EMBODIED LEARNING IN THE LOOP: CO-ADAPTING POLICIES AND ENVIRONMENT DISTRIBUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We devise a learner-aware framework for embodied PointNav that treats the scene distribution as a policy-conditioned component of training. Existing policies are trained on static corpora or samples from pretrained generators, neither of which responds to the learner's evolving competence. Moreover, scene-centric properties (*e.g.*, realism, reachability, geometric complexity) do not identify scenes that promote learning. Our framework adapts a layout diffusion generator using scene-level policy learning progress. It generates indoor layouts for PointNav and compares policy performance on the same episodes before and after each policy update. Changes in success, path efficiency, and goal distance define a utility signal that weights the denoising records that produced each scene, steering updates to the pretrained generator. This closes the loop between scene generation and policy learning. Averaged across five distributions, our pipeline improves PointNav performance (**5.84** and **4.44** percentage-point gains in SR and SPL, respectively, over Frozen Generator). Policy feedback shifts the generator to a distinct distribution (MMD^2=**0.752**), with the separation persisting across later adaptation intervals. Matched-success trajectories and same-input comparisons further reveal distinct navigation behaviors and action choices under identical observation and action histories. Our code will be released.

1 INTRODUCTION

Embodied agents learn navigation through interaction with 3D environments. The training environments define the geometry, observations, and action consequences encountered by a policy, thereby substantially influencing the skills it acquires. Simulators (*e.g.*, AI2-THOR [1] and Habitat [2–4]), together with indoor environments spanning real-world scans [5–7], digital reconstructions [8], and synthetic or procedurally constructed scenes [3, 9–11], have made large-scale navigation learning feasible. Yet training strong point-goal navigation (PointNav) policies can still consume billions of frames [12], making it important to curate and generate training environments that are both diverse and informative.

A common response is to generate more environments. Procedural [10, 13, 14], language-guided [15], and learned [16–22] scene generators can produce interactive indoor environments at scale, from complete houses to structured object layouts. However, environments are often generated independently of policy training, leaving their distribution largely independent of the learner's changing abilities. This separation raises four issues. ❶ Scene-centric properties, such as physical validity, reachability, planning scores, or visual realism, do not directly measure potential training utility. ❷ Training utility is learner-dependent and non-stationary: as the policy evolves, the relative utility of scenes can change across training updates. ❸ Static scene corpora define a fixed training distribution. Generators trained only on such corpora can increase the number of available scenes, but their outputs remain anchored to the same data prior; thus, scaling scene generation alone does not ensure exposure to the environmental variation needed for generalization beyond the training distribution [11]. ❹ Existing task-oriented generators generally rely on predefined objectives rather than measured policy learning progress, providing limited mechanisms for policy-learning feedback to directly adapt the generator.

To address these challenges, we propose a learner-aware framework for adaptive 3D environment generation. Rather than treating a pretrained generator as a fixed source of training data, we adapt its output distribution throughout training using policy learning progress. At each outer iteration, a layout diffusion model generates structured object layouts that are instantiated with ProcTHOR assets in a navigation simulator. A PointNav policy is then trained on episodes sampled from these scenes. Our central feedback signal is scene-level learning progress, inspired by progress-based exploration and

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.

learner-adaptive training [23–25]: we evaluate fixed probe episodes before and after the policy update, using changes in success rate, path efficiency, and distance-to-goal as an empirical proxy for each scene's training utility. We use this proxy to weight the denoising records associated with generation, a design that accommodates feedback from non-differentiable simulator interaction and policy updates.

This design offers four advantages. *First*, it avoids the mismatch between scene-centric proxies and training value: a scene's utility is determined by the current policy's improvement on matched evaluation episodes, rather than predefined physical or geometric attributes alone (*issue* ❶). *Second*, it tracks *non-stationary, scene-specific feedback*: evaluating identical episodes before and after an update removes episode-resampling effects and exposes heterogeneous gains and regressions across scenes. The generator therefore follows current policy feedback rather than a fixed scene ranking (*issue* ❷). *Third*, it provides a mechanism to adapt the generator rather than only sampling more scenes from a fixed prior: scenes associated with higher utility receive larger weights during diffusion adaptation, biasing subsequent generation toward high-utility regions of the learned scene distribution (*issue* ❸). *Fourth*, it turns policy feedback into a generator-training signal: after a generated scene is evaluated, its utility is attached to the denoising record used to produce it and weights the standard diffusion loss (*issue* ❹). Together, these components form a closed loop in which policy-conditioned scene generation and navigation learning co-adapt over time.

For examination, we evaluate our framework on PointNav across synthetic, generated, and external evaluation distributions. Experimental results demonstrate that our feedback-adapted pipeline improves generalization. As seen in §4.2, our pipeline achieves the highest average performance across five evaluation domains (**5.84** SR and **4.44** SPL percentage points improvements over Frozen Generator). In §4.3, we show that successive policy-feedback updates move the generator away from its frozen counterpart: their MMD^2 separation rises to **0.752** and remains high across subsequent adaptation intervals. Matched-success analyses show that this distributional change is accompanied by distinct navigation behaviors (§4.4), while controlled comparisons establish that the resulting policies select different actions under identical observation and action histories, showing that feedback adaptation reshapes the policy's underlying action-selection function (§4.5).

2 Related Work

3D Scene Generation. Early 3D scene synthesis [26–32] treats layout construction primarily as a constrained design problem. As explicit design become difficult to scale, data-driven methods [33, 34] learn spatial and semantic regularities from scenes and structured datasets such as 3D-FRONT [35]. Later work represents these regularities as structured scene distributions [36, 37] and makes them increasingly controllable through floor plans [16, 38], scene graphs [39–41], partial layouts [42], or language [19, 43–45], supported by autoregressive and diffusion models [17, 18, 42, 46]. Across this progression, the primary objective has remained the generation of realistic, structurally coherent, and controllable scenes. Embodied learning [10, 20] additionally requires physically plausible and interactive layouts, and recent work [47, 48] extends scene generation to simulation-ready assets and executable worlds. However, these works evaluate scenes through environment-level validity or predefined objectives rather than the training improvement induced in an embodied policy (*e.g.*, navigation). Our framework instead uses this measured improvement as feedback to adapt the generator toward scene distributions that support the policy's evolving learning needs.

Environments for Embodied Navigation. Embodied navigation relies on simulators and benchmarks that provide 3D geometry, visual observations, and action-dependent state transitions. These infrastructures have evolved from multimodal indoor spaces and human-designed houses [49, 50] to reconstructed scenes for standardized evaluation [2, 5, 51]. Physics-rich platforms [1, 9, 52, 53] further support household tasks and object-centric interaction. Together, they provide the basis for navigation training and evaluation [12, 54–59], while increasing the need for varied and informative training environments. Generated and procedurally instantiated scenes expand the training data for navigation. Existing systems provide scalable procedural houses and realistic simulatable layouts [10, 14, 60], connect generated scenes with real-world settings and large-scale navigation learning [13, 61], and target scalable scene and data synthesis with neural simulators [62]. However, these environments are generally sampled from a preconstructed or externally specified distribution, so their quantity and diversity do not indicate which scenes are useful for the current policy [63].

Task-Oriented Environment Design for Embodied Learning. Task-oriented environment design connects scene generation with embodied learning by incorporating downstream requirements into the generation process. Existing approaches consider physical validity and reachability [20], language or

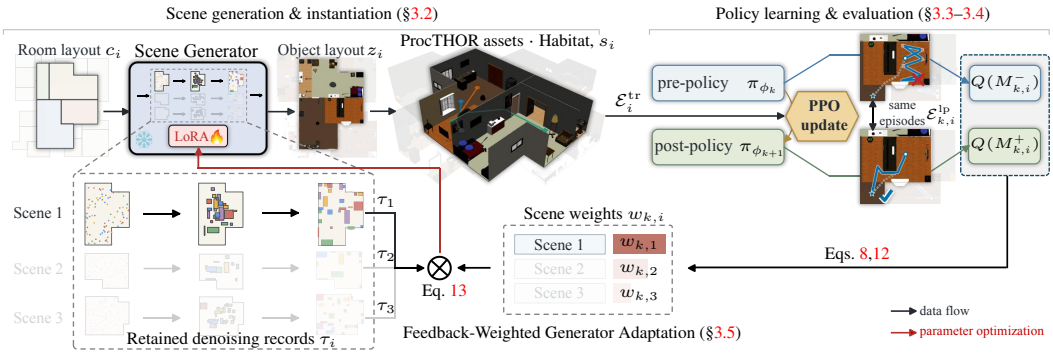


Figure 1: **Framework Overview** (§3). A room-layout-conditioned diffusion generator produces object layouts, instantiated with ProcTHOR assets in Habitat for PointNav training. Before and after each PPO update, the policy is evaluated on the same per-scene episodes, sampled separately from training. The change in navigation score measures scene-level learning progress, which is shaped and normalized into weights for the denoising losses of the corresponding retained generation records. Only LoRA adapters are updated, while the diffusion backbone remains frozen. The adapted generator supplies scenes for the next iteration. Denoising states are schematic.

spatial constraints [15], navigation-oriented planning signals [21], and robotic objectives for adapting procedural scene priors [22]. More generally, curriculum learning and teacher-based methods organize experiences by task difficulty [64–66], adapt task parameters as the learner changes [25, 67–72], or estimate training potential from task representations, replay information, and learned adversaries [73–78]. Domain randomization provides a complementary way to broaden training variation [79]. Although these studies establish a forward connection from environment design to embodied learning, the guidance is typically specified before training, with policy training feedback not used to generator adaption.

