FROM PLAUSIBLE SCENES TO EXECUTABLE WORLDS: SCRIPT-BASED 3D INDOOR SCENE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Script-based 3D indoor scene generation aims to construct 3D environments that faithfully realize the behaviors described in textual narratives. Existing scene generation methods, however, primarily focus on visual plausibility and semantic consistency, leaving an important question largely unexplored: *can the generated scene actually support the prescribed behaviors?* We argue that a behaviorally meaningful scene should constitute an *executable world*, where scripted actions are feasible not only individually but also as a coherent sequence. Based on this perspective, we identify three levels of executable requirements: *spatial executability*, which requires characters to reach the locations needed for consecutive actions; *functional executability*, which requires scene objects to geometrically and functionally support their intended interactions; and *sequential executability*, which requires successive actions to remain coherently executable over time. These requirements motivate us to move beyond one-shot scene generation toward an explicit generation-diagnosis-repair paradigm. To study it, we first construct FilmScene, a character behavior-centric dataset containing script-3D scene pairs, with scene-grounded interaction objects and action chains, together with real-world out-of-distribution test sets. Building upon this benchmark, we propose **Script-to-Scene (S2S)**, a closed-loop framework that first generates a plausible scene, explicitly diagnoses violations of executable constraints, and then performs targeted scene repair while preserving valid scene structure. By combining spatial reachability analysis, functional compatibility assessment, behavioral sequence verification, S2S progressively transforms plausible scenes into executable worlds via iterative local scene editing. Extensive experiments on FilmScene and OOD test sets demonstrate that S2S improves both overall scene quality and the ability of generated environments to support script-driven character behaviors.

1 INTRODUCTION

In film and theater production Rao et al. (2023); He et al. (2024), scenography, the art of designing stage and set environments, serves as the essential bridge between textual narrative and visual presentation. A well-crafted 3D indoor scene does more than define the space where a story unfolds: the arrangement, geometry, and functionality of objects determine what characters can actually do within the scene. This motivates Script-based 3D Indoor Scene Generation (SCR-SG), which aims to generate 3D environments that faithfully realize the behaviors prescribed by a textual script.

However, existing text-to-3D scene generation methods Fu et al. (2021a); Tang et al. (2023); Huang et al. (2023); Fang et al. (2023); Lin & Mu (2024); Yang et al. (2024) primarily focus on visual plausibility, semantic consistency, and spatial arrangement, without explicitly considering whether the generated scene can *execute* the behavior described in a script. This creates a fundamental gap between a *plausible scene* and an *executable world*: a scene may look reasonable as a static composition while failing once a character is required to interact with it. We therefore formulate SCR-SG from the perspective of *behavioral executability*, where a generated scene should not only look plausible, but also make the script-prescribed actions feasible in space, function, and time.

Specifically, we identify three levels of executability induced by the behavioral structure of a script: (1) ***Spatial Executability:*** Each action requires a character to reach an appropriate interaction location. Thus, the scene should provide feasible spatial transitions between the locations required by

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.

consecutive actions. For example, in "character retrieves the book from the shelf and sits down at the desk.", the character should be able to reach the bookshelf and subsequently move to the desk without being blocked by surrounding objects. (2) ***Functional Executability:*** Reaching an object does not guarantee that the intended action can be performed. Objects in a script serve as functional elements whose geometry and placement must support the corresponding actions. For example, in "character sits at the table and types on the computer," the table should provide an appropriate height, size, and spatial configuration for sitting and typing. (3) ***Sequential Executability:*** Scripted behavior is temporally ordered, and the arrangement of individually valid objects should also support their coherent interaction sequence. For example, a chair may have multiple reasonable placements around a table, but if the subsequent action is "stands up and opens the refrigerator," the action chain favors a placement that leaves the character naturally oriented and positioned toward the refrigerator. Thus, scene layout should account for the dependencies between successive actions.

These action-dependent requirements make SCR-SG more than producing a plausible object arrangement. A scene that is visually coherent may still contain localized violations of the spatial, functional, or sequential requirements imposed by the script. Since such violations are often confined to a local part of the scene, regenerating the entire scene can unnecessarily modify objects, spatial relationships, and visual semantics that are already valid. This observation motivates a *generate-diagnose-repair* paradigm: we first identify which executable constraint is violated, and then minimally modify the responsible part of the scene while preserving the remaining valid structure. We instantiate this paradigm as **Script-to-Scene (S2S)**, an agentic closed-loop framework consisting of a *Generator*, an *Executable Constraint Diagnoser*, and a *Minimal Repair* module. The central principle of S2S is *repair locally, preserve globally*: improving executable behavior while retaining the valid structure established by the initial generation. Given a script, the system first generates a plausible initial scene, then evaluates it against script-derived executable requirements to identify violations in spatial reachability, functional compatibility, and sequential behavior, together with basic physical validity. The resulting structured diagnosis serves as guidance to perform minimal modifications targeted at the detected violations, after which the repaired scene is re-evaluated to verify the behavioral constraints, forming the closed-loop generation process.

To support systematic study of this task, we construct **FilmScene**, a character behavior-centric dataset containing 11k paired scripts and high-quality 3D indoor scenes. FilmScene is characterized by three properties that are critical for executable scene generation but largely absent from existing 3D indoor scene datasets. *(1) Actionable beyond inhabitable.* Besides large furniture that affords static poses such as sitting and lying, each scene is enriched with diverse interactive objects that characters can actually manipulate, yielding a substrate for actions rather than a mere backdrop for occupancy. *(2) Deeply coupled and reasoning-demanding scripts.* Scripts are bound to the underlying scene through an explicit action chain: each action is anchored to a concrete interaction object and location, and consecutive actions are constrained by feasible spatial transitions, so that the script and the scene are aligned at the level of executable behavior. Recovering such grounded script-scene alignment requires reasoning over what can be done on which object, where and in what order, making the scripts substantially more demanding than conventional scene descriptions. *(3) Grounded in real needs.* Beyond large-scale in-distribution data, FilmScene includes out-of-distribution test sets derived from real-world application scenarios, covering game scenarios that demand complex sequential object interactions and long-horizon behavioral reasoning, and theatrical scenarios with distinct visual and aesthetic conditions, thereby enabling evaluation of generalization beyond the training scene distribution.

Our contributions are four-fold. (1) We formulate Script-based 3D Indoor Scene Generation (SCR-SG) task from the perspective of *executable worlds*, highlighting three script-induced requirements that go beyond conventional scene plausibility. (2) We construct FilmScene, a character behavior-centric dataset with script-3D scene pairs, providing scene-grounded action chains and interaction annotations, together with out-of-distribution test sets that cover complex interaction logic and distinct visual domains. (3) We propose Script-to-Scene (S2S), a closed-loop generation-diagnosis-repair framework that explicitly identifies violations of executable constraints and performs targeted, minimal scene modifications, thereby preserving valid scene structure while improving behavioral executability. (4) Experiments on FilmScene and the OOD test sets demonstrate that S2S improves the ability of generated scenes to support script-driven behaviors, validating the effectiveness of explicit executable constraint diagnosis and targeted repair.

2 RELATED WORK

2.1 TEXT-BASED 3D SCENE GENERATION

Early studies Chang et al. (2014; 2015); Ma et al. (2018); Wang et al. (2021); Paschalidou et al. (2021); Liu et al. (2023) rely on simple text descriptions that specify only object categories, offering limited control over the resulting scene. More recent methods Lin & Mu (2024); Sun et al. (2024) accept longer descriptions and typically predict a scene layout, which is then composed by retrieving pre-set 3D models Fu et al. (2021b). Other works Liu et al. (2025); Bai et al. (2025b); Deng et al. (2025); Li et al. (2025) further rely on extra visual conditions, such as scene images, to provide priors for generation. Despite this progress, their text remain focused on the scene itself while excluding narrative elements such as character conversation, and contain no information about human activities, thus omitting the executable demands. Our SCR-SG instead casts generation as constructing an *executable world* that must support the ordered behaviors prescribed by the script, shifting the objective from visual plausibility alone to three script-induced executability requirements.

2.2 LLM-BASED 3D SCENE GENERATION

Several works use advanced LLMs to generate 3D scenes, mainly by prompting the LLM to produce scene configurations from simple descriptions. LayoutGPT Feng et al. (2023) directly prompts an LLM to predict layouts in CSS format. HoloDeck Yang et al. (2024) introduces pre-defined rules as constraints to optimize the predicted scene. AnyHome Fu et al. (2024) and SceneWeaver Yang et al. (2025b) apply iterative refinement to adjust scenes. Though promising, these methods mainly consider spatial and physical rules, and overlook whether a scene is feasible for specific activities. Particularly, they conduct less consideration about functional executability, i.e., whether object attributes match the actional intent, nor sequential executability, i.e., whether the layout sustains coherent transitions along a temporally ordered action chain, and thus remains only *visually plausible*. Our method further incorporates requirement from spatial, functional, and sequential executability perspective to support its script filming usage.

3 DATASET: FILMSCENE

To support the study of script-based 3D executable scene generation, we construct **FilmScene**, a character behavior-centric dataset containing paired scripts and high-quality 3D indoor scenes. FilmScene is organized around three properties critical for executable scene generation: *actionable rather than merely inhabitable* environments, *deeply coupled and reasoning-demanding* scripts, and evaluation scenarios *grounded in real needs*. Our construction pipeline is shown in Figure.1.

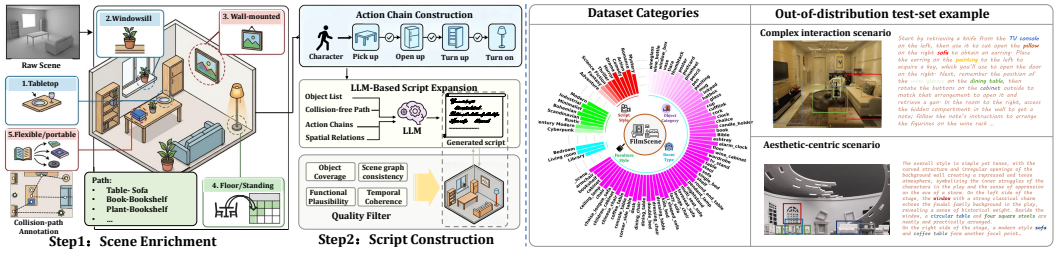


Figure 1: **Construction process and Information about FilmScene.** *Left:* FilmScene construction process. *Right:* Statistics of FilmScene and examples of out-of-distribution real world tests.

