence of the intervention and allows alternative actions to receive positive feedback even when they differ from the demonstrated inverse action.

We optimize these candidate decisions with Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which compares sampled actions within the same repair state and regularizes the updated policy toward the SFT reference policy. The standard group-relative advantage, clipping objective, KL regularization, and complete optimization hyperparameters are provided in the appendix. Importantly, policy optimization is performed at the level of individual repair decisions rather than complete multi-round trajectories, providing localized credit assignment to the scene transition directly induced by each action.

At inference time, these learned atomic decisions are composed through the conservative closed-loop procedure in Sec. 3.2. After each retained modification, the scene is re-canonicalized and re-rendered before the next policy decision. SceneReverse therefore provides generator-independent process supervision for learning *how to repair*, while realized-improvement optimization allows the policy to move beyond imitation of a single synthetic recovery path.

## 4 Experiments

We evaluate ScenePilot around four questions. **RQ1:** Can one shared repair policy improve scenes produced by heterogeneous 3D generators without backend-specific policy tuning? **RQ2:** Does the learned policy transfer to a held-out generator through only the canonical adapter and capability interface? **RQ3:** How much do process supervision, GRPO, capability conditioning, and conservative closed-loop verification contribute? **RQ4:** Does ScenePilot improve scenes conservatively, rather than obtaining better metrics through excessive or harmful edits?

### 4.1 Experimental Setup

We evaluate ScenePilot on SceneEval (Tam et al., 2026) using five heterogeneous text-driven 3D scene generators: ReSpace, LayoutVLM, Reason-3D, LayoutGPT, and InstructScene. None of these generators or their outputs are used to train the repair policy. At evaluation time, the same frozen ScenePilot checkpoint is applied to all five generators without backend-specific policy fine-tuning; only lightweight adapters map each native scene representation and editing interface to the shared

Table 1: **Plug-and-play repair on SceneEval.** Parentheses in the Overall column denote relative percentage improvements over the corresponding Base row. All official SceneEval submetrics are reported under the two benchmark groups, Fidelity and Plausibility. The Overall score is computed from the nine core metrics in Eqs. (18–20); $\mathrm{COL}_{\mathrm{sc}}$ is reported for completeness but excluded from Overall to avoid double-counting collision.

| | | SceneEval Fidelity | | | | SceneEval Plausibility | | | | | | Overall↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Generator | Method | CNT↑ | ATR↑ | OOR↑ | OAR↑ | $\mathrm{COL}_{\mathrm{ob}}$ ↓ | $\mathrm{COL}_{\mathrm{sc}}$ ↓ | SUP↑ | NAV↑ | ACC↑ | OOB↓ | |
| ReSpace | Base | 80.18 | 49.47 | 17.89 | 18.24 | 31.44 | 55.56 | 12.07 | 100.00 | 61.12 | 5.78 | 52.77 |
| | + ScenePilot | 83.47 | 49.73 | 36.76 | 34.60 | 18.28 | 37.78 | 10.84 | 100.00 | 59.67 | 2.66 | 59.79 **(+13.3%)** |
| LayoutVLM | Base | 41.19 | 22.26 | 8.60 | 23.29 | 36.09 | 69.00 | 67.96 | 98.75 | 85.91 | 4.14 | 44.34 |
| | + ScenePilot | 41.19 | 39.53 | 16.90 | 24.71 | 33.21 | 54.00 | 68.03 | 97.88 | 87.42 | 2.89 | 50.52 **(+13.9%)** |
| Reason-3D | Base | 62.19 | 44.31 | 25.98 | 33.76 | 14.43 | 28.28 | 40.70 | 99.95 | 75.65 | 13.85 | 56.79 |
| | + ScenePilot | 65.32 | 47.63 | 32.88 | 34.10 | 12.86 | 25.91 | 38.47 | 99.87 | 73.12 | 9.64 | 59.15 **(+4.2%)** |
| LayoutGPT | Base | 19.54 | 18.98 | 2.87 | 5.24 | 12.96 | 30.00 | 28.24 | 100.00 | 47.29 | 73.11 | 25.98 |
| | + ScenePilot | 28.61 | 23.77 | 7.63 | 6.32 | 8.31 | 27.00 | 29.04 | 100.00 | 63.81 | 23.61 | 34.60 **(+33.2%)** |
| InstructScene | Base | 23.39 | 27.01 | 9.01 | 6.06 | 36.32 | 83.84 | 28.56 | 99.55 | 76.92 | 16.10 | 33.97 |
| | + ScenePilot | 27.49 | 30.91 | 18.41 | 23.63 | 31.03 | 77.24 | 28.56 | 99.62 | 75.81 | 13.07 | 42.51 **(+25.1%)** |

Table 2: **Repair-policy comparison.** Parentheses in the Overall column denote relative percentage improvements over Qwen3-VL-4B. All policies are evaluated on the same frozen ReSpace generator outputs using the full official SceneEval submetrics. Overall is computed from the same nine core metrics as in Table 1.

| | SceneEval Fidelity | | | | SceneEval Plausibility | | | | | | Overall↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Policy | CNT↑ | ATR↑ | OOR↑ | OAR↑ | $\mathrm{COL}_{\mathrm{ob}}$ ↓ | $\mathrm{COL}_{\mathrm{sc}}$ ↓ | SUP↑ | NAV↑ | ACC↑ | OOB↓ | |
| Qwen3-VL-4B | 79.12 | 48.83 | 6.30 | 24.54 | 29.25 | 47.14 | 15.41 | 100.00 | 62.95 | 3.97 | 52.35 |
| ScenePilot-SFT | 82.11 | 51.89 | 29.87 | 27.66 | 22.00 | 44.98 | 18.77 | 99.75 | 55.66 | 3.33 | 57.80 **(+10.4%)** |
| ScenePilot-SFT+GRPO | 83.47 | 49.73 | 36.76 | 34.60 | 18.28 | 37.78 | 10.84 | 100.00 | 59.67 | 2.66 | 59.79 **(+14.2%)** |
| w/o verify | 78.74 | 47.69 | 27.91 | 25.70 | 21.70 | 42.13 | 17.42 | 100.00 | 53.45 | 3.92 | 55.75 **(+6.5%)** |
| Expert-GPT-5.4† | 83.22 | 59.62 | 43.71 | 47.64 | 15.40 | 18.74 | 23.69 | 100.00 | 69.17 | 1.69 | 66.33 **(+26.7%)** |

canonical interface. Because different generators may produce different object inventories and scene distributions, we evaluate ScenePilot through within-generator paired comparison, i.e., $\mathrm{Base}_g$ versus $\mathrm{Base}_g + \mathrm{ScenePilot}$, rather than comparing absolute scores across generators. We report the official SceneEval Fidelity and Plausibility metrics together with a compact Overall summary score; the aggregation definitions are provided in Appendix F.1. Visual-semantic quality is evaluated separately using a blind GPT-5.4 (OpenAI, 2026) judge on layout coherence, relation fidelity, and object completeness, complemented by a blinded human study under the same criteria.

### 4.2 Main Results

Table 1 reports the complete SceneEval fidelity and plausibility metrics, organized under the two benchmark categories, together with the Overall summary score. Because different generators may instantiate different object inventories, we evaluate ScenePilot through paired improvement over each generator's native output rather than by directly comparing absolute scores across generators. Importantly, the repair policy is trained without generator-specific fine-tuning and is applied to all evaluated generators through their canonical interface adapters. These results therefore evaluate whether a single repair policy can transfer across heterogeneous 3D scene generation pipelines.

Figure 4 provides a qualitative illustration of this cross-generator repair behavior under the same instruction. Additional qualitative comparisons across generators are provided in Appendix G.7.

**Effect of policy learning.** Table 2 isolates the effect of policy learning by comparing the pretrained Qwen3-VL-4B backbone, ScenePilot after SFT, the final SFT+GRPO policy, and an expert GPT-5.4 repair agent on the same frozen ReSpace outputs.