Adaptive Environment Design. Recent work moves beyond fixed training environments by adapting them to the learner's abilities. Unsupervised environment design uses learner-derived signals, such as regret [80–82] or replay information [83], to adapt existing levels through selection or mutation [84, 85]. Generative and language-guided methods instead focus on constructing new compositional tasks and curricula, with the generation process not necessarily tied to policy-training feedback [86, 87]. This idea has recently extended to embodied learning through propose–generate–learn loops [88], feedback-driven environment evolution [89, 90], and task-driven world construction [91, 92]. In our framework, the scene generator produces scenes that are used to train the navigation policy. We measure how policy performance changes on each scene's fixed evaluation episodes across the update, and use this scene-level feedback to weight the denoising records that guide subsequent scene generation.

3 Methodology

Overview. Our framework couples simulator-compatible scene generation and embodied navigation through a scene-level feedback loop. A diffusion scene generator produces object layouts from a ProcTHOR-compatible prior and instantiates them as PointNav training environments (§3.2). The navigation policy is updated with PPO on episodes from generated scenes (§3.3), while fixed per-scene episodes measure policy changes before and after training (§3.4). We use this learning progress to adapt the generator (§3.5), which then produces scenes for the next iteration. An overview is in Fig. 1.

3.1 Problem Formulation

We study PointNav in generated indoor environments. Let θ and ϕ denote the scene-generator and navigation-policy parameters, respectively. Given a room-layout representation c_i and an initial noise sample $\xi_i \sim \mathcal{N}(0, I)$, the generator G_θ produces a structured object layout

$$z_i = G_\theta(c_i, \xi_i), \tag{1}$$

where z_i contains object categories, translations, sizes, and orientations. An asset-based builder converts this layout into a simulator scene $s_i = \text{Instantiate}(z_i; \mathcal{A})$, where $\mathcal{A}$ is the simulator asset set. At outer iteration k, we sample B candidate layouts $\{z_i\}_{i=1}^B$ from the current generator. Let $\mathcal{I}_k \subseteq \{1, \dots, B\}$ denote the indices of candidates that pass the instantiation checks. For each instantiated scene s_i with $i \in \mathcal{I}_k$, a PointNav task generator samples a training episode set

$$\mathcal{E}_i^{\text{tr}} = \text{PointNav}(s_i; \mathcal{C}_{\text{tr}}) = \{e_{i,j}\}_{j=1}^{N_i}, \tag{2}$$

where $\mathcal{E}$ denotes a set of PointNav episodes, the superscript tr marks episodes used for policy training, and the subscript i associates the set with scene s_i. The constraint set $\mathcal{C}_{\text{tr}}$ specifies how training

episodes are sampled, and N_i is their number. The training data and feedback records at iteration k are

$$\mathcal{D}_k \triangleq \bigcup_{i \in \mathcal{I}_k} \mathcal{E}_i^{\mathrm{tr}}, \qquad \mathcal{R}_k \triangleq \{(s_i, \tau_i, R_{k,i})\}_{i \in \mathcal{I}_k}, \tag{3}$$

where τ_i records the diffusion denoising information used to generate s_i, and $R_{k,i}$ is the empirical feedback from the policy update on scene s_i. At each outer iteration, we first update the navigation policy with PPO and then adapt the scene generator using the resulting scene-level feedback:

$$\phi_{k+1} = \mathrm{PPO}(\phi_k, \mathcal{D}_k), \qquad \theta_{k+1} = \mathrm{FeedbackUpdate}(\theta_k, \mathcal{R}_k), \tag{4}$$

where PPO updates the policy on $\mathcal{D}_k$, and FeedbackUpdate adapts the generator using $\mathcal{R}_k$. The first iteration establishes a navigation checkpoint. Subsequent iterations continue from the previous checkpoint and enable before–after feedback, tracking the same learner as its competence changes.

3.2 Simulator-Compatible Scene Prior

Simulator-Compatible Training Corpus. We construct a simulator-compatible training corpus from ProcTHOR [10] scene descriptions and object assets. Existing scene generators commonly train on 3D-FRONT [35] room layouts and 3D-FUTURE [93] furniture assets, which are not directly compatible with our navigation simulator. By using ProcTHOR instead, our generated layouts can be instantiated with ProcTHOR assets for PointNav training. We align object records with asset metadata, normalize object and floor-plan coordinates, and discard object records that cannot be matched to simulator instances or have clearly out-of-range coordinates. Detailed preprocessing is provided in *supp.* §A.1.

Generator Training and Scene Instantiation. We train DiffuScene [18] on this ProcTHOR-derived corpus in its room-layout-conditioned setting, without language or object geometry features. It represents scenes as unordered sets of object attributes and retains an intermediate denoising record for each layout. These records later link generated scenes to their navigation feedback for generator adaptation. For instantiation, we map each predicted object to the closest-sized ProcTHOR asset in its predicted category, write assets and poses to a scene-instance description, and load it in Habitat [2]. We retain only layouts that load successfully with valid asset assignments before constructing PointNav episodes.

3.3 Scene Generation and PointNav Training

For each scene that passes the instantiation checks, we use the Habitat PointNav generator [2, 54] to construct a training episode set $\mathcal{E}_i^{\mathrm{tr}}$. The sampling process is governed by fixed task-generation constraints, and scenes that cannot provide the required number of episodes are omitted from the current policy update. These episodes form the current iteration's training data $\mathcal{D}_k$ in Eq. 3. We initialize the policy from the latest checkpoint and update it with PPO using this data. This lets the policy evolve as the generated scene distribution changes across iterations.

3.4 Scene-Level Learning Progress

Within-Iteration Matched Evaluation Episodes. At iteration k, we separately sample a scene-specific evaluation set for each generated scene s_i, $i \in \mathcal{I}_k$:

$$\mathcal{E}_{k,i}^{\mathrm{lp}} = \{\bar{e}_{k,i,j}\}_{j=1}^{K_{k,i}}, \tag{5}$$

where the superscript lp denotes learning-progress evaluation. These episodes are sampled separately from $\mathcal{E}_i^{\mathrm{tr}}$ and are held fixed for the paired evaluations around the current PPO update. The pre-update policy π_{ϕ_k} and post-update policy $\pi_{\phi_{k+1}}$ are evaluated on the same set:

$$M_{k,i}^{-} = \mathrm{Eval}(\pi_{\phi_k}, \mathcal{E}_{k,i}^{\mathrm{lp}}), \qquad M_{k,i}^{+} = \mathrm{Eval}(\pi_{\phi_{k+1}}, \mathcal{E}_{k,i}^{\mathrm{lp}}), \tag{6}$$

where Eval returns success rate, SPL, and distance-to-goal for each scene. Holding the episodes fixed within the iteration removes episode-resampling effects from the before–after comparison. Later iterations generate new scenes and construct new evaluation sets.

Scene Utility. We summarize task completion, path efficiency, and remaining goal distance with a scalar navigation score:

$$Q(M) = w_{\mathrm{sr}}\,\mathrm{SR}(M) + w_{\mathrm{spl}}\,\mathrm{SPL}(M) - w_{\mathrm{dtg}}\widetilde{D}(M), \tag{7}$$

where w_{sr}, w_{spl}, and w_{dtg} weight the three metrics, and $\widetilde{D}(M)$ is the clipped, normalized distance-to-goal. We define the primary feedback for scene s_i at iteration k as the change in this score:

$$R_{k,i} = Q(M_{k,i}^{+}) - Q(M_{k,i}^{-}). \tag{8}$$

This signal measures the change in policy performance on navigation tasks sampled within s_i across the current policy update. It provides scene-associated feedback for how the update transfers to that scene's task instances. When $R_{k,i} < 0$, the updated policy performs worse on these episodes, so the corresponding generation record receives a reduced weight in the subsequent generator update.

3.5 Feedback-Weighted Generator Adaptation

Feedback-Driven Diffusion Adaptation. Early approaches [94] incorporate downstream feedback into diffusion fine-tuning through reward-weighted likelihood or related supervised updates. To optimize downstream rewards more directly, subsequent methods [95, 96] formulate the reverse diffusion process as a sequential decision problem and apply KL-regularized policy-gradient updates to its denoising transitions. This formulation can be written as

$$\max_{q} \left(\mathbb{E}_{s\sim q(s|c)}[R(s)] - \frac{1}{\beta} D_{\mathrm{KL}}(q(s|c) \,\|\, p_0(s|c))\right), \tag{9}$$

where $p_0(s|c)$ is the distribution over simulator-compatible scenes induced by the pretrained diffusion generator, $q(s|c)$ is the adapted distribution, and $R(s)$ is the utility of scene s. The expectation measures the utility under the adapted diffusion generator, while $D_{\mathrm{KL}}(q\|p_0)$ penalizes deviation from the pretrained generator and $\beta > 0$ controls the tradeoff. Later work [97–99] has explored larger-scale RL and more feedback-efficient updates within this family. In parallel, differentiable-reward fine-tuning and preference-based objectives [100–102] avoid explicit policy-gradient estimation by backpropagating through the sampler or directly optimizing diffusion likelihoods.

