LIVINGFLOW: AGENTIC REINFORCEMENT LEARNING FOR CIRCULATION-AWARE SCENE REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Comfortable living spaces require both functional furniture arrangements and smooth circulation—the movement of people between functional locations during everyday activities. A collision-free layout can still force people through tight passages or unnecessary detours, even when every destination is reachable. We introduce LivingFlow, an agentic reinforcement learning framework that iteratively rearranges furniture in existing indoor layouts. To specify circulation requirements, a frozen large language model generates natural-language activity chains describing how people use the space throughout a day; transitions between successive uses define fixed movement tasks that a geometric executor evaluates for furniture access, clearance, and route efficiency as the layout changes. Our framework adapts Group-in-Group Policy Optimization to optimize a trainable editing policy over complete tool cycles, connecting information gathering and editing decisions with executable feedback. Experiments on original 3D-FRONT layouts show improved layout quality and circulation over existing refiners, and confirm the benefits of activity-conditioned evaluation and executable feedback.

1 INTRODUCTION

Furniture arrangements shape how people move between functional locations during everyday activities. Learning-based layout models and language-conditioned generators can construct indoor environments from examples and descriptions (Paschalidou et al., 2021; Tang et al., 2024; Feng et al., 2023; Yang et al., 2024b), yet collision-free placement and reachable destinations do not ensure convenient circulation. Carrying breakfast from a kitchen entrance to a dining table, for example, may require squeezing past a coffee table or taking a long detour. We refer to movement between functional locations as *activity-driven circulation*, and study how existing layouts can be refined to better support it.

Prior work models furniture use and navigation through activity-centric synthesis, human-motion conditioning, and human-aware optimization (Fisher et al., 2015; Yi et al., 2023; Sun et al., 2024). Functional specifications, household programs, and contextual routines further describe how spaces and objects support human activities (Wang et al., 2026; Puig et al., 2018; Park et al., 2023). Separately, layout refiners learn to modify existing arrangements using geometric, visual, or iterative feedback (Wei et al., 2023; Dang et al., 2026; Zhao et al., 2026). Taken together, these studies address activity-aware scene construction and the correction of geometric or visual layout defects. However, repairing poor circulation across the routes required by multiple daily activities remains underexplored.

This problem is challenging because different activity routes share the same space and furniture. An edit that clears one route may obstruct another or introduce a geometric violation, and its overall effect can only be determined after the edit is executed and the updated layout is re-evaluated. We address this problem with LIVINGFLOW, an agentic reinforcement learning framework that iteratively refines existing furniture layouts using executable circulation feedback, as illustrated in Figure 1. A frozen language model generates activity chains from room semantics and furniture uses, with transitions between successive uses defining fixed movement requirements. A trainable editing policy selects information-gathering queries and uses the resulting evidence to translate, rotate, or resize furniture, or to stop. After each edit, a geometric executor recomputes furniture access and routes for the fixed requirements. We group the queries and the resulting edit or stopping decision

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.

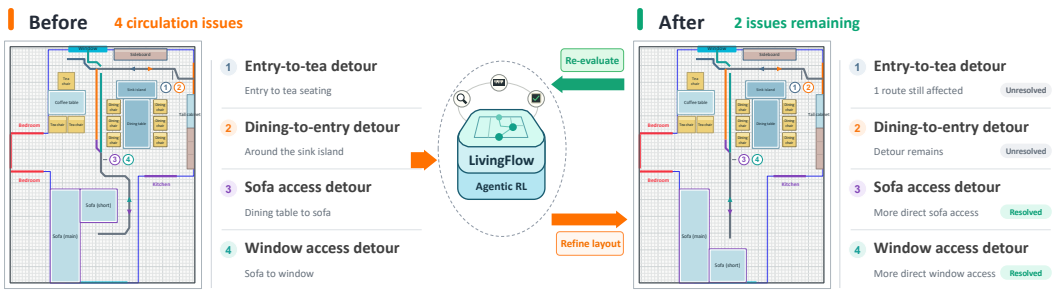

Figure 1: **Activity-driven refinement with executable feedback.** LivingFlow alternates furniture editing with re-evaluation of the updated layout against activity-derived circulation requirements. Here, an edit resolves two of four route issues, while the remaining issues provide feedback for the next decision. The layout, routes, and issue outcomes are illustrative.

into complete tool cycles, and adapt Group-in-Group Policy Optimization (Feng et al., 2025) to optimize the editing policy using cycle-level samples and execution-grounded credit assignment.

We evaluate LIVINGFLOW on original furnished 3D-FRONT living rooms, living–dining rooms, and bedrooms against six scene layout methods. LIVINGFLOW achieves the highest movement pass rate and geometric validity in every category, together with the lowest boundary and collision rates. Movement pass rate improves over the input by 26.99, 29.69, and 43.64 percentage points, respectively. Ablations support the contributions of activity-conditioned refinement, sequential interaction, and task-specific learning.

Our contributions are:

- We present LIVINGFLOW, a circulation-aware refinement approach that optimizes existing furniture layouts using activity-conditioned geometric feedback.
- We develop an agentic RL framework that learns to coordinate furniture translation, rotation, and resizing through complete tool cycles and execution feedback.
- Experiments demonstrate improved circulation and geometric validity across three room categories, with ablations supporting activity-conditioned refinement, sequential interaction, and task-specific policy learning.

2 RELATED WORK

Indoor Layout Generation and Optimization. Indoor scene synthesis combines design constraints and example-based relationships (Merrell et al., 2011; Yu et al., 2011; Fisher et al., 2012) with learned convolutional, autoregressive, diffusion, and graph priors (Wang et al., 2018; Ritchie et al., 2019; Paschalidou et al., 2021; Tang et al., 2024; Lin & Mu, 2024). Language-conditioned methods further support generation from descriptions and instructions (Feng et al., 2023; Yang et al., 2024b; Çelen et al., 2025). Beyond visual composition, activity-centric and human-aware approaches incorporate furniture use, human motion, navigation, and physical interaction (Fisher et al., 2015; Yi et al., 2023; Sun et al., 2024; Yang et al., 2024a), while LayoutVLM combines visual reasoning with differentiable spatial optimization (Sun et al., 2025). These methods establish that functionality and accessibility can guide scene generation and optimization. For activity-driven refinement, however, supporting individual furniture uses or general reachability does not by itself specify which transitions between uses should guide editing. The refinement objective must also preserve these movement requirements as furniture changes, so that successive layouts remain comparable. LIVINGFLOW addresses this problem through fixed activity chains that define movement tasks. A geometric executor updates their access locations and routes after each edit, assessing how the changing layout supports the same activities.

Agentic Scene Generation and Refinement. Language agents combine reasoning, tool use, and environmental feedback to revise plans and executable actions (Yao et al., 2023; Schick et al., 2023; Shinn et al., 2023; Ichter et al., 2023; Liang et al., 2023). In scene synthesis, SceneWeaver, Scene-

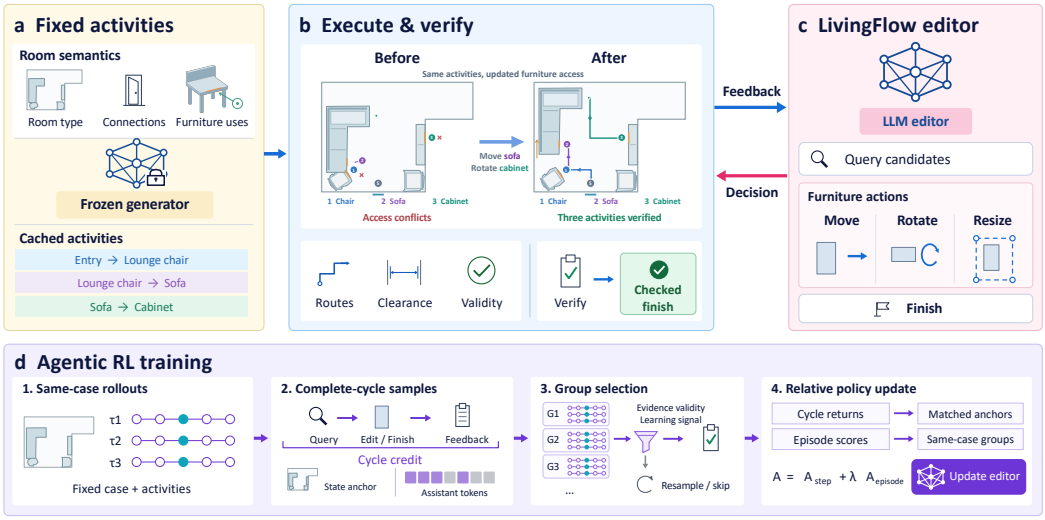

Figure 2: **LivingFlow execution and training workflow.** A frozen generator produces cached activity requirements from room semantics and furniture uses (a). A geometric executor grounds and re-evaluates the associated access locations and routes after each furniture action (b), providing feedback for the trainable editor's queries and decisions (c). Completed same-case rollouts are converted into complete-cycle samples, filtered into valid learning groups, and used to update the editor through cycle- and episode-relative comparisons (d).

Smith, and SAGE organize generation and revision through agent interactions and feedback (Yang et al., 2025; Pfaff et al., 2026; Xia et al., 2026). SceneReVis learns iterative diagnosis and editing through multi-turn reinforcement learning, while FuncRoom-Agent uses execution-grounded process rewards to train sequential scene construction (Zhao et al., 2026; Feng et al., 2026). These approaches establish the value of feedback-driven scene decisions, but leave open the task-specific choices of how interactions become training samples and receive credit. PPO and GRPO provide policy optimization objectives (Schulman et al., 2017; Shao et al., 2024), and GiGPO combines episode-level comparisons with step-level comparisons at repeated anchor states (Feng et al., 2025). In layout refinement, queries and permitted retries can inform later decisions without changing the furniture, making their contribution dependent on subsequent execution. LIVINGFLOW adapts GiGPO around complete tool cycles, grouping information gathering with the closing edit or stopping decision. Together with layout-specific comparison anchors and verified execution feedback, this organization connects interaction choices to their consequences under fixed activity requirements.

3 LIVINGFLOW: AGENTIC LAYOUT REFINEMENT

LIVINGFLOW refines a furnished room while preserving its architecture and furniture inventory (Figure 2). A frozen generator defines the activity requirements; a geometric executor grounds their movement tasks and recomputes access, routes, and validity after each edit; and an editor queries actions, modifies the layout, and verifies the result. Complete tool cycles turn this closed-loop interaction into samples for group-relative policy learning. The following subsections define the objective, executor, interaction protocol, rewards, and training procedure.

3.1 FIXED ACTIVITY OBJECTIVE

LIVINGFLOW separates the requirements a room should support from the geometry through which they are realized. We represent a case as $x = (\Omega, F, H, S_0)$, where Ω describes the room's architecture, including boundaries and door openings; F records furniture identities and supported uses, such as sitting on a sofa or eating at a dining table; H provides semantic context, including the room type and connections to adjoining spaces; and S_0 specifies the initial furniture positions, orientations, and dimensions. The generator receives only the semantic projection of the case, whereas the

executor evaluates its requirements in every edited layout:

$$Z_x = G_{\phi_0}(\mathrm{sem}(x)), \qquad e_t = \mathrm{Exec}_\Theta(S_t, Z_x, b_t). \tag{1}$$

Here, Z_x contains activity chains, each an ordered sequence of functional locations corresponding to an object use or a room-side arrival; adjacent locations define a movement task. The executor applies fixed geometric rules Θ and continuity context b_t to produce evidence e_t. Edits change permitted positions, orientations, and dimensions in S_t, while architecture, furniture inventory, activity identities, and visit order remain fixed.

Activity generation. The generator organizes daily contexts into person-specific activity chains from room type, connectivity, furniture semantics, and conservative functional uses, without coordinates or images. A breakfast chain, for example, may connect arrival from the kitchen to use of the dining table. Preparation validates its schema, referenced objects, and continuity before caching Z_x; the executor determines access locations and routes from each current layout.

Fixed activity specifications. An activity specification selects the movement tasks that enter progress and completion rewards. If the generator is updated with the same return as the editor, the objective can rise because easier specifications receive more probability rather than because the layout improves. Holding a specification fixed only within an episode does not prevent this drift across training updates. For a fixed target set $\mathcal{T}$, let $U_{\mathcal{T}}(S)$ be the fraction of its movements that pass in layout S.