**Perceptual evaluation.** Table 3 reports 1–10 perceptual scores from two independent VLM judges, GPT-5.4 and Gemini-3-Flash, on *layout coherence*, *relation fidelity*, and *object complete-*

Table 3: **Average perceptual scores from two independent VLM judges.** Scores range from 1 to 10, with higher values indicating better perceptual quality. GPT-5.4 and Gemini-3-Flash independently evaluate each scene on layout coherence, relation fidelity, and object completeness without observing method identities or paired before/after results.

| | GPT-5.4 | | | | Gemini-3-Flash | | | |
|---|---|---|---|---|---|---|---|---|
| Policy | Layout↑ | Relation↑ | Completeness↑ | Average↑ | Layout↑ | Relation↑ | Completeness↑ | Average↑ |
| ReSpace | 7.10 | 6.57 | 7.05 | 6.91 | 6.33 | 6.24 | 7.05 | 6.54 |
| w/ ScenePilot (ours) | **7.24** | **7.19** | **7.57** | **7.33** | **6.81** | 6.86 | **7.48** | **7.05** |
| LayoutVLM | 6.29 | 6.10 | 6.33 | 6.24 | 6.14 | 6.19 | 6.24 | 6.19 |
| w/ ScenePilot (ours) | 6.73 | 6.82 | 6.35 | 6.63 | 6.57 | **7.12** | 6.27 | 6.65 |

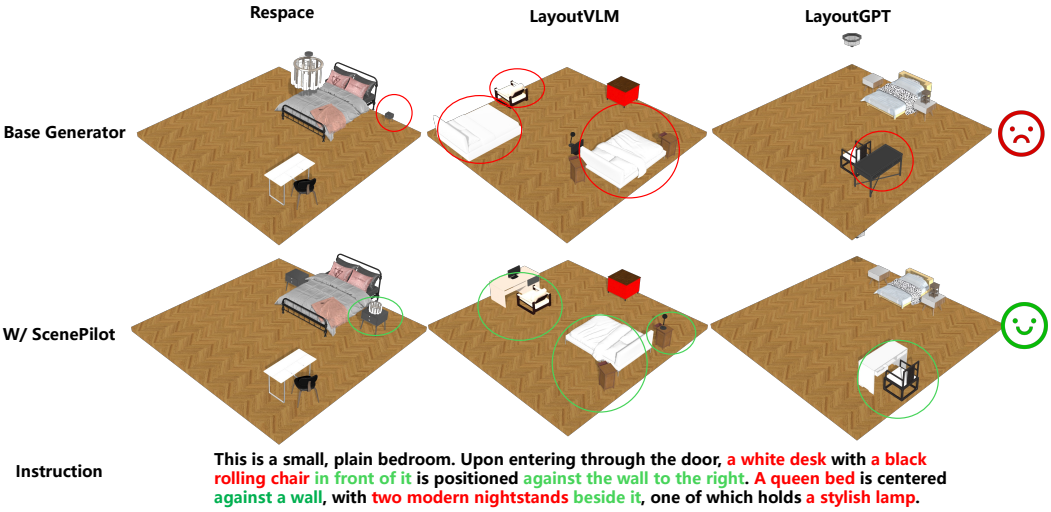


Figure 4: **Qualitative examples of plug-and-play repair across generators.** Each column compares the native generator output with the result after applying the same frozen ScenePilot policy under the same instruction. Highlighted regions illustrate representative local repairs without regenerating the scene from scratch.

*ness*. Higher is better. We also conduct a blinded human study under the same criteria; the rank-based analysis is detailed in Appendix F.2.

## 5 Conclusion

We presented **ScenePilot**, a plug-and-play framework for generator-agnostic 3D scene repair. By decoupling repair reasoning from generator-specific execution through a canonical interface, learning transferable repair behavior from SceneReverse, and applying conservative closed-loop verification, ScenePilot enables one shared policy to refine scenes from heterogeneous generators without backend-specific policy fine-tuning. Experiments on SceneEval show consistent improvements across five generators unseen during policy training, supporting 3D scene repair as a reusable capability rather than a generator-specific component.

**Limitations and Future Work.** ScenePilot currently focuses on a compact set of object-level geometric edits, limiting its ability to address errors that require richer structural changes. In addition, each new generator still requires a lightweight adapter for representation conversion and action execution. Future work could explore more expressive canonical operations, more automated adapter construction, and more efficient verification, while extending evaluation to broader scene domains and generation systems.

### AI USE STATEMENT

In this work, we used generative AI tools for language editing and assistance with code development. All AI-assisted content, mathematical derivations, experimental implementations, and generated text were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work does not introduce new training data or collect human-subject data. The proposed method inherits potential risks and biases from the underlying pretrained generative models.

### REPRODUCIBILITY STATEMENT

We provide the complete mathematical formulation, optimization procedure, and implementation details required to reproduce the proposed method. Hyperparameter settings, evaluation protocols, and additional experimental results are provided in the main paper and appendix. Code will be released upon publication.

## REFERENCES

Shuai Bai et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, Afshin Dehghan, and Joshua Susskind. Gaudi: A neural architect for immersive 3d scene generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 25102–25116, 2022.

Frédéric Berdoz, Luca A. Lanzendörfer, Nick Tuninga, and Roger Wattenhofer. Text-to-scene with large reasoning models. *arXiv preprint arXiv:2509.26091*, 2025.

Martin JJ Bucher and Iro Armeni. Respace: Text-driven 3d scene synthesis and editing with preference alignment. *arXiv e-prints*, pp. arXiv–2506, 2025.

Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-design: Personalized llm interior designer. In *European Conference on Computer Vision*, pp. 217–234. Springer, 2024.

Minglin Chen, Rongkun Yang, Qibin Hu, Kaiwen Xue, Shunbo Zhou, and Yulan Guo. Graph2scene: Versatile 3d indoor scene generation with interaction-aware scene graph. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 11313–11320, 2025.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied ai using procedural generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5982–5994, 2022.

Wei Deng, Mengshi Qi, and Huadong Ma. Global-local tree search in vlms for 3d indoor scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Chuan Fang, Yuan Dong, Kunming Luo, Xiaotao Hu, Rakesh Shrestha, and Ping Tan. Ctrl-room: Controllable text-to-3d room meshes generation with layout constraints. In *International Conference on 3D Vision*, pp. 692–701. IEEE, 2025.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sreyashi Basu, Xin Eric Wang, William Yang Ma, Ranjay Krishna, et al. Layoutgpt: Compositional visual planning and generation with large language models. *arXiv preprint arXiv:2305.15393*, 2023.

Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Cao Wang, Hongbo Li, Yiyun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

Zeqi Gu, Yin Cui, Zhaoshuo Li, Fangyin Wei, Yunhao Ge, Jinwei Gu, Ming-Yu Liu, Abe Davis, and Yifan Ding. Artiscene: Language-driven artistic 3d scene generation through image intermediary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Chenguo Lin and Yadong Mu. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. *arXiv preprint arXiv:2402.04717*, 2024.

Guying Lin, Kemeng Huang, Michael Liu, Ruihan Gao, Hanke Chen, Lyuhao Chen, Beijia Lu, Taku Komura, Yuan Liu, Jun-Yan Zhu, and Minchen Li. Pat3d: Physics-augmented text-to-3d scene generation. In *International Conference on Learning Representations*, 2026.

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. *arXiv preprint arXiv:2505.02836*, 2025.

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 *Advances in Neural Information Processing Systems*, 2023.

Léopold Maillard, Nicolas Sereyjol-Garros, Tom Durand, and Maks Ovsjanikov. Debara: Denoising-based 3d room arrangement generation. In *Advances in Neural Information Processing Systems*, 2024.

Başak Melis Öcal, Maxim Tatarchenko, Sezer Karaoğlu, and Theo Gevers. Sceneteller: Language-to-3d scene generation. In *European Conference on Computer Vision*, pp. 362–378. Springer, 2024.

OpenAI. GPT-5.4 Thinking System Card. https://openai.com/index/gpt-5-4-thinking-system-card/, March 2026.

Zhenyu Pan and Han Liu. Metaspatial: Reinforcing 3d spatial reasoning in vlms for the metaverse. *arXiv preprint arXiv:2503.18470*, 2025.

Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Xingjian Ran, Yixuan Li, Linning Xu, Mulin Yu, and Bo Dai. Direct numerical layout generation for 3d indoor scene synthesis via spatial reasoning. *arXiv preprint arXiv:2506.05341*, 2025.

Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yukai Shi, Weiyu Li, Zihao Wang, Hongyang Li, Xingyu Chen, Ping Tan, and Lei Zhang. Scenemaker: Open-set 3d scene generation with decoupled de-occlusion and pose estimation model. *arXiv preprint arXiv:2512.10957*, 2025.

Liangchen Song, Liangliang Cao, Hongyu Xu, Kai Kang, Feng Tang, Junsong Yuan, and Yang Zhao. Roomdreamer: Text-driven 3d indoor scene synthesis with coherent geometry and texture. *arXiv preprint arXiv:2305.11337*, 2023.

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. *arXiv preprint arXiv:2412.02193*, 2025.

Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. SceneEval: Evaluating semantic coherence in text-conditioned 3D indoor scene synthesis. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2026.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Cheng Wan, Yongsen Mao, Wenzheng Wu, Yuxuan Xie, Chucheng Xiang, Runze Wang, Xiang Zhang, Zhongyuan Liu, Rushi Dai, and Yuan Liu. Nala: A 3d native llm layout agent for high-quality 3d scene generation. *arXiv preprint arXiv:2606.29395*, 2026.

Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Trans. Graph.*, 37(4):70, 2018.

Kai Wang, Yu-An Lin, Benjamin Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. PlanIT: Planning and instantiating indoor scenes with relation graph and spatial prior networks. In *ACM Transactions on Graphics*, volume 38, pp. 1–15, 2019.

Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *International Conference on 3D Vision*, 2021.

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. *arXiv preprint arXiv:2602.10116*, 2026.

Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024a.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent. *arXiv preprint arXiv:2509.20414*, 2025a.

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

Zhifei Yang, Keyang Lu, Chao Zhang, Jiaxing Qi, Hanqi Jiang, Ruifei Ma, Shenglin Yin, Yifan Xu, Mingzhe Xing, Zhen Xiao, Jieyi Long, Xiangde Liu, and Guangyao Zhai. Mmgdreamer: Mixed-modality graph for geometry-controllable 3d indoor scene generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 9391–9399, 2025b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Hongwei Yi, Chun-Hao P. Huang, Shashank Tripathi, Lea Hering, Justus Thies, and Michael J. Black. MIME: Human-aware 3D scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12965–12976, 2023.

Meisheng Zhang, Shizhao Sun, Yang Zhao, Ziyuan Liu, Zhijun Gao, and Jiang Bian. Orchestrating spatial semantics via a zone-graph paradigm for intricate indoor scene generation. *arXiv preprint arXiv:2605.02537*, 2026.

Qihang Zhang, Yinghao Xu, Yujun Shen, Bo Dai, Bolei Zhou, and Ceyuan Yang. Berfscene: Bev-conditioned equivariant radiance fields for infinite 3d scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

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. *arXiv preprint arXiv:2602.09432*, 2026.

Haiyang Zhou, Xinhua Cheng, Wangbo Yu, Yonghong Tian, and Li Yuan. Holodreamer: Holistic 3d panoramic scene generation from text descriptions. *IEEE Transactions on Visualization and Computer Graphics*, 2025a.

Mengqi Zhou, Yuxi Wang, Jun Hou, Shougao Zhang, Yiwei Li, Chuanchen Luo, Junran Peng, and Zhaoxiang Zhang. Scenex: Procedural controllable large-scale scene generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 10806–10814, 2025b. doi: 10.1609/aaai.v39i10.33174.

# SUPPLEMENTARY CONTENTS


**A Experimental Details** **15**

A.1 Model and Training Setup . . . . . 15

A.2 SceneReverse Data Statistics . . . . . 15

A.3 Capability-Conditioned SFT Sampling . . . . . 16

A.4 Benchmark and Evaluation Protocol . . . . . 16

**B Additional Quantitative Results** **17**

B.1 Detailed VLM-Judge Results . . . . . 17

B.2 Human–VLM Agreement . . . . . 17

**C Additional Details of SceneReverse** **17**

C.1 Canonical Scene Representation . . . . . 17

C.2 SceneReverse Construction and Filtering Details . . . . . 18

C.3 Terminal Supervision . . . . . 19

**D Generator Adapters and Canonical Interface** **19**

D.1 Backend Capability Matrix . . . . . 19

D.2 Adapter Responsibilities . . . . . 19

**E Training and Inference Details** **19**

E.1 Step-Level GRPO Credit Assignment . . . . . 19

E.2 Conservative Closed-Loop Inference . . . . . 20

**F Additional Evaluation Details** **20**

F.1 SceneEval Aggregation . . . . . 20

F.2 Perceptual Evaluation Protocol . . . . . 21

**G Prompt and Output Schemas** **21**

G.1 Prompt Audit and Frozen ScenePilot Policy Prompt . . . . . 21

G.2 Multimodal Observation Contract . . . . . 22

G.3 Canonical Atomic Output Schema . . . . . 22

G.4 Prompt Versioning and Excluded Historical Prompts . . . . . 23

G.5 GRPO VLM Reward Judge . . . . . 23

G.6 VLM-based Evaluation Protocol . . . . . 27

G.7 Additional Qualitative Results . . . . . 27

# A Experimental Details

## A.1 Model and Training Setup

**Base model and observation format.** We instantiate the shared repair policy with Qwen3-VL-4B-Instruct. Each policy input contains two rendered observations of the current scene—a diagonal view and an annotated top-down view—together with the canonical structured scene state and the capability descriptor exposed by the current backend. The model predicts exactly one atomic canonical decision from

$$\mathcal{A} \cup \{\texttt{STOP}\}, \qquad \mathcal{A} = \{\texttt{translate}, \texttt{rotate}, \texttt{scale}, \texttt{delete}, \texttt{add}\}.$$

The output is constrained to a strict JSON schema and contains no free-form explanation. Generator identity is not provided as an explicit policy input.

**Supervised fine-tuning.** The final SFT stage uses the room-disjoint SceneReverse SFT partition described in Sec. Appendix A.2. We train for one epoch with LoRA rank 64, LoRA scaling factor 128, and dropout 0.05. Optimization uses AdamW with learning rate $2 \times 10^{-5}$, weight decay 0.1, cosine learning-rate decay, warmup ratio 0.03, and bf16 precision. The per-device batch size is 1, and gradient accumulation is chosen to maintain an effective global batch size of approximately 32. We use a maximum sequence length of 5632 tokens and FlashAttention-2. Only assistant output tokens are supervised; image tokens and input prompt tokens are masked from the language-model loss.

**GRPO initialization.** The GRPO stage is initialized from the final SFT checkpoint and uses only the disjoint GRPO partition of SceneReverse. Each sampled candidate is parsed through the same canonical schema and capability-aware execution interface used at inference. The final GRPO reward is

$$R = \lambda_{\text{fmt}} R_{\text{format}} + \lambda_{\text{phy}} \Delta Q_{\text{PHY}} + \lambda_{\text{spa}} \Delta Q_{\text{SPA}} + \lambda_{\text{vlm}} \Delta Q_{\text{VLM}}, \tag{13}$$

where $R_{\text{format}}$ jointly measures schema validity and executability, while the remaining terms measure the realized scene-quality change after applying the candidate edit. We do not use a ground-truth action-matching reward. The final rollout and optimizer hyperparameters are reported from the frozen training run in Table 4.

Table 4: GRPO hyperparameters. Values that depend on the final frozen training run are left as placeholders until training is completed.

| Setting | Value |
|---|---|
| GRPO states | 8,574 |
| Rollouts per state | 4 |
| Learning rate | $5 \times 10^{-6}$ |
| Epochs | 1 |
| $\lambda_{\text{fmt}}$ | 0.30 |
| $\lambda_{\text{phy}}$ | 0.40 |
| $\lambda_{\text{spa}}$ | 0.15 |
| $\lambda_{\text{vlm}}$ | 0.15 |
| KL coefficient | 0.04 |

**Implementation environment.** Training is conducted on a Linux server with two NVIDIA RTX A6000 GPUs (48 GB each), while evaluation is performed on a separate Linux server with four NVIDIA RTX 4090D GPUs (24 GB each). Rendering uses the backend-native scene assets and the same renderer for the unrepaired and repaired versions of each generator output. Unless otherwise noted, the reported ScenePilot refinement time excludes the runtime of the original scene generator.