3.1 FROM INHABITABLE TO ACTIONABLE ENVIRONMENTS

Existing 3D indoor scene datasetsFu et al. (2021a) are visually complete but dominated by large furniture that affords only static behaviors like sitting or lying, offering little for character to manipulate. FilmScene instead treats each scene as an actionable substrate. Starting from a 3D indoor scenes, we enrich them with diverse interactive assets(derived from ObjaverseDeitke et al. (2023)), organized by supporting patterns ranging from tabletop, windowsill, wall-mounted, floor/standing to flexible/portable configurations, with placement and scale adjusted to their supporting structures. For

each interaction-relevant object, we annotate an interaction point where a character can perform the corresponding action, and use a top-view occupancy grid with A* search to ensure a collision-free paths exists between them, which provide the spatial substrate for constructing and later evaluating executability grounded action sequences.

3.2 Deeply Coupled and Reasoning-Demanding Scripts

To align scripts with scenes, FilmScene binds each script to its scene via an explicit action chain that aligns narrative, interaction objects, and spatial layout at the level of executable behavior. This chain is built from the enriched scene: from a designated entry point, we sample a sequence of interaction points whose consecutive pairs are connected by feasible collision-free paths, associating each point with its object and a plausible action determined by the object's category and supporting pattern. The chain thus guarantees that each action has a concrete object and location, consecutive actions are spatially connected, and objects are functionally compatible with their actions, with lengths varied from 4 to 15 to capture different behavioral complexity. The chain is then given to GPT-4oAchiam et al. (2023) as a behavioral skeleton to produce a fluent script, acting as an explicit constraint rather than a generation hint, preserving actions and their temporal order while keeping the narrative grounded in the scene. Finally, we randomly sample a subset of 200 generated scripts for manual inspection by rendering multi-view images of the scene, checking the plausibility of each action-object pair, the consistency between described movements and annotated paths, and the temporal coherence of consecutive actions, and turns out with 99.5% path rate, ensuring generated scenes are behaviorally grounded and executable.

3.3 Grounded in Real Needs: Statistics and Out-of-Distribution Evaluation

FilmScene ultimately contains 11k paired scripts and scenes, covering diverse object categories, script lengths, interaction types, and action-chain structures. Beyond the in-distribution data, we construct two out-of-distribution test sets derived from real world application scenarios. *Escape-The-Room* draws on real escape-room game scenarios and focuses on complex, interdependent object interactions and long-horizon action chains, such as "find key $\rightarrow$ unlock drawer $\rightarrow$ retrieve code," requiring reasoning over both functional dependencies and behavioral continuity. *Theatrical Stage Design* is adapted from StagePro-V1Gan et al. (2025) and introduces a distinct visual and aesthetic setting with audience-centric viewpoint constraints, where objects must support the prescribed actions while remaining appropriately visible from a fixed front-facing perspective. Together, these subsets evaluate whether executable scene generation can generalize beyond the training distribution, covering both increased interaction complexity and substantial visual-domain variation.

4 Method: Script-to-Scene (S2S)

4.1 Overview

The overall pipeline of S2S is illustrated in Fig. 2. Given a script T_{scr}, our goal is to generate a 3D indoor scene that not only appears visually plausible, but also serves as an *executable world* that supports the prescribed behavior, which must satisfies the script-induced constraints as follows:

$$S \models \mathcal{C}_{spa}(T_{scr}) \wedge \mathcal{C}_{fun}(T_{scr}) \wedge \mathcal{C}_{seq}(T_{scr}) \wedge \mathcal{C}_{phy} \tag{1}$$

where each $\mathcal{C}(\cdot) \in \{0, 1\}$ is a binary indicator of whether the corresponding requirement is satisfied: $\mathcal{C}_{spa}$ requires feasible movement between interaction locations, $\mathcal{C}_{fun}$ requires objects to support their intended actions, and $\mathcal{C}_{seq}$ requires successive interactions to remain coherently executable. We additionally enforce general physical validity $\mathcal{C}_{phy}$. To construct such an executable world, S2S adopts a *generate-diagnose-repair* paradigm that explicitly evaluates and refines the generated scene against the script-induced constraints. Specifically, the *Generator* $\mathcal{M}_G$ first constructs a plausible scene; the *Executable Constraint Diagnoser* then identifies violations from spatial, functional, sequential, and physical perspectives; and the *Minimal Repair* module modifies the scene elements relevant to the diagnosed violations. This design follows the principle of *repair locally, preserve globally*, avoiding unnecessary changes to scene elements that already satisfy the required constraints. The refined scene is subsequently re-validated, forming a closed loop that progressively transforms a plausible scene into an executable world.

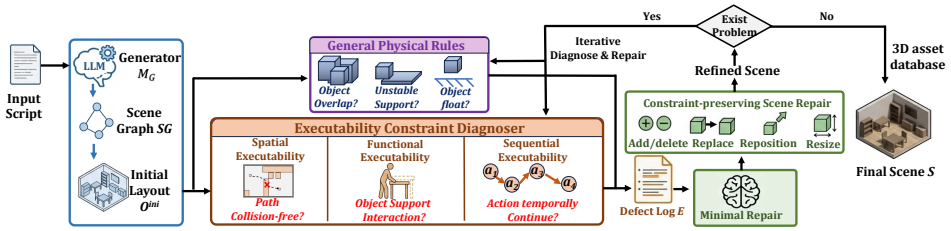


Figure 2: **Overall pipeline of S2S.** Given a script, the Generator constructs a plausible initial scene. The Executable Constraint Diagnoser evaluates it against script-induced executable constraints and physical validity, producing structured violation feedback. The Minimal Repair module then performs targeted scene editing, and the refined scene is re-validated in a closed loop.

4.2 Plausible Initial Scene Construction

We first employ an LLM-based generator $\mathcal{M}_G$ to translate the semantic information in T_{scr} into a plausible initial scene. Given the script, $\mathcal{M}_G$ produces a scene graph $SG = \{O_{SG}, R_{SG}\}$, where O_{SG} denotes the set of objects and R_{SG} denotes their spatial and semantic relations. Crucially, O_{SG} contains not only explicitly mentioned objects, but also objects implicitly required by the described behaviors. For example, "sit down" implies the existence of a suitable seat. This behavior-aware scene graph provides an intermediate representation that connects script semantics with subsequent geometric scene construction. Conditioned on SG, $\mathcal{M}_G$ predicts an initial scene $\mathcal{O}^{ini}$ in JSON format, specifying object categories and their continuous attributes $(\mathbf{c}_j, \mathbf{s}_j, \theta_j)$. We treat this stage as the construction of a *plausible initial world*, followed by subsequent executable constraint diagnosis and repair to transform it into executable world.

4.3 Executable Constraint Diagnosis

The initial scene $\mathcal{O}^{ini}$ is visually and semantically plausible starting point without executability assurrance. We therefore introduce an *Executable Constraint Diagnoser* $\mathcal{D}$ to identify violations induced by the script. Specifically, we consider three script-induced executable constraints below.

Spatial Executability. A scripted action requires the character to reach the corresponding interaction location. Therefore, we evaluate reachability conditioned on the ordered interaction sequence extracted from the script.We first extract an ordered sequence of *action-object pairs* from T_{scr}, $\mathcal{P} = \{(a_1, o_1), (a_2, o_2), \ldots, (a_K, o_K)\}$, where each pair associates an action a_k with the interaction object o_k that the action operates on, and k denotes its the temporal order. Then acquire the locations p_i (corresponds to (a_1, o_1)) for theses objects from initial scene $\mathcal{O}^{ini}$. For each pair of consecutive interaction locations implied by the action chain, we construct a 2D occupancy grid and perform A* search to determine whether a collision-free path exists. Formally, for interaction locations p_i and p_{i+1}, we evaluate

$$\mathcal{R}(p_i, p_{i+1} \mid \mathcal{O}^{ini}) = \begin{cases} 1, & \text{if a collision-free path exists,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

When $\mathcal{R} = 0$, the corresponding transition is recorded as a spatial executability violation. The diagnosis identifies the disconnected interaction objects and provides correction directions such as repositioning or resizing relevant objects to restore a feasible path.

Functional Executability. Spatial reachability alone does not guarantee that an intended action can actually be performed. Each action-object pair therefore imposes functional requirements on the geometry and configuration of the target object. For example, an object intended for sitting should have an appropriate height. For each action-object pair, we employ an affordance classifier $\mathrm{CLS}(a_i, o_i, s_i) \in [0, 1]$ trained on FilmScene, which predicts the compatibility between an action a_i, an object category o_i, and its geometric attributes s_i. The classifier captures functional criteria such as appropriate surface height, orientation and other action-dependent geometric requirements. When the predicted affordance score falls below a threshold τ_{aff}, we record a functional executability violation. The resulting diagnosis indicates whether the target object should be resized, repositioned, or replaced with a more suitable functional alternative.

Sequential Executability. The above two constraints evaluate whether individual transitions and interactions are feasible, but a script specifies an ordered chain of behaviors whose actions must remain

mutually compatible. A scene can therefore satisfy individual spatial and functional requirements while still producing an infeasible action sequence. To evaluate such sequence-level requirements, we render the generated scene from 16 camera viewpoints to expose the spatial context of the interaction objects. These views are provided to a VLM-based diagnoser Bai et al. (2025a), which evaluates the behavioral chain from two complementary aspects: (a) whether the interaction region of a target object, such as a cabinet door or refrigerator handle, is obstructed by neighboring objects; and (b) whether consecutive interaction objects are spatially separated to an extent that makes the prescribed transition impractical given the narrative context. Detected violations are associated with the corresponding action transitions and scene objects, producing sequence-level executable feedback for subsequent repair.

Beyond script-induced executable constraints, we also enforce general physical rules, ensuring the generated scene remains structurally plausible. Specifically, we check object non-overlap and non-anti-gravity constraints, including whether an object's center of mass projects within its supporting footprint and whether unsupported objects exhibit an excessive vertical gap from their supporting surfaces. All detected violations are converted into a structured defect representation $E = \{e_i\}_{i=1}^{M}$, where each e_i records the violated constraint type, the relevant action or action transition, the involved objects, and the observed deficiency. This representation serves as the interface between numerical/visual validation and semantic scene repair.