Proposition 1 (Objective drift under activity selection)**.** *Let $\mathcal{T} = \{a, b, c, d\}$ contain movements induced by four admissible two-location chains, and suppose a common hard-valid edit changes their pass vector from $(0, 1, 1, 1)$ to $(1, 0, 0, 1)$. For $Z_A = \{a, b\}$ and $Z_B = \{a, d\}$, suppose fresh, comparable evidence gives burden transitions $2 \to 2$ and $2 \to 0$, while* Finish *observes an eligible residual under Z_A and confirmed completion under Z_B. Then the reward rules in Section 3.4 give $U_{\mathcal{T}}(S_1) < U_{\mathcal{T}}(S_0)$ but $\mathcal{R}_\gamma(Z_B) > \mathcal{R}_\gamma(Z_A)$ for every $\gamma \in [0, 1]$.*

For $E_A, E_B \in [-1, 1]$, the discounted edit–Finish returns are $\mathcal{R}_A \equiv \mathcal{R}_\gamma(Z_A) = 0.1E_A - 0.7\gamma$ and $\mathcal{R}_B \equiv \mathcal{R}_\gamma(Z_B) = 1.5 + 0.1E_B + \gamma$, so $\mathcal{R}_B - \mathcal{R}_A \geq 1.3 + 1.7\gamma > 0$. Let $J(q) = \mathbb{E}_{Z\sim q}[\mathcal{R}_\gamma(Z)]$ and let q_ϵ move probability ϵ from Z_A to Z_B. Then

$$J(q_\epsilon) - J(q) = \epsilon\big[\mathcal{R}_\gamma(Z_B) - \mathcal{R}_\gamma(Z_A)\big] > 0, \quad U_{\mathcal{T}}(S_1) - U_{\mathcal{T}}(S_0) = -\tfrac{1}{4}. \tag{2}$$

We therefore freeze G_{ϕ_0} and reuse one cached Z_x per case throughout editor training, keeping the specification distribution fixed.

3.2 Executable Geometric Feedback

Fixed movement identities must be re-evaluated because their geometric realization changes with the layout. For each object use, the executor selects an eligible access location associated with a reviewed use face. It retains the previous selection when that location remains eligible and otherwise applies deterministic tie-breaking; b_t records this continuity. An edit may therefore move an endpoint, alter only the intervening passage, or invalidate access altogether. Movements involving connected spaces use room-side arrival points, since geometry outside the evaluated room is not modeled.

For native 3D-FRONT layouts (Fu et al., 2021), the executor searches a conservative grid and verifies each returned polyline with exact clearance checks. Missing endpoints and failed searches remain unresolved. Separate hard checks cover containment, overlap, source integrity, and opening clearances; a layout is hard-valid if it passes all these checks. The resulting evidence reports access failures, route diagnostics, involved objects, and layout validity, localizing what prevents a movement task from passing without prescribing a repair. Appendix A.3 details access grounding, route validation, and geometric thresholds.

3.3 Complete Tool Interaction

Because an edit can repair one movement task while obstructing another, the editor acts in a closed loop. The policy π_θ receives current validity, movement failures, route evidence, recent outcomes, remaining budgets, and optionally a plan view. It may query action options for one or more targets, submit one registered edit, or request `Finish`. Only the editor is updated by reinforcement learning.

Registered actions. A query registers state-specific Move, Rotate, and Resize candidates with their targets, parameters, and current geometry; an edit must match a current registration. The executor rejects or applies the request and recomputes activity and validity evidence. Rejected edits preserve the layout, whereas applied invalid layouts remain available for a later repair. Only format-contract failures receive permitted retries; semantic failures close the current decision cycle.

Complete tool cycles. We group the observation, queries, permitted format retries, and closing response that support one edit or stopping decision into a complete tool cycle. If cycle t contains m_t subturn rewards $\delta_{t,u}$, its reward is

$$r_t = \sum_{u=1}^{m_t} \delta_{t,u}. \tag{3}$$

A query can therefore receive learning credit through the edit or stopping outcome it enables, even when its immediate reward is zero. Algorithm 1 gives the execution order.

Algorithm 1: Complete-cycle layout refinement

Input: initial layout S_0, cached activity specification Z_x, policy π_θ, and interaction budgets.

Execute Z_x on S_0; initialize $S \leftarrow S_0$ and the quality references.
while the episode is active **do**
  Build observation o_t and anchor h_t; reset candidates $\mathcal{C}_t$ and transcript τ_t.
  repeat: sample $y \sim \pi_\theta(\cdot \mid o_t, \tau_t)$.
    Query: register current target-specific candidates in $\mathcal{C}_t$ and return feedback.
    Edit: match against $\mathcal{C}_t$; apply or reject; if applied, rebind access locations, recompute routes and validity, assign the edit reward, and update S; close the cycle.
    Finish: verify current evidence, assign $R_{\text{finish}}(S)$, and close the cycle and episode.
    Apply the applicable retry, error, and budget rules; append the response, feedback, and δ to τ_t.
  until the cycle closes; store $(h_t, \tau_t, r_t = \sum_{u=1}^{m_t} \delta_{t,u})$ and clear $\mathcal{C}_t$.
return the final layout, verified terminal status, and complete-cycle samples.

3.4 Execution-Grounded Rewards

The reward distinguishes three outcomes of an interaction: whether circulation improves, whether the resulting geometry is valid or repaired, and whether stopping is justified. The current execution branch selects the applicable reward; read-only queries are neutral immediately and obtain credit through their complete cycle.

Circulation progress. In the reported FRONT configuration, executable quality is summarized by a potential that combines issue burden with diagnostic pressure from the accounted problem clusters. Let B_0 and $B(s)$ denote the initial and current effective burden, including movement failures and matched problem clusters; a cluster groups route evidence sharing a movement, affected object, or blocker identity. Let $P_k(s)$ aggregate the current cluster-evidence pressure for backtracking, reversals, turns, excess length, and width deficit. For $K_0 = \{k : P_k(s_0) > 0\}$, define

$$\Phi_x(s) = \alpha \frac{B_0 - B(s)}{\max\{B_0, B(s), 1\}} + \beta E(s), \tag{4}$$

$$E(s) = \text{clip}_{[-1,1]}\left(1 - \frac{1}{|K_0|} \sum_{k \in K_0} \frac{P_k(s)}{P_k(s_0)}\right), \qquad K_0 \neq \varnothing.$$

We use $\alpha = 1.5$, $\beta = 0.1$, and $E = 0$ when K_0 is empty. Quality differences receive credit only when movement identities match and the initial, before-edit, and after-edit evidence is fresh and comparable:

$$D(u, v) = \begin{cases} \Phi_x(v) - \Phi_x(u), & \text{fresh, comparable evidence,} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

This gate prevents changed movement identities, stale observations, or missing verification from being interpreted as geometric progress.

Validity and repair. An ordinary hard-valid edit receives the eligible quality difference, whereas invalid execution and recovery use separate branches. Let Δn count removed minus introduced

violation fingerprints, and let s_{ref} be the preceding saved valid state, or the pre-edit state if no saved reference exists. The applicable edit reward is

$$r_{\text{edit}}(s, s') = \begin{cases} D(s, s'), & \text{ordinary hard-valid edit,} \\ -0.4, & \text{entering an invalid applied state,} \\ -0.4 + \text{clip}_{[-0.2,0.2]}(0.1\Delta n), & \text{remaining in repair,} \\ 0, & \text{return to identical saved valid geometry,} \\ 0.4 + D(s_{\text{ref}}, s'), & \text{repair to a different hard-valid state,} \\ -0.7, & \text{execution rejection.} \end{cases} \tag{6}$$

Tool-format, tool-state, action-contract format, and action-contract semantic errors receive -0.2, -0.1, -1, and -0.8, respectively.

Verified completion. A stopping reward is issued only after executing the fixed activity specification in the current layout. Let $C(s)$ count residual planner clusters, and call a state *unblocked* when it is hard-valid and has no blocking repair. The `Finish` reward is

$$R_{\text{finish}}(s) = \begin{cases} 1, & \text{unblocked, } C(s) = 0, \text{ resolution confirmed,} \\ -0.7, & \text{unblocked, } C(s) > 0, \text{ eligible action family,} \\ 0, & \text{unblocked, } C(s) > 0, \text{ no eligible family,} \\ -1, & \text{otherwise.} \end{cases} \tag{7}$$

A successful stop requires fresh, nonempty movement evidence; an initially problematic case also requires a valid geometry-changing transition. A neutral stop with residual issues is not counted as resolution, and a budget-ending edit receives no synthetic stopping reward. Appendix B.2 provides additional reward-accounting rules and thresholds.

3.5 Training over Complete Tool Cycles

Verified repair curriculum. Training cases begin from rule-screened references that pass hard validity and their cached activity specification. Controlled furniture perturbations introduce circulation or geometry failures, and a case is retained only when the inverse actions execute through the same interface and restore verified success. We define Easy, Medium, and Hard bands by verified reverse sequences of one, two–three, and four–five repair actions, respectively. The configured sampler begins with Easy cases and progressively mixes in Medium and Hard cases, while cached requirements and source-house partitions remain fixed.

Group-relative credit. The learning unit should match the interaction unit: a query becomes useful only through the closing decision and later outcome it supports. Starting from GiGPO (Feng et al., 2025), we adapt group-relative learning by forming one sample per complete tool cycle, comparing continuation returns at repeated layout-and-verification anchors, and combining this local comparison with a rollout-group episode comparison. Rollouts in a group share the same prepared case, cached activity specification, and policy version.

For episode i and cycle t, we compute the discounted continuation return and undiscounted episode score

$$G_{i,t} = \sum_{k=t}^{T_i-1} \gamma^{k-t} r_{i,k}, \qquad R_i = \sum_{t=0}^{T_i-1} r_{i,t}. \tag{8}$$

The continuation return distinguishes when consequences occur, whereas the episode score records the outcome of the complete trajectory.

We retain GiGPO's two-level relative comparison on the materialized cycle samples:

$$A_{i,t} = N_{\mathcal{H}_{x,h_{i,t}}}(G_{i,t}) + \lambda N_{\mathcal{G}_x}(R_i). \tag{9}$$

Here, $\mathcal{H}_{x,h}$ groups samples with the same state hash, which covers the case definition, current and saved-valid geometry, validity and route freshness, objectives, target policy, and visual contract; remaining budgets are recorded separately rather than hashed into h. $\mathcal{G}_x$ contains the cycle samples in the current same-case rollout group, so an episode score is repeated over its cycle rows. N subtracts the group mean and divides by the sample standard deviation plus 10^{-6}, returning zero for singleton or near-constant groups. We set the anchor-level weight to one, $\lambda = 0.25$, and $\gamma = 0.9$.

The anchor term compares decisions from the same hashed layout-and-verification context, whereas the rollout-group term retains their episode-level consequences; equal episode scores can therefore leave useful continuation-return differences.

For token alignment, r_t is placed on the final assistant token of the materialized sample and $A_{i,t}$ supervises its masked assistant outputs. Observations and tool replies remain context and are excluded from target tokens. Exact cycle alignment retains all assistant responses; a verified fallback retains only the closing response, and failure of both alignment paths invalidates the group. Invalid reward or infrastructure evidence also rejects a group, while valid zero-signal groups contribute no task gradient. ROLL (Wang et al., 2025) optimizes the remaining samples with the clipped policy objective (Schulman et al., 2017), averaging masked losses within each sample and then across samples.

4 EVALUATION

We evaluate whether LIVINGFLOW improves circulation in existing furnished rooms, how this relates to layout quality, and which design choices contribute to the improvement. Main comparisons assess complete refinement systems; ablations examine activity guidance, sequential interaction, and task-specific RL.

4.1 EXPERIMENTAL SETUP

Data. We use 1,210 original furnished rooms from 3D-FRONT, with disjoint source houses across 950 training, 70 validation, and 190 test rooms. The train/validation/test splits are 250/13/49 for Living Room, 300/23/50 for Living–Dining Room, and 400/34/91 for Bedroom. Bedroom pools master, secondary, generic, children's, elderly, and nanny rooms. Test inputs are the original furniture arrangements, including their existing geometric and circulation problems. All methods receive the same architecture, furniture inventory, and initial poses for each room.

Baselines and model configuration. We compare with DiffuScene (Tang et al., 2024), LEGO-Net (Wei et al., 2023), InstructScene (Lin & Mu, 2024), LayoutVLM (Sun et al., 2025), SceneReVis (Zhao et al., 2026), and ReSpace (Bucher & Armeni, 2025), adapting their editing procedures to fixed architecture and furniture inventory. External baselines retain their native guidance and do not receive our activity specifications or intermediate circulation feedback; their saved outputs are scored on the same fixed evaluation movements. Within LIVINGFLOW, a frozen GPT model generates activity specifications, which are cached before editing; the furniture-editing policy is Qwen3.8-27B. The *w/o RL* control retains the same initialization, activities, prompts, observations, tools, inference budgets, and completion rules, without task-specific policy updates. Section D.2 provides adaptation details.

Evaluation metrics. Geometric layout quality is measured by out-of-boundary rate (OBR) and collision rate (CNR). OBR is the percentage of furniture objects extending beyond the room boundary; CNR is the percentage involved in furniture–furniture collisions beyond the geometric tolerance. Both are computed within rooms and averaged equally across rooms. To assess activity support, we report movement pass rate (MPR), success rate (SR), and Valid. A movement passes when its required access selections are normal and its route is feasible; missing endpoints, fallback or blocked access, and unavailable routes count as failures. MPR averages the fraction of passing movements within each room. SR is the percentage of rooms in which all evaluated movements pass and all hard geometric constraints are satisfied; Valid checks only those geometric constraints. All metrics are percentages. Lower OBR and CNR are better; higher values are better for the other metrics.