## A.2 SceneReverse Data Statistics

The final SceneReverse pool contains 9,582 unique rooms, 20,012 validated recovery trajectories, and 53,599 atomic recovery states. We split the pool by room identity, so no room appears in both SFT and GRPO. Table 5 summarizes the final split.

Table 5: Statistics of the final room-disjoint SceneReverse training split.

| Item | Full pool | SFT | GRPO |
|---|---|---|---|
| Rooms | 9,582 | 8,050 | 1,532 |
| Trajectories | 20,012 | 16,810 | 3,202 |
| Recovery steps | 53,599 | 45,025 | 8,574 |
| Terminal `STOP` samples | N/A | 4,246 | N/A |
| Total SFT samples | N/A | 49,271 | N/A |

Table 6 reports the decision distribution in the final SFT set. Each sample contains exactly one atomic target decision.

Table 6: Atomic decision distribution in the final SceneReverse SFT set.

| Decision | Samples |
|---|---|
| `translate` | 15,161 |
| `rotate` | 13,783 |
| `scale` | 2,297 |
| `delete` | 6,892 |
| `add` | 6,892 |
| `STOP` | 4,246 |
| Total | 49,271 |

### A.3 Capability-Conditioned SFT Sampling

To prevent the SFT policy from assuming that every backend supports the full canonical action set, we augment each SFT sample with a capability descriptor. The final capability sampler produces approximately equal numbers of full-capability and partial-capability examples. Of the $49{,}271$ SFT samples, $24{,}806$ expose all five edit operations and $24{,}465$ expose a four-operation subset. For a non-terminal target, partial-capability sampling disables exactly one non-target operation, ensuring that the demonstrated target action remains executable. For a terminal `STOP` sample, one edit operation is disabled while `STOP` remains available. This construction exposes the policy to heterogeneous action interfaces without revealing generator identity.

### A.4 Benchmark and Evaluation Protocol

**SceneEval protocol.** The final cross-generator benchmark uses a fixed set of $100$ text prompts, denoted S ce ne Ev a l. The same prompts and logical room specification are given to ReSpace, LayoutVLM, Reason-3D, and LayoutGPT. Each generator retains its native object-selection mechanism, asset inventory, and scene representation. ScenePilot is applied only after the native scene has been generated. LayoutGPT is held out from policy training and policy-level tuning.

**Paired comparison.** Because heterogeneous generators may instantiate different object inventories for the same prompt, our primary comparison is paired within each backend:

$$\mathrm{Base}_g \quad \text{vs.} \quad \mathrm{Base}_g + \mathrm{ScenePilot}.$$

We therefore emphasize generator-level improvements and macro-average these improvements across backends rather than treating absolute scores from incompatible object sets as directly interchangeable.

**SceneEval metrics.** We use the official SceneEval evaluator. The paper-level Fidelity, Plausibility, and Overall scores are computed exactly by Eqs. (18–20). The Fidelity aggregate uses CNT, ATR, OOR, and OAR. The Plausibility aggregate uses object-level collision, support, navigability, accessibility, and out-of-boundary rate, with lower-is-better terms complemented before averaging. We additionally retain $\mathrm{COL}_{\mathrm{sc}}$ in the detailed results below but exclude it from the aggregate to avoid assigning collision two weights.

**VLM-based evaluation.** For final visual-semantic evaluation, we use GPT-5.4 at temperature 0 in a blind single-scene protocol. Each request contains only the annotated top-down and diagonal renders of one scene and does not expose the generator, policy, method identity, paired baseline, or expected improvement. The judge assigns an integer score from 0 to 10 for layout coherence, relation fidelity, and object completeness. These absolute scores are retained for diagnostic reporting, while per-scene scores are converted offline to ranks for the perceptual comparison in Table 8. Judge failures are recorded as missing values rather than silently replaced by heuristic scores.

# B Additional Quantitative Results

## B.1 Detailed VLM-Judge Results

Table 7: Detailed blind VLM-judge scores before conversion to ranks. Scores are integers in $[0, 10]$ at the scene level; the table reports means across valid scenes.

| Policy | Layout↑ | Relation↑ | Completeness↑ | Avg.↑ |
|---|---|---|---|---|
| Qwen3-VL-4B | 6.92 | 6.59 | 6.83 | 6.78 |
| ScenePilot-SFT | 7.35 | 6.65 | 6.85 | 6.95 |
| ScenePilot-SFT+GRPO | 7.57 | 6.93 | 7.01 | 7.17 |
| w/o ScenePilot | 7.21 | 6.81 | 6.92 | 6.98 |
| Expert-GPT-5.4$^{\dagger}$ | 7.24 | 7.19 | 7.57 | 7.33 |

## B.2 Human–VLM Agreement

We use a blinded human study as an auxiliary validation of the VLM-based evaluator. We recruit 15 graduate students to rank anonymized policy outputs under the same three criteria used by the VLM judges: *layout coherence*, *relation fidelity*, and *object completeness*. We report average human ranks and assess agreement with the VLM evaluator using Pearson's $r$ and Spearman's $\rho$, measuring linear and rank-order correlation, respectively.

Table 8: **Average ranks based on blinded user ratings and VLM scores.** Lower rank indicates better perceptual quality.

| | **User** | | | **VLM** | | |
|---|---|---|---|---|---|---|
| Policy | Layout↓ | Relation↓ | Completeness↓ | Layout↓ | Relation↓ | Completeness↓ |
| ReSpace | 2.31 | 2.19 | 2.07 | 2.21 | 2.48 | 2.18 |
| w/ ScenePilot (ours) | 1.62 | 1.52 | 1.73 | 1.73 | 1.68 | 1.68 |
| LayoutVLM | 3.34 | 3.27 | 3.08 | 3.08 | 3.38 | 3.10 |
| w/ ScenePilot (ours) | 2.72 | 3.02 | 3.11 | 2.98 | 2.46 | 3.05 |

Table 9: **Agreement between blinded user ratings and VLM scores.**

| Criterion | Pearson $r$ | Spearman $\rho$ |
|---|---|---|
| Layout coherence | 0.950 | 1.000 |
| Relation fidelity | 0.914 | 1.000 |
| Object completeness | 0.994 | 0.800 |

# C Additional Details of SceneReverse

## C.1 Canonical Scene Representation

SceneReverse and the repair policy use the same generator-independent, object-centric representation. A canonical scene is written as

$$z = (B, \mathcal{O}), \qquad \mathcal{O} = \{o_i\}_{i=1}^{N},$$

where $B$ denotes the room boundary and each object

$$o_i = (\mathrm{id}_i, c_i, \mathbf{p}_i, \theta_i, \mathbf{s}_i)$$

contains a stable object identifier, semantic category, metric position, yaw angle, and physical size. Backend adapters normalize native coordinate frames, units, yaw conventions, and room-boundary formats before the state is exposed to the policy.

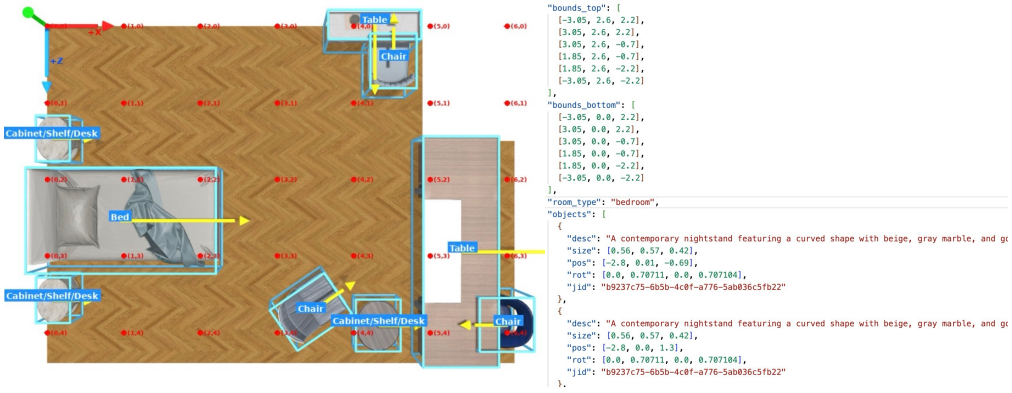


Figure 5: Example of the canonical object-centric scene representation used by ScenePilot.