4.4 Minimal Constraint-Preserving Scene Repair

Given the current scene $\mathcal{O}^{ini}$, script T_{scr}, and the diagnosed violations E, the Repair Module aims to produce a refined layout that eliminates these violations while preserving scene elements that are already valid. This objective is instantiated as a constrained minimization over modification,

$$\mathcal{O}^* = \arg\min_{\mathcal{O}'} D(\mathcal{O}', \mathcal{O}^{ini}) \quad \text{s.t.} \quad \mathcal{O}' \models \mathcal{C}_{spa} \wedge \mathcal{C}_{fun} \wedge \mathcal{C}_{seq} \wedge \mathcal{C}_{phy}, \tag{3}$$

where $D(\mathcal{O}', \mathcal{O}^{ini})$ denotes the deviation of the refined scene from the current one, which follows the *repair locally, preserve globally* principle with two rationale: First, the initial scene already captures most required objects and their coarse spatial relations, with violations caused only by a few local decisions; regenerating the whole scene would discard this largely valid structure and risk introducing new inconsistencies elsewhere. Second, the diagnosed violations are local and constraint-specific, precisely identifying the objects and relations responsible for the failure, so targeted modification is both sufficient and more reliable than unconstrained regeneration. The Repair Module therefore conditions on the defects in E and edits only the objects implicated by the detected violations, satisfying all executable constraints while retaining already valid configurations.

To realize this objective, the Repair Module performs semantic-level editing operations guided by the defect log, each targeting a specific class of violation. Objects may be repositioned to restore reachability between consecutive interaction locations, resized to satisfy functional requirements such as appropriate surface height or accessible area, or replaced by a more suitable category when the existing object cannot support the intended action. Objects may also be added or removed when the action chain requires interaction targets that are missing or extraneous. All operations are conditioned on the script and the surrounding scene context, so that resolving a local violation does not disturb already valid configurations. In practice, Eq. 3 is realized through these operations rather than solved explicitly, with the defect log providing the constraint-level signals that determine which objects to edit and how. After correction, the refined layout $\mathcal{O}^*$ is returned to the Executable Constraint Diagnoser for re-validation. If violations persist, the newly diagnosed feedback is provided to the Repair Module for another repair step. This closed-loop process continues until the scene satisfies required constraints or reaches the maximum number of refinement iterations.

5 Experiments

5.1 Experimental Setup

Implementation Details. For the Generator, we adopt Qwen3-8B Yang et al. (2025a) and fine-tune it on the FilmScene training set using LoRA Hu et al. (2021). For executable constraint diagnosis, we employ Qwen3-VL-8B-Instruct Bai et al. (2025a) to inspect scene-level behavioral violations, while a lightweight MLP is trained on FilmScene as the affordance classifier for functional executability assessment. More implementation details are provided in the appendix.

Table 1: **Results on FilmScene and OOD testset. Bold** for the best, underline for second-best.

Method	FilmScene Test Set								Real-scene Scenario Test Set					
	Semantic		Physical		Executability				Physical		Executability			
	FID↓	*CLIP-Sim*↓	*CR*↓	*UOR*↓	*RC*↑	*OC*↑	*FS*↑	*SC*↑	*CR*↓	*UOR*↓	*RC*↑	*OC*↑	*FS*↑	*SC*↑
LayoutGPTFeng et al. (2023)	167.55	20.19	22.93	58.41	73.65	94.23	67.23	54.11	25.7	34.1	79.94	73.06	58.22	60.03
HoloDeckYang et al. (2024)	140.54	21.77	4.65	2.88	99.93	98.82	77.11	73.38	10.8	12.88	92.06	95.7	69.04	65.12
LayoutVLMSun et al. (2024)	143.55	21.56	2.88	0.93	97.84	96.54	85.50	82.71	2.15	1.04	94.56	94.42	73.05	68.03
AnyHomeFu et al. (2024)	133.66	22.92	7.85	10.66	98.12	97.23	88.04	73.82	4.87	1.77	92.83	96.87	72.11	64.76
StageDesignerGan et al. (2025)	130.65	22.83	3.68	1.77	99.87	**100**	89.18	80.76	3.96	1.56	93.65	95.11	83.42	77.05
SceneWeaverYang et al. (2025b)	133.72	22.07	**2.05**	0.66	98.73	**100**	87.54	81.00	**1.87**	0.66	97.03	97.12	82.16	77.64
S2S (Ours) (*zero-shot*)	129.88	22.53	3.08	0.83	98.66	99.99	89.72	82.45	2.91	0.53	97.55	96.02	83.18	79.05
S2S (Ours) (*fine-tuned*)	**123.04**	**23.01**	2.48	**0.42**	99.98	**100**	**91.08**	**84.75**	2.06	**0.28**	**99.24**	**98.15**	**85.07**	**82.99**

Compared Methods. We compare with open-sourced text-to-3D indoor scene generation methods, including LayoutGPT Feng et al. (2024), LayoutVLM Sun et al. (2024), AnyHome Fu et al. (2024), HoloDeck Yang et al. (2024), StageDesigner Gan et al. (2025), SceneWeaver Yang et al. (2025b).

Evaluation Metrics. We evaluate generated scenes from three complementary perspectives: ***(1)Semantic Fidelity.*** *FID*: measures the distribution distance between generated and ground-truth scenes by rendering top-down views; *CLIP-Sim*: measures the semantic similarity between the input script and the rendered generated scene. ***(2)Physical Validity.*** *Collision Rate (CR)*: the ratio of object overlap occurrences; *Unstable Object Rate (UOR)*: the ratio of floating or otherwise unstably supported objects. ***(3)Executability.*** *Room Connectivity (RC)*: measures whether the scene provides connected paths between required interaction locations using A* search. *Object Completeness (OC)*: evaluates whether the scene contains the objects necessary for the scripted activities. *Functional Support (FS)*: uses VLM Lu et al. (2024) to assess whether the generated objects can functionally support the intended actions, with scores ranging from 0 to 100. *Script Compliance (SC)*: assess whether the generated scene can smoothly support the character actions specified by the complete script.

5.2 Comparison with State-of-the-Art

Quantitative Comparison. Table 1 compares S2S with existing methods on both the FilmScene test set and the real-scene scenario test set, reporting a *zero-shot* variant that performs executable diagnosis and repair without fine-tuning the Generator, and a *fine-tuned* variant that additionally fine-tunes the Generator on FilmScene. The results support our formulation that generating a plausible scene and making it executable are complementary objectives, and explicit executable diagnosis bridges the two from following observations: (1) **Executability is achieved beyond visual plausibility.** S2S attains consistently strong executability-related performance, the fine-tuned version obtains the best Functional Support and Script Compliance; even the zero-shot version remains competitive, indicating that the gains stem not only from learning the data distribution but also from explicitly diagnosing and repairing executablility violations, while the further improvement from zero-shot to fine-tuned S2S shows that a better initial plausible scene provides a stronger starting point for refinement. (2) **Executability does not compromise scene quality.** The fine-tuned S2S obtains the best FID and CLIP-Sim among the compared methods while maintaining low collision and unstable-object rates, confirming that we improve executability by minimal repair while preserving the visual and geometric properties of the generated scene. (3) **S2S generalizes to unseen scenarios.** On the real-scene scenario test set, S2S remains strong across both physical and executability metrics with the best RC, OC, FS, and SC performance, demonstrating that the executable-world formulation is not restricted to the FilmScene distribution and transfers to scenarios with different script requirements, showing generalization ability and practicality.

Quantitative Comparison. Fig. 3 provides qualitative comparisons. The examples illustrate the gap between a scene that is visually plausible and one that is executable under the given script. (1) Existing methods tend to arrange objects according to generic commonsense rather than the specific behavioral requirements of the script. For example, in Line 1, Column 6, the compared method places the vase on the nightstand instead of on the cabinet as required by the script. In another example, nightstands are placed beside the head of the bed (Line 1, Columns 1–4), even though the script requires the character to step on them to reach the vase. These layouts are individually plausible, but fail to realize the intended action sequence. (2) Examples expose failures at different levels of executability. Some baselines place an incorrect object sub-category (Line 3, Column 3, an office chair in a living room) or orient an object incorrectly (Line 3, Column 5), resulting in functional incompatibility. Other scenes contain blocked internal paths (Line 2, Column 2), preventing the character from reaching required interaction locations. Physical violations such as floating or penetrating ob-



Figure 3: **Visualized Comparison.** For scripts, text denotes key behavioral clues, and text denotes spatial relationship hints. For generated scenes, boxes denote physical errors (e.g., floating and penetration), boxes denote missing objects that break the behavior chain, boxes denote blocked paths, and boxes denote functional incompatibility.

jects are also observed (Line 2, Columns 1-2). In contrast, S2S explicitly diagnoses these different constraint violations and performs targeted modifications, producing scenes that better support the prescribed behaviors while retaining plausible scene structure.

OOD Scenario Visualization. Fig. 4 visualizes results on the two real-scene scenario test sets. For the complex interaction game scenarios, S2S explicitly checks path connectivity and object accessibility, helping maintain realizable transitions across sequential interactions. For the audience-centric theatrical scenes, the multi-view VLM-based diagnosis evaluates object accessibility and contextual arrangement from different viewpoints, allowing S2S to maintain key scene structures under a distinct visual domain. These examples qualitatively demonstrate that the diagnosis-and-repair formulation can be transferred beyond the training distribution.

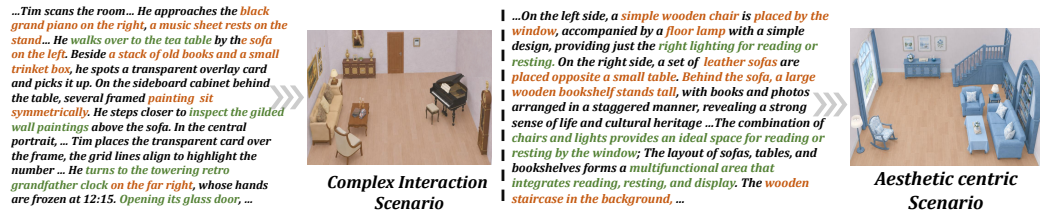


Figure 4: **Generalization to OOD scenarios** on *complex interaction game* and *aesthetic-centric theater* settings. (Assets are generated by TRELLIS Xiang et al. (2024) for better visual quality.)