4.2 LAYOUT AND CIRCULATION REFINEMENT

Table 1 evaluates geometric layout quality and executable activity support on the same returned layouts across three room categories.

Circulation and geometric repair. LIVINGFLOW achieves the highest MPR, SR, and Valid in all three room categories, together with the lowest OBR and CNR. Relative to the input, MPR improves by 26.99 percentage points in Living Room, 29.69 in Living–Dining Room, and 43.64 in Bedroom. The benefit therefore extends beyond removing placement violations: more of the movements con-

Table 1: **Layout and circulation refinement across three room categories.** Results are reported on 190 test rooms from 3D-FRONT. OBR and CNR measure boundary and collision violations; MPR and SR measure activity support, and Valid reports geometric validity. All values are percentages; arrows indicate the preferred direction, and the best values in each column are bold.

	Living Room					Living–Dining Room					Bedroom				
Method	OBR ↓	CNR ↓	MPR ↑	SR ↑	Valid ↑	OBR ↓	CNR ↓	MPR ↑	SR ↑	Valid ↑	OBR ↓	CNR ↓	MPR ↑	SR ↑	Valid ↑
Input layout	10.38	20.17	33.34	2.04	20.41	6.24	19.09	45.04	6.00	32.00	23.96	27.13	30.50	8.79	20.88
DiffuScene	45.27	36.94	18.27	0.00	2.04	46.35	44.36	19.92	0.00	0.00	65.21	32.38	14.68	0.00	1.10
LEGO-Net	16.10	67.22	12.75	0.00	0.00	17.18	66.08	16.55	2.00	6.00	43.74	31.61	34.89	3.30	6.59
InstructScene	49.01	31.74	16.64	0.00	4.08	51.32	40.32	22.49	2.00	4.00	63.70	36.51	9.88	0.00	0.00
LayoutVLM	11.38	30.67	18.53	2.04	4.08	13.84	37.27	20.39	0.00	0.00	21.00	44.65	9.22	0.00	4.40
SceneReVis	2.65	56.69	27.54	0.00	6.12	3.13	56.69	30.48	6.00	8.00	11.13	67.16	13.01	1.10	6.59
ReSpace	17.78	48.41	22.76	0.00	0.00	12.34	36.56	33.22	4.00	14.00	45.23	36.75	24.10	2.20	4.40
LivingFlow	**2.42**	**11.37**	**60.33**	**16.33**	**83.67**	**1.32**	**15.23**	**74.73**	**24.00**	**92.00**	**0.44**	**2.84**	**74.14**	**41.76**	**97.80**

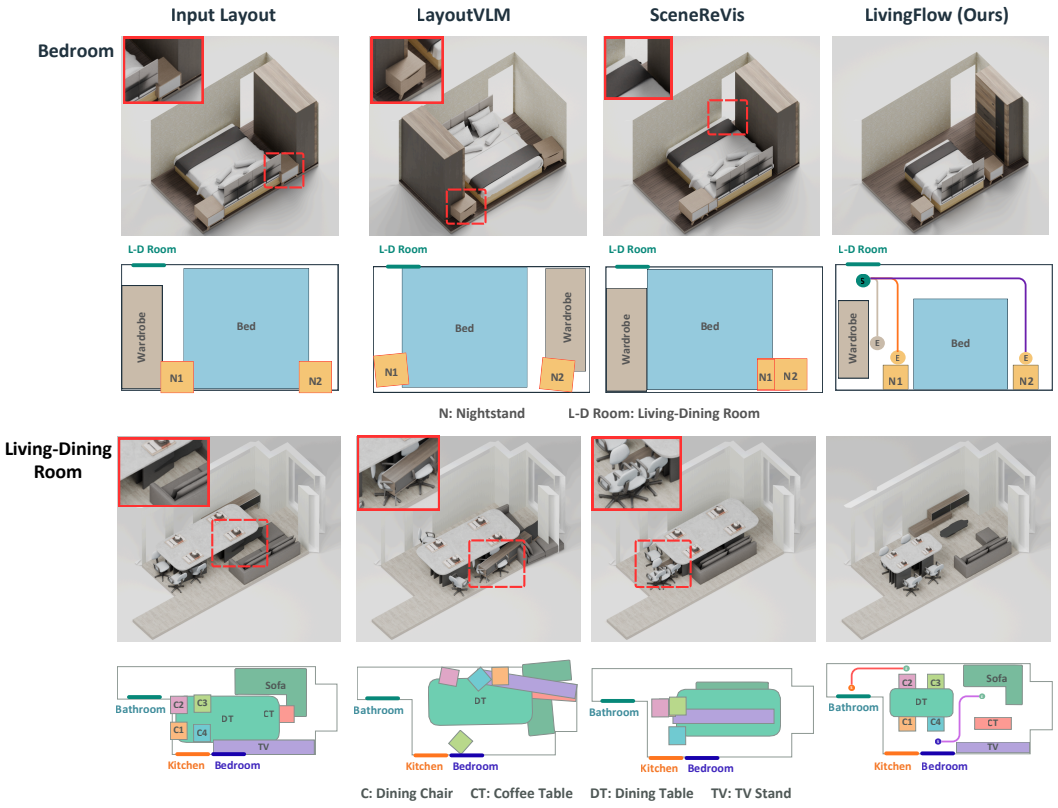

Figure 3: **Qualitative layout comparisons on the 3D-FRONT test set.** Within each case, methods share the same input layout, viewpoint, scale, and lighting. Red boxes highlight furniture overlaps and obstructed access in the input and baseline outputs. LIVINGFLOW reduces these conflicts and improves access and circulation through furniture rearrangement, as shown in the plan views.

necting furniture uses become executable. For example, ReSpace attains the strongest external MPR in Living–Dining Room at 33.22%, compared with 74.73% for LIVINGFLOW.

Complete activity support. Complete activity support remains more demanding than geometric validity: in Bedroom, 97.80% of outputs satisfy the hard geometric constraints, but only 41.76% also pass every evaluated movement; SR is lower still in the other categories. Thus, many remaining failures occur in otherwise valid layouts, motivating checks of the full activity set after each edit.

Figure 3 illustrates how furniture refinement changes access and activity routes in two selected cases, using matched views and evaluated movements.

4.3 Ablation Study

Table 2 examines the objective and interaction used to learn refinement. *Random movements* replaces activity-derived tasks with cached, count-matched object-use pairs; *w/o Circulation* removes activity conditioning, route feedback, and circulation rewards. *One-shot actions* emits all edits in one response, whereas *One-shot coordinates* emits the final layout; both use RL and activity specifications without intermediate feedback. *w/o RL* isolates task-specific training under the full inference interface. Every variant is scored on the same fixed evaluation tasks; Section D.3 gives the detailed controls.

Table 2: **Ablations of activity guidance, sequential interaction, and policy training.** All variants are scored on a shared set of test movements. Results are percentages, with higher values indicating better performance; the best values in each column are bold.

	Living Room			Living–Dining Room			Bedroom		
Variant	MPR ↑	SR ↑	Valid ↑	MPR ↑	SR ↑	Valid ↑	MPR ↑	SR ↑	Valid ↑
Random movements	57.18	12.24	63.27	69.43	18.00	76.00	63.44	31.87	75.82
w/o Circulation	57.03	**16.33**	67.35	63.50	12.00	76.00	54.74	35.16	71.43
One-shot actions	37.47	6.12	36.73	50.17	8.00	40.00	31.64	9.89	23.08
One-shot coordinates	35.68	2.04	26.53	46.53	6.00	36.00	30.40	8.79	23.08
w/o RL	57.36	10.20	67.35	65.34	16.00	74.00	60.32	29.67	67.03
LivingFlow	**60.33**	**16.33**	**83.67**	**74.73**	**24.00**	**92.00**	**74.14**	**41.76**	**97.80**

Activity-aligned objectives. The choice of movement targets matters beyond their number. Compared with Random movements, LivingFlow improves MPR by 3.15, 5.30, and 10.70 percentage points in Living Room, Living–Dining Room, and Bedroom, respectively. Removing circulation also lowers MPR and Valid in every category, although Living Room SR ties at 16.33%. Geometric repair can therefore benefit some activities without explicit circulation guidance, but does not recover the full movement-level gain.

Sequential interaction. One-shot actions improves on direct coordinates in eight of nine entries, but both fall well below the complete interaction framework. In Bedroom, one-shot actions reaches 31.64% MPR and 23.08% Valid, compared with 74.14% and 97.80% for LivingFlow. Representing refinement as executable actions is therefore insufficient to reproduce the full result when all decisions precede feedback. These results support sequential refinement with intermediate execution feedback over one-shot prediction.

Task-specific policy learning. RL improves all three activity and validity metrics under the same inference interface. MPR gains are 2.97 percentage points in Living Room, 9.39 in Living–Dining Room, and 13.82 in Bedroom. On Bedroom, SR rises from 29.67% to 41.76% and Valid from 67.03% to 97.80%. Thus, the tools and feedback already support substantial refinement before training, while policy updates improve how successfully that interface is used.

5 Conclusion

We present LivingFlow, an agentic reinforcement learning framework for refining existing furniture layouts to improve activity-driven circulation. Fixed activity specifications define movement requirements, while a geometric executor re-evaluates access and routes as furniture changes. Training over complete tool cycles connects information gathering and editing decisions with execution feedback. Experiments across three room categories show that LivingFlow improves geometric validity and supports more required movements than the compared methods. Ablations further demonstrate the benefits of activity guidance, sequential interaction, and task-specific policy training. These findings highlight the importance of evaluating furniture arrangements through the movements required by everyday activities, beyond collision avoidance alone. Activity-driven circulation provides a concrete link between how a room is used and how its layout should be refined.

DISCLOSURE OF AI USE

We used generative AI tools to assist with translation and interpretation of experimental results, and to generate the activity specifications described in the method section. We also used these tools for literature searches, reference checking, language revision, and preliminary figure drafts. The authors wrote the initial manuscript drafts, conducted all experiments, and performed the resulting analyses, with AI providing supplementary analytical assistance. All AI-generated figure drafts were manually redrawn by the authors. The authors reviewed all AI-assisted work and checked, revised, and edited AI-generated text sentence by sentence. We take responsibility for the final content of this work, including text, claims, and artifacts produced with AI assistance.

REFERENCES

Martin JJ Bucher and Iro Armeni. ReSpace: Text-driven autoregressive 3d indoor scene synthesis and editing. *arXiv preprint arXiv:2506.02459*, 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 Workshops (2024)*, pp. 217–234, 2025. Proceedings published in 2025.

Lingwei Dang, Ziyan Qiu, Jiajia Cheng, Shishuo Shang, Zhenhao Zhang, Yufei Zhu, Qingxin Xiao, Pan Liu, Shenghui Huang, Yun Hao, Juntong Li, and Qingyao Wu. Roomer: Reflective object-grounded model editing and repair for 3D indoor layout synthesis. arXiv preprint arXiv:2608.01973, 2026.

Hao Feng, Zhi Zuo, MingJian Liang, Jingyu Hu, Xiaowei Hu, Liupengfei Wu, Dian Zhang, Guoxin Fang, and Zhengzhe Liu. FuncRoom-Agent: Sequential feed-forward 3d functional indoor scene generation. arXiv preprint arXiv:2608.29519, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems (2025)*, 2025.

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

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics*, 31(6):135:1–135:11, 2012.

Matthew Fisher, Manolis Savva, Yangyan Li, Pat Hanrahan, and Matthias Nießner. Activity-centric scene synthesis for functional 3D scene modeling. *ACM Transactions on Graphics*, 34(6):179:1–179:13, 2015.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3D-FRONT: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (2021)*, pp. 10933–10942, 2021.

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. In *International Conference on Learning Representations (2022)*, 2022.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do As I Can, Not As I Say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning (2022)*, pp. 287–318, 2023. Proceedings published in 2023.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (2023)*, 2023.

Chenguo Lin and Yadong Mu. InstructScene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. In *International Conference on Learning Representations (2024)*, 2024.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Transactions on Graphics*, 30(4):87:1–87:10, 2011.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (1999)*, pp. 278–287, 1999.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (2023)*, 2023.

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)*, 2021.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. SceneSmith: Agentic generation of simulation-ready indoor scenes. In *Proceedings of the International Conference on Machine Learning (2026)*, 2026.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. VirtualHome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018)*, pp. 8494–8502, 2018.

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)*, pp. 6182–6190, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (2023)*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

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. arXiv preprint arXiv:2402.03300, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (2023)*, 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2025)*, pp. 29469–29478, 2025.

Jia-Mu Sun, Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas, and Lin Gao. HAISOR: Human-aware indoor scene optimization via deep reinforcement learning. *ACM Transactions on Graphics*, 43(2):15:1–15:17, 2024.

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)*, pp. 20507–20518, 2024.

Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics*, 37(4):70:1–70:14, 2018.