### C.2 SceneReverse Construction and Filtering Details

SceneReverse starts from a clean scene $S^\star$ and applies a sequence of structured perturbations in the canonical edit space. Each perturbation has a known inverse, allowing the construction of an executable recovery path. The resulting inverse action should be interpreted as one verifiable recovery action induced by the corruption process, rather than as the unique optimal repair for the degraded scene.

Given a clean scene $S^\star$, we generate degraded intermediate states through executable perturbations:

$$\hat{S}_0 = S^\star, \qquad \hat{S}_t = c_t(\hat{S}_{t-1}; \beta_t), \qquad c_t \sim \mathcal{C}, \tag{14}$$

where $\mathcal{C}$ denotes the set of reversible perturbations defined in the canonical action space, and $\beta_t$ controls the perturbation magnitude.

Starting from the most degraded state $S^{\mathrm{r}}_0 = \hat{S}_T$, the recovery trajectory is obtained by reversing the applied perturbations:

$$a^\star_t = \mathrm{Inv}(c_{T-t}), \qquad S^{\mathrm{r}}_{t+1} = f_{\mathrm{apply}}\left(S^{\mathrm{r}}_t, a^\star_t\right), \quad t = 0, \ldots, T-1, \tag{15}$$

To remove nearly unchanged transitions, we measure the normalized magnitude of the state change. For matched objects, the per-step geometric change can be written as

$$V_t = \frac{1}{|\mathcal{O}_t|} \sum_{o_i \in \mathcal{O}_t} \left[ w_p \frac{\|\Delta \mathbf{p}_{i,t}\|_2}{d_B + \epsilon} + w_r \frac{|\Delta \theta_{i,t}|}{180^\circ} + w_s \left\| \log \frac{\mathbf{s}_{i,t+1}}{\mathbf{s}_{i,t}} \right\|_1 \right], \tag{16}$$

where $d_B$ is a room-scale normalization factor derived from the floor-plan extent. This normalization prevents translation, rotation, and scale changes from being compared in incompatible units.

We additionally characterize how far an intermediate state remains from the clean reference. For each matched object, we normalize the remaining error by the magnitude of the original corruption:

$$D_t = \frac{1}{|\mathcal{O}_t|} \sum_{o_i \in \mathcal{O}_t} \left[ \alpha_p \frac{\|\mathbf{p}^\star_i - \mathbf{p}_{i,t}\|_2}{\|\mathbf{p}^\star_i - \mathbf{p}^0_i\|_2 + \epsilon} + \alpha_r \frac{|\theta^\star_i - \theta_{i,t}|}{|\theta^\star_i - \theta^0_i| + \epsilon} + \alpha_s \frac{\|\mathbf{s}^\star_i - \mathbf{s}_{i,t}\|_2}{\|\mathbf{s}^\star_i - \mathbf{s}^0_i\|_2 + \epsilon} \right], \tag{17}$$

where $(\mathbf{p}_i^0, \theta_i^0, \mathbf{s}_i^0)$ denotes the initial degraded state and $(\mathbf{p}_i^\star, \theta_i^\star, \mathbf{s}_i^\star)$ denotes the clean target. We use this residual measure only to organize recovery difficulty and data coverage; it is not used as a ground-truth action-matching reward during GRPO.

### C.3 Terminal Supervision

Recovery trajectories provide executable supervision until the clean terminal state is reached. We pair a subset of recovered terminal states with `STOP`, yielding 4,246 terminal SFT examples. The terminal examples teach the policy to preserve a satisfactory state rather than applying unnecessary edits.

## D Generator Adapters and Canonical Interface

### D.1 Backend Capability Matrix

Each generator adapter exposes only the canonical operations that can be grounded into executable native edits. Table 10 is filled from the final frozen adapters rather than inferred from generator names.

Table 10: Distribution of atomic ScenePilot decisions during one repair run over the 100 SceneEval cases for each generator. Values denote the number of executed actions; "–" indicates that the corresponding operation is not exposed by the backend adapter.

| Generator | Translate | Rotate | Scale | Delete | Add | Stop |
|---|---|---|---|---|---|---|
| ReSpace | 44 | 35 | 14 | 2 | 9 | 3 |
| LayoutVLM | 48 | 40 | – | – | – | 12 |
| Reason-3D | 41 | 30 | – | 4 | 8 | 7 |
| LayoutGPT | 23 | 15 | 8 | 8 | 29 | 6 |
| InstructScene | 34 | 31 | 13 | 8 | 11 | 2 |

### D.2 Adapter Responsibilities

For backend $g$, the interface

$$\mathcal{I}_g = (\phi_g, \psi_g, C_g)$$

has three responsibilities. First, $\phi_g$ canonicalizes the native room geometry, object identifiers, poses, physical sizes, and semantic categories. Second, $\psi_g$ translates one validated canonical action into the backend's native editing representation. Third, $C_g$ exposes the supported operation subset and valid parameter ranges. No generator-specific policy checkpoint is introduced: the same repair-policy parameters are reused across all backends.

The adapter is intentionally restricted to representation conversion and action execution. It does not contain generator-specific repair heuristics, hidden prompts that decide which object should be changed, or backend-specific policy fine-tuning. This separation is essential for interpreting held-out-backend performance as transfer of the shared repair policy rather than as a collection of separately engineered optimizers.

## E Training and Inference Details

### E.1 Step-Level GRPO Credit Assignment

For each GRPO state $S_t$, the policy samples a group of candidate atomic decisions. A candidate is first parsed and validated. If it is executable and non-terminal, it is applied to a temporary copy of the scene, producing $\widetilde{S}_{t+1}^{(k)}$. Reward components are computed from realized scene deltas

$$\Delta Q_m^{(k)} = Q_m(\widetilde{S}_{t+1}^{(k)}) - Q_m(S_t), \qquad m \in \{\mathrm{PHY}, \mathrm{SPA}, \mathrm{VLM}\}.$$

The group-relative advantage is obtained by normalizing rewards within the sampled group, and the policy is optimized with the standard clipped GRPO objective and KL regularization to the SFT reference policy. Because credit is assigned to individual executable edits, the RL stage does not require an oracle final layout or a unique ground-truth repair action.

### E.2 Conservative Closed-Loop Inference

Algorithm 1 summarizes the final generator-agnostic inference loop. Each round proposes at most one atomic intervention. The candidate edit is committed only when it passes schema/capability validation, executes successfully, and satisfies the conservative verifier. The next observation is always reconstructed from the retained scene after acceptance or rollback.