5.3 Ablation Study

Ablation on the Generate–Diagnose–Repair Pipeline.

We analyze the contribution of the components in our framework. As shown in Table 2, (1) Removing Generator fine-tuning(*w/o* Generator FT) decreases CLIP-Sim, indicating that domain-specific training improves the quality of the initial plausible scene, providing a stronger starting point for executable diagnosis and repair. (2) Removing the Diagnoser (*w/o* Diagnoser), i.e., the generated scene is directly passed to the repair module without structured diagnosed report, shows a degradation across all major metrics. It proves that scene repair benefits from explicit and interpretable diagnosis that identifies which executable constraints are violated and where the violations occur. (3) Removing the iterative repair process (*w/o* Iterative Repair) decreases all metrics as well, proves that iterative re-validation and repair can further resolve residual violations.

We further compare several alternative strategies for scene repair. The *w/ Preference Learning* variant constructs flawed scenes paired with corresponding correct scenes as training samples, explicitly guiding the repair module to learn how to exert repair. The *w/ Score-based RL* variant replaces structured defect log with scalar scores and uses them as rewards for reinforcement learning. The

Table 2: **Ablation on main components.**

Method	*CLIP-Sim* ↑	*CR* ↓	*RC* ↑	*FS* ↑	*SC* ↑
Pipeline Ablation					
w/o Generator FT	22.53	3.08	98.66	89.72	82.45
w/o Diagnoser	21.87	7.24	96.45	80.52	72.16
w/o Iterative Repair	23.12	2.79	99.31	90.12	83.58
Ours (Full)	**23.01**	**2.48**	**99.98**	**91.08**	84.75
Alternative Repair Strategies					
w/ Full Regeneration	21.92	7.05	96.58	80.74	72.39
w/ Preference Learning	22.07	4.88	99.02	89.15	**85.02**
w/ Score-based RL	22.31	3.25	99.21	89.63	83.27

Table 3: **Executable constraint analysis.**

Diagnosis Dimension	*CLIP-Sim* ↑	*CR* ↓	*RC* ↑	*FS* ↑	*SC* ↑
w/o Spatial Exec.	22.31	3.42	97.12	86.15	80.24
w/o Functional Exec.	22.08	2.96	99.56	82.47	77.63
w/o Sequential Exec.	22.24	3.18	98.85	84.92	78.41
w/o Physical Valid.	22.47	6.85	98.41	87.36	81.05
Full	**23.01**	**2.48**	**99.98**	**91.08**	**84.75**

w/ Full Regeneration variant discards the current layout and regenerates the entire scene whenever violation is detected. As shown in Table 2, Preference Learning and Score-based RL alternatives remain below our strategy on most executable metrics, suggesting that explicit constraint-level diagnosis combined with targeted repair provides a more direct signal for scene correction than learning from paired corrections or scalar rewards alone. Full regeneration discards the largely correct initial structure, yielding a layout that is again merely plausible rather than executable, with no mechanism to enforce the script-induced constraints beyond initial generation, thus shows sub-optimal results compared to our minimal contraint-preserving repair.

Executable Constraint Analysis. We further analyze individual constraints effect in Executable Constraint Diagnoser. As shown in Table 3, we separately remove spatial executability, functional executability, sequential executability, and physical validity checks. We observe (1) Removing spatial executability check (*w/o Spatial Exec.*) mainly decreases RC, reflects that without explicitly checking paths between required interaction locations, the repair process can leave blocked transitions unresolved. (2) Removing the functional executability check (*w/o Functional Exec.*) produces the largest degradation in FS. This confirms that spatially reachable objects are not necessarily functionally usable, and that affordance-oriented diagnosis is necessary to ensure that object geometry and configuration support their intended actions. (3) Removing the sequential executability check (*w/o Sequential Exec.*) reduces SC. This highlights the distinction between satisfying individual actions and satisfying the complete action chain: without sequence-level diagnosis, problematic transitions and unfavorable arrangements between successive interactions can remain unresolved. (4) Removing the general physical rules (*w/o Physical Valid.*) also witness corresponding physical metrics drop, demonstrateing that physical validity serves as a necessary constraint for generation.

5.4 Human Evaluation

We conduct a human evaluation with 10 participants, asking them rate 60 randomly sampled scenes from FilmScene. Following a 5-point scale(1 the poorest, 5 the best), they judge results from three aspects: *Visual Plausibility*: whether the rendered scene looks reasonable. *Functional/Narrative Affordance*: whether the objects can support the actions, and whether the scene can smoothly support the sequential actions in the script. *Physical/Geometric Rationality*: whether the scene layout is physically plausible and free of overlaps or floating objects. Figure 5 reports results. Our S2S outperforms baselines on all four functional affordance and visual quality, and show second-best layout ratinality. Particularly, the improvement on Functional/Narrative Affordance is most significant, confirming that explicit executable constraint diagnosis provides information that conventional scene generation objectives do not directly capture.

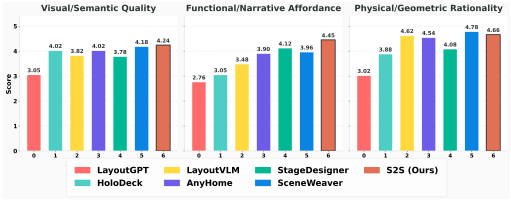


Figure 5: **Human Evaluation Results.**

6 Conclusion

We presented **SCR-SG**, a task that shifts 3D indoor scene generation from visual plausibility toward executable worlds that support script-prescribed behaviors. To facilitate this task, we introduced **FilmScene**, a excecutability-centric dataset with script-scene pairs with interactive objects, grounded action chains, and challenging real-world out-of-distribution scenarios. We further proposed **S2S**, a closed-loop *generate-diagnose-repair* framework that explicitly diagnoses spatial, functional, and sequential executability constraints and performs targeted minimal repair. Experiments demonstrate that S2S improve executability, encouraging future research on 3D scene generation that goes beyond plausible appearance toward worlds that are genuinely usable.

AI USE STATEMENT

In this work, we used generative AI tools to assist with paper formatting, such as table formatting and typesetting. We have not used generative AI tools for experimental design, result analysis, or technical writing. We have reviewed all AI-assisted work and take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies script-based 3D indoor scene generation task that carries significantly broader impact potential from both positive and negative sides. By generating 3D scenes, this work contributes to diverse applications like film scripting. However, it also raises ethical concerns related to privacy and misinformation, which must be carefully con sidered and addressed to ensure responsible deployment and minimize potential negative consequences.

REPRODUCIBILITY STATEMENT

For reproducibility, we provide the details needed to reproduce our results in the appendix. Dataset details, including object and relation category lists, distribution statistics, and data examples, are given in Sec. A, and the full prompt templates used for initial scene construction, executable violation detection, and minimal repair are provided in Sec. D. Method implementation details are given in Sec. F, including general implementation details in Sec. F.1, the affordance classifier in Sec. F.2, the preference learning and score-based reinforcement learning variants of the minimal repair in Sec. F.3 and Sec. F.4, and the iterative refinement procedure in Sec. F.5. Visualization of intermediate results is provided in Sec. G, and the efficiency analysis is reported in Sec. H. We will release our dataset and code publicly.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Tongyuan Bai, Wangyuanfan Bai, Dong Chen, Tieru Wu, Manyi Li, and Rui Ma. Freescene: Mixed graph diffusion for 3d scene synthesis from free prompts, 2025b. URL https://arxiv.org/abs/2506.02781.

Angel Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2028–2038, 2014.

Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. Text to 3d scene generation with rich lexical grounding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 53–62, 2015.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Wei Deng, Mengshi Qi, and Huadong Ma. Global-local tree search in vlms for 3d indoor scene generation, 2025. URL https://arxiv.org/abs/2503.18476.

Chuan Fang, Xiaotao Hu, Kunming Luo, and Ping Tan. Ctrl-room: Controllable text-to-3d room meshes generation with layout constraints. *arXiv preprint arXiv:2310.03602*, 2023.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Reddy Akula, Xuehai He, S Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional visual planning and generation with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun 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*, pp. 10933–10942, 2021a.

Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, 129: 3313–3337, 2021b.

Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. Anyhome: Open-vocabulary generation of structured and textured 3d homes, 2024. URL https://arxiv.org/abs/2312.06644.

Zhaoxing Gan, Mengtian Li, Ruhua Chen, Zhongxia Ji, Sichen Guo, Huanling Hu, Guangnan Ye, and Zuo Hu. Stagedesigner: Artistic stage generation for scenography via theater scripts, 2025. URL https://arxiv.org/abs/2503.02595.

Liu He, Yizhi Song, Hejun Huang, Daniel Aliaga, and Xin Zhou. Kubrick: Multimodal agent collaborations for synthetic video generation, 2024. URL https://arxiv.org/abs/2408.10453.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16750–16761, 2023.

Zehuan Huang, Yuan-Chen Guo, Xingqiao An, Yunhan Yang, Yangguang Li, Zi-Xin Zou, Ding Liang, Xihui Liu, Yan-Pei Cao, and Lu Sheng. Midi: Multi-instance diffusion for single image to 3d scene generation, 2025. URL https://arxiv.org/abs/2412.03558.

Haoran Li, Yuli Tian, Kun Lan, Yong Liao, Lin Wang, Pan Hui, and Peng Yuan Zhou. Dreamscene: 3d gaussian-based end-to-end text-to-3d scene generation. *arXiv preprint arXiv:2507.13985*, 2025.

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

Jingyu Liu, Wenhan Xiong, Ian Jones, Yixin Nie, Anchit Gupta, and Barlas Oğuz. Clip-layout: Style-consistent indoor scene synthesis with semantic furniture embedding. *arXiv preprint arXiv:2303.03565*, 2023.

Xinhang Liu, Yu-Wing Tai, and Chi-Keung Tang. Agentic 3d scene generation with spatially contextualized vlms, 2025. URL https://arxiv.org/abs/2505.20129.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. Deepseek-vl: Towards real-world vision-language understanding, 2024.

Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. Language-driven synthesis of 3d scenes from scene databases. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

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.

Anyi Rao, Xuekun Jiang, Yuwei Guo, Linning Xu, Lei Yang, Libiao Jin, Dahua Lin, and Bo Dai. Dynamic storyboard generation in an engine-based virtual environment for video production, 2023. URL https://arxiv.org/abs/2301.12688.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

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, 2024. URL https://arxiv.org/abs/2412.02193.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis. *arXiv preprint arXiv:2303.14207*, 2023.

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

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent, 2025b. URL https://arxiv.org/abs/2509.20414.

Yixuan Yang, Zhen Luo, Wanshui Gan, Jinkun Hao, Junru Lu, Jinghao Yan, Zhaoyang Lyu, and Xudong Xu. Code-as-room: Generating 3d rooms from top-down view images via agentic code synthesis, 2026. URL https://arxiv.org/abs/2605.18451.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2024)*, volume 30, pp. 20–25. IEEE/CVF, 2024.

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

APPENDIX FOR FROM PLAUSIBLE SCENES TO EXECUTABLE WORLDS: SCRIPT-BASED 3D INDOOR SCENE GENERATION

This appendix provides extra information organized as follows.

- Dataset details, including object and relation category lists, distribution statistics, script-scene examples, and out-of-distribution examples. (in Sec. A)
- Prompt templates for initial scene construction, executable violation detection, and minimal repair. (in Sec. D)
- Extended related work on image-to-3D indoor scene generation and its relation to the SCR-SG setting. (in Sec. E)
- Method details, including training configuration, affordance classifier, minimal repair variants, and iterative refinement. (in Sec. F)
  - General Implementation Details. (in Sec.F.1)
  - Affordance classifier, including input representation, architecture, and training data. (in Sec. F.2)
  - Minimal repair preference learning, including the corruption operator, training instances, and supervised fine-tuning. (in Sec. F.3)
  - Score-based reinforcement learning for minimal repair, including reward definition and GRPO training. (in Sec. F.4)
  - Iterative refinement details, including the alternating detection-repair loop and its algorithm. (in Sec. F.5)
- Visualization of intermediate results in each stage, including the initial scene, defect log, and refined scene. (in Sec. G)
- Efficiency analysis, including generation speed and quality comparison against baselines. (in Sec. H)

A DATASET DETAILS

A.1 OBJECT CATEGORY LIST

FilmScene contains 71 object categories, comprising 35 large furniture categories and 36 fine-grained small objects:

Large Furniture (35):

armchair, bookshelf, cabinet, ceiling lamp, chair, chaise longue sofa, children cabinet, chinese chair, coffee table, console table, corner side table, desk, dining chair, dining table, double bed, dressing chair, dressing table, kids bed, L-shaped sofa, lazy sofa, lounge chair, loveseat sofa, multi-seat sofa, nightstand, pendant lamp, round end table, shelf, single bed, sofa, stool, table, tv stand, wardrobe, wine cabinet

Small Objects (36):

Bible, alarm clock, ashtray, beer bottle, book, candle holder, cd, chalice, clock, cork, cufflink, cup, drink container, hand glass, hand towel, hatbox, inkpad, laptop, mug, oven, painting, pen, pencil, pinwheel, postcard, poster, table lamp, teddy, thumbtack, tiara, towel, trash can, vase, window box for plants, wine bottle, wineglass

A.2 RELATION CATEGORY LIST

We annotate **15 spatial relation types** between objects:

left, right, front, behind, close by, above, standing on, bigger than, smaller than, taller than, shorter than, symmetrical to, same style as, same super category as, same material as

B DISTRIBUTION STATISTICS

Figure 6 shows the distribution of object categories across FilmScene. The dataset provides both large furniture and small interactable objects, mirroring the natural pattern of real indoor scenes while ensuring that rare interaction targets remain represented for behavioral reasoning. Figure 7 presents the distribution of script lengths. Scripts range from 0–500 words, with a peak around 150–200 words. This distribution reflects our design choice to balance narrative richness against annotation reliability: scripts long enough to contain multi-step action chains, yet short enough to maintain precise scene-script alignment. Figure 8 shows the distribution of action chain lengths, i.e., the number of consecutive interaction points in each script. The majority of scripts contain chains of length 4–7, corresponding to moderately complex behavioral sequences. This distribution ensures that the dataset covers a wide spectrum of behavioral complexity, from simple single-object interactions to extended multi-object action chains, providing comprehensive coverage for training models to reason about sequential executability.

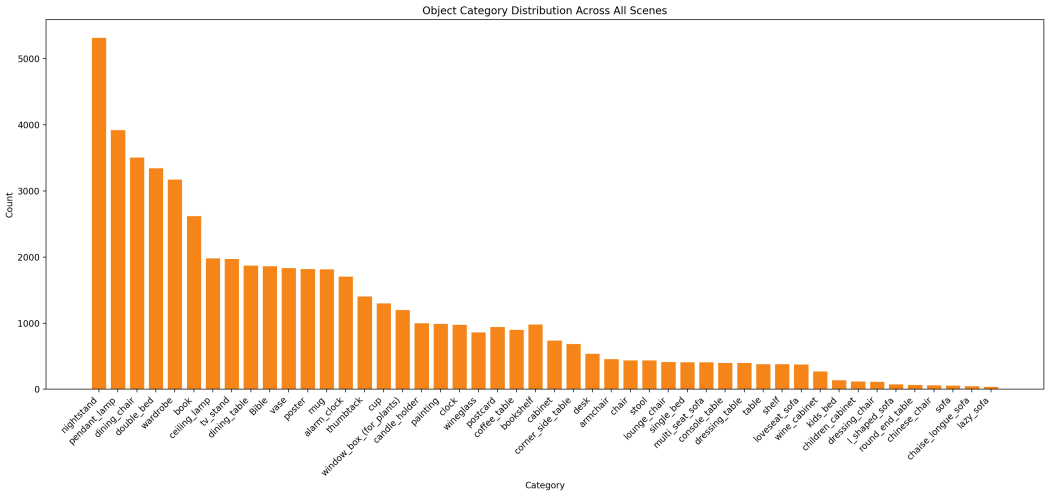


Figure 6: Object category distribution across FilmScene.

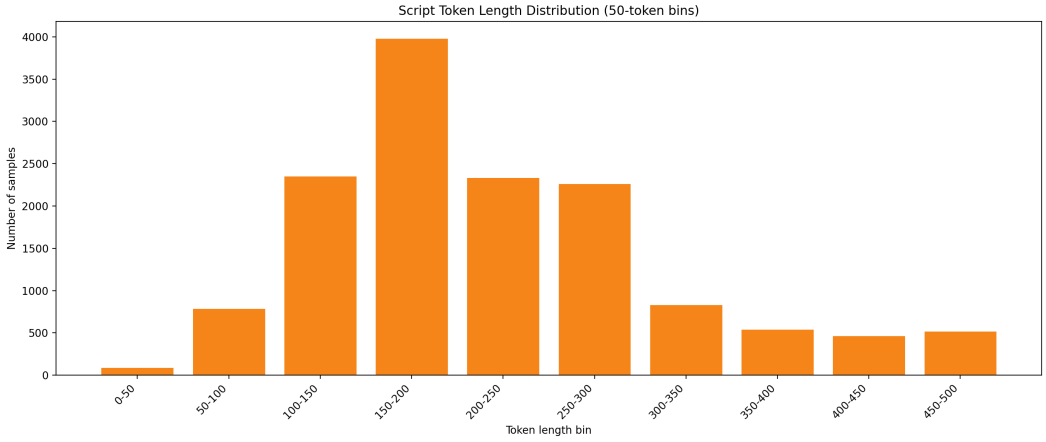


Figure 7: Script Length distribution across FilmScene.

B.1 SCRIPT-SCENE EXAMPLES

We provide data samples in FilmScene in Figure 9 and Figure 10, which include the paired script and scene, the action chain that the character executes within the script, and the navigation map between interactive points from a top-down view.

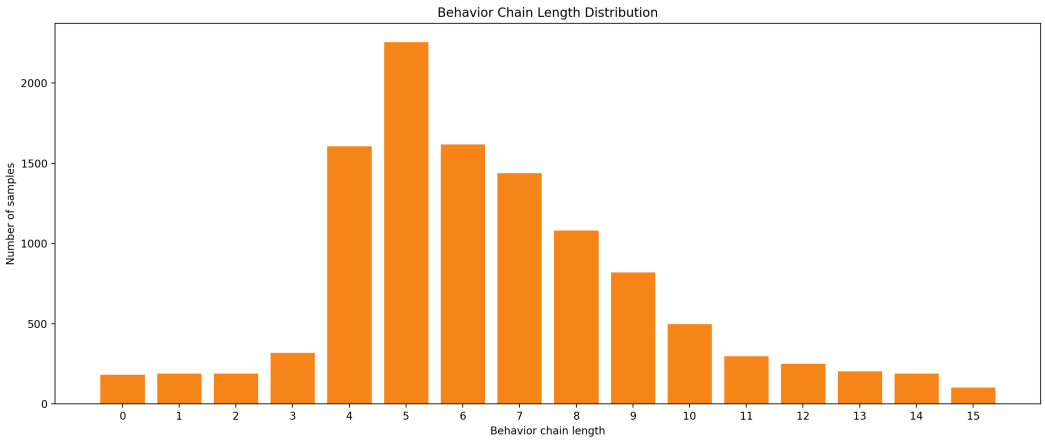


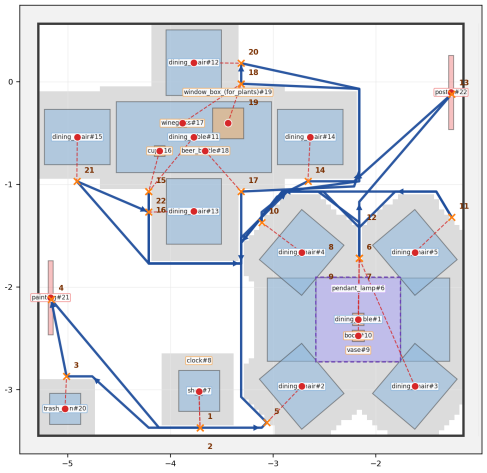
Figure 8: Behavior Chain Length distribution across FilmScene.