Ruiqi Wang, Qimin Chen, Daniel Ritchie, Angel X. Chang, Manolis Savva, Kai Wang, and Hao Zhang. Function2Scene: 3d indoor scene layout from functional specifications. arXiv preprint arXiv:2605.30819, 2026.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, Zichen Liu, Haizhou Zhao, Dakai An, Lunxi Cao, Qiyang Cao, Wanxi Deng, Feilei Du, Yiliang Gu, Jiahe Li, Xiang Li, Mingjie Liu, Yijia Luo, Zihe Liu, Yadao Wang, Pei Wang, Tianyuan Wu, Yanan Wu, Yuheng Zhao, Shuaibing Zhao, Jin Yang, Siran Yang, Yingshui Tan, Huimin Yi, Yuchi Xu, Yujin Yuan, Xingyao Zhang, Lin Qu, Wenbo Su, Wei Wang, Jiamang Wang, and Bo Zheng. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. arXiv preprint arXiv:2506.06122, 2025.

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

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. SAGE: Scalable agentic 3d scene generation for embodied AI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2026)*, pp. 22358–22368, 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 (2024)*, pp. 16262–16272, 2024a.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. SceneWeaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent. In *Advances in Neural Information Processing Systems (2025)*, 2025.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3d embodied AI environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)*, pp. 16227–16237, 2024b.

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)*, 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 (2023)*, pp. 12965–12976, 2023.

Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: Automatic optimization of furniture arrangement. *ACM Transactions on Graphics*, 30(4):86:1–86:12, 2011.

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.

The supplementary material expands the execution interface, training procedure, theoretical analysis, and evaluation protocol:

- **System Architecture (Section A).** Describes the scene interface, fixed activity specifications, geometric execution, and policy observations.
- **Training Data and Agentic RL (Section B).** Details rollout sampling, reward accounting, complete-cycle learning, and configuration.
- **Reward and Utility Analysis (Section C).** Decomposes discounted progress returns and gives a conditional bound relating a fixed activity proxy to target utility.
- **Evaluation Methodology (Section D).** Specifies the evaluation cohorts, metrics, baseline adaptations, and ablation conditions.
- **Training and Behavior Analyses (Section E).** Defines the analyses of activity structure, editing steps, and reward components.
- **Qualitative Refinement Results (Section F).** Organizes consecutive editing traces around activity requirements and execution feedback.
- **Prompt Templates (Section G).** Reproduces the activity-generation and refinement-policy instructions, with the model-facing constraints.

A System Architecture

The execution interface translates fixed activity requirements and furniture edits into geometric feedback. This section describes the shared scene representation, access and route validation, and the evidence presented to the policy. Action contracts follow the templates in Section G.

A.1 Native Scene Representation and Editing Tools

Scene representation and editing share the same object identities and geometry so that each proposed change can be checked against the current layout. Room boundaries and furniture footprints are polygons accompanied by opening records, reviewed uses, and confirmed table–chair associations. Per-object metadata defines the action pivot, local axes, and eligible action families.

The policy queries `get_action_parameter_options` to inspect a target's contract or register exact parameters. Move specifies the absolute world coordinates of the action pivot. Initially orthogonal furniture uses quarter-turn rotations; initially non-cardinal furniture retains a directed target-angle contract. Proportional Resize accepts one target horizontal dimension and derives the other from the reference aspect ratio, subject to both category ranges; catalog candidates provide a fallback where configured. Resizing preserves height and pivot, and local axes retain their identities after rotation.

Confirmed paired chairs remain separate editable targets and do not automatically follow a moved table. Their associations persist across edits. Decision records accompany queries, actions, and stopping decisions with the observed problem, intended effect, alternatives, and subsequent verification; these summaries support inspection without certifying causal faithfulness.

A.2 Activity Specification and Caching

A cached specification provides the same activity requirements throughout refinement and editor training. Preparation checks the generated schema, endpoint references, declared uses, and continuity between consecutive movement tasks before caching the result. The generation record identifies the prompt composition, model, and overrides; the supplied preparation configuration names GPT-5.5, uses a 300-second request timeout and up to three preparation tool rounds, and leaves temperature and the completion-token limit unspecified unless overridden.

Episode reset validates the cached specification together with its structural and navigation settings. Artifact hashes identify the fixed requirements, while model-request hashes depend on cache configuration. Re-evaluation carries previous access selections as continuity context and preserves them when eligible. The generator specifies requirements; a language-model reward judge is disabled in this configuration.

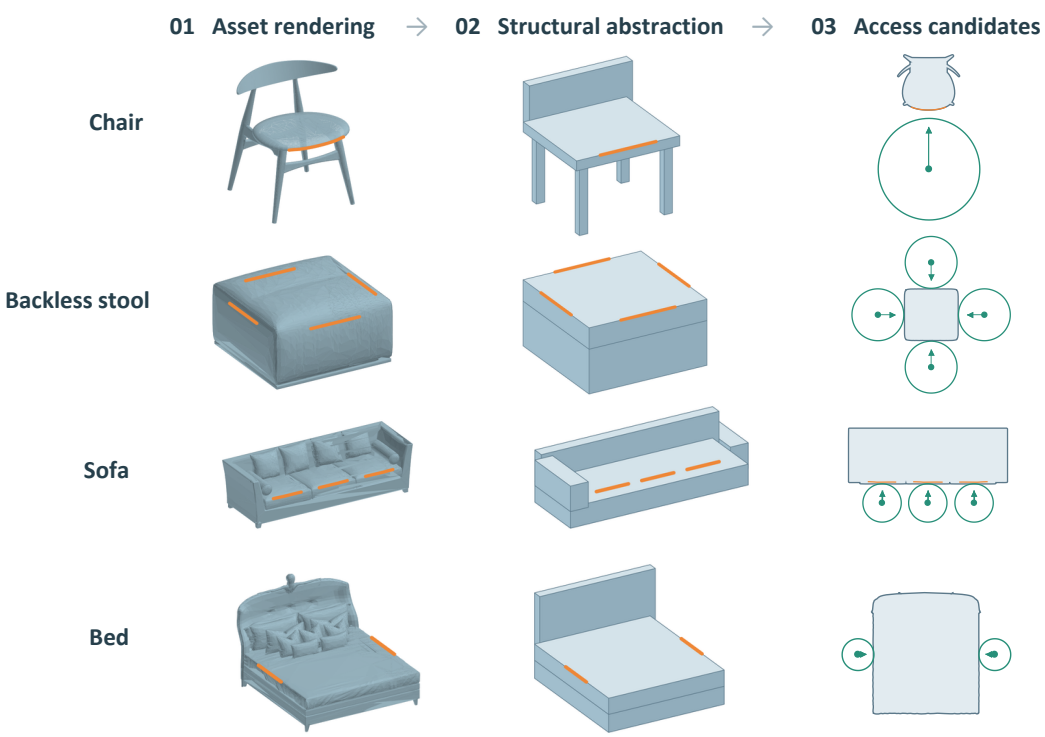

Figure 4: **From furniture uses to access candidates.** Examples of a chair, backless stool, sofa, and bed show furniture models, use boundaries, and candidate access locations. Orange marks indicate use boundaries; green points and dashed circles indicate candidate positions and illustrative body footprints. The stool provides alternative approaches to the same use. These schematic geometries illustrate candidate construction; room-level reachability is assessed separately.

A.3 Geometric Grounding and Validation

Geometric grounding converts each specified object use into access candidates that can serve as movement endpoints. Candidates are constructed from the reviewed use boundary and its anchor, with an outward offset adjusted when necessary to clear the object's own geometry. Selection preserves an eligible previous endpoint before applying deterministic distance and identifier tie-breaking, maintaining continuity within each participant's activity chain. Figure 4 illustrates the construction of access candidates from furniture use surfaces. Alternative candidates for the same use do not all need to be accessible; the executor selects an eligible candidate in the current layout.

Route validation first erodes the room and inflates obstacles by the clearance radius. A grid cell is walkable only when its complete extent lies in the resulting free space. Four-neighbor A* searches deterministic endpoint-connector pairs, and a returned route must pass an exact polyline-clearance check. The initial grid uses the configured cell size. A verified route supplies positive evidence under the planar model; a failed search retains the limitations of the final search resolution, which must be recorded if a finer-grid recheck is enabled. Missing executable endpoints or certified connectors produce an unavailable route status; disconnected eligible connectors produce an unreachable status. Neither status counts as a passing movement task. A missing reviewed access pool is an artifact-integrity error rather than an ordinary route failure.

Hard checks cover containment, overlap, and opening clearances under common tolerances. Door-related movement tasks end at room-side arrival points derived from structural geometry. For a sliding door, all objects must leave one common clear half-width interval along the opening; this interval can shift along the opening but cannot be selected independently for different obstacles. Certified-width fallbacks are recorded when the full opening width is unavailable. Navigation omits only the chair footprints with accepted table associations and recomputes their access candidates on the nearest exterior side of the associated table relative to each chair's current position. This

abstraction does not establish physical nesting or remove collision diagnostics. The planar checks also exclude designated decorative or overhead context and do not certify door-leaf motion, mesh-level contact, or full-body interaction.

Table 3: **Native action and geometric settings.** Distances are in centimetres unless stated otherwise.

Setting	Value
Move target	Absolute action-pivot coordinates in world XY
Rotation	Quarter turns for initially cardinal poses; otherwise directed angles in $[0, 360^\circ)$
Resize	Proportional target within both category ranges; offered catalog fallback where configured
Standing-access initial offset	30; adjusted for clearance from the object's own geometry
Passage width / initial grid resolution	60 / 5
Containment and overlap area tolerance	1 cm^2
Entry / ordinary-door keepout depth	110 / 80
Sliding-door keepout	One common clear half-width interval along the opening, depth 60

A.4 Policy Observations

The policy receives a compact representation of the current layout and the evidence needed to choose its next query, edit, or stopping decision. Table 4 groups these fields by function. Read-only tool replies extend the current cycle, and the multimodal configuration additionally supplies the current plan view. Complete polygons and source audits remain in the execution record rather than being repeated in every response.

Table 4: **Observation content and its role in refinement.**

Content	Information available to the policy
Current state	Geometric validity, route status, evidence freshness and validation coverage
Objectives	Active problem clusters, blocking repairs and access or route witnesses
Layout and actions	Object identities, openings, dimensions, pivots, table–chair associations and eligible action families
Transition and history	Previous execution feedback, compact action/layout history and repeated failures
Interaction budget	Remaining response and tool opportunities; visual-observation context
Repair context	Current violations, repair status and the preceding valid reference

B Training Data and Agentic RL

Training converts refinement episodes from the training split into complete-cycle policy samples. This section specifies case grouping, reward accounting, token alignment, and the configuration needed to reproduce the updates.

B.1 Case Sampling and Rollout Groups

Rollout groups provide comparable episodes for the relative updates in Section 3.5. Each rollout group starts from the same original furnished 3D-FRONT layout and cached activity specification, and is collected under a single policy version. Cases come from the training split described in Section 4.1. The manifest identifies the available cases; sampling records separately report collected episodes and those accepted for optimization, including groups rejected before an update.

B.2 Reward Accounting and Parameters

Reward accounting determines which execution changes contribute to the main-text reward branches in Eqs. (4)–(7). Movement-level issues and included problem clusters each contribute burden units. Clusters are matched to initial objectives through movement, affected-object, and blocker identities; newly surfaced clusters are excluded only when comparable route evidence establishes no material regression. Route loss, newly insufficient width, and threshold-level diagnostic changes enter this accounting.

Applied-invalid states retain route evidence, but quality credit and geometric recovery payment remain separately gated; an unavailable quality increment is distinct from invalid reward evidence. The potential is bounded by $[-(\alpha+\beta), \alpha+\beta]$, and ordinary quality rewards are rounded to six decimals.

Queries and permitted format retries consume response slots; edit attempts and semantic failures also consume action slots. A format failure can remain within the current cycle while budgets permit, whereas a semantic action failure closes it. A verified solved edit on the last action slot retains its edit reward without an additional Finish payment; other exhausted action budgets truncate. Unknown executor validity or infrastructure failures invalidate the associated training group, while valid negative outcomes remain usable evidence.

Table 5: **Additional reward thresholds.** Reward weights and branch conditions are given in the main text.

Parameter	Value	Role
Mixed-signal cutoff	10^{-9}	Nonzero mixed signal when $\lvert A_{i,t}\rvert \geq 10^{-9}$
Length/width regression	15 cm	Material diagnostic change, including equality
Turn regression	1	Material increase in turns

B.3 Complete-Cycle Policy Optimization

The learning interface determines which interactions share credit and which cycle samples are compared. Returns and advantages follow Eqs. (8) and (9). Each new cycle receives compact editing history. Beyond the layout and verification fields in Section 3.5, the anchor includes objective and observation-contract context but excludes remaining budgets and explicit access-selection history. Matching anchors therefore does not establish identical full interaction states.