These methods mainly optimize image-level rewards, human preferences, or other properties of individual samples. Our feedback instead comes from navigation-policy improvement in a non-differentiable simulator and is sparse across scenes, making direct policy-gradient estimation over a long denoising trajectory unsuitable. We therefore use a bounded feedback weight in the standard denoising objective to transfer learner feedback while adapting the pretrained generator. For each generated scene s_i, we retain a denoising record

$$\tau_i = \{x_{i,t_i}, t_i, \epsilon_i, c_i\}, \tag{10}$$

where x_{i,t_i} is a noisy state, t_i is a diffusion timestep, ϵ_i is the target noise, and c_i is the condition associated with s_i. For each record, the denoising loss [18, 103] is

$$\ell_i(\theta) = \|\hat{\epsilon}_\theta(x_{i,t_i}, t_i, c_i) - \epsilon_i\|_2^2, \tag{11}$$

where $\hat{\epsilon}_\theta$ denotes the noise predicted by the denoising network parameterized by θ. Let $\widetilde{R}_{k,i}$ denote the clipped learning-progress feedback $R_{k,i}$, and let $\mathcal{P}_k$ index the scene records used for generator adaptation. We convert these feedback values into weights:

$$w_{k,i} = \frac{\exp(\widetilde{R}_{k,i})}{\frac{1}{|\mathcal{P}_k|}\sum_{j\in\mathcal{P}_k}\exp(\widetilde{R}_{k,j})}, \tag{12}$$

where $i \in \mathcal{P}_k$. Higher utility yields a larger relative weight, while normalization keeps the average weight equal to one. The feedback-weighted denoising loss is

$$\mathcal{L}_{\mathrm{fb}}(\theta) = \frac{1}{|\mathcal{P}_k|}\sum\nolimits_{i\in\mathcal{P}_k} w_{k,i}\ell_i(\theta), \tag{13}$$

where higher utility increases a scene's relative contribution to the denoising update.

Parameter-Efficient Adaptation. We adapt the generator with LoRA adapters [104], following parameter-efficient diffusion adaptation [105, 106]. The diffusion backbone remains frozen, and only the low-rank adapters are optimized, reducing the cost of full-parameter fine-tuning.

4 Experiments

Experiments examine our loop between scene generation and navigation learning. We first show that the complete pipeline improves generalization across synthetic, generated, and external evaluation distributions (§4.2). We then show that policy feedback produces a sustained shift in the generator distribution (§4.3). We further show that policies trained with and without feedback adaptation exhibit distinct navigation behaviors (§4.4). Finally, we localize these behavioral differences to the learned action-selection process under matched input histories and navigation states (§4.5).

4.1 Experimental Setup

Datasets. We train DiffuScene [18] on the ProcTHOR [10] training split. The online loop uses only ProcTHOR scenes and assets. Gibson [6] and Matterport3D (MP3D) [5] are fixed external evaluation domains, excluded from training, feedback, generator adaptation, and model selection.

Implementation Details. We adopt Habitat's [2] PointNav PPO implementation and use a fixed policy architecture and PPO protocol across all comparisons. Each policy uses a ResNet-18 visual encoder trained end-to-end from random initialization and a single-layer GRU with hidden size 512 in the RGBD setting. All policies are trained for 75 million simulator steps; online iterations warm-start from the

Table 1: **PointNav Performance Across Evaluation Distributions** (§4.2) with RGB-D observations. Gray rows denote domain-specialized reference policies trained on Gibson or MP3D training sets.

Metrics	Training Policy	ProcTHOR	Initial Generator	Feedback Generator	Gibson [6]	MP3D [5]	Overall Avg.
SR ↑	ProcTHOR	**0.9920**	0.7860	0.9560	0.9145	0.7778	0.8853
	Frozen Generator	0.9480	0.8800	0.9420	0.9175	0.7394	0.8854
	Ours Full	0.9800	**0.9620**	**0.9940**	**0.9467**	**0.8364**	**0.9438**
	Gibson-trained	0.9200	0.7440	0.8500	0.9608	0.8444	0.8638
	MP3D-trained	0.9660	0.7940	0.8620	0.9557	0.8929	0.8941
SPL ↑	ProcTHOR	**0.9213**	0.6722	0.8321	**0.7309**	0.5607	0.7434
	Frozen Generator	0.7903	0.7403	0.8133	0.7121	0.5138	0.7140
	Ours Full	0.8159	**0.7958**	**0.8732**	0.7092	**0.5979**	**0.7584**
	Gibson-trained	0.7512	0.5824	0.6707	0.8008	0.6598	0.6930
	MP3D-trained	0.7202	0.5396	0.6027	0.7218	0.6604	0.6489

preceding policy checkpoint. Fixed ProcTHOR, frozen-generator, and feedback-adapted-generator policies use matched policy architecture, PPO configuration, and training budget. Additional PointNav, generator, and feedback-adaptation hyperparameters are in *supp.* §B. Our code will be released.

4.2 PointNav Performance across Evaluation Distributions

Baselines. We compare three ProcTHOR-only protocols: *i)* ProcTHOR (ProcTHOR training scenes); *ii)* Frozen Generator (initial DiffuScene samples without feedback); and *iii)* Ours Full (the complete feedback-adapted pipeline). All use the same policy architecture, PPO configuration, and training budget. We additionally train domain-specialized reference policies on Gibson [6] and MP3D [5].

Metrics. We report standard PointNav Success Rate (SR) and Success weighted by Path Length (SPL) [54] across five held-out distributions: ProcTHOR test scenes; layouts sampled from the initial and feedback-adapted DiffuScene models; and the fixed external Gibson and MP3D domains. The initial-generator layouts characterize the prior-derived distribution, whereas feedback-adapted layouts characterize the distribution induced by policy feedback. For ProcTHOR, we sample 10 held-out scenes once and construct 50 PointNav episodes per scene. Both generator domains use layouts from the same 10 held-out conditioning IDs, with 50 independently sampled episodes per layout; Gibson and MP3D use all episodes in their official validation sets.

Performance Across Distributions. Table 1 compares the Frozen Generator and Ours Full policies across the five evaluation distributions. Compared with Frozen Generator, Ours Full improves SR across all five evaluation distributions and improves SPL on four of them, yielding the highest overall average SR and SPL. While the ProcTHOR-, Gibson-, and MP3D-trained policies achieve the best performance in their respective training domains, Ours Full attains higher average SR and SPL across all five distributions, indicating stronger overall generalization.

Learning Dynamics. Figs. 2(a–e) show the SR learning curves across the five evaluation distributions. Our full pipeline maintains a consistent advantage over Frozen Generator throughout training, with the separation emerging early and persisting to the final checkpoint. Both generated-scene policies learn rapidly: their success rates increase sharply during the initial training stage and then remain high across all distributions. Importantly, the advantage of Ours Full is retained as the generator is continually adapted, without sustained degradation at later stages.

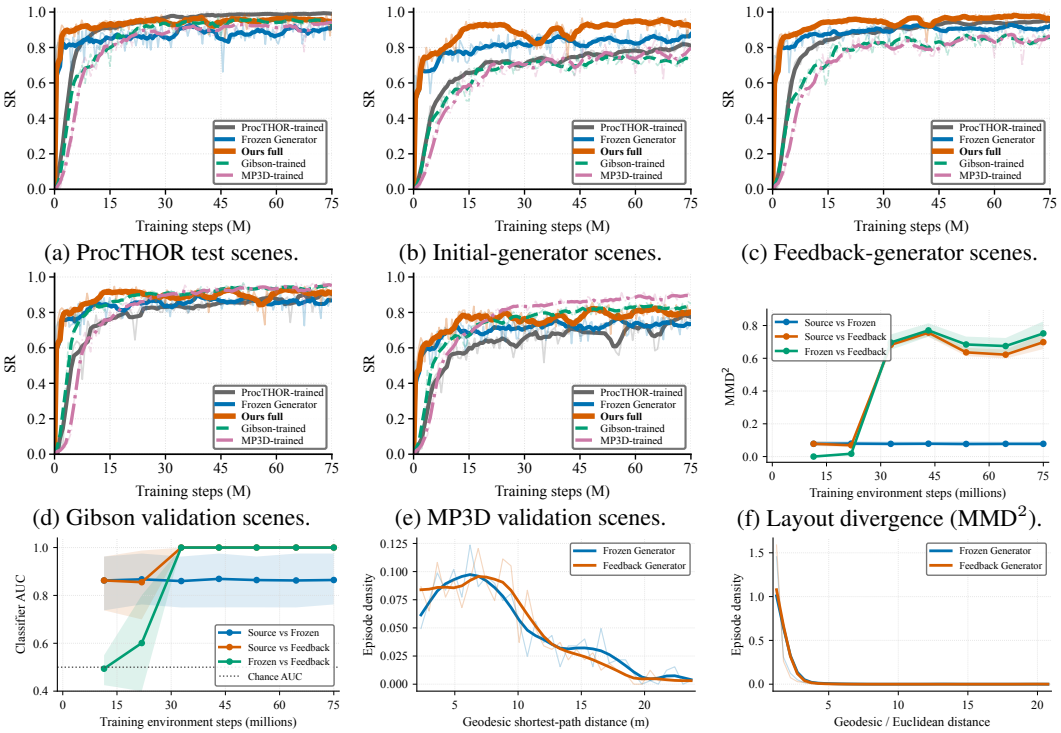
Training-Loop Ablations. Selection-only uses learning-progress feedback to select training scenes without adapting the generator. Shuffled-feedback permutes the scene-to-feedback assignments before generator updates, retaining the feedback values but breaking their association with the scenes that produced them. Table 2 compares these controls on the two external domains using the evaluation protocol of Table 1. Full achieves the highest SR and SPL on both domains, supporting the use of scene-specific feedback to adapt the generator.