C Out-of-distribution Real-world Examples

We provide out-of-distribution data samples in Figure 11 and Figure 12, respectively comprising samples within interaction-intensive game scenarios and aesthetic-centric theater scenarios.

The dining room keeps one dim light burning. Above the first dining table the pendant lamp stutters, and the shadows it throws along the walls do not quite match the furniture. The character crosses to the shelf in the far corner, runs a hand along its edge, and lifts the clock down from it. The hands have stopped at midnight, and the ticking they remember is only in their own head. In the corner they lift the trashcan, shake it once, and set it back down. On the wall the painting holds a dark room painted inside a darker room, and they look at it until their eyes begin to water. The nearest dining chair is still pulled out. They sit, and the seat is colder than the room; from there the dining table is close enough to touch, and they set it the way it was set an hour ago. The vase in the middle of the table holds water that has not moved all evening. They inspect the rim, the base, the thin dark line of the glass, and then turn to the book lying open beside it. They read a page, and another, and the words keep sliding back into the same sentence. Above them the pendant lamp buzzes softly, so they reach up and dim it until the room is almost honest. One by one they walk the chairs of the first table back into place: the second dining chair pushed in, the third straightened until its back runs parallel with the table, the fourth slid back under the edge. On the far wall the poster shows a bright room that does not exist here. They look at it for a long moment, then turn away. The second dining table is set for a meal nobody is having. They sit at the dining chair beside it and look down the length of the table: a cup, a wineglass, a beer bottle, and the window box whose leaves have gone grey. They pick up the cup and set it down again, stand the beer bottle upright, lift the wineglass into the light, and run two fingers through the soil of the window box. At last they pull out the far dining chair, straighten the one at the end, and push the last dining chair in, and the room is exactly as still as it was when they came in.



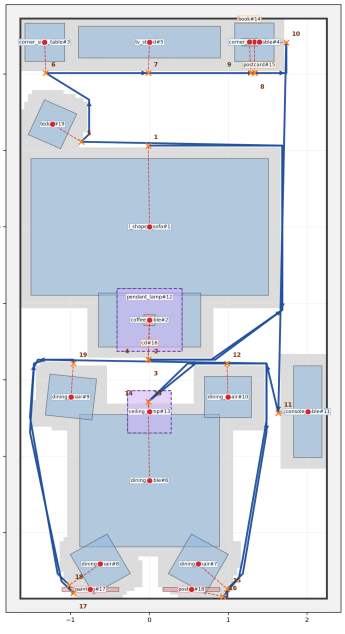


#	object	action	walk	off	
1	shelf#7	pick from	0.00	0.36	
2	clock#8	check	1.49	0.35	
3	trash_can#20*	empty	0.76	0.32	
4	painting#21*	look at	2.70	0.00	
5	dining_chair#2	sit on	3.68	0.49	
6	dining_table#1	set the table	0.00	0.60	
7	vase#9	inspect	0.00	0.60	
8	book#10	read	0.00	0.76	
9	pendant_lamp#6	dim	1.57	0.60	
10	dining_chair#4	push in	2.16	0.49	
11	dining_chair#5	straighten	1.49	0.50	
12	dining_chair#3	slide back	1.93	1.36	
13	poster#22*	look at	1.72	0.00	
14	dining_chair#14*	sit on	2.71	0.43	
15	dining_table#11*	set the table	0.00	0.69	
16	cup#16*	pick up	2.30	0.41	
17	beer_bottle#18*	pick up	3.21	0.53	
18	wineglass#17*	pick up	0.00	0.69	
19	window_box_(for_plants)#19*	inspect		0.20	0.40
20	dining_chair#12*	pull out	5.64	0.46	
21	dining_chair#15*	straighten	0.76	0.43	
22	dining_chair#13*	push in	-	0.44	

Figure 9: FilmScene Example Visualization.

The evening light in the living room is warm and unhurried. The character sinks into the l shaped sofa and lets the cushions take the weight of the day, then leans forward and rests a hand on the coffeetable in front of them. A cd has been lying on it untouched for weeks; they pick it up, turn it once in the low light, and set it down again. Overhead the pendantlamp burns a little too dim, so they reach up and nudge it until the seating corner glows properly. A teddy lies tipped onto its side beside the sofa's left arm. They scoop it up, smooth one ear flat, and carry it across to the cornersidetable that stands beside the tvstand, setting it on the tabletop to face the room. Then they crouch at the tvstand itself, run a thumb along its edge, and switch it on, and the wall behind it takes on a pale, shifting glow. On the other cornersidetable, at the far end of the tvstand, a book lies open with a postcard keeping its place. They lift the book and read a page standing up, then ease the postcard free and look at the faded ink - a beach, a date, two names. Both items travel with them to the consoletable along the side wall, where they lay them down side by side. At the dining end of the room they pull out a diningchair and sit; the ceilinglamp overhead is brighter than they want, so they stand again and dim it. The diningtable takes the rest of their attention. They straighten the far diningchair that has been left pushed out of line, then stop in front of the poster on the wall and look at the bright city print, and afterwards stand a while at the painting beside it, letting its darker mood settle over them. Finally they push the other far diningchair neatly back into place, walk around to the nearer diningchair on the left, and sit down in a room gone quiet.





#	object	action	walk	off
1	l_shaped_sofa#1	sit on	5.76	1.06
2	coffee_table#2	rest on	0.00	0.52
3	cd#16*	pick up	0.00	0.52
4	pendant_lamp#12	switch on	6.62	0.52
5	teddy#19*	pick up	1.24	0.43
6	corner_side_table#3	set down on	1.30	0.40
7	tv_stand#5	switch on	1.35	0.40
8	corner_side_table#4	take book from	0.05	0.40
9	book#14	read	0.85	0.40
10	postcard#15	look at	4.85	0.34
11	console_table#11	lay items down on	1.17	0.37
12	dining_chair#10	pull out and sit in	1.21	0.44
13	ceiling_lamp#13	dim	0.00	0.13
14	dining_table#6	set the table	4.74	1.03
15	dining_chair#7	straighten	0.15	0.49
16	poster#18*	look at	9.22	0.41
17	painting#17*	look at	0.12	0.22
18	dining_chair#8	push in	3.55	0.48
19	dining_chair#9	sit in	-	0.44

object standing on the ground (blocks walking)
hanging object: pendant_lamp / ceiling_lamp (above head, walkable)
object on a surface: book / postcard / cd (walkable below)
wall art: painting / poster (1.24 m up, walkable below)
free floor removed by the 0.15 m inflation
A* path on free floor (arrow = walking direction)
interaction point = object bbox center
landing cell = nearest walkable cell
stand-off connector (bbox center → landing cell)

Figure 10: FilmScene Example Visualization.

The stage shows a realistic indoor space, with few but refined furniture, which fully embodies a strong sense of history and vicissitudes, and instantly brings the audience into a decadent old feudal family. The overall design skillfully uses old architectural elements, the background wall is mottled and old, and a delicate and dilapidated carved bed is displayed in the center of the stage. The eldest son, dressed in mourning, slowly walks in from the right side of the stage, pauses briefly before the carved bed, then reaches out to touch the bed frame with a trembling hand , the bed is not only a symbol of traditional Beijing home culture, but also cleverly divides the stage space and guides the audience to focus on the core area. On the left side of the stage, a mottled black display cabinet stands still with ornaments in it. The traces of its surface use seem to tell the story of the past, symbolizing the deposition of history and the rise and fall of the family. The son turns to the washstand, picks up the white towel, and wipes his face slowly. Next to the carved bed is this old wooden washstand with a white washbasin and white towels. After hanging the towel back on the rack, he walks over to the single Chinese sofa on the right side of the front area of the stage, which carries a blanket with a sense of time. He sits down heavily and stares blankly at the black display cabinet on the left. On the right side of the sofa is a small edge with exquisite ornaments on it. Although the furniture is exquisite, it always reveals a gloomy and rotten atmosphere. After a long pause, he stands up, approaches the cabinet, opens its door, then he returns to the sofa, sits down again, and gazes at it in silence. Meanwhile, the youngest daughter quietly enters from the left, walks to the edge of the carved bed, kneels down slowly, and rests her head on the bed frame. She then rises, walks to the washstand, pours water from the basin into a small bowl, and carries it carefully to her brother on the sofa. He takes the bowl, drinks, and places it on the small side table next to the sofa. The two remain motionless. The prominent black pane on the right background board looks like the carved flowers on the side of the coffin in ancient times. It is a metaphor that the whole space is a mourning hall, and it points out the theme of the play: the feudal family in old China is bound to decline until it dies. As the lights slowly fade, the two figures remain seated in silence.

Script



Scene

The stage is a fine double-layer wooden house model. The first floor is the thematic space of the drama, the "public kitchen," with a large wooden table in the middle and three stoves in a row on the right. A square table and three chairs are placed against the wall in the dining area on the left, and several layers of posters are pasted on the wall. There are several layers of shelves on the main wall of the kitchen, filled with bottles and jars, revealing the triviality and delicacy of life. A woman in an apron stands at the stove, stirring a pot with one hand while reaching for a jar on the shelf with the other. A young man sits at the square table, flipping through a newspaper, occasionally glancing toward the stove. An elderly man slowly walks down the L-shaped wooden staircase on the far right, which connects the upper and lower floors, pausing midway on the resting platform where a staggered door leads to more rooms at the back. On the second floor, there is a row of square glass sliding door balconies. The old-fashioned clothes drying racks are set up outdoors, drying scattered clothes such as skirts and long-sleeved jackets. A young girl steps onto the balcony, takes a skirt off the rack, folds it carefully, and places it into a locker. There is a row of half-person-high lockers in the room. Another man emerges from the staggered door on the resting platform, carrying a stack of bowls, and descends to the kitchen, placing them on the large wooden table. The woman at the stove turns, takes a bowl from the table, and ladles soup into it, handing it to the young man at the square table. The design exudes a strong sense of life. The double-layer structure makes the stage space more three-dimensional, providing more scheduling possibilities for the actors' performances.

Script

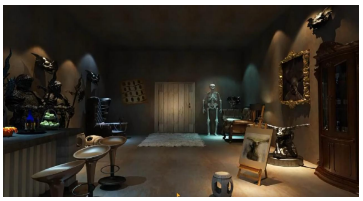


Scene

Figure 11: Out-of-distribution Data Visualization in aesthetic-centric scenarios.