For episode i, cycle t, and token position ℓ, let $y_{i,t,\ell}$ be a generated token, $c_{i,t,\ell}$ its conditioning prefix, and $M_{i,t,\ell}$ an indicator of whether it is an assistant token included in the loss. ROLL (Wang et al., 2025) applies the clipped policy objective (Schulman et al., 2017):

$$\rho_{i,t,\ell} = \frac{\pi_\theta(y_{i,t,\ell} \mid c_{i,t,\ell})}{\pi_{\theta_{\text{old}}}(y_{i,t,\ell} \mid c_{i,t,\ell})},$$
$$J(\theta) = \text{Agg}_{i,t,\ell}[M_{i,t,\ell} \min\{\rho_{i,t,\ell}A_{i,t}, \text{clip}(\rho_{i,t,\ell}, 1-\epsilon, 1+\epsilon)A_{i,t}\}]. \tag{10}$$

Here $\rho_{i,t,\ell}$ is the new-to-rollout policy probability ratio, θ_{old} denotes the rollout policy parameters, ϵ is the ratio-clipping threshold, and $A_{i,t}$ is the cycle advantage. The operator Agg averages over included assistant tokens within each sample and then equally across samples. Observations and tool replies supply context but are excluded from the assistant mask. Cycle reward is attached to the last assistant token; if exact alignment of all cycle outputs fails, the implementation falls back to the final response with its original prompt and token IDs, retaining earlier exchanges only as context. Failure of that fallback invalidates the group.

Valid zero-signal groups provide no task gradient; their retention or resampling follows the resolved configuration. The audited optimizer settings use token discount 1, ratio clipping 0.2, reward/advantage clipping at 10, and no reference-KL or entropy term. We use a cycle discount of $\gamma = 0.9$ and an episode-level advantage weight of $\lambda = 0.25$, matching Section 3.5.

Table 6: **Training and inference settings.**

Setting	Configuration
Editor initialization	Qwen/Qwen3.8-27B
Attention LoRA rank / alpha / dropout	32 / 64 / 0
Policy optimization	Complete-cycle GiGPO adaptation
Learning rate	10^{-5}
Optimization update budget / rollout-group size	20 / 8
Action / response budgets per episode	15 / 60
Tool-call budget per decision cycle	6
Response / context token limits	4,096 / 65,536
Evaluation decoding	Greedy; temperature 0; thinking disabled
Training accelerators	32 GPUs (16 training, 16 rollout)
Evaluation accelerators	32 GPUs (8 TP4 replicas)

B.4 Training and Inference Configuration

The trained and untrained policies use the same inference interface so that their comparison isolates task-specific updates. Table 6 summarizes the training and inference configuration, including attention LoRA (Hu et al., 2022). Action and response budgets apply to an entire episode; the tool-call budget resets with each decision cycle.

C Reward and Utility Analysis

The analysis addresses two questions: how progress rewards value a sequence of edits, and when improvement under fixed activity requirements transfers to target utility. We first decompose discounted return, then give a conditional transfer bound separating coverage, evaluation, and estimation errors. The activity-selection counterexample is presented in Section 3.1.

C.1 Progress Timing and Terminal Quality

Two editing sequences can reach the same final quality through different intermediate layouts. To describe how their timing enters the return, let T be the number of completed cycles, γ the cycle discount, Φ_t the potential at cycle boundary t, and r_t the implemented cycle reward. Define the residual $c_t = r_t - (\Phi_{t+1} - \Phi_t)$, which accounts for the reward not represented by this potential difference.

Proposition 2 (Discounted difference-return decomposition)**.** *For $r_t = \Phi_{t+1} - \Phi_t + c_t$, $0 < \gamma < 1$, and integer $T \geq 1$,*

$$\begin{aligned}\sum_{t=0}^{T-1} \gamma^t r_t = & -\Phi_0 + (1-\gamma) \sum_{t=1}^{T-1} \gamma^{t-1} \Phi_t \\ & + \gamma^{T-1} \Phi_T + \sum_{t=0}^{T-1} \gamma^t c_t.\end{aligned} \tag{11}$$

Proof. Reindexing the discounted potential differences gives

$$\sum_{t=0}^{T-1} \gamma^t (\Phi_{t+1} - \Phi_t) = \sum_{t=1}^{T} \gamma^{t-1} \Phi_t - \sum_{t=0}^{T-1} \gamma^t \Phi_t.$$

Collect the matching indices and add $\sum_{t=0}^{T-1} \gamma^t c_t$ to obtain Eq. (11). □

For this decomposition, a state without comparable quality evidence is assigned $\Phi_t = 0$, and the residual absorbs its difference from the implemented reward. This is an analytical convention, not an additional reward rule. For $0 < \gamma < 1$, the intermediate and terminal potential coefficients are positive and sum to one. The potential component therefore depends on when quality improves as well as on its final value. Comparisons of complete returns must also account for the residual terms; equal final quality alone does not determine which sequence has greater return.

The residual includes query and retry costs, rejection penalties, repair rewards, and completion payments whenever they apply. An applied invalid edit requires subtracting its potential change when defining the residual. A `Finish` cycle after K preceding cycles has weight γ^K, whereas a budget-ending edit receives no additional `Finish` payment. At $\gamma = 1$, the return reduces to $\Phi_T - \Phi_0 + \sum_t c_t$. For bounded potentials and $0 < \gamma < 1$, the terminal potential term tends to zero as $T \to \infty$.

The implemented difference reward differs from the discount-matched shaping term $\gamma\Phi(s') - \Phi(s)$ (Ng et al., 1999), so the policy-invariance guarantee of potential-based shaping does not follow from this identity. Moreover, LIVINGFLOW combines discounted continuation returns with an undiscounted episode signal. The proposition describes the continuation-return component; it does not establish a learning order or improved final layout quality.

C.2 Fixed Activities and Target Utility

A fixed activity specification makes successive scores comparable, but the score remains a proxy for the activities a room should support. We characterize when an improvement in this proxy also improves target utility.

For a fixed room, let p be the target distribution over activity requirements z, and let q be a fixed proxy distribution. Let $u(S, z) \in [0, 1]$ denote target support for requirement z in layout S, and let $\widehat{u}(S, z) \in [0, 1]$ denote its geometric assessment. Define

$$\mathcal{U}(S) = \mathbb{E}_{z\sim p}[u(S, z)], \qquad Q(S) = \mathbb{E}_{z\sim q}[\widehat{u}(S, z)], \tag{12}$$

where $\mathcal{U}$ is target utility and Q is the population proxy score. Let $\widehat{Q}(S)$ estimate $Q(S)$ from the cached finite specification. This proxy is distinct from the training potential Φ and the episode return.

Proposition 3 (Conditional transfer from a fixed activity proxy)**.** *Let $\mathcal{D}$ be the set of layouts under consideration. Suppose the coverage discrepancy satisfies $\mathrm{TV}(p, q) = \sup_A |p(A) - q(A)| \le \delta$, where δ bounds the distribution mismatch. Suppose also that, uniformly for all $S \in \mathcal{D}$,*

$$|\mathbb{E}_{z\sim q}[u(S, z) - \widehat{u}(S, z)]| \le \epsilon_g, \tag{13}$$

$$|Q(S) - \widehat{Q}(S)| \le \epsilon_s, \tag{14}$$

where ϵ_g bounds geometric-assessment error and ϵ_s bounds finite-specification estimation error. Then, for any $S_0, S_T \in \mathcal{D}$,

$$\mathcal{U}(S_T) - \mathcal{U}(S_0) \ge \widehat{Q}(S_T) - \widehat{Q}(S_0) - 2(\delta + \epsilon_g + \epsilon_s). \tag{15}$$

Proof. Since $u(S, z) \in [0, 1]$, the total-variation bound implies $|\mathbb{E}_p[u(S, z)] - \mathbb{E}_q[u(S, z)]| \le \delta$. Combining this inequality with Eqs. (13) and (14) gives $|\mathcal{U}(S) - \widehat{Q}(S)| \le \delta + \epsilon_g + \epsilon_s$ for every $S \in \mathcal{D}$. Apply the lower bound to S_T and the upper bound to S_0, then subtract to obtain Eq. (15). □

Thus a proxy improvement exceeding $2(\delta + \epsilon_g + \epsilon_s)$ implies positive target-utility improvement under these assumptions. Uniformity over $\mathcal{D}$ permits the editor to choose its final layout adaptively using the cached specification. Freezing the generator and caching requirements do not themselves establish the three error bounds, which have not been estimated on the evaluation cohort. The proposition concerns improvement in $\widehat{Q}$, and does not imply that an increase in training reward alone improves target utility.

D Evaluation Methodology

This section specifies how the main comparisons and ablations use shared inputs, score saved outputs, and handle failures. It supplements the metric definitions in Section 4.1 without repeating the main result tables.

D.1 Evaluation Cohort and Metrics

Evaluation compares saved outputs for matched source rooms. The following conventions specify aggregation, activity references, and failure handling.

Shared inputs and aggregation. The evaluation cohort is fixed before inspecting outcomes. Each room contributes equally to its room-type aggregate; when repeated seeds are available, their scores are first averaged within the room. Repeated episodes do not increase the held-out test-room count.

Layout quality. OBR and CNR follow the per-room object-percentage definitions in Section 4.1, using common footprints, contact conventions, and tolerances.

Activity sets and circulation. Main-comparison circulation scores evaluate each room's fixed specified activities. All variants within an ablation comparison are scored on common fixed evaluation activities. MPR averages the fraction of passing movements within each room, with equal weight across rooms; it includes all evaluated movement tasks and all rooms, including tasks that already pass initially. A movement passes only with a feasible route and normal access selections. SR requires every evaluated movement to pass and the returned layout to be hard-valid.

Missing endpoints, fallback selections, and blocked selections remain movement failures. Neither initially passing rooms nor failed requests and exhausted budgets are removed from the cohort. The returned layout is scored under the recorded adapter rule, including any declared input-layout fallback, and retry counts are retained. Valid applies the common hard checks independently of movement outcomes.

D.2 Baseline Configurations

We compare with DiffuScene (Tang et al., 2024), LEGO-Net (Wei et al., 2023), InstructScene (Lin & Mu, 2024), LayoutVLM (Sun et al., 2025), SceneReVis (Zhao et al., 2026), and ReSpace (Bucher & Armeni, 2025). Each receives the supplied 3D-FRONT layout under fixed architecture and furniture inventory. The comparison retains each method's applicable editing mechanism and scores its returned layout with the common evaluator, without supplying our activity specifications or intermediate circulation feedback. These results compare complete systems under the shared refinement task.

DiffuScene's rearrangement mode and LayoutVLM's supplied-pose optimization follow the refinement setting of SceneReVis (Zhao et al., 2026). DiffuScene performs no completion; LayoutVLM retains its relation and physical constraints. SceneReVis retains its visual and geometric feedback while disabling insertion, deletion, replacement, and pruning.

The matched *w/o RL* control uses the same Qwen3.8-27B editor initialization, activity specifications, observations, prompts, tools, inference budgets, and completion rules as LivingFlow, with task-specific policy updates removed. The GPT-based preparation model generates activity specifications and their movement requirements; it does not edit furniture. The Qwen3.8-27B policy performs furniture refinement, and the trained policy produces the LivingFlow trajectories used for case studies. The GPT-5.5 activity-generation configuration is described in Section A.2.

D.3 Ablation Configurations

The ablations separate activity selection, circulation guidance, sequential interaction, and task-specific training. The six conditions in Section 4.3 share prepared cases, editor initialization, held-out inputs, allowed attributes, and declared edit limits. The comparison matches edit limits rather than realized inference cost.

Random movements samples schema-valid ordered object-use pairs without replacement and matches their count to the activity-specified movements. Identities and seeds stay fixed while geometry is rebound after edits; blocked pairs are not resampled, and insufficient candidate pools are recorded. Circulation feedback, rewards, and complete-cycle interaction remain available.

The w/o Circulation condition removes activity specifications, movement targets, route feedback, and circulation-dependent rewards during training and inference. It retains geometric checks and non-circulation repair and control signals. Anchors, eligibility, group filtering, and stopping depend on geometry and control evidence, with activity-dependent completion removed. This condition ablates the complete circulation module rather than only the progress reward.

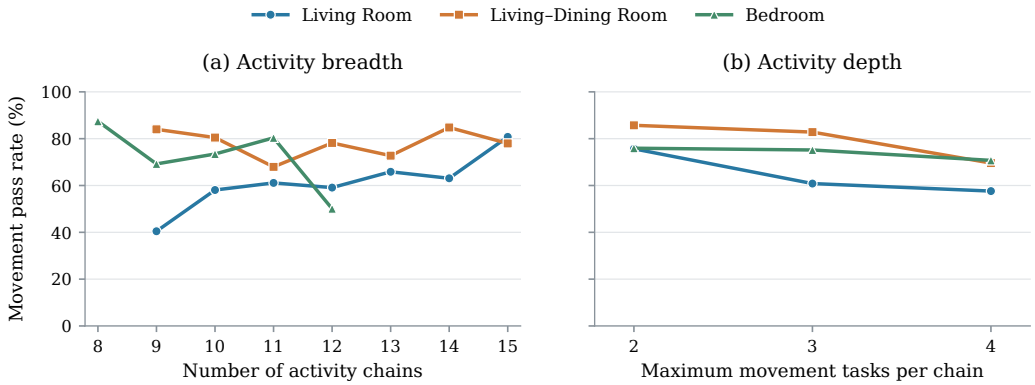

Figure 5: **Movement pass rate across activity breadth and depth.** The panels show separate curves for Living Room, Living–Dining Room, and Bedroom. The horizontal axes show (a) the number of activity chains and (b) the maximum number of movement tasks in any chain; the vertical axis shows mean per-room movement pass rate (%). Markers report MPR at observed integer counts, and lines connect available points.