Table 2: **Training-Loop Ablations** (§4.2). Selection-only uses learning progress to select scenes with a frozen generator; Shuffled-feedback permutes scene-to-feedback assignments before generator updates.

Metric	Selection-only	Shuffled-feedback	Full
Gibson SR	0.8994	0.9225	**0.9467**
Gibson SPL	0.6234	0.5964	**0.7092**
MP3D SR	0.7758	0.7879	**0.8364**
MP3D SPL	0.5339	0.5027	**0.5979**

4.3 Feedback-Induced Generator Distribution Shift

Metrics. We use the squared maximum mean discrepancy (MMD^2) [107] and the cross-validated AUC of a classifier [108] trained to distinguish layouts drawn from the two distributions.



(a) ProcTHOR test scenes. (b) Initial-generator scenes. (c) Feedback-generator scenes.

(d) Gibson validation scenes. (e) MP3D validation scenes. (f) Layout divergence (MMD^2).

(g) Layout separability (AUC). (h) Induced geodesic distance. (i) Induced path detour ratio.

Figure 2: **Learning Dynamics, Generator Distribution Shift, and Induced Task Geometry** (§4.2,§4.3). (a–e) Success-rate learning curves for ProcTHOR-trained, Frozen Generator, Full, and domain-specialized reference policies. Faint traces are raw evaluations and bold traces are 3M-step moving averages. (f–g) MMD^2 and classifier AUC show that feedback separates the generator from both source and Frozen distributions after 32.8M steps. (h–i) Feedback induces modestly shorter paths, while task-distance and path-geometry distributions broadly overlap.

Distribution Analysis. We compare held-out source ProcTHOR layouts, Frozen Generator layouts, and feedback-adapted layouts in a shared descriptor space. Each comparison uses 20 layouts from each distribution and is repeated over 200 balanced selections. Table 3 reports the mean MMD^2 and AUC, and Figs. 2(f–g) show the 2.5–97.5 percentile selection intervals. The Frozen Generator has a stable offset from the source layouts ($MMD^2 \approx 0.078$, AUC≈ 0.86). At 11.4M and 21.9M steps, the feedback-adapted generator remains close to Frozen. By 32.8M steps, feedback-adapted layouts are clearly separated from both Frozen and source layouts. This separation persists through the final checkpoint. Figs. 2(f–g) further show that this shift is robust to the sampled layouts. Therefore, feedback produces a sustained shift in the generator distribution.

Table 3: **Pairwise Distribution Comparisons** (§4.3). Results average 200 balanced selection draws with 20 layouts per distribution.

Metric	Comparison	11.4M	21.9M	32.8M	75.0M
MMD^2	Source–Frozen	0.078	0.079	0.078	0.078
	Source–Full	0.078	0.069	0.682	0.698
	Frozen–Full	0.000	0.017	0.696	0.752
AUC	Source–Frozen	0.86	0.87	0.86	0.86
	Source–Full	0.86	0.86	1.00	1.00
	Frozen–Full	0.49	0.60	1.00	1.00

Induced Navigation-Task Distribution. We additionally characterize the PointNav task distributions induced by the two generators. Figs. 2(h–i) compare 500 episodes per generator, sampled from the same 10 held-out conditioning IDs. Feedback yields a modestly shorter mean geodesic distance (8.25 m versus 8.78 m), but the distance distributions strongly overlap (KS statistic 0.066, $p = 0.226$), and their geodesic-to-Euclidean ratios are similarly distributed. Thus, feedback changes the induced task distribution only modestly, without large changes in either path length or path geometry.

4.4 Navigation Behavior of Learned Policies

Paired Navigation Behavior on External Domains. We compare the behavior of the final Full and Frozen Generator policies on the external episodes solved by both policies (878 Gibson; 329 MP3D). Table 4 asks how the two learned policies traverse the same task, rather than which tasks they solve. Table 4 reports SPL, the efficiency of a successful path; action count, the trajectory length; turn rate, the fraction of actions that are turns; and position repetition, the fraction of trajectory positions that

Table 4: **Paired navigation behavior** (§4.4). Means on jointly solved episodes (Gibson: 878; MP3D: 329). The colored annotation next to Full is its difference from Frozen: green is better and red is worse. *: the 95% bootstrap CI crosses zero.

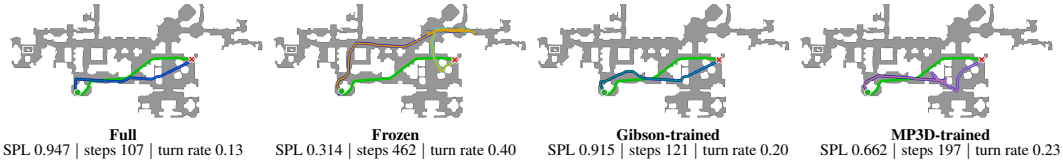
Domain	Metric	Frozen	Ours Full
Gibson	SPL ↑	**0.783**	0.756 ↓0.028
	Actions ↓	**60.3**	63.6 ↑3.27*
	Turn rate ↓	33.64%	**32.06%** ↓1.58%
	Position repetition ↓	36.50%	**34.99%** ↓1.52%
MP3D	SPL ↑	0.702	**0.743** ↑0.041
	Actions ↓	115.8	**99.7** ↓16.08
	Turn rate ↓	33.22%	**28.20%** ↓5.02%
	Position repetition ↓	36.45%	**32.46%** ↓3.98%

Table 5: **First-turn alignment with initial goal bearing** (§4.4). Entries give the fraction of episodes whose first turn is left when the goal begins on the left, or right when it begins on the right. The front-facing group is omitted.

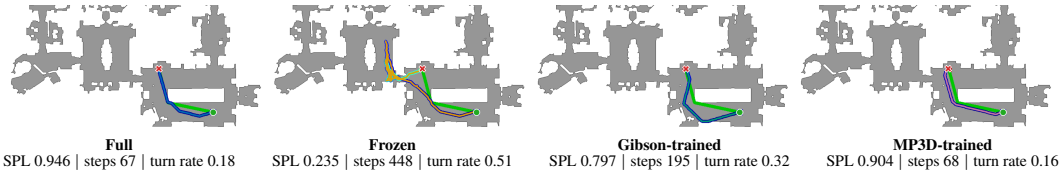
Domain	Policy	Goal left	Goal right
Gibson	Frozen Generator	87.1%	82.6%
	Ours Full	80.2%	86.7%
	Gibson-trained	100.0%	0.0%
	MP3D-trained	0.0%	100.0%
MP3D	Frozen Generator	84.9%	79.5%
	Ours Full	78.5%	91.5%
	Gibson-trained	100.0%	0.0%
	MP3D-trained	0.0%	100.0%



(a) Gibson, Greigsville, episode 431 (geodesic distance 5.80 m).



(b) MP3D, 2azQ1b91cZZ, episode 36 (geodesic distance 22.25 m).



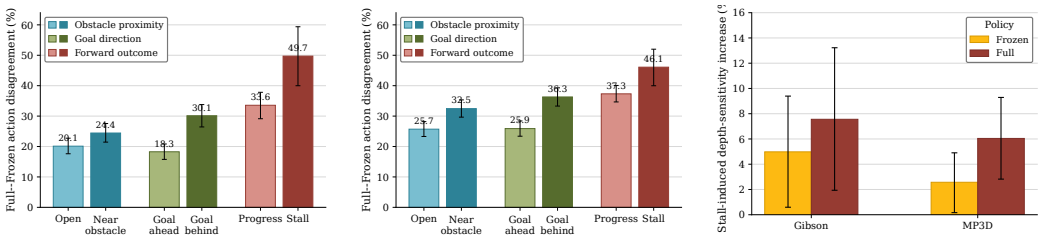
(c) MP3D, Z6MFQCViBuw, episode 39 (geodesic distance 12.80 m).

Figure 3: **Illustrative Matched-Success Navigation Trajectories** (§4.4). All four policies solve each shown episode. Green curves denote shortest geodesic paths; blue and orange curves show the executed trajectories. Examples are selected from matched-success episodes with at least 3 m geodesic distance and ranked by trajectory separation, action count, turning, and position repetition.

revisit an earlier location. On Gibson, our Full pipeline turns and revisits less. Notably, its 3.27-action increase is inconclusive because the 95% bootstrap CI crosses zero. On MP3D, our Full pipeline improves all four measures: it achieves higher SPL while using fewer actions, turning less often, and revisiting fewer positions. These results demonstrate that the learned policies exhibit distinct navigation behaviors beyond their differences in aggregate task success.

Action-Selection Patterns. We group episodes by initial goal bearing and examine the first turn. Table 5 shows that Full and Frozen select turns by bearing, but Full matches right-sided goals more often than Frozen, reflecting an asymmetric initial action selection. In contrast, the two in-domain reference checkpoints exhibit opposite single-direction turning conventions: across both validation sets, Gibson-trained executes 63,777 left turns and no right turns, whereas MP3D-trained executes 53,894 right turns and no left turns. Detailed turn-action counts appear in *supp.* §C.1.

Qualitative Trajectory Comparison. Fig. 3 compares policies on the same matched-success episodes. Full commits to a route early and follows near-geodesic paths with few corrective turns, whereas Frozen repeatedly revises its route around obstacles and junctions, causing backtracking and loops. Gibson-trained and MP3D-trained policies instead begin with left and right turns, respectively, rather than selecting the direction from the initial goal bearing (Table 5).