Steeped in oppressive shadows and lit only by dim spotlights, the dark stone room immediately draws the player toward an easel in the foreground, where they examine a landscape painting to discover faint grid patterns woven into the horizon. Turning to the bar counter on the left, flanked by two menacing suits of black armor. The player counts the six green apples in the bowl and cross-references them with a patterned cloth on the back wall to deduce a four-digit numeric code. They use this code to unlock the ornate wooden cabinet on the right, retrieving a velvet box that holds a sharp silver dagger. Crossing back to the suits of armor, the player slots the dagger into the left knight's empty gauntlet, triggering a hidden gear that causes the wall-mounted gargoyle head above to drop a heavy iron key. The player retrieves the key and carries it to the illuminated skeleton standing by the back wall, inserting it into a subtle keyhole hidden within its ribcage; turning the key activates the skeleton's mechanical joints, causing its right arm to swing up and point directly at the gilded portrait on the right wall. Following its gesture, the player tilts the bottom of the gold frame to reveal a hidden press-button, which unbolts the heavy central wooden door with a loud metallic clunk, swinging it open to reveal the escape path.

Script



Scene

Bathed in the warm, dim radiance of a grand crystal chandelier suspended from the high ceiling, the long corridor is wrapped in intricate, damask-patterned wall panels and lined with a brown-and-tan diamond tiled floor. Deep in the room, the player's attention is immediately drawn to the large, gilded mirror leaning precariously against the left wall, its dusty surface painted with the word "STYLE" in stark red lettering that hides a word-puzzle key. Searching through the messy stack of cardboard boxes and open crates piled underneath the mirror, the player uncovers a small combination lockbox. The lockbox's tumbler requires a three-digit sequence, which the player decodes by matching the color scheme of the red "STYLE" text to the bright red, yellow, and blue Greek letters rendered over the classic oil painting on the right wall. Using the decoded sequence to open the box reveals a small brass key. The player carries it over to the seating nook on the right, where two heavy, plush brown leather armchairs rest over an ornate, yellow-and-navy Persian rug, flanking a dark wood side table topped with decorative glass decanters. Inserting the key into a hidden keyhole beneath the right armchair's cushion releases a spring mechanism in the side table, opening its small cabinet to reveal a heavy stone horse figurine. Walking back across the room, the player slots the figurine into the empty brass horse-head wall sconce mounted beside the mirror, which engages an internal pulley system that pops open the wall-mounted red emergency box near the back wall. Inside the brightly painted emergency box sits a high-security magnetic keycard; swiping it through the digital reader beside the closed, beige elevator doors at the end of the hallway disengages the heavy iron bolts, allowing the doors to slide open for the final escape.

Script



Scene

Figure 12: Out-of-distribution Data Visualization in game scenarios.

D PROMPT TEMPLATES

We provide the detailed prompts for initial scene construction in Table 4, executable violation detection in Table 5, and minimal repair in Table 6.

Table 4: Initial scene construction prompt template.

Prompt Template for Initial Scene Construction

Given the following script, design a {room_type} scene that matches the story and output the complete 3D scene layout.

First predict the coarse scene graph matching the requirements, then based on it return the output. The output must be a JSON object listing every object in the scene with its attributes, in exactly this format:

```
{
  "room_type": "<room type>",
  "objects": [
    {
      "object_id": 1,
      "category": "<object category>",
      "size": [sx, sy, sz],
      "position": [cx, cy, cz],
      "rotation_angle": <angle>
    }
  ]
}
```

Field meanings:
- size: bounding-box dimensions of the object in meters, [x, y, z].
- position: center of the object in world coordinates (meters), [x, y, z].
- rotation_angle: rotation angle around the y-axis, in radians.

Scene script: {script}

Output only the JSON object without any extra text.

Table 5: Executable violation detection prompt template.

Prompt Template for Executable Violation Detection

You are an interior scene quality inspector. Below are multi-view renderings of the same 3D scene (the first few images are surrounding views, the last one is a top-down view), together with the object information of the scene.
{obj_text}
{chain_text}

Carefully inspect the images and list ALL unreasonable layout / defect issues, focusing on:
1. Whether cabinet-type objects (wardrobe / wine cabinet / tv stand / children cabinet etc.) have their front face or door facing the interior of the room (toward usable space), instead of being flush against a wall or facing the wall directly;
2. Whether chairs on the same side of a table (dining table / desk) share a consistent orientation, all facing the table edge (not facing other directions);
3. Whether adjacent objects in the behavior chain are too far apart (more than 3 meters is considered too far), making it hard for the character to perform consecutive interactions;
4. Any other obviously unreasonable object orientation or layout;
5. Whether any interactive region of objects is blocked by other objects.
IMPORTANT: ONLY report actual problems. Do NOT mention objects that are placed correctly. For every reported problem, also provide a concrete, actionable fix suggestion (e.g. "rotate the wardrobe 90 degrees so its door faces the room interior", "move chair #2 to align with the table edge and face the table", "move object A closer to object B", "lower object C to rest on the table top").

Output format: one issue per line, in the form "- [CATEGORY] description (involving object_id: ...). Fix: <specific fix suggestion>". If everything is reasonable, output exactly one line: "No issues". Do not output anything else.

Table 6: Repair prompt template.

Prompt Template for Minimal Repair

You are a professional 3D interior scene layout editor. You are given a predicted scene layout and its defect report. You must edit the scene so that the scene becomes reasonable and all the reported defects are eliminated, while preserving the parts of the scene that are already valid.

— Predicted scene —
{scene_text}

— Defect report —
{defect_log_text}

You may perform these operations:
1. resize: modify an object's size (keep physically plausible dimensions);
2. re-position: modify an object's position (keep ground objects on the floor with position.y = 0, keep small objects resting on their support surface);
3. rotation: modify an object's rotation_angle (radians around the Y axis);
4. category replacement: replace an object's category with a similar suitable one;
5. delete: remove an object from the list.

Editing principles:
- Repair minimally: modify only the objects implicated by the defect report, and leave all other objects unchanged in category, size, position, and rotation.
- Eliminate all [Intersection] pairs: move / rotate / resize / replace / delete objects so they no longer penetrate each other.
- Eliminate all [Floating] issues: make sure every non-hanging object rests on the floor or on a supporting surface.
- Eliminate all [VLM] issues: rotate objects to make them functionally usable (e.g. chairs facing tables, cabinets facing counters).
- Keep behavior-chain objects reachable and not too far from each other.
- Prefer the smallest edit that resolves each reported defect: when several operations can fix a defect, choose the one that changes the least (e.g., prefer re-position or rotation over category replacement or deletion).
- DO NOT introduce new problems.
- Keep the scene realistic for the room type.

Output ONLY the corrected scene as a JSON object with exactly this format:
{
  "room_type": "<room type>",
  "objects": [
    {"object_id": <int>, "category": "<category>", "size": [sx, sy, sz], "position": [cx, cy, cz], "rotation_angle": <radians>}
  ]
}

Keep every remaining object's object_id unchanged.

E Extended Related Work

Image-to-3D Indoor Scene Generation. Recent advances Huang et al. (2025); Yang et al. (2026); Yin et al. (2026) in image-to-3D indoor scene generation have enabled the creation of 3D scenes from single or multi-view images. While promising, their outputs are often designed for reconstruction or visual synthesis rather than the structured generation of a complete indoor room. As a result, they are not well-suited for modeling room-scale scene elements such as small objects to interact with. As for the SCR-SG setting, scripts contain rich details and implicit semantic requirements that cannot be faithfully conveyed through a single image. Much of the detailed semantic information is implicit and may be overlooked by a single-view image. Reconstructing a scene from one image inevitably loses this behavioral layer, as the image captures only static appearance while discarding the action-oriented semantics that SCR-SG must preserve.

F METHOD DETAILS

F.1 IMPLEMENTATION DETAILS

For Generator training, the detailed configuration is provided in Table 7.

Table 7: Generator training configuration.

Parameter	**Value**
Fine-tuning method	LoRA
LoRA rank r	16
LoRA alpha α	32
LoRA dropout	0.05
Learning rate	1×10^{-4}
Learning rate schedule	Cosine decay
Warmup ratio	0.03
Batch size	8
Gradient accumulation steps	4
Training epochs	5
Weight decay	0.01
Max sequence length	8192
Optimizer	AdamW
GPUs	8 $\times$ NVIDIA RTX 4090

F.2 AFFORDANCE CLASSIFIER

Input representation. The affordance classifier $\mathrm{CLS}(a, cat, s)$ takes as input a triple consisting of an action a, an object category cat, and the object's attributes $s \in \mathbb{R}^4$. Both a and cat are drawn from predefined discrete vocabularies: the action set $\mathcal{A} = \{a_1, \ldots, a_{N_a}\}$ ($N_a = 18$, covering actions such as *sit*, *write*, *dine*, *cook*, *sleep*, *store*, *reach*, *open*, *place*, *read*, *wash*, *turn_on*, *lie_on*, *lean_on*, *step_on*, *hang*, *display*, *contain*) and the category set $\mathcal{C} = \{c_1, \ldots, c_{N_c}\}$ ($N_c = 71$, matching the object categories in FilmScene). Each action is encoded as a one-hot vector $\mathbf{e}_a \in \{0,1\}^{N_a}$ and each category as a one-hot vector $\mathbf{e}_c \in \{0,1\}^{N_c}$. These sparse one-hot vectors are projected into dense embeddings via learned linear layers:

$$\mathbf{h}_a = \mathbf{W}_a \mathbf{e}_a \in \mathbb{R}^{d_a}, \qquad \mathbf{h}_c = \mathbf{W}_c \mathbf{e}_c \in \mathbb{R}^{d_c}, \tag{4}$$

where $d_a = 32$ and $d_c = 64$ are the embedding dimensions. The dimension vector $s = [s_x, s_y, s_z, angle]$ is normalized by the room bounding box, the angle is transformed into radians, serving as $\tilde{s} \in [0,1]^3$, which are concatenated with the embeddings:

$$\mathbf{z} = [\mathbf{h}_a; \mathbf{h}_c; \tilde{s}] \in \mathbb{R}^{d_a + d_c + 4}. \tag{5}$$

Architecture. The concatenated representation $\mathbf{z}$ is passed through a 3-layer MLP with hidden dimensions (256, 128, 64) and ReLU activations. The final layer produces a scalar logit, which is mapped to a compatibility score via a sigmoid:

$$\mathrm{CLS}(a, cat, s) = \sigma\big(\mathrm{MLP}(\mathbf{z})\big) \in (0,1). \tag{6}$$

Training data. We construct triplets (a, cat, s) with continuous labels $y \in [0,1]$. Positive samples ($y = 1$) are extracted from FilmScene where an object is explicitly used for an action. Negative samples ($0 < y < 1$) are synthesized by perturbing valid object dimensions:

$$y = \exp\big(-\alpha \|s - s_{gt}\|^2\big), \quad \alpha = 0.5, \tag{7}$$

where s_{gt} is the ground-truth dimension of the object in its original valid configuration. This yields a smooth decay of compatibility as the object deviates from its functional dimensions. We also append negative samples with mismatched actions and object categories with labels $= 0$, guiding the classifier to prioritize category-wise affordance.