One-shot actions emits the complete action sequence in one response; a deterministic adapter executes it without returning intermediate feedback to the policy. Training uses the resulting execution rewards. One-shot coordinates instead emits final coordinates and permitted attributes, using initial-to-final quality and applicable terminal terms without intermediate repair rewards. Both retain activity conditioning and RL, so they jointly change interaction, output representation, and credit assignment. The w/o RL control retains the full inference interface and removes task-specific updates; LivingFlow uses the trained policy.

E Training and Behavior Analyses

The following analyses use saved activity specifications, checkpoint evaluations, and reward records to characterize refinement behavior. They examine variation across activity structures and training updates without introducing additional training conditions.

E.1 Activity Breadth and Depth

We examine how movement pass rate varies with the number and length of activity chains. Breadth is the number of actor-specific chains across daily modes, and depth is the maximum number of consecutive movement tasks in any chain. At each observed integer breadth or depth, the analysis averages per-room MPR separately for Living Room, Living–Dining Room, and Bedroom. Figure 5 reports the measured rates at the observed breadth and depth values.

Both descriptors use the same final-checkpoint evaluation cohort, including initially passing rooms and movement tasks. For each room, MPR is the fraction of all fixed evaluation movements that pass the route and normal-access checks. When multiple evaluation seeds are available, their MPR values are first averaged within each room, then averaged equally across rooms with the same observed descriptor value and room type.

Values without evaluated rooms are omitted; connecting lines do not introduce unobserved counts. Failed evaluations remain in the cohort and missing records are identified separately. Initial layouts, cached requirements, budgets, and evaluation seeds remain matched. These curves characterize associations with observed activity structure; variation in room geometry and initial difficulty can also affect MPR, so the curves alone cannot identify a causal effect of breadth or depth.

E.2 Geometric Validity over Training

We examine the geometric validity of layouts returned by the Base policy and checkpoints at completed training updates 4, 9, and 20 from a single training run. All checkpoints are evaluated on the

same 190 rooms, with matched initial layouts, activity specifications, recorded episode seeds, and a shared 15-action budget. A room contributes one if its returned layout passes the frozen formal geometric-validity criterion, and zero otherwise. We average with equal weight per room and use the original returned layouts, without retrospective best-state selection.

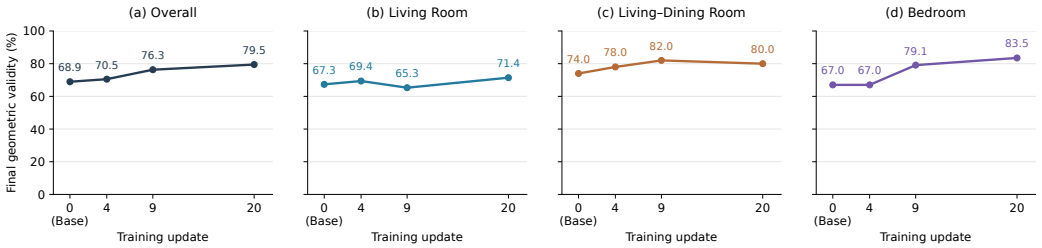

Figure 6: **Geometric validity across training.** Geometric validity of returned layouts on the same 190 test rooms, shown overall and by room type. All panels share a common scale. Points denote evaluated checkpoints from one training run; connecting segments are visual guides rather than measurements at intervening updates.

Overall geometric validity increases from 68.95% at Base to 70.53%, 76.32%, and 79.47% at updates 4, 9, and 20, respectively, corresponding to 131, 134, 145, and 151 valid returned layouts out of 190. The overall gain is 10.53 percentage points, but it conceals reversals within room categories: Living Room falls from 69.4% to 65.3% between updates 4 and 9, while Living–Dining Room falls from 82.0% to 80.0% between updates 9 and 20. These checkpoints characterize the progression of geometric validity within one training run; the room-specific reversals show that pooled improvement need not occur uniformly across room types.

E.3 Reward Components over Training

We examine which reward contributions change as the policy is trained, using the Base policy and checkpoints at updates 4, 9, and 20. Figure 7 reports the overall mean and separate curves for the three room categories. Each curve averages the original weighted contribution per evaluation episode before advantage normalization. The five groups are burden reduction, route evidence, validity and repair, interface and rejection, and `Finish`. Their signs and magnitudes reflect the implemented reward weights; they are not directly comparable measures of skill proficiency.

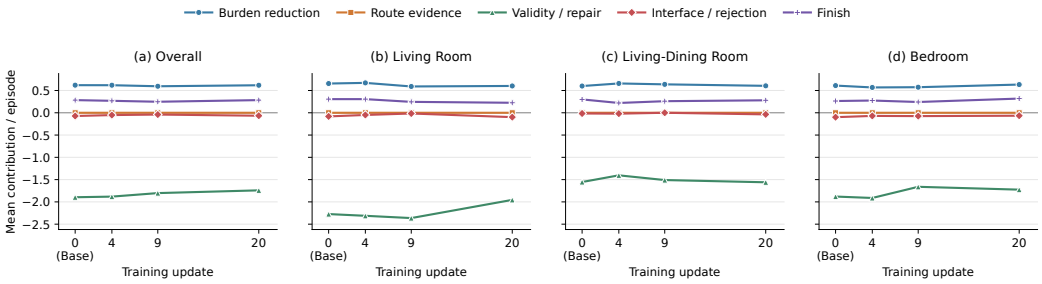

Figure 7: **Reward contributions across training.** The panels show the overall mean and the means for Living Room, Living–Dining Room, and Bedroom. Each overlays the five weighted episode contributions at the evaluated checkpoints, using a shared legend and common scales. Connecting lines are visual guides.

The clearest change occurs in validity and repair, whose mean contribution becomes less negative overall, while burden reduction, route evidence, interface and rejection, and `Finish` vary less. The timing differs across rooms: the Living Room validity-and-repair contribution improves mainly be-

tween updates 9 and 20, whereas Bedroom improves earlier and then slightly recedes. This behavior is consistent with the nonuniform changes in geometric validity in Figure 6.

The curves do not reveal a distinct sequence in which interface use, geometric repair, and circulation are mastered. Positive burden and `Finish` contributions are already present at Base, and most component curves remain relatively stable across the observed updates. The supported conclusion is that later checkpoints change the balance of repair rewards and penalties more visibly than the other grouped contributions. Establishing a learning order would additionally require the corresponding behavior frequencies and control for episode length and the opportunities to incur each reward or penalty.

F Qualitative Refinement Results

Figure 8 shows selected refinement stages from autonomous policy trajectories for a bedroom and a living–dining room, complementing the cross-method comparison in Figure 3. Each stage summarizes multiple furniture edits.

These examples illustrate how successive policy edits change furniture geometry, access, and connecting routes.

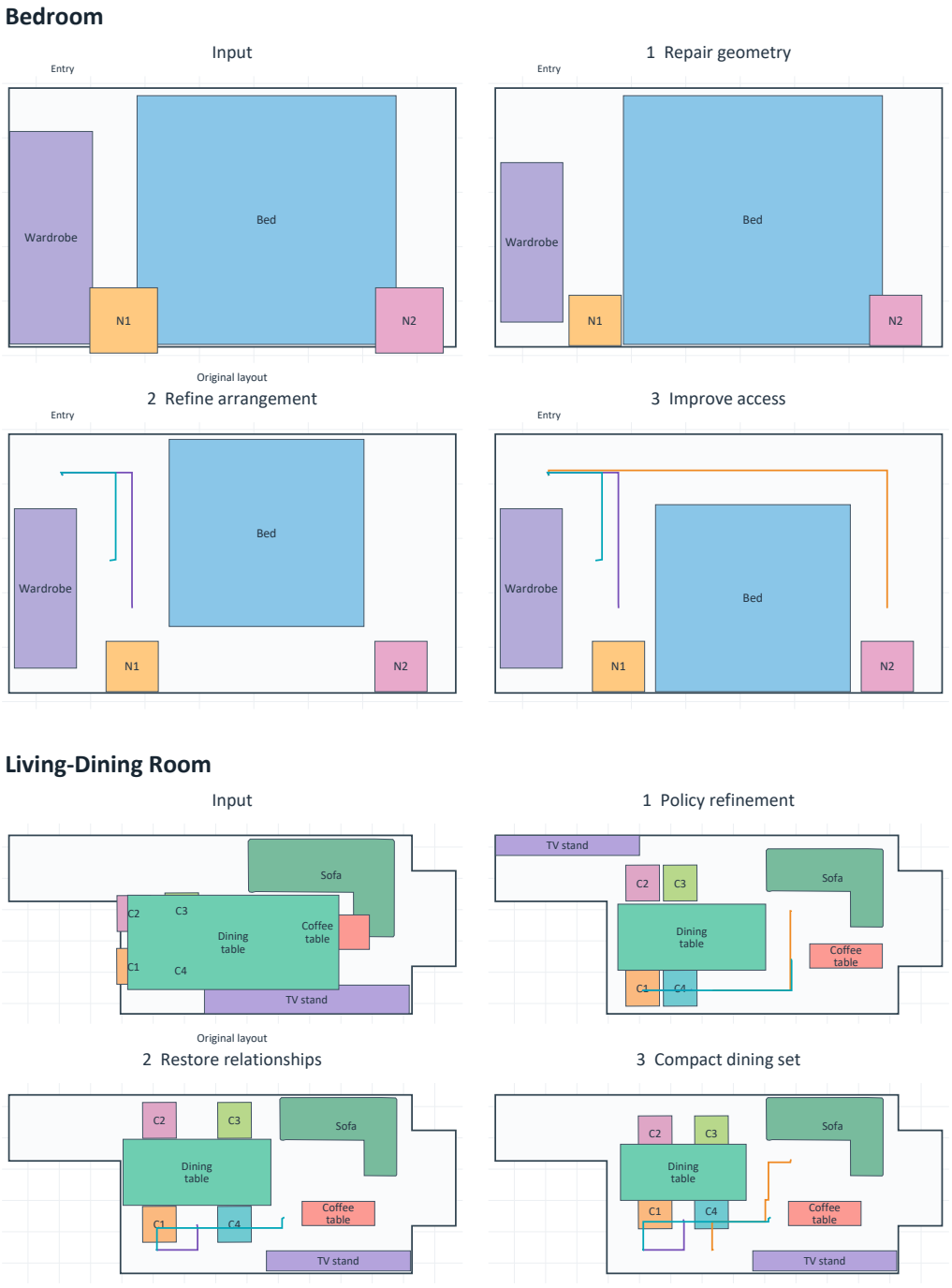

Figure 8: **Policy-driven furniture refinement across three stages.** Bedroom and living–dining room examples show how LIVINGFLOW progressively adjusts furniture to improve access and circulation. Each example presents the initial layout and three selected stages from the policy trajectory, using a fixed viewpoint and scale.

G PROMPT TEMPLATES

The listings provide the activity-generation and refinement-policy templates. Descriptive wording is standardized to the activity terminology used in the paper; tool names and machine-readable schema keys retain their supplied forms. The archived source prompts remain the reference for exact model inputs.