(a) Gibson action disagreement. (b) MP3D action disagreement. (c) Depth response during recovery.

Figure 4: **State-Conditioned Policy Diagnosis** (§4.5). (a–b) Each bar is the percentage of identical-input states in which Full and Frozen select different highest-probability actions. (c) Each bar tests whether replacing the current depth image has a larger effect on the action distribution after a stalled rather than progressing forward action. Error bars are 95% trajectory-clustered bootstrap intervals. A stalled forward is an executed forward action with less than 1 cm change in PointGoal distance.

4.5 Where Do the Learned Policies Differ?

So far, we have shown that feedback adaptation changes both the generated scene distribution and the navigation behavior of the resulting policy. The observed trajectory differences, however, do not reveal whether Full and Frozen differ only because they visit different states or because they map the same observation history to different actions. We next examine whether these behavioral differences reflect distinct decision rules and, if so, where they emerge during navigation.

Same-Input Policy Comparison. One policy controls the environment while the other acts as a shadow policy. Both receive the same observations and executed-action history while maintaining separate recurrent states; we repeat the comparison in both rollout directions. As shown in Table 6, their highest-probability actions disagree on 32.5% of Gibson steps and 39.2% of MP3D steps. Full also has lower action entropy in both domains. Thus, their trajectory differences reflect distinct observation-history-to-action mappings rather than state visitation alone.

Table 6: **Same-Input Policy Comparison** (§4.5). Action disagreement and entropy are averaged over Full-controlled and Frozen-controlled rollouts.

Domain	Disagreement	Full entropy	Frozen entropy
Gibson	0.325	0.242	0.383
MP3D	0.392	0.291	0.484

Where Policies Disagree. We sample 200 episodes for each external domain and each primary-policy rollout direction, evaluating both policies on the same state every fifth step. This yields 6,710 Gibson states and 14,430 MP3D states. Figs. 4(a–b) report the percentage of identical-input states in which Full and Frozen select different highest-probability actions. On Gibson, disagreement is higher near obstacles than in open space, when the goal is behind rather than ahead, and after a stalled rather than progressing forward action. MP3D exhibits the same pattern. Across both domains, Full and Frozen differ significantly more in navigation states requiring corrective control, indicating that their learned decision rules diverge primarily during challenging local adjustments.

Depth Use During Recovery. At each sampled state, we replace the current depth image with three genuine donor observations without advancing the simulator and measure the total variation (TV) of the action distribution. Fig. 4(c) directly compares this depth-replacement response during normal progress and after a stalled forward action. Full exhibits greater depth sensitivity after stalled forward actions, suggesting that it relies more strongly on local depth cues during recovery.

Summary. Experiments in this section show that the policy trained with our Full pipeline selects different actions under identical observation and action histories. Thus, scene distribution changes reshape the policy's underlying action-selection function, altering how observation histories are translated into actions rather than only the states it visits or its aggregate success.

5 Conclusion

We present a learner-aware framework that closes the loop between 3D scene generation and embodied navigation. The framework evaluates matched navigation episodes before and after each policy update, uses the resulting scene-level learning progress to adapt a layout diffusion generator, and trains the policy on the evolving generated distribution. Experiments show that feedback adaptation improves PointNav generalization across five evaluation distributions, produces a sustained shift in the generator distribution, and is associated with distinct navigation behaviors and action-selection mappings. We believe that our results suggest a learner-aware direction for embodied generalization. Scene generation can thus adapt to a policy's evolving needs, shaping the training experiences from which it learns.

AI USE STATEMENT

In this work, generative AI tools were used for manuscript language editing, and for preliminary discussion of possible experimental settings. All final experimental settings were determined by the authors. Generative AI was not used to implement the proposed method, conduct experiments, or perform any other task requiring disclosure under the ICLR 2027 policy. All AI-assisted content was reviewed by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY

Our framework is implemented in PyTorch. We run experiments on a single NVIDIA GeForce RTX 4090 GPU with 48 GB memory. Our code will be released. More details are in the **Appendix**.

REFERENCES

[1] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3d environment for visual ai. *CoRR*, 2017. 1, 2

[2] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 1, 2, 4, 5, 18, 20

[3] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. In *NeurIPS*, 2021. 1

[4] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars, and robots. In *ICLR*, 2024. 1

[5] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pages 667–676, 2017. 1, 2, 5, 6, 18, 20

[6] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. 5, 6, 18, 20

[7] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. HM3D: 1000 large-scale 3d environments for embodied ai. In *NeurIPS*, 2021. 1

[8] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, 2019. 1

[9] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An open simulation-to-real embodied ai platform. In *CVPR*, 2020. 1, 2

[10] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied ai using procedural generation. In *NeurIPS*, 2022. 1, 2, 4, 5, 18, 20

[11] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat synthetic scenes dataset (HSSD-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. In *CVPR*, 2024. 1

[12] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2020. 1, 2

[13] Matt Deitke, Rose Hendrix, Ali Farhadi, Kiana Ehsani, and Aniruddha Kembhavi. Phone2Proc: Bringing robust robots into our chaotic world. In *CVPR*, 2023. 1, 2

[14] Weipeng Zhong, Peizhou Cao, Yichen Jin, Luo Li, Wenzhe Cai, Jingli Lin, Hanqing Wang, Zhaoyang Lyu, Tai Wang, Xudong Xu, Bo Dai, and Jiangmiao Pang. Internscenes: A large-scale simulatable indoor scene dataset with realistic layouts. In *NeurIPS*, 2025. 1, 2

[15] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3d embodied ai environments. In *CVPR*, 2024. 1, 3

[16] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *3DV*, 2021. 1, 2

[17] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. ATISS: Autoregressive transformers for indoor scene synthesis. In *NeurIPS*, 2021. 2

[18] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *CVPR*, 2024. 2, 4, 5, 18, 20

[19] Xiuyu Yang, Yunze Man, Jun-Kun Chen, and Yu-Xiong Wang. Scenecraft: Layout-guided 3d scene generation. In *NeurIPS*, 2024. 2

[20] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *CVPR*, 2024. 2

[21] Bowei Jiang, Tongyuan Bai, Peng Zheng, Tieru Wu, and Rui Ma. Nav2scene: Navigation-driven fine-tuning for robot-friendly scene generation. *Graph. Models*, 2025. 3

[22] Nicholas Ezra Pfaff, Hongkai Dai, Sergey Zakharov, Shun Iwase, and Russ Tedrake. Steerable scene generation with post training and inference-time search. In *CoRL*, 2025. 1, 3

[23] Pierre-Yves Oudeyer, Frédéric Kaplan, and Verena V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE TEVC*, 2007. 2

[24] Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *RAS*, 2013.

[25] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *CoRL*, 2020. 2, 3

[26] 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 TOG*, 2011. 2

[27] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM TOG*, 2012.

[28] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *CVPR*, 2018.

[29] Song-Hai Zhang, Shao-Kui Zhang, Yuan Liang, and Peter Hall. A survey of 3d indoor scene synthesis. *JCST*, 2019.

[30] Akshay Gadi Patil, Supriya Gadi Patil, Manyi Li, Matthew Fisher, Manolis Savva, and Hao Zhang. Advances in data-driven analysis and synthesis of 3d indoor scenes. *CGF*, 2024.

[31] Beichen Wen, Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. 3d scene generation: A survey. *IJCV*, 2026.

[32] Tianwei Ye, Yifan Mao, Minwen Liao, Jian Liu, Chunchao Guo, Dazhao Du, Quanxin Shou, Fangqi Zhu, and Song Guo. 3d generation for embodied ai and robotic simulation: A survey. *CoRR*, 2026. 2

[33] Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM TOG*, 2018. 2

[34] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. GRAINS: Generative recursive autoencoders for indoor scenes. *ACM TOG*, 2019. 2

[35] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Jiaming Wang, Cao Li, Zengqi Xun, Chengyue Sun, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3D-FRONT: 3d furnished rooms with layouts and semantics. In *ICCV*, 2021. 2, 4

[36] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *ACM TOG*, 2020. 2

[37] Haitao Yang, Zaiwei Zhang, Siming Yan, Haibin Huang, Chongyang Ma, Yi Zheng, Chandrajit Bajaj, and Qixing Huang. Scene synthesis via uncertainty-driven attribute synchronization. In *ICCV*, 2021. 2

[38] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. PlanIT: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM TOG*, 2019. 2

[39] Lin Gao, Jia-Mu Sun, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas, and Jie Yang. Scenehgn: Hierarchical graph networks for 3d indoor scene generation with fine-grained geometry. *IEEE TPAMI*, 2023. 2

[40] Guangyao Zhai, Evin Pınar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. CommonScenes: Generating commonsense 3d indoor scenes with scene graph diffusion. In *ICCV*, 2023.

[41] Gege Gao, Weiyang Liu, Anpei Chen, Andreas Geiger, and Bernhard Schölkopf. GraphDreamer: Compositional 3d scene synthesis from scene graphs. In *CVPR*, 2024. 2

[42] Siyi Hu, Diego Martín Arroyo, Stephanie Debats, Fabian Manhardt, Luca Carlone, and Federico Tombari. Mixed diffusion for 3d indoor scene synthesis. In *WACV*, 2026. 2

[43] Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Xuehai He, S. Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional visual planning and generation with large language models. In *NeurIPS*, 2023. 2

[44] Qihang Zhang, Chaoyang Wang, Aliaksandr Siarohin, Peiye Zhuang, Yinghao Xu, Ceyuan Yang, Dahua Lin, Bo Dai, Bolei Zhou, Sergey Tulyakov, and Hsin-Ying Lee. SceneWiz3D: Towards text-guided 3d scene composition. In *CVPR*, 2024.