F.3 Minimal Repair Preference Learning Details

We provide the details regarding the preference learning variant for the Minimal Repair. The intention is to construct paired scenes that contain certain flaws and their corrected versions, using which to train the Minimal Repair to more deliberately correct the initial scene.

Notation. Let $\mathcal{O}^{gt}$ denote a ground-truth scene layout from FilmScene, and let $\mathcal{O}^{noise}$ denote a corrupted version of $\mathcal{O}^{gt}$ obtained by applying a random perturbation. We define the corruption operator $\mathcal{N}(\cdot)$ that introduces one or more of the following error types:

- **Path-blocking displacement:** randomly translate an object o_i by a displacement $\delta \in \mathbb{R}^3$ such that the resulting position obstructs a previously valid path between two interaction points.
- **Affordance-degrading rescaling:** scale an object o_i by a factor $\gamma \in [0.5, 1.5]$ such that its dimensions fall outside the functional range for its associated action (e.g., shrink a dining table below the minimum height for dining).
- **Chain-breaking removal/substitution:** either delete an object o_i that participates in a behavior chain, or replace its category with a functionally incompatible alternative (e.g., replace a dining chair with a stool that lacks back support).

Each corruption produces a corrupted layout $\mathcal{O}^{noise} = \mathcal{N}(\mathcal{O}^{gt})$. We then run the Executable Constraint Diagnoser on $\mathcal{O}^{noise}$ to obtain the corresponding defect log E^{noise}, which records the specific violations introduced. The full training instance is the tuple $(\mathcal{O}^{noise}, E^{noise}, \mathcal{O}^{gt})$, which provides the model with both the corrupted scene, its diagnosed defects, and the target correct output. We train the Minimal Repair with supervised fine-tuning, serving $\mathcal{O}^{noise}$ and E^{noise} as the input and the corresponding correction towards $\mathcal{O}^{gt}$ as the labels.

F.4 Score-Based Reinforcement Learning for Minimal Repair

We provide the details regarding the Score-Based Reinforcement Learning variant for the Minimal Repair. The intention is to replace the explicit defect log with numerical judgments as the metric to guide scene optimization.

Reward definition. We define a continuous reward that quantifies the overall quality of a scene along four dimensions: spatial executability, functional executability, sequential executability, and physical plausibility. Each dimension is assigned a maximum score of 5 points, and specific violations incur deductions.

1. Spatial Executability (max 5).

- Path blocked between a pair of consecutive interaction points: -2 per blocked pair.

2. Functional Executability (max 5).

- Affordance score below τ_{aff} for an action-object pair: -2 per pair.

3. Sequential Executability (max 5).

- Consecutive interaction objects separated by large spatial distance: -2 per transition.
- Interaction region occluded by adjacent objects: -1 per occurrence.
- Chain broken (missing object or incompatible relation): -3 per break.

4. Physical Validity (max 5).

- Object-object intersection: -1 per intersecting pair.
- Floating object (no supporter): -1 per object.
- Anti-gravity placement (center of mass outside supporter): -1 per object.

The total reward is the sum of the four dimension scores, clipped to $[0, 20]$. Formally, for a scene $\mathcal{O}$:

$$R(\mathcal{O}) = \max\left(0, \sum_{d\in\{\text{spa,fun,seq,phy}\}} \left(5 - \sum_{v\in\mathcal{V}_d(\mathcal{O})} \text{pen}(v)\right)\right), \tag{8}$$

where $\mathcal{V}_d(\mathcal{O})$ is the set of violations detected in dimension d for scene $\mathcal{O}$, and $\text{pen}(v)$ is the penalty associated with violation v as defined above.

We train the Minimal Repair using Group Relative Policy Optimization (GRPO) Shao et al. (2024). For each input scene $\mathcal{O}^{noise}$ (a corrupted layout from the same corruption pipeline as in Appendix F.3), we sample a group of $G = 8$ candidate corrections $\{\mathcal{O}_1, \ldots, \mathcal{O}_G\}$ from the current policy. Each candidate is scored by $R(\cdot)$, then we update the policy with the GRPO objective:

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(\mathcal{O}_i \mid \mathcal{O}^{noise}, E^{noise})}{\pi_{\text{old}}(\mathcal{O}_i \mid \mathcal{O}^{noise}, E^{noise})} A_i,\ \text{clip}\left(\frac{\pi_\theta}{\pi_{\text{old}}}, 1-\epsilon, 1+\epsilon\right) A_i\right)\right], \tag{9}$$

with clipping parameter $\epsilon = 0.2$. Training data consists of 5k corrupted scenes generated by the same corruption operator $\mathcal{N}(\cdot)$ used for preference learning.

F.5 Iterative Refinement Details

We provide the details regarding the iterative refinement procedure. It alternates between executable violation detection and constraint-preserving scene editing. Starting from the initial layout $\mathcal{O}^{ini}$ produced by the Generator, at each iteration t we run the Executable Constraint Diagnoser to obtain the defect log E_t, and if $E_t \neq \emptyset$, we apply the Minimal Repair to produce an updated layout $\mathcal{O}_{t+1}$. The loop terminates when either no defects are detected or the maximum number of iterations $T = 3$ is reached.

Algorithm 1 Iterative Refinement Loop

Require: Script T_{scr}, Initial layout $\mathcal{O}^{ini}$, Max iterations T
Ensure: Refined layout $\mathcal{O}^*$

$\mathcal{O} \leftarrow \mathcal{O}^{ini}$
for $t = 1$ to T **do**
  $E \leftarrow$ Executable Constraint Diagnoser$(\mathcal{O}, T_{scr})$
  if $E = \emptyset$ **then**
    break
  end if
  $\mathcal{O} \leftarrow$ Minimal Repair$(\mathcal{O}, E, T_{scr})$
end for

return $\mathcal{O}$

G Visualization of Intermediate Results in Each Stage

Figure 13 visualizes the progression of our framework and the corresponding intermediate results: the initial scene generated by the Generator, the defect log produced by the Executable Constraint Diagnoser, and the final refined scene after correction. In the initial scene, we observe violations ranging from physical issues such as object penetration and floating, to behavioral issues such as lack of usage space and excessive distance for consecutive interactions. The corresponding defect log precisely captures these issues in natural language and provides tailored fixing suggestions. After correction, the refined scene accurately resolves these errors, yielding scenes more suitable for script filming while maintaining physical plausibility and narrative coherence.

H Efficiency Analysis

Analysis. As shown in Table 8, our S2S framework demonstrates a favorable efficiency–quality trade-off compared to existing methods. Without iterative refinement, the full S2S pipeline (Generator, Executable Constraint Diagnoser, and Minimal Repair) requires moderate generation time

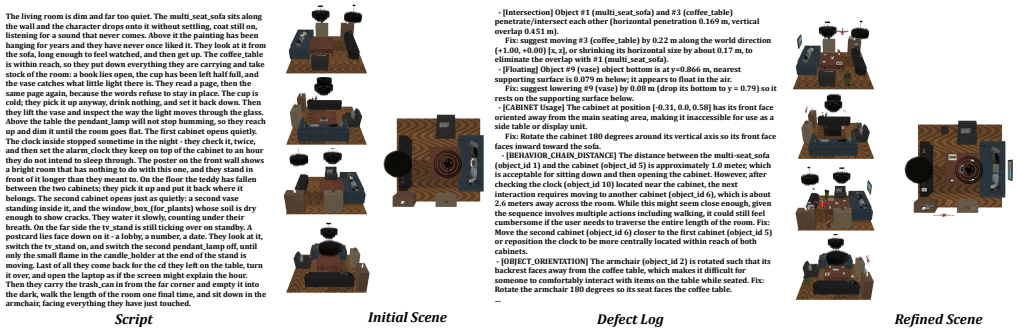


Figure 13: The intermediate output of each stage of S2S.

Table 8: Generation speed comparison. FS and SC are reported on the FilmScene test set. Tests are conducted on NVIDIA RTX 4090 GPUs.

Method	Time per Scene (s)	FS↑	SC↑	Notes
LayoutGPT	7.8	67.23	54.11	Single pass
InstructScene	9.6	77.02	74.34	Single pass
LayoutVLM	8.2	85.50	82.71	Scene Parameter Optimization
AnyHome	∼200	88.04	73.82	Generate-then-refine
StageDesigner	∼48	89.18	80.76	Generate-then-refine
SceneWeaver	∼580	87.54	81.00	Iterative Refinement
S2S (Generator/Diagnoser/Minimal Repair/Full)	3.9/14.5/5.6/∼24	91.08	84.75	Generate-then-repair
S2S + Iterative Refinement	∼53	**91.64**	**85.42**	Iterative Refinement

yet achieves an FS of 91.08 and an SC of 84.75, outperforming all baselines on both behavioral metrics. While methods like SceneWeaver and AnyHome require more time to optimize the generated scenes, even StageDesigner, which is specifically designed for script-driven stage generation, requires ∼48s (2× slower than our full pipeline) yet achieves only 89.18 FS and 80.76 SC. Further refinement through iterative correction (∼53s, approximately 2.2× the single-pass cost) yields additional gains, boosting SC to 85.42 and FS to 91.64. These results demonstrate that our method achieves state-of-the-art behavioral performance with efficiency.