**Algorithm 1** ScenePilot Conservative Closed-Loop Repair

**Require:** Native scene $S_0^{(g)}$, adapter $\mathcal{I}_g = (\phi_g, \psi_g, C_g)$, repair policy $\pi_\theta$, maximum rounds $T_{\max}$
**Ensure:** Repaired native scene $S_T^{(g)}$

$H_0 \leftarrow \varnothing$
**for** $t = 0$ to $T_{\max} - 1$ **do**
  $z_t \leftarrow \phi_g(S_t^{(g)})$
  Render $I_t^{\text{diag}}$ and $I_t^{\text{ann-top}}$
  $o_t \leftarrow (I_t^{\text{diag}}, I_t^{\text{ann-top}}, z_t, C_g, H_t)$
  $\hat{a}_t \sim \pi_\theta(\cdot \mid o_t)$
  **if** $\hat{a}_t =$ `STOP` and `STOP` is valid **then**
    **break**
  **end if**
  $a_t \leftarrow \text{ValidateAndProject}(\hat{a}_t; \mathcal{A}_g, S_t^{(g)})$
  **if** $a_t$ is invalid **then**
    Update $H_t$ with validation failure; **continue**
  **end if**
  $\widetilde{S}_{t+1}^{(g)} \leftarrow F_g(S_t^{(g)}, \psi_g(a_t; S_t^{(g)}))$
  Evaluate realized improvement $\Delta V_t$
  **if** $\Delta V_t > \epsilon$ **then**
    $S_{t+1}^{(g)} \leftarrow \widetilde{S}_{t+1}^{(g)}$
    Record ACCEPT
  **else**
    $S_{t+1}^{(g)} \leftarrow S_t^{(g)}$
    Record ROLLBACK
  **end if**
  Update history $H_{t+1}$ from the realized outcome
**end for**
**return** retained scene $S_T^{(g)}$

## F Additional Evaluation Details

### F.1 SceneEval Aggregation

For compact cross-metric reporting, we compute Fidelity and Plausibility aggregates from the official SceneEval metrics. All constituent metrics are percentages in $[0, 100]$.

Since $\text{COL}_{\text{ob}}$ and OOB are lower-is-better metrics, we complement them before aggregation so that all terms follow the same higher-is-better direction:

$$S_{\text{Fid}} = \frac{\text{CNT} + \text{ATR} + \text{OOR} + \text{OAR}}{4}, \tag{18}$$

$$S_{\text{Plaus}} = \frac{(100 - \text{COL}_{\text{ob}}) + \text{SUP} + \text{NAV} + \text{ACC} + (100 - \text{OOB})}{5}, \tag{19}$$

We then report a balanced overall score:

$$S_{\text{Overall}} = \sqrt{S_{\text{Fid}}\, S_{\text{Plaus}}}, \tag{20}$$

The Overall score is a summary derived from official SceneEval outputs rather than a replacement for the original metrics. We retain all original SceneEval metrics in the main results and appendix tables.

### F.2 Perceptual Evaluation Protocol

For visual-semantic evaluation, each VLM judge independently scores every scene on *layout coherence*, *relation fidelity*, and *object completeness*. For criterion $c$ and policy $m$, the independent score assigned to frozen scene $i$ is denoted by $s^{\text{VLM}}_{i,m,c}$. We convert the scores obtained for the same scene into policy ranks and compute the average rank as

$$\bar{r}^{\text{VLM}}_{m,c} = \frac{1}{N}\sum_{i=1}^{N} \text{rank}_m\left(-s^{\text{VLM}}_{i,m,c}\right), \tag{21}$$

where lower rank is better and tied scores receive average ranks.

For the human evaluation, participants are shown anonymized and randomly ordered outputs produced from the same frozen input and rank them according to the same three criteria. Let $r^{\text{Human}}_{i,m,c,j}$ denote the rank assigned by participant $j$. We report the average human rank across scenes and valid raters:

$$\bar{r}^{\text{Human}}_{m,c} = \frac{1}{N|\mathcal{J}|}\sum_{i=1}^{N}\sum_{j\in\mathcal{J}} r^{\text{Human}}_{i,m,c,j}, \tag{22}$$

where $\mathcal{J}$ denotes the set of valid human raters. This produces directly comparable aggregate rankings for the VLM and human evaluations while keeping the VLM judgments independent and blind.

## G Prompt and Output Schemas

### G.1 Prompt Audit and Frozen ScenePilot Policy Prompt

We audit prompts against the final protocol-clean SceneReverse materialization rather than historical training or demo artifacts. The active policy contract uses the five canonical edit operations `translate`, `rotate`, `scale`, `delete`, and `add`, with `STOP` as the only terminal decision. The final SFT materialization contains no generator identity, diagnostic/RAG/prior field, or free-form explanation.

The static text of the final SFT policy prompt is shown below. At materialization time, `SCENE_JSON` and `HOST_CAPABILITIES` are replaced by compact serialized JSON. The line breaks below are introduced only for readability; the runtime materializer concatenates the same text into a single prompt string.

**Frozen ScenePilot policy prompt used for final SFT materialization**

```
You are a generator-agnostic ScenePilot policy.
Make exactly one atomic decision per round.
STOP is the only terminal control.
Use stable object IDs and obey host_capabilities.
Canonical actions are translate, rotate, scale, delete, add;
return exactly one JSON object and no explanation.
SCENE_JSON=<canonical current scene JSON>
HOST_CAPABILITIES=<backend capability JSON>
```

### G.2 MULTIMODAL OBSERVATION CONTRACT

The final multimodal SFT record augments the text prompt above with two images bound to the same current state: an annotated top-down rendering and a diagonal rendering. Thus the policy observation is

$$o_t = (I_t, z_t, C_g),$$

where $I_t$ contains the two current-state views, $z_t$ is the canonical scene JSON, and $C_g$ is the backend capability descriptor. The training prompt does not expose future states, target-action hints, oracle history, generator identity, or diagnostic labels. At inference, the closed-loop runtime may retain execution history for bookkeeping and verification, but the canonical action contract itself is unchanged.

**Capability-conditioned multimodal input template**

```
Image 1: current annotated top-down view
Image 2: current diagonal view

You are a generator-agnostic ScenePilot policy.
Make exactly one atomic decision per round.
STOP is the only terminal control.
Use stable object IDs and obey host_capabilities.
Canonical actions are translate, rotate, scale, delete, add;
return exactly one JSON object and no explanation.

SCENE_JSON=<canonical current scene JSON>
HOST_CAPABILITIES={
  "translate": <true|false>,
  "rotate": <true|false>,
  "scale": <true|false>,
  "delete": <true|false>,
  "add": <true|false>,
  "STOP": true
}
```

### G.3 CANONICAL ATOMIC OUTPUT SCHEMA

Every policy round returns exactly one JSON object. The following schemas match the final canonical action interface; field names are part of the executable contract.

**Canonical repair decision schema**

```
# STOP
{"decision":"STOP"}

# translate
{"object":"<stable_object_id>",
 "operation":{"type":"translate",
              "delta_position_m":[dx,dy,dz]}}

# rotate
{"object":"<stable_object_id>",
 "operation":{"type":"rotate",
              "delta_yaw_deg":degrees}}

# scale
{"object":"<stable_object_id>",
 "operation":{"type":"scale",
              "scale_factor":[fx,fy,fz]}}
```

```
# delete
{"object":"<stable_object_id>",
 "operation":{"type":"delete"}}

# add
{"object":"<new_stable_object_id>",
 "operation":{"type":"add",
              "category":"<category>",
              "reference_object_ids":["<existing_id>"],
              "position_m":[x,y,z],
              "yaw_deg":degrees,
              "physical_size_m":[sx,sy,sz]}}
```

Existing edit targets and `add.reference_object_ids` must resolve to stable IDs visible in the current canonical state; `add.object` introduces the new stable ID. A backend may expose only a subset of the five edit operations. Unsupported proposals are rejected by VALIDATEANDPROJECT rather than silently converted into another action. Legacy `move` is normalized to `translate` only at the compatibility boundary.

### G.4 PROMPT VERSIONING AND EXCLUDED HISTORICAL PROMPTS

Earlier development artifacts contain prompts for retrieval-augmented relation planning, functional-group generation, diagnostic reasoning, multi-action repair, and repeated-scale constraints. These prompts correspond to superseded Grow-and-Repair or diagnostic pipelines and are not used to define the final ScenePilot policy. We therefore omit them from the reproducibility specification instead of mixing historical interfaces with the final checkpoint protocol.

### G.5 GRPO VLM REWARD JUDGE

During GRPO, we use GPT-5.4 as a separate VLM judge to provide the visual-semantic component $Q_{\mathrm{VLM}}$ of the realized-improvement reward. For each scene state, the judge receives rendered views of the same room, typically including a top view and a diagonal view. The top view primarily exposes global spatial arrangement, circulation, alignment, and object overlap, while the diagonal view provides complementary evidence about object scale, accessibility, and functional usability. Optional textual context, such as the room type or natural-language design request, may also be provided, but the judge is explicitly instructed to treat it only as auxiliary information and to base its primary assessment on the rendered observations.