[45] Chenguo Lin and Yadong Mu. InstructScene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. *CoRR*, 2024. 2

[46] Daniel Ritchie, Kai Wang, and Yu-An Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *CVPR*, 2019. 2

[47] Xinjie Wang, Liu Liu, Yu Cao, Ruiqi Wu, Wenkang Qin, Dehui Wang, Wei Sui, and Zhizhong Su. EmbodiedGen: Towards a generative 3d world engine for embodied intelligence. *CoRR*, 2025. 2

[48] Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. SAGE: Scalable agentic 3d scene generation for embodied ai. In *CVPR*, 2026. 2

[49] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *CoRR*, 2017. 2

[50] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *CoRR*, 2018. 2

[51] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 2

[52] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *CVPR*, 2020. 2

[53] Chengshu Li, Fei Xia, Roberto Martin-Martin, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. iGibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *CoRL*, 2022. 2

[54] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents. *CoRR*, 2018. 2, 4, 6

[55] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, Saurabh Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020.

[56] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *CVPR*, 2022.

[57] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martin-Martin, Fei Xia, Kent Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. BEHAVIOR: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *CoRL*, 2021.

[58] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martin-Martin, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, et al. BEHAVIOR-1K: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. In *CoRL*, 2024.

[59] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of household tasks for generalist robots. In *RSS*, 2024. 2

[60] Jing Yu Koh, Harsh Agrawal, Dhruv Batra, Richard Tucker, Austin Waters, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Simple and effective synthesis of indoor 3d scenes. In *AAAI*, 2023. 2

[61] Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-HAO Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, et al. SPOC: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. In *CVPR*, 2024. 2

[62] Zihan Wang, Seungjun Lee, Yinghao Xu, and Gim Hee Lee. Image2Sim: Scaling embodied navigation via generative neural simulator. *CoRR*, 2026. 2

[63] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, 2020. 2

[64] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009. 3

[65] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *JMLR*, 2020.

[66] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. In *IJCAI*, 2020. 3

[67] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *CoRL*, 2017. 3

[68] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, 2018.

[69] Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Self-paced deep reinforcement learning. In *NeurIPS*, 2020.

[70] Pascal Klink, Haoyi Yang, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *ICML*, 2022.

[71] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *TNNLS*, 2020.

[72] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *ICLR*, 2018. 3

[73] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. In *NeurIPS*, 2020. 3

[74] Abdus Salam Azad, Izzeddin Gur, Jasper Emhoff, Nathaniel Alexis, Aleksandra Faust, Pieter Abbeel, and Ion Stoica. CLUTR: Curriculum learning via unsupervised task representation learning. In *ICML*, 2023.

[75] Clément Romac, Rémy Portelas, Katja Hofmann, and Pierre-Yves Oudeyer. TeachMyAgent: A benchmark for automatic curriculum learning in deep rl. In *ICML*, 2021.

[76] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *ICML*, 2021.

[77] Ishita Mediratta, Minqi Jiang, Jack Parker-Holder, Michael Dennis, Eugene Vinitsky, and Tim Rocktäschel. Stabilizing unsupervised environment design with a learned adversary. In *CoLLAs*, 2023.

[78] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents. *CoRR*, 2021. 3

[79] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017. 3

[80] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart J. Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *NeurIPS*, 2020. 3

[81] Samuel Garcin, James Doran, Shangmin Guo, Christopher G. Lucas, and Stefano V. Albrecht. DRED: Zero-shot transfer in reinforcement learning via data-regularised environment design. In *ICML*, 2024.

[82] Michael Beukman, Samuel Coward, Michael Matthews, Mattie Fellows, Minqi Jiang, Michael D. Dennis, and Jakob Nicolaus Foerster. Refining minimax regret for unsupervised environment design. In *ICML*, 2024. 3

[83] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In *NeurIPS*, 2021. 3

[84] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *ICML*, 2022. 3

[85] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. In *ICLR*, 2019. 3

[86] Izzeddin Gur, Natasha Jaques, Yingjie Miao, Jongwook Choi, Manoj Tiwari, Honglak Lee, and Aleksandra Faust. Environment generation for zero-shot compositional reinforcement learning. In *NeurIPS*, 2021. 3

[87] William Liang, Sam Wang, Hung-Ju Wang, Osbert Bastani, Dinesh Jayaraman, and Yecheng Jason Ma. Environment curriculum generation via large language models. In *ICML*, 2025. 3

[88] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation. In *ICML*, 2024. 3

[89] Teresa Yeo, Dulaj Weerakoon, Dulanga Weerakoon, and Archan Misra. Towards adaptive environment generation for training embodied agents. *CoRR*, 2026. 3

[90] Haoqiang Kang, Xiaokang Ye, Yuhan Liu, Siddhant Hitesh Mantri, Lingjun Mao, James Fleming, Drishti Regmi, and Lianhui Qin. SimWorld Studio: Automatic environment generation with evolving coding agent for embodied agent learning. *CoRR*, 2026. 3

[91] Zixuan Bian, Ruohan Ren, Yue Yang, and Chris Callison-Burch. HOLODECK 2.0: Vision-language-guided 3d world generation with editing. *CoRR*, 2025. 3

[92] Xinjie Wang, Liu Liu, Taojun Ding, Andrew Choi, Chaodong Huang, Mengao Zhao, Ziang Li, Jackson Jiang, Chunlei Yu, Shengxiang Liu, Wei Xu, and Zhizhong Su. EmbodiedGen V2: An agentic, simulation-ready 3d world engine for embodied ai. *CoRR*, 2026. 3

[93] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Stephen J. Maybank, and Dacheng Tao. 3D-FUTURE: 3d furniture shape with texture. *IJCV*, 2021. 4

[94] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *CoRR*, 2023. 5

[95] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. In *NeurIPS*, 2023. 5

[96] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *ICLR*, 2024. 5

[97] Yinan Zhang, Eric Tzeng, Yilun Du, and Dmitry Kislyuk. Large-scale reinforcement learning for diffusion models. In *ECCV*, 2024. 5

[98] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M. Tseng, Sergey Levine, and Tommaso Biancalani. Feedback efficient online fine-tuning of diffusion models. In *ICML*, 2024.

[99] Zichen Miao, Jiang Wang, Ze Wang, Zhengyuan Yang, Lijuan Wang, Qiang Qiu, and Zicheng Liu. Training diffusion models towards diverse image generation with reinforcement learning. In *CVPR*, 2024. 5

[100] Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. *CoRR*, 2023. 5

[101] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *CVPR*, 2024.

[102] Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Yusuke Kato, and Kazuki Kozuka. Aligning diffusion models by optimizing human utility. In *NeurIPS*, 2024. 5

[103] Jonathan Ho, Ajay N. Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 5

[104] 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 *ICLR*, 2022. 5

[105] Ligong Han, Yinxiao Li, Han Zhang, Peyman Milanfar, Dimitris Metaxas, and Feng Yang. SVDiff: Compact parameter space for diffusion fine-tuning. In *ICCV*, 2023. 5

[106] Enze Xie, Lewei Yao, Han Shi, Zhili Liu, Daquan Zhou, Zhaoqiang Liu, Jiawei Li, and Zhenguo Li. DiffFit: Unlocking transferability of large diffusion models via simple parameter-efficient fine-tuning. In *ICCV*, 2023. 5

[107] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 2012. 6

[108] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *ICLR*, 2017. 6

[109] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *JRSSB*, 1995. 19

APPENDIX

This appendix contains additional details for the ICLR 2027 submission, titled *Embodied Learning in the Loop: Co-Adapting Policies and Environment Distributions*. The appendix is organized as follows:

- §A provides additional method details.
- §B details the experimental protocol.
- §C presents additional experimental results.
- §D provides reproducibility materials and data licenses.
- §E discusses our limitations and future work.

A Additional Method Details

A.1 ProcTHOR-to-DiffuScene Data Preparation

We construct the training corpus from ProcTHOR scene descriptions and asset metadata. Preprocessing preserves the ProcTHOR asset domain while converting scene layouts into the representation required by DiffuScene.

Scene Matching and Asset Metadata. Each source JSON file describes rooms, walls, and hierarchical object instances. We identify each object's ProcTHOR asset by its asset ID and retrieve its category ID and axis-aligned bounding-box dimensions from the asset metadata. The simulator scene-instance description aligns object placements with the source records. When several instances share an asset ID, we match them by horizontal position and remove each matched candidate from the available set to prevent duplicate assignments. We retain the object hierarchy and simulator metadata where applicable and remove unmatched objects.

Object Record Validation. We retain the ProcTHOR scenes in the training split. During preprocessing, we remove individual object records with non-finite or clearly out-of-range position coordinates; this check does not exclude their parent scenes. We map the remaining object categories to the generator's 109-category ProcTHOR vocabulary. The stored one-hot representation has 111 dimensions: 109 object categories followed by the reserved `start` and `end` symbols. DiffuScene's diffusion encoding removes `start` and uses `end` to mark empty padded slots, yielding 110 denoised class channels (109 object categories plus one empty-slot channel). Empty slots are removed before scene instantiation.