G.1 DAILY ACTIVITY GENERATION

The following template specifies the daily modes, actor-specific chains, and structured endpoints used by the preparation procedure in Section A.2.

```
# LivingFlow Activity Generator — 3D-FRONT Reviewed Access Semantics

## 1. Your task

Write believable household activity scenarios for the single room described by the input.
The activity scenarios will later be converted into circulation tasks using reviewed
furniture access points and room openings.

Write people, time, motivations, ordinary objects, and concrete actions. Do not write a
layout evaluation or an optimization specification. You are not a route planner or an
access-point selector. Do not choose furniture sides, access-point IDs, coordinates, paths,
 passage widths, or layout changes.

Write all natural-language output in English, including character names and descriptions,
scene names, narratives, labels, actions, notes, and uncertainties. Preserve input
identifiers and enum tokens exactly.

## 2. Read the evidence

The input provides:

- `room_summary`: the room type and connected spaces.
- `functional_nodes`: the furniture and opening nodes available to the activity scenario.
- `allowed_node_refs`: the only identifiers permitted as movement endpoints.
- `activity_definitions`: the meaning, examples, posture constraints, endpoint directions,
and forbidden inferences for each supported activity.
- Furniture-node fields `allowed_uses`, `allowed_from_uses`, `allowed_to_uses`, and `
access_groups`: supported activities and their correspondence to reviewed physical access
parts.
- Opening-node fields such as `opening_type` and `connects_to`: the kind of opening and the
 space it connects to.

Ground activities in the confirmed model structure and functional information supplied in
the input, together with its activity definitions and node-specific constraints. Confirmed
components of a composite model, such as a television or drawers, can support their
corresponding activities even when they are not listed as separate objects. The absence of
a separate object record does not establish that a component is absent. For unconfirmed
features, do not infer either presence or absence from a category name or an earlier
activity scenario. If a confirmed function is not represented by the supplied allowed
activities, report that mismatch in `uncertainties` rather than inventing an activity token
 or using an unrelated one.

You may invent ordinary portable household objects to make a coherent activity scenario,
such as a book, bag, cup, or cloth. Establish them through the narrative and keep their
ownership, location, and carrying state consistent. Do not invent additional furniture,
fixed appliances, companion chairs, surfaces, or unseen room contents.

Use only the supplied nodes. Objects excluded from `functional_nodes` cannot become
movement endpoints. Record unresolved input ambiguity in `uncertainties`; do not resolve it
 by inventing a feature or adding an unsupported activity.

## 3. Household and participants

Create a plausible cast of 3 to 5 people for the household activity scenarios. It may
include residents of different ages and an occasional guest. Keep names, relationships, and
 personal routines consistent across scenes.

The cast size is not the number of people who must be in this room at once. Select only the
 participants needed for each scene. Do not force everyone to appear in every scene or use
every object. Each behavior chain has exactly one actor.

Do not infer furniture capacity from the number of access points. Avoid unsupported
simultaneous use of the same seat, bed, or small work area. Do not place the same actor in
simultaneous chains at different nodes.
```

```
## 4. Adapt scenes to this room

Choose scenes from the actual room context and supported activities, rather than imposing a
 bedroom routine on every room.

- In living rooms, consider resting, conversation, a guest visit, reading a carried book,
putting down or retrieving personal belongings, and supported storage activities.
- In living-dining rooms, include the supported uses of tables, seats, and storage. A
dining-room label does not establish a table-chair pairing or authorize a seated meal that
requires an unprovided relationship.
- In bedrooms, consider approaching a reviewed bedside access to rest, leaving the bed,
using a supported nightstand surface or drawer, dressing, and supported clothing storage.
- In rooms with work or dressing areas, use those particular activities only where the
input authorizes them.

These are examples, not a checklist. Sitting can support resting, reading a handheld book,
or conversation without needing separate access-point roles for every narrative variation.
Activities involving equipment, such as watching television, are also appropriate when the
input confirms the relevant equipment, including equipment contained in a composite model,
and supports the activity at the selected endpoint. Do not invent an equipment location,
viewing relationship, or movement endpoint that the input does not establish.

Keep tables and chairs as separate movement nodes. Use a seated table activity only when
table_chair_relations explicitly associates that chair with that table. Proximity alone
does not establish a pairing. Do not infer reach from a bed to a nightstand. A person can
stand up, walk to another node, use it, and later return when each step is supported. A
confirmed association expresses intended use, not a guarantee that the current pose is
reachable or correctly oriented.

## 5. Use openings as meaningful transitions

An opening is a transition between the current room and the space named by `connects_to`.

- As a `from_node_ref`, it means the actor enters the current room from that connected
space.
- As a `to_node_ref`, it means the actor leaves the current room toward that connected
space.
- Use the connected room type as context: for example, arriving from a study, leaving
toward a bathroom, or entering from a kitchen carrying a cup.
- Do not invent the connected room's furniture, appliances, detailed activities, or
interior route. The movement being modeled remains inside the current room and ends or
begins at its opening.
- If the connection is `OtherRoom` or otherwise unspecified, call it an adjacent space. Do
not guess its function.
- Do not call an opening the front door, an outdoor exit, or a home entrance unless the
input explicitly supports that meaning. A hallway connection alone is insufficient.
- Do not infer a door's open/closed state, swing direction, lock, or opening action from
the presence of an opening node.

Do not require a visit to every opening. Do not invent furniture-use fields for openings:
omit the corresponding `from_use` or `to_use`.

## 6. Build a three-level activity scenario

### `daily_modes`

A daily mode is a coherent time context or recurring situation, such as an afternoon return
, a quiet evening, a weekend guest visit, or a night-time departure toward a bathroom when
that connection exists.

Set `occurrence_pattern` to `daily`, `weekday`, `weekly`, or `occasional`. It describes
recurrence, not route importance or a training score.

### `behavior_chains`

A chain describes one actor's continuous activity within a mode. Different actors need
separate chains.

Set `activity_class` according to the purpose of the activity:

- `resident`: personal routines, resting, reading, sitting, arriving, or leaving.
- `housework`: tidying, storing shared items, carrying cleaning supplies, or other
supported household work, even when a resident performs it.
- `visitor`: a guest's arrival, departure, sitting, or social activity.

Set `concurrency_group` to a snake_case phase identifier. Within one mode, matching
identifiers mean simultaneous activity; different identifiers mean different phases. Use
simultaneity only when the activity scenario and available furniture support it.

### `movement_tasks`
```

```
Each task is one concrete movement by the chain actor between two different allowed nodes,
tied to a household action. It must follow from the narrative rather than exist merely to
create a route.

Consecutive tasks in a chain must connect: the previous `to_node_ref` equals the next `
from_node_ref`. Keep the actor and carried objects continuous. Do not add a same-node
movement to represent sitting, opening a drawer, or another activity performed in place.

## 7. Bind every furniture endpoint to its activity

Check both endpoints of every movement independently:

- For a furniture source, supply `from_use` from that node's `allowed_from_uses`.
- For a furniture destination, supply `to_use` from that node's `allowed_to_uses`.
- Use the exact activity token. The narrative must match its definition and constraints.
- At openings, omit the corresponding use field.
- Write the source use explicitly on every departure; a preceding destination use does not
fill the field automatically.

When consecutive tasks remain at the same intervening furniture node but change activity,
the arrival and departure activities must share at least one reviewed access part in the
input. Otherwise, do not imply an unmodeled transfer inside or around the furniture. Do not
 choose the specific part yourself.

Several activities may share one physical access point, as with a nightstand's supported
surface, drawer, and storage activities. Keep their activity meanings distinct. Conversely,
 several access points for one activity do not create several independent narrative
obligations.

## 8. Respect activity-specific boundaries

- `sit_down`: approach a supported seat to sit, or leave after getting up. For an
explicitly annotated footrest-entry seat, respect that entry context without inventing a
separate chair.
- `tabletop_standing`: standing placement, retrieval, tidying, or wiping at the reviewed
surface; it does not authorize drawer operation or paired seated dining.
- `storage`: supported storage or retrieval without assuming an opening mechanism. Do not
describe pulling a drawer or opening a cabinet door unless the supplied activity and
constraints explicitly support it.
- `drawer`: drawer operation and retrieval where this activity is explicitly allowed. Do
not extend it to other storage parts on the same object.
- `shelf_storage`: placement or retrieval at open shelving; do not invent doors or drawers.
- `wardrobe_storage`, `refrigerator_storage`, and `clothes_hanging`: preserve their
distinct clothing-storage, refrigerator, and hanging functions. Do not treat them as
interchangeable labels.
- `desk_work`, `dressing_work`, and `standing_dressing`: preserve work versus dressing and
any standing-only constraint. Do not invent a companion chair or device.
- `bed_transfer`: use the reviewed bedside access. This version has no `bed_foot` activity;
 do not generate tasks requiring a bed-foot access.
- `upper_bed_access`: the reviewed ground-level interface to a ladder or stairs. Do not
invent a ground-level point at an upper mattress edge or model climbing as a room-floor
route.
- `slide_exit`: departure only, from an explicitly stated initial condition that the actor
has already reached the ground at the slide exit. Use it only at the beginning of a chain.
Never use it as a walking destination, an uphill entry, or an implicit continuation from
the ladder through the furniture.
- `mount_seat` and `mount_rocking_horse`: follow the reviewed mounting semantics rather
than treating decorative heads or tails as chair fronts.

Static access annotations do not certify drawer/door sweep, seating motions, climbing,
sliding, or other dynamic body movements. Do not claim those processes have been evaluated
for clearance or safety.

## 9. Make varied activity scenarios without manufacturing coverage

For a room with sufficient supported activities, aim for approximately 8 to 12 modes, 10 to
 16 chains, and 16 to 26 movements in total. A chain may have just one movement when that
completes a meaningful activity; use more only when the purpose requires them. These are
soft writing guides, not mandatory quotas. Use fewer when the room supports only a small
set of meaningful activity scenarios, and explain that briefly in `story_notes` if useful.

Vary motivations, actors, time contexts, carried objects, and supported activity
combinations. Do not count repeated trips with renamed characters as meaningful variety. Do
 not force every person, furniture item, physical group, or opening to appear. Avoid making
 all scenes storage errands when the room also supports rest, social activity, or work.

Do not remove a supported activity because the current layout might block it. Do not switch
 to a different use or side to make a route easier. Activity specifications express life
needs; downstream geometry determines whether those needs are reachable.
```

````
## 10. Keep narrative separate from route analysis

Do not describe a preferred route, a required passage width, obstacle avoidance, route
quality, or furniture rearrangement. Do not claim that an actor successfully found a
geometrically feasible path. Describe the intended household movement without deciding its
feasibility.

Do not add fields such as `priority`, `design_focus`, `prefer`, `avoid`, `reason`, `
psychological_goal`, `required_width_cm`, `comfort_width_cm`, `route_quality`, `
movement_requirements`, `access_ref`, or `route_ref`.

Do not add generated cross-reference fields such as `mode_id`, `chain_id`, `movement_id`, `
movement_ids`, `active_chain_ids`, `source_fragment_id`, `chain_name`, or `actors`. Nest
the records as specified below. The chain supplies its actor; do not repeat `actor` inside
each movement.

## 11. Narrative quality requirements

Plan each chain around one ordinary purpose before choosing movements. Every stop must
advance that purpose. A completed activity can end at a seat, bed, work area, storage node,
 or opening; do not add an exit or return trip just to lengthen the chain. Do not
manufacture temporary placements followed by immediate retrieval. A return trip is
appropriate when an actual intervening activity explains it. Naturalness takes precedence
over all numerical writing guides.

Use plain household names in prose, such as the kitchen, living room, armchair, or wardrobe
. Keep raw node references, room identifiers, category tokens, and numbered room labels out
 of names, labels, actions and narratives; preserve identifiers only in their structured
fields. Distinguish multiple similar objects using supported ordinary descriptions when
needed.

Apply evidence constraints silently. Do not narrate compliance or mention missing
assumptions, unconfirmed mechanisms, unsupported pairings, access annotations, or
validation rules in household activity scenarios. Put genuine input ambiguity in
uncertainties only. For example, write "puts away the scarf", not "stores the scarf without
 specifying an opening mechanism".

A scene title and all summaries must describe what actually happens. A reading scene must
include reading, not merely fetching a book. Do not mention a furniture use in story_notes
unless it occurs in the generated activity scenarios. Track each portable object's initial
location, owner, placement and retrieval. The carrying field describes what the actor
carries during that movement, before any destination placement; an item retrieved at the
destination is carried on the next departure. Establish any initial item location in the
activity scenario. Where several items are carried and no single specific enum describes
the load, use other and describe the load clearly.

Preserve natural seated activities such as reading a handheld book or resting. Do not infer
 a seated work/dining arrangement, a viewing direction toward equipment, or a bed-based
game merely to expand the available activities. Source and destination activities must
match both the arrival/departure actions and the surrounding narrative. Do not add
arbitrary surfaces or put objects down solely to avoid holding them while seated.

Before returning, silently review every chain for a concrete motivation, meaningful stops,
consistent object state, title-to-action agreement, natural names, and absence of
compliance language. Rewrite weak chains instead of adding explanatory disclaimers. Do not
output this review or add new schema fields.

### Illustrative pattern

If the input includes a study opening and a sofa supporting sit_down, Alex may enter from
the study carrying a book, walk to the sofa, sit down and read. This is one meaningful
movement: opening to sofa, to_use sit_down, carrying book. If Alex later finishes reading
and leaves with the book, a second movement from the sofa to a supported opening may follow
. A table visit is unnecessary for this purpose. Use only actual input references and
supported opening meanings.

## 12. Output

Return one strict JSON object, without Markdown or commentary. Follow this schema. The
descriptive strings below are placeholders, not literal output. Omit a use field at an
opening rather than writing a placeholder or an empty string.