The judge evaluates each scene independently along three dimensions: *layout correctness* ($lc$), which measures geometric and spatial validity, including collisions, boundary violations, spacing, circulation, alignment, orientation, and grouping; *semantic plausibility* ($spa$), which measures whether object relationships, scales, orientations, and functional groupings form a believable indoor arrangement; and *functional completeness* ($fc$), which measures whether the scene contains sufficient core furniture and functional structure to support its intended use. Each dimension receives a continuous score in $[0, 10]$ together with a short justification grounded in visible evidence. The scalar visual-semantic quality score is defined as the arithmetic mean of the three dimensions:

$$Q_{\mathrm{VLM}}(S) = \frac{s_{\mathrm{lc}}(S) + s_{\mathrm{spa}}(S) + s_{\mathrm{fc}}(S)}{3}, \tag{23}$$

which corresponds directly to the `overall` field returned by the judge. During GRPO, we use the realized improvement

$$\Delta Q_{\mathrm{VLM}} = Q_{\mathrm{VLM}}(\widetilde{S}_{t+1}) - Q_{\mathrm{VLM}}(S_t), \tag{24}$$

rather than the absolute scene score. Therefore, the VLM reward assigns credit to the visual-semantic consequence of the sampled intervention rather than to the pre-existing quality of the input scene. This reward-side judge is used only for GRPO policy optimization and is distinct from the independent VLM evaluator used for final test-time reporting.

For reproducibility, the complete frozen prompt used by the GRPO VLM reward judge is provided below.

Frozen VLM judge prompt used for GRPO reward

```
You are an expert reviewer for indoor 3D scene layout and furniture
    arrangement.

You will be given rendered images of the SAME room, typically
    including:
- a top view, which is most reliable for judging global spatial
    arrangement, circulation, alignment, and object overlap;
- a diagonal view, which is most reliable for judging realism,
    object scale, accessibility, and whether furniture placement
    looks usable in perspective.

You may also be given optional textual context, such as the room
    type or a natural-language design request. If textual context is
     provided, use it only as auxiliary information. The primary
    judgment must come from the rendered images.

Your task is to evaluate the scene on exactly three criteria:

1) lc: Layout correctness
2) spa: Semantic plausibility
3) fc: Functional completeness

General evaluation principles:
- Be generous .
- Judge only what is visible in the provided images and optional
    textual context.
- Do not assume hidden furniture, invisible functional areas, or
    unshown geometry.
- If the two views disagree, use both views together and penalize
    uncertainty or visible inconsistency.
- Penalize physical or spatial problems such as:
  - obvious object-object collisions or severe overlap,
  - furniture intersecting walls or being partly outside the room,
  - blocked circulation or inaccessible furniture,
  - awkward spacing, unnatural placement, or unusable arrangements,
  - severe misalignment between related furniture pieces,
  - implausible scale or proportion,
  - missing core furniture required for the room's apparent function
    .
- Prefer concise but informative reasons.
- Use the full score range from 7 to 10 when appropriate.
- Scores do not need to be integers; use floats.

Detailed scoring criteria:

A) lc: Layout correctness
Definition:
Evaluate the geometric and spatial quality of the arrangement itself
    , regardless of style preference.

Focus on:
- whether objects are placed in reasonable positions inside the room
     boundary;
- whether there are collisions, overlaps, wall intersections, or out
    -of-bound placements;
- whether there is clear and usable free space for movement;
- whether furniture alignment, orientation, spacing, and grouping
    are spatially coherent;
- whether the overall composition looks organized rather than
    chaotic or arbitrarily scattered.
```

```
High lc score (8-10):
- Objects are well placed, mostly collision-free, inside the room,
    and support clear circulation.
- Relative positions and orientations are coherent.
- Major furniture is arranged in a spatially sensible and usable way
    .

Medium lc score (6-7):
- Layout is partially reasonable but has noticeable spacing issues,
    weak alignment, mild obstruction, or questionable placement.

Low lc score (3-5):
- Serious collisions, blocked walkways, unusable access, severe
    boundary violations, or obviously broken placement.

B) spa: Semantic plausibility
Definition:
Evaluate whether the scene makes semantic sense as a believable room
     layout for its apparent room type and intended use.

Focus on:
- whether object relationships are semantically appropriate;
- whether furniture types and pairings make sense together;
- whether items are positioned in functionally meaningful relations
    (e.g., seating around a table, bedside furniture near a bed,
    desk and chair pairing, TV facing seating, storage placed
    sensibly);
- whether scales, orientations, and usage relationships look
    realistic;
- whether the room appears like a plausible human-designed interior
    rather than a random collection of objects.

High spa score (9-10):
- Furniture relationships are natural and believable.
- The room reads clearly as an intended functional space.
- Object placement supports typical human use patterns.

Medium spa score (7-8):
- Scene is somewhat believable but contains odd pairings, weak
    semantic grouping, or several unnatural relations.

Low spa score (5-6):
- Scene appears semantically confused, implausible, or obviously
    unrealistic for the room type.

C) fc: Functional completeness
Definition:
Evaluate whether the room contains enough of the key furniture and
    arrangement structure needed to support its intended function.

Focus on:
- whether essential furniture for the apparent room type is present;
- whether the scene supports the main activity of the room;
- whether the functional zones look complete rather than partial or
    under-specified;
- whether the room feels missing major items that would normally be
    necessary.

Examples:
- A bedroom should usually include a bed and enough supporting
    furniture to make the room feel usable.
- A living room should usually include core seating and a coherent
    social or media focus.
```

```
- A dining room should usually include a dining table and
    appropriate seating.
- A study/workspace should usually include a desk or work surface
    and seating.
These are examples only; judge based on what is visible and any
    provided textual context.

High fc score (8-10):
- The room includes the major furniture needed for its purpose and
    feels functionally usable and reasonably complete.

Medium fc score (5-7):
- Some core functionality is present, but important supporting
    furniture or functional structure is missing.

Low fc score (3-4):
- The room is missing essential furniture and does not adequately
    support its intended use.

Scoring instructions:
- Score each criterion independently on a continuous scale from 0.0
    to 10.0.
- Start from the visible evidence, not from a default high score.
- 8.0-10.0: strong quality with no major visible problems.
- 6.0-7.9: usable or plausible, with noticeable but non-critical
    problems.
- 3.0-5.9: major problems that substantially limit quality or use.
- 0.0-2.9: severely broken, unusable, or missing the essential room
    function.
- Apply these bands separately to lc, spa, and fc. A clean-looking
    render must not compensate for a missing function or an
    implausible arrangement.
- Do not infer problems or furniture that cannot be established from
     the images.
- Use decimal scores when the evidence supports a value between
    anchors.

Reason instructions:
- For each criterion, provide a short reason grounded in visible
    evidence.
- Mention the most important positive or negative factors only.
- Avoid long explanations, speculation, or restating the rubric.

Output instructions:
Return ONLY one JSON object with exactly this schema:
{
  "lc": {"score": 0.0, "reason": ""},
  "spa": {"score": 0.0, "reason": ""},
  "fc": {"score": 0.0, "reason": ""},
  "overall": 0.0
}

Additional output constraints:
- Do not output markdown.
- Do not output code fences.
- Do not output any text before or after the JSON object.
- The "overall" field must be the arithmetic average of lc.score,
    spa.score, and fc.score.
- Ensure the JSON is valid and directly parseable.
```

### G.6 VLM-BASED EVALUATION PROTOCOL

For VLM-based evaluation, each rendered scene is judged independently without access to the generator identity, paired counterpart, or generation history. The judge receives only the rendered image and the corresponding task specification. We evaluate three dimensions: *layout*, *relation*, and *completeness*. The exact prompt is shown below.