Coordinate and Orientation Alignment. We express floor plans and object layouts in a common simulator coordinate system. Floor-plan vertices are converted to Habitat-compatible horizontal coordinates, and the midpoint of their axis-aligned bounds defines the scene center. We subtract this center from object translations and use it to align the floor plan when rasterizing a 256×256 room mask; the stored floor-plan vertices remain in the transformed simulator coordinates. Object yaw angles are transformed consistently with the coordinate convention and normalized to $[-\pi, \pi)$.

We store both the transformed floor-plan vertices and the scene center. During generation, the center restores object translations to simulator coordinates before instantiation.

DiffuScene-Compatible Output. For each processed scene, the object representation is

$$\mathcal{Z}_i = \{\mathbf{c}_{i,n}, \mathbf{t}_{i,n}, \mathbf{s}_{i,n}, \mathbf{a}_{i,n}\}_{n=1}^{N_i^{\text{obj}}}, \tag{14}$$

where N_i^{obj} counts retained objects, $\mathbf{c}_{i,n}$ is a one-hot category vector, $\mathbf{t}_{i,n}$ is the centered translation, $\mathbf{s}_{i,n}$ is the asset bounding-box size, and $\mathbf{a}_{i,n}$ is the normalized yaw angle. An NPZ file stores these attributes as `class_labels`, `translations`, `sizes`, and `angles`, together with `room_layout`, `floor_plan_vertices`, and `floor_plan_centroid`.

We also export object records as text and room masks as PNG images for inspection and training-pipeline compatibility. Dataset statistics summarize translation, size, and angle ranges, class frequencies, object counts, and the coordinate center for each processed corpus.

A.2 Generator Configuration and Scene Instantiation

We train a new layout diffusion model on this corpus using DiffuScene's room-layout-conditioned scene-generation branch, without language input or object geometry features. At inference time, predicted categories are matched to ProcTHOR assets and centered translations are restored to simulator coordinates before serialization.

A.3 PointNav Episode Construction

For each instantiated scene, we use Habitat's PointNav episode generator to sample navigable start–goal pairs. The generator is controlled by the start-distance range, the geodesic-to-Euclidean distance constraint, the retry budget, the shortest-path success criterion, and the maximum episode length. Episodes that do not satisfy the configured constraints are not retained, and a scene is omitted from the corresponding policy update when the required number of valid episodes cannot be generated. Training episodes and learning-progress episodes are generated separately so that the latter can remain fixed before and after the policy update.

A.4 Learning-Progress Feedback and Generator Adaptation

Distributional Interpretation of Learning-Progress Feedback. This section interprets the feedback objective in §3.5 as adaptation of a scene distribution, rather than unrestricted generation of novel scenes. Let $p_0(s \mid c)$ denote the pretrained scene prior and let $q(s \mid c)$ denote an adapted distribution. For this idealized analysis, let $R_k(s)$ denote a scene-utility function whose observed values on evaluated scenes s_i are $R_{k,i}$ from the matched pre- and post-update evaluations in §3.4. An idealized feedback objective is

$$\mathcal{J}_k(q) = \mathbb{E}_{s\sim q(s|c)}[R_k(s)] - \frac{1}{\beta} D_{\mathrm{KL}}\big(q(s \mid c) \,\|\, p_0(s \mid c)\big), \tag{15}$$

where $\beta > 0$ controls the preference for utility relative to preservation of the pretrained prior. Holding c fixed, optimizing this objective over normalized distributions gives the exponentially tilted distribution

$$q_k^\star(s \mid c) = \frac{p_0(s \mid c)\exp\big(\beta R_k(s)\big)}{Z_k(c)}, \qquad Z_k(c) = \int p_0(s' \mid c)\exp\big(\beta R_k(s')\big)\, ds'. \tag{16}$$

Equivalently, the relative density correction satisfies

$$\frac{q_k^\star(s \mid c)}{p_0(s \mid c)} \propto \exp\big(\beta R_k(s)\big). \tag{17}$$

This idealized tilt favors higher-utility scenes relative to lower-utility scenes, with the pretrained prior as its reference distribution. Since R_k depends on the current policy, the preferred distribution changes across outer iterations. This objective provides a policy-conditioned distributional interpretation; our implementation does not optimize it directly.

Practical Surrogate. The ideal distribution is not directly available: scene utility is observed only after simulator interaction and a policy update, is non-differentiable with respect to the diffusion parameters, and is measured for only a finite set of sampled scenes. For a generated scene s_i and its retained denoising record τ_i, the standard diffusion denoising loss $\ell_i(\theta)$ provides a surrogate for fitting the corresponding conditional diffusion trajectory. Reweighting these records by a bounded, normalized monotone transformation of $R_{k,i}$ fits the adapted model more strongly to trajectories associated with higher observed utility. The exponential weights in *supp.* §A.4 mirror the utility dependence of the idealized tilt at the sample level, but weighted denoising neither recovers $q_k^\star$ nor explicitly enforces its KL penalty. It is a practical feedback-weighted surrogate, not a direct policy-gradient estimator over the full diffusion trajectory.

Scope and Stabilization. This interpretation does not claim exact optimization of the KL-regularized objective or guarantee out-of-distribution generalization. It describes how learning-progress weighting favors sampled generation records with higher observed utility while adaptation starts from the pretrained generator.

Utility-Based Feedback Weighting. We convert clipped learning-progress feedback into positive, mean-one weights, as in Eq. 12. For brevity, we omit the iteration index k here: $\widetilde{R}_i$ denotes clipped feedback, and $\mathcal{P}$ denotes the set of scene indices used for generator adaptation:

$$w_i = \frac{\exp(\widetilde{R}_i)}{\frac{1}{|\mathcal{P}|}\sum_{j\in\mathcal{P}}\exp(\widetilde{R}_j)}. \tag{18}$$

The resulting weights preserve the ordering of scene utilities and give scenes with higher utility a larger contribution to the denoising update. The corresponding weighted loss is

$$\mathcal{L}_{\mathrm{fb}}(\theta) = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} w_i \ell_i(\theta), \tag{19}$$

where $\mathcal{P}$ denotes the scene indices used for generator adaptation.

Table 7: **Implementation and Sampling Hyperparameters** (*supp.* §B). Settings used in the reported experiments.

Component	Setting
PPO optimizer	Adam; learning rate 5×10^{-5}; $\epsilon = 10^{-5}$
PPO update	16 environments; 128 rollout steps; 4 mini-batches; 1 epoch
PPO objective	clip 0.10; $\gamma = 0.99$; GAE $\lambda = 0.95$; normalized advantages
PPO regularization	value-loss coefficient 0.5; entropy coefficient 0.002; gradient norm 0.2
Generator adapters	LoRA rank 16; $\alpha = 32$; dropout 0; Adam learning rate 3×10^{-5}
Generator update	batch size 8; 2 epochs; at most 12 optimizer steps per outer iteration; gradient norm 1.0
Training episode sampling	300 valid episodes per retained scene; at most 5 sampling rounds and 20 retries per target
Start–goal constraints	geodesic distance in $[2, 24]$ m; geodesic/Euclidean ratio at least 1.1
Episode and feedback evaluation	at most 500 actions; success distance 0.2 m; 8 paired evaluation episodes per retained scene

B Detailed Experimental Setup

B.1 Data Splits and Domain Isolation

We train the initial DiffuScene [18] layout generator on the ProcTHOR [10] training split. During the online loop, predicted layouts are instantiated exclusively with ProcTHOR assets, and the PointNav policy is trained on episodes sampled from these generated environments. Consequently, the generator prior, scene instantiation, policy training, and learning-progress feedback all access only ProcTHOR scenes and assets. Gibson [6] and MP3D [5] serve as fixed external evaluation domains. Their validation episodes are used only to measure policy performance; no Gibson or MP3D scenes, assets, episodes, or evaluation metrics are exposed to any component of the online loop, including PPO training, learning-progress computation, and generator adaptation. This restriction does not apply to the separately trained Gibson- and MP3D-trained reference policies: those policies use their respective in-domain training splits only to provide domain-specific comparison points, remain outside the online loop, and never influence generator adaptation or model selection.

B.2 Policy, Generator, and Training Configuration

We adopt Habitat's [2] PointNav PPO implementation and use a fixed policy architecture and PPO protocol across all comparisons. Each policy uses a ResNet-18 visual encoder trained end-to-end from random initialization and a single-layer GRU with hidden size 512. Under the RGBD setting, RGB and depth observations have resolution 256×256 with a horizontal field of view of $90°$; the agent additionally receives a two-dimensional polar PointGoal sensor with GPS and compass. The action space comprises a 0.25 m forward action, $30°$ turns, and STOP; episodes last for at most 500 steps and use a success distance of 0.2 m. All policies are trained for 75 million simulator steps. In the online pipeline, each outer iteration warm-starts policy parameters from the preceding checkpoint. At each outer iteration, the current generator produces 25 candidate ProcTHOR-asset scenes. A scene is retained for policy training only if the sampler obtains all 300 valid PointNav training episodes; otherwise, that scene is omitted from the current policy update. Eight independently sampled episodes per retained scene are reserved for the paired learning-progress evaluation. The resulting feedback updates only LoRA adapters of the ProcTHOR-trained DiffuScene model, while its diffusion backbone remains frozen. Each generated layout contains at most 250 object slots and uses a 110-dimensional diffusion class representation (109 object categories plus one empty-slot channel). We use DiffuScene without language or partial-scene conditioning and omit its mesh-derived latent shape features. Fixed ProcTHOR, frozen-generator, and feedback-adapted-generator policies are trained with matched policy architecture, PPO configuration, and training budget. Gibson and MP3D validation results are not used for hyperparameter selection, checkpoint selection, or any online update. Table 7 reports the remaining optimization and sampling hyperparameters.