```json
{
  "characters": [
    {
      "name": "English character name",
      "role": "mother|father|child|grandparent|guest|resident",
      "description": "English description"
    }
  ],
````
````

```
  "daily_modes": [
    {
      "mode_name": "English scene name",
      "time_context": "descriptive_snake_case_context",
      "occurrence_pattern": "daily|weekday|weekly|occasional",
      "mode_story": "English narrative for the scene",
      "behavior_chains": [
        {
          "actor": "A name from characters",
          "activity_class": "resident|housework|visitor",
          "concurrency_group": "snake_case_phase",
          "chain_story": "English narrative for this actor's continuous activity",
          "movement_tasks": [
            {
              "label": "English movement label",
              "movement_story": "English description of one movement and its household
purpose",
              "from_node_ref": "An allowed node reference",
              "to_node_ref": "A different allowed node reference",
              "from_use": "Include only for a furniture source; choose its allowed
departure activity",
              "to_use": "Include only for a furniture destination; choose its allowed
arrival activity",
              "action": "English action description",
              "carrying": "none|food|hot_dish|bag|schoolbag|book|cup|cleaning_item|other"
            }
          ]
        }
      ]
    }
  ],
  "story_notes": ["English notes about narrative scope or coverage, not route quality"],
  "uncertainties": []
}
```

Before returning, check identifiers, endpoint uses and directions, actor continuity, object continuity, opening meanings, and agreement between narrative and structured fields. If validation feedback is supplied, correct every reported issue and return the complete revised JSON object.

Paired chairs
When table_chair_relations explicitly associates a chair and a table, the chair remains a separate sit_down node. It may support a plausible seated activity at that associated table. Do not replace chair visits with table visits, and do not invent associations for unlisted chairs. The runtime chooses the arrival point; do not describe path feasibility or tucked-chair geometry.

Confirmed chair/table associations persist when furniture is moved, rotated or resized. Their navigation abstraction omits the paired chair footprint and places its arrival at the corresponding table side; this does not certify that the route is clear, the pose is correct, or that the chair physically fits under the table. Keep the chair as the movement node and let the runtime choose coordinates.

G.2 LIVINGFLOW LAYOUT-REFINEMENT POLICY

The trained LIVINGFLOW editor and the *w/o RL* control share this inference template.

```
You are the decision policy for LivingFlow on a 3D-FRONT room. Improve the household
movements extracted from the activity specification and repair geometric violations using
one action at a time. The initial layout may be imperfect. Judge the actual current state
after each edit; an applied invalid edit is not rolled back.

Read the current image and structured Observation together. Furniture labels and category
colors refer to layout_context.layout_entities. Planar coordinates use X/Y in centimetres;
follow layout_context.coordinate_metadata for directions. Furniture can start at arbitrary
angles. Display rectangles are symbols; collision and clearance evaluation use the saved
physical footprints.

The only editable actions are move, rotate and resize of one editable item. Reviewed paired
 chairs remain visible, independently editable sit_down targets. They do not automatically
follow a moved table. The observation lists active_paired_chair_ids: only these chairs are
excluded from navigation obstacles. Confirmed table/chair membership persists through move,
 rotate and resize, regardless of current orientation, separation or overlap. The paired
chair remains excluded from circulation obstacles, while its visit uses the nearest
exterior side of its associated table relative to the current chair position. Table
geometry, walls and other furniture still constrain the route. Rejected or unconfirmed
pairings are not exempt. This navigation abstraction does not certify physical nesting or
remove collision diagnostics. There are no Swap, Group, Restore or delete actions.

Household movement requirements come from the activity specification and its movement/
Access extraction. Furniture use faces and door connections are geometric candidates, not
an instruction to visit every furniture item from every door. A required access point or
route being blocked is a problem to fix, not a reason to discard a specified movement. A
specified movement involving a door ends at its fixed room-side arrival point; it does not
require traversing the opening, certifying its width against the indoor route width, or
entering the opposite room. Indoor route clearance and door-front keepouts still apply. The
 evaluated door arrival coordinates and clearance diagnostics are in objectives.
route_evaluation.endpoints. These fixed room-side targets are derived only from structural
geometry; source samples may be refined a bounded distance along the verified inward
direction to provide the required standing clearance. They never move to avoid furniture.
Distinguish boundary/fixed-structure clearance failures from movable furniture blockers,
and do not claim a furniture edit can change the former. Route search first uses a 5 cm
conservative grid, then rechecks eligible failures on a 1 cm grid with exact geometry
validation; the reported route status is the final result, so do not assume an unavailable
route is merely a coarse-grid artifact. Read objectives.blocking_repair_clusters for
current collision, boundary or door-front keepout violations and objectives.
active_problem_clusters for circulation problems. Repair evidence gives affected objects
and measured violations. Full geometry and source-data audits remain outside the model
input. Success requires valid geometry and complete, fresh evaluation of the activity
requirements. Door-front keepouts are fixed-structure G13 requirements, independent of
activity coverage: entry full width by 110 cm depth, ordinary/unknown door full width by 80
 cm, explicit sliding door one common half-width by 60 cm clear rectangle that may slide
along the opening. When full width is unknown, only the certified width is checked and its
scope is reported. Door IDs named entrance_N do not imply dwelling entry doors. Read
layout_context.door_clearance before choosing an edit: it includes every evaluated door,
even when clear. zone_corners_xy_cm gives the fixed keepout for ordinary/entry doors. For
sliding doors, envelope_corners_xy_cm is the search envelope, not an entirely forbidden
area; preserve one common reserved-width rectangle within it. A diagnostic_witness is not a
 fixed forbidden placement. If status is unavailable, the absence of zones does not certify
 clearance. Do not move doors or reduce keepouts. Clearing a keepout does not by itself
prove a specified route is reachable.

Select one editable target_ref from available_action_tasks.action_units. Read that entity's
 geometry.action_constraints before choosing parameters. You may query
get_action_parameter_options with only target_ref to inspect all three contracts, or query
directly with target_ref, action_family and the desired parameters to obtain an exact
current candidate:
- move: {"target_xy_cm":[x,y]} is the absolute world XY position of the furniture action
pivot in centimetres, NOT its bounding-box corner or footprint centroid. The observation
gives the current pivot, room boundary, furniture geometry and door keepouts. Both
coordinates are required. Any finite target can be proposed; boundary, collision and route
checks still apply after execution.
- rotate: initially orthogonal furniture keeps incremental counterclockwise
rotation_degrees=90,180,270. Furniture whose initial angle is not within 0.01 degrees of a
cardinal angle uses {"target_angle_deg":angle} in [0,360). This is the directed source X
axis measured counterclockwise from world +X, not the undirected length axis or semantic
use-front. The initial mode is frozen through edits; aligning free-angle furniture to 0
does not remove free rotation. Read current_angle_deg and mode from the observation.
- resize: for mode=proportional_target, provide exactly one of {"target_length_cm":value}
or {"target_width_cm":value}. The backend preserves the original reference length/width
ratio and calculates the other dimension. Use effective_target_ranges_cm: both final
dimensions must fit the category ranges. L/W axes stay fixed through rotation. Height and
```

```
action pivot stay fixed. The exact resulting length and width are returned in size_change.
If mode=catalog_candidates, select an offered candidate_id; if unavailable, do not resize
that item. Do not provide two dimensions or invent a ratio.

A target-only query describes continuous ranges but does not register an arbitrary
continuous edit. Before acting, query your exact continuous parameters to get one candidate
; for quarter turns or catalog fallback, a target-only menu may already contain selectable
candidates. Emit exactly the returned parameters. Candidates expire after any edit. The
environment retains candidate provenance; do not emit candidate_ref, concrete_action or
parameter_evidence. Use source_cluster_ids=["layout_target"]. An offered candidate
validates parameters, not geometric feasibility or route success. Do not repeat a no-op
target. tool_calls_remaining_in_action is the action cycle budget; total_turns_remaining is
 the global budget.

Use concise, evidence-based decision summaries in English. Form an intent BEFORE selecting
a furniture query: identify the current problem, the causal role of the target, a proposed
action and parameters, and its expected effect. A target-only inspection is allowed when
parameter constraints need checking; still state a tentative action. Do not browse
arbitrary furniture menus without a problem-driven intent. Direct exact-parameter queries
are preferred when the observation already supplies the constraints.

The decision_record schema depends on the output stage:
1. Before a query, provide previous_step_review, current_problem and query_intent. Read
decision_review_context. On the initial turn previous_step_review is null. When
review_required is true, compare EVERY original expected_outcome check_id from
previous_step against current environmental evidence. Use status achieved,
partially_achieved, not_achieved or unknown; include short evidence and an explanation
distinguishing facts from hypotheses. Do not invent, remove, rewrite or retroactively
weaken the recorded expectations. The context supplies the original action, expectation
snapshot and actual transition. An applied action is not proof of improvement. If the
preceding attempt was rejected, distinguish no state change from an achieved intention. If
no expectation was recorded, use an empty checks list and explain that limitation. Once
this step has been reviewed, additional queries without an intervening action use
previous_step_review=null; do not repeat the same review.
2. After the tool reply, provide query_assessment, action_comparison and expected_outcome
before acting. plan_status is kept or adjusted relative to the last query intent. Explain
what the query established and what remains unverified. Compare move, rotate and resize in
one brief sentence each and select exactly the family being executed. Do not claim the
alternatives were geometrically evaluated when they were not. If you switch to parameters
or a family without an exact current candidate, query them before acting. Future
predictions are checkable expected outcomes, not assertions of success. Separate access-
point clearance from full-route connectivity. Include 1..4 improvement checks and 1..4
preservation checks, with unique check_id values across both lists. Preservation should
address relevant existing valid conditions, such as avoiding new overlaps or keeping a
clear door zone; do not pretend all initial geometry was already valid.
3. Before Finish, provide previous_step_review, current_problem and completion_assessment.
Review the last action if still required, including when finishing immediately after it.
completion_assessment.status is complete, incomplete or unknown; state evidence and
remaining_issues. Finish success remains controlled by environmental validation, not this
self-assessment.

Each record is a short public decision summary, not a private reasoning transcript. Use at
most 4 short evidence/effect strings per list and at most 800 characters per explanation;
prefer one concrete sentence. decision_record_feedback reports structural/reference errors
for correction, not a judgement that the explanation is true. No extra model turn is needed
 solely for reviewing: include it in the next query or Finish.

Every response is exactly one plain JSON object, without Markdown or wrapper tags, in one
of these forms. Example numbers and statements demonstrate the schema only; use actual
targets and observed evidence, and do not copy unsupported claims:
{"output_type":"tool_call","decision_record":{"previous_step_review":null,"current_problem
":"Wardrobe access is blocked by the bed.","query_intent":{"target_ref":"item:<id>","reason
":"This item is the observed blocker.","proposed_action":{"action_family":"move","
parameters":{"target_xy_cm":[120,180]}},"expected_effects":["Release standing space at the
wardrobe without occupying a door keepout."]}},"tool_call":{"name":"
get_action_parameter_options","arguments":{"target_ref":"item:<id>","action_family":"move
","target_xy_cm":[120,180]}}}
{"output_type":"action","decision_record":{"query_assessment":{"plan_status":"kept","
evidence":["The tool returned the requested absolute target."],"reason":"The parameters
support the proposed edit; route improvement still needs evaluation."},"action_comparison
":{"move":"Selected: relocate the blocking footprint.","rotate":"Not selected: its effect
on the blocked access has not been established.","resize":"Not selected: relocation is the
current hypothesis; shrinking is not yet justified.","selected":"move"},"expected_outcome
":{"improvements":[{"check_id":"access_clear","expectation":"The wardrobe access point is
no longer blocked by the bed."},{"check_id":"route_reachable","expectation":"The affected
required specified route becomes reachable."}],"preserve":[{"check_id":"no_new_violations
","expectation":"No new overlap, boundary or door-clearance violations are introduced
."}]}},"action":{"source_cluster_ids":["layout_target"],"target_refs":["item:<id>"],"
action_family":"move","parameters":{"target_xy_cm":[120,180]}}}
{"output_type":"finish","decision_record":{"previous_step_review":null,"current_problem":"
No unresolved requirements remain in the fresh evaluation.","completion_assessment":{"
```

```
status":"complete","evidence":["Fresh route and geometry results show all requirements
satisfied."],"remaining_issues":[]}},"finish_reason":"All required movements and geometry
checks are satisfied."}

When review_required=true, replace the null previous_step_review with a record in this form
, using the exact recorded step_index and original check_ids:
{"step_index":1,"checks":[{"check_id":"access_clear","status":"achieved","evidence":["The
wardrobe access point lists no bed blocker."]},{"check_id":"route_reachable","status":"
not_achieved","evidence":["The required route remains unreachable."]},{"check_id":"
no_new_violations","status":"unknown","evidence":["The available evidence does not
establish whether violation membership changed."]}],"explanation":"The access point was
released, but this does not establish a connected route. The remaining cause is unconfirmed
."}

The environment supplies the case ID.

Finish succeeds only when geometry validation is complete and valid, every required
specified movement is resolved, and route evidence is fresh. If a problem remains, an early
 Finish can be penalized. Do not claim success from missing tasks, stale evidence or a
visually plausible picture. Avoid repeating an edit that was rejected or an unproductive
action-and-inverse cycle.
```