**Frozen VLM judge prompt used for final evaluation**

```
You are a blind judge of one rendered 3D scene. Evaluate only the
    attached
image and the task specification. Do not guess the generation method
     or
compare this scene with other scenes.

Score each requested metric independently with an INTEGER from 1 to
    10.
Higher is better. Use these bands consistently:

1-2: Severe failure; most relevant requirements are visibly unmet.
3-4: Poor; several major problems.
5-6: Mixed; the main intent is recognizable, but noticeable problems
     remain.
7-8: Good; most requirements are met, with limited problems.
9-10: Excellent; all or nearly all visible requirements are met.

Within each band, choose the lower score for more numerous or more
    serious
problems, and the higher score for fewer or less serious problems.

Metrics:
- layout: Physical and spatial plausibility, including collisions,
  room-boundary violations, spacing, and circulation.
- relation: Whether the explicitly specified positional and facing
  relationships are satisfied.
- completeness: Whether the requested object categories and
    quantities
  are present.

Do not score texture, lighting, or artistic style. Do not count an
    occluded
object as definitely missing. If the image provides insufficient
    evidence
for a metric, return null for its score and explain why.

For each scored metric, pass = (score >= its pass_threshold).
Return valid JSON only.
```

### G.7 ADDITIONAL QUALITATIVE RESULTS

Figures 6–8 provide additional qualitative comparisons across different scene generators. For each example, we show the native generator output together with the scene repaired by the same frozen ScenePilot policy. Across generators, ScenePilot performs localized edits to improve object placement, orientation, spacing, and prompt-specified spatial relations while preserving the underlying generated scene. The highlighted regions illustrate representative corrections to geometric conflicts and relation inconsistencies.

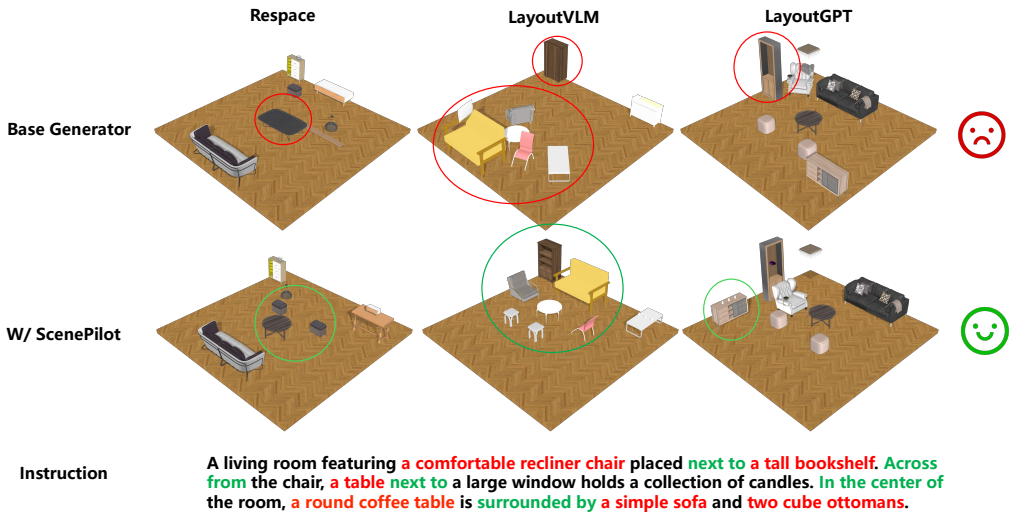


Figure 6: **Additional qualitative results on a living-room scene.** We compare the native outputs of ReSpace, LayoutVLM, and LayoutGPT with the corresponding scenes repaired by the same frozen ScenePilot policy. The highlighted regions show representative corrections to object placement and prompt-specified spatial relations.

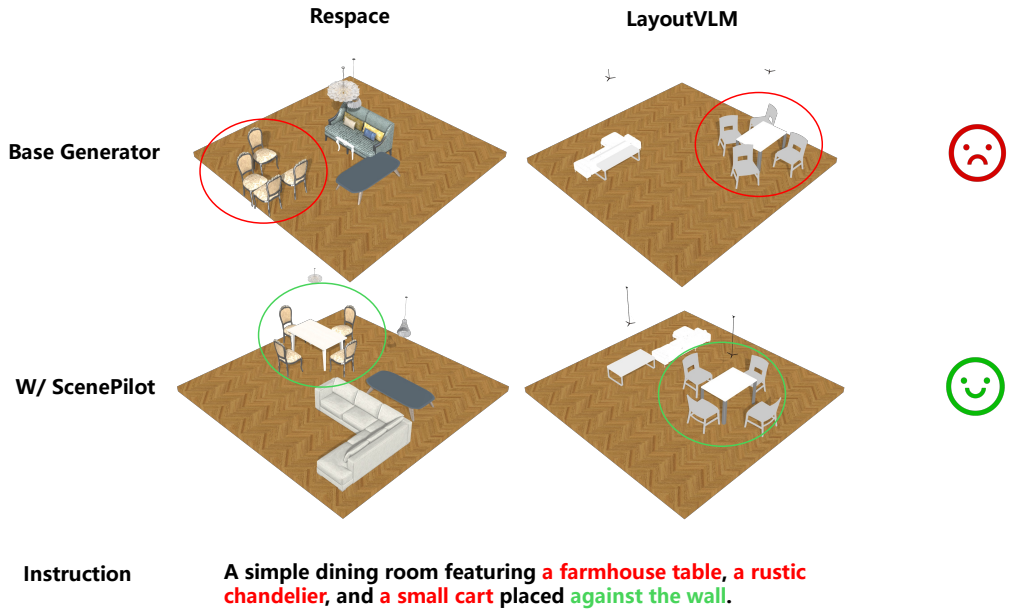


Figure 7: **Additional qualitative results on a dining-room scene.** We compare the native outputs of ReSpace and LayoutVLM with the corresponding scenes repaired by the same frozen ScenePilot policy. ScenePilot improves local object arrangement and spatial consistency through localized edits without regenerating the scene from scratch.

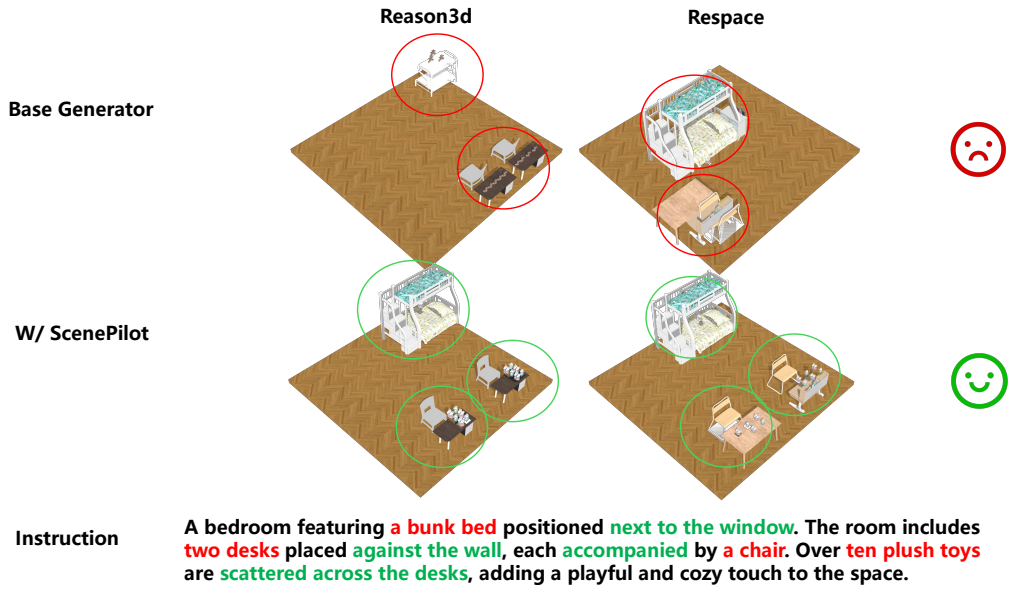


Figure 8: **Additional qualitative results on a bedroom scene.** We compare the native outputs of Reason3D and ReSpace with the corresponding scenes repaired by the same frozen ScenePilot policy. ScenePilot improves prompt-specified relations and object arrangement, including the placement of the bunk bed, desks, chairs, and plush toys, through localized scene edits.