Table 8: **Raw Turn-Action Counts for Four Policies on Two External Domains** (*supp.* §C.1). "First" columns count each episode's initial turn; "All" columns count turns over complete trajectories. All policies are evaluated on the same 994 Gibson and 495 MP3D episodes.

Domain	Policy	Episodes	First Left	First Right	All Left	All Right
Gibson	Frozen Generator	994	525	469	17,593	15,648
	Ours Full	994	475	519	14,217	17,648
	Gibson-trained	994	994	0	30,012	0
	MP3D-trained	994	0	994	0	31,437
MP3D	Frozen Generator	495	257	238	23,836	22,616
	Ours Full	495	212	283	17,900	16,320
	Gibson-trained	495	495	0	33,765	0
	MP3D-trained	495	0	495	0	22,457

B.3 Computational Resources

All reported runs used a single NVIDIA GeForce RTX 4090 GPU with 48 GB of device memory. The navigation policy has 5.82M parameters, and generator adaptation updates 1.72M LoRA parameters (approximately 0.94% of the full generator). For one completed Full run, the recorded cumulative runtime was 141.15 hours, including scene generation, episode construction, PPO training, feedback evaluation, and generator updates; benchmark evaluations also ran asynchronously on the same GPU. Downtime between interrupted sessions is excluded. The maximum sampled device-memory usage in the monitored resumed segment was 28.25 GiB.

B.4 Evaluation Protocol

We report SR and SPL across ProcTHOR test scenes, layouts sampled from the initial and feedback-adapted generators, and the Gibson and MP3D validation domains. ProcTHOR uses 10 held-out scenes with 50 episodes per scene; both generator domains use the same 10 held-out conditioning IDs with 50 independently sampled episodes per layout; Gibson and MP3D use their official validation episodes.

C Additional Experimental Results

C.1 Turn-Action Audit

Table 8 provides the raw counts underlying the first-turn alignment results in Table 5. The analysis covers the same official validation episodes for all four policies: 994 episodes from 14 Gibson scenes and 495 episodes from 11 MP3D scenes. Every evaluated trajectory contains at least one turn. Counts are computed from the actions actually executed by the Habitat environment under stochastic policy evaluation; in the standard action mapping, action 2 is Turn Left and action 3 is Turn Right. Gibson-trained uses only left turns and MP3D-trained uses only right turns in every episode and every evaluated scene.

C.2 Descriptor-Level Distributional Tests

We complement the aggregate MMD^2 and classifier-AUC results in §4.3 with descriptor-level tests between Frozen and feedback-adapted layouts. These tests use 109 category-composition and 48 layout features. Descriptor-wise KS tests compare 40 layout pairs at each checkpoint, with Benjamini–Hochberg-adjusted q-values. At 11.4M and 21.9M training steps, none of the tested descriptors differs significantly after correction. At 32.8M and 75.0M steps, many do. These counts are consistent with the later separation seen in the aggregate metrics, but they quantify neither the magnitude nor the training benefit of any individual descriptor change. The number of significant layout tests need not increase monotonically with training time.

Table 9: **Descriptor-Level Tests** (*supp.* §C.2). Counts significant after Benjamini–Hochberg correction [109] ($q < 0.05$).

Training steps	Categories	Layout
11.4M	0/109	0/48
21.9M	0/109	0/48
32.8M	48/109	44/48
75.0M	57/109	40/48

C.3 Within-Condition Layout Variation

The distributional comparisons in §4.3 measure separation between generators. Here, we examine variation among layouts generated under the same room condition. We use 20 held-out conditioning IDs and two diffusion seeds per condition, yielding 40 layouts per generator. Conditions and seeds are matched between Frozen and the final analyzed Full generator checkpoint. For each condition c,

let $d_{\text{Frozen}}(c)$ and $d_{\text{Full}}(c)$ be the distance between its two layouts from each generator. We report

$$\rho = \frac{\frac{1}{20}\sum_{c=1}^{20} d_{\text{Full}}(c)}{\frac{1}{20}\sum_{c=1}^{20} d_{\text{Frozen}}(c)}. \tag{20}$$

A ratio above one indicates greater mean within-condition variation in the measured descriptors. We obtain 95% percentile intervals from 20,000 paired bootstrap resamples of the conditions. These intervals reflect variation across conditions, not across independently trained generators.

We measure distances using scene descriptors. Category-composition distance is half the ℓ_1 distance between category-proportion vectors. Layout distance is the root-mean-square difference after standardizing descriptors by their sample standard deviations across the 40 Frozen layouts; the same scales are used for Full. We exclude Frozen-constant features, room-mask occupancy, floor area, outside-floor fraction, and category entropy and count. The spatial-only variant retains object-position statistics, inter-object distances, spatial extents, and orientation concentration, but excludes count, size, footprint, and overlap descriptors.

Table 10: **Within-Condition Layout Variation** (*supp.* §C.3). Mean Full/Frozen distance ratios with gray 95% paired-bootstrap intervals.

Descriptor	Full/Frozen
Category composition	0.952 [0.858, 1.067]
Layout	1.612 [1.283, 1.995]
Spatial only	1.350 [1.073, 1.686]

Table 10 shows larger within-condition layout and spatial-descriptor distances for Full, with paired intervals above one. The category-composition ratio is near one, and its interval includes one. Thus, the sampled conditions show no overall contraction in these spatial descriptors.

D Reproducibility Materials

D.1 Framework Pseudocode

Algorithm 1 provides the pseudocode for our framework.

D.2 Data and Asset Licenses

We conduct our experiments with three scene datasets (*i.e.*, ProcTHOR [10], Gibson [6], and MP3D [5]) and two open-source software resources (*i.e.*, Habitat-Sim [2] and DiffuScene [18]), each used under its respective license or access terms. We do not redistribute the original dataset assets or scans. ProcTHOR (`https://github.com/allenai/procthor`) is released under the Apache License 2.0. Gibson (`https://github.com/StanfordVL/GibsonEnv`) requires users to sign its database license agreement before accessing the scene database; the corresponding access instructions are provided here. MP3D (`https://niessner.github.io/Matterport/`) is available for academic, non-commercial research under Matterport's terms and conditions. Habitat-Sim (`https://github.com/facebookresearch/habitat-sim`) is released under the MIT License. DiffuScene (`https://github.com/tangjiapeng/DiffuScene`) is released under its non-commercial research, education, and evaluation license.

E Limitations and Future Work

Our experiments study learner-aware scene generation for RGB-D PointNav. ProcTHOR assets make the generated layouts executable in the simulator, but the navigation task does not require the agent to manipulate objects or change their states. The reported gains therefore establish benefits for navigation; they do not evaluate whether feedback-adapted scene generation also benefits tasks in which success depends on object affordances and interaction outcomes.

Future work can use the interactive ProcTHOR assets to extend the feedback loop to manipulation and multi-stage embodied tasks. Such an extension would require task-specific episode construction and learning-progress measures that reflect object states and action consequences as well as navigation. It would allow us to study how learner-aware scene generation transfers across different embodied tasks.

Algorithm 1 Pseudo-code of learner-aware scene generation and navigation training.

```
# theta: scene-generator parameters; phi: navigation-policy parameters
# k: outer iteration; C: room-layout conditions
initialize theta, phi

for k in range(num_outer_iterations):
    current_scenes = []
    current_episodes = []
    current_records = []

    # Generate and instantiate simulator-compatible scenes
    for c_i in sample_layout_conditions():
        z_i, tau_i = GENERATE_LAYOUT(theta, c_i, return_trajectory=True)
        z_i = POST_PROCESS_LAYOUT(z_i)

        # Match each object to the closest asset within its category
        assets_i = MATCH_ASSETS_BY_CATEGORY_AND_SIZE(z_i)
        scene_i = INSTANTIATE_SCENE(z_i, assets_i)
        if not SERIALIZABLE_AND_LOADABLE(scene_i):
            continue

        # Construct training and learning-progress episodes separately
        train_eps_i = POINTNAV_EPISODES(scene_i, task_constraints)
        lp_eps_i = FIXED_LP_EPISODES(scene_i, task_constraints)
        if not ENOUGH_EPISODES(train_eps_i, lp_eps_i):
            continue

        current_scenes.append(scene_i)
        current_episodes.extend(train_eps_i)

        # Keep the same episodes for before/after evaluation
        score_before_i = EVALUATE(phi, lp_eps_i)
        current_records.append((scene_i, tau_i, lp_eps_i, score_before_i))

    # Train the navigation policy on current episodes
    phi = PPO(phi, current_episodes)

    # Measure scene-level learning progress after the policy update
    feedback_records = []
    for scene_i, tau_i, lp_eps_i, score_before_i in current_records:
        score_after_i = EVALUATE(phi, lp_eps_i)
        progress_i = SCORE(score_after_i) - SCORE(score_before_i)
        feedback_records.append((scene_i, tau_i, progress_i))

    # Adapt the generator using current scene-level feedback
    theta = FEEDBACK_WEIGHTED_UPDATE(theta, feedback_records)
```