2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

One Prompt, Many Rooms: A Force-Directed Approach to 3D Scene Generation for Robotics Simulation

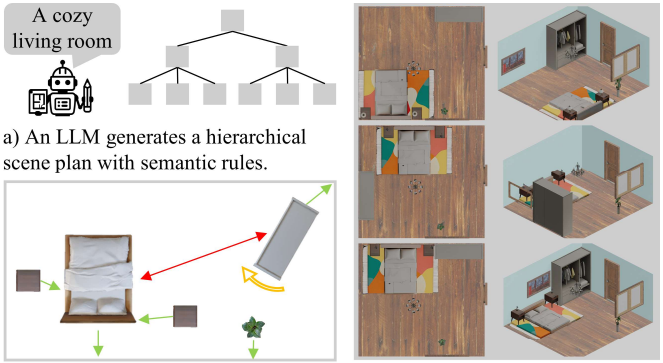
Christopher May[1], Peter Hetzner[1], Matthias Kalenberg[1], Ashwin Sajith Nambiar[1], Jörg Franke[1], and Sebastian Reitelshöfer[1]

***Abstract*— Training generalizable robotic agents requires large datasets of diverse, physically consistent 3D scenes, yet their generation remains a critical bottleneck. Current text-to-scene methods are inefficient for this task; generating diverse layouts requires repeated, expensive sampling that fails to maintain the semantic consistency required for robust policy learning. In this paper, we address this with our "one-plan-to-many-layouts" method. A Large Language Model (LLM) generates a single declarative plan, which a force-directed physics simulation then realizes into multiple layouts that share semantics but differ in geometry. We validate our method by transfer to photorealistic 3D reconstructions of real environments (Replica) within simulation, where a navigation agent trained on our scenes attains a Success Rate of 0.84. These results establish our pipeline as a scalable method for producing the controlled, diverse data required for embodied AI training.**

I. Introduction

Training generalizable robotic agents requires large datasets of 3D environments that are not only visually convincing but also structurally diverse and physically consistent. Unlike graphics or entertainment applications, where visual plausibility alone may suffice, robotic learning depends on accurate geometry, collision-free placement, and consistent semantics to ensure that the policies learned in simulation transfer to the real world. Robust navigation and manipulation policies therefore require exposure to a wide range of spatial layouts, furniture configurations, and constraint interactions, far beyond what can be curated manually [1], [2].

Generative AI promises to ease the data bottleneck in robotics, however, while current text-to-scene systems can produce high-quality individual scenes, their architectures are not suited for the large-scale generation required for training robust agents. These methods typically treat a Large Language Model's (LLM) output as a set of soft constraints to be resolved into a single layout through corrective optimization or require repeated, computationally expensive sampling of a generative model [3], [4], [5]. In both cases, producing N diverse scenes requires running the entire pipeline N times, which becomes prohibitive at the dataset sizes needed for policy learning. Moreover, re-prompting to obtain variation fails to preserve semantic consistency, as object sets, relations, and styles drift unpredictably between runs. This is problematic for robotics because experiments to isolate the effect of spatial structure on performance require controlled variation: the ability to vary geometry while holding semantics fixed.



a) An LLM generates a hierarchical scene plan with semantic rules.

b) A force-directed method arranges the scene in a series of 2D simulations.

c) A single plan yields a distribution of valid 3D scenes.

Figure 1. Our pipeline generates a distribution of 3D scenes from a single prompt by decoupling planning from realization. An LLM first creates one semantic plan (a), which a force-directed simulation then realizes by translating rules into attractive (green) and repulsive (red) forces, along with rotational torques (yellow) (b). The simulation acts as a generative engine; it can converge to multiple stable equilibria, efficiently producing many distinct, valid layouts (c) from the single plan.

We address this limitation by reframing layout generation as a "one-plan-to-many-layouts" sampling task that decouples high-level semantic planning from geometric realization (Fig. 1). Our method uses an LLM only to generate a single declarative scene specification, consisting of a scene graph and abstract placement rules. This plan is then realized through a force-directed simulation that translates these rules into a system of forces and torques. Because this generative, not corrective, simulation can converge to multiple distinct stable equilibria, it efficiently produces a distribution of layouts that are all physically plausible and share the exact same semantic content, differing only in their geometry. This approach amortizes the expensive LLM call and, most importantly, provides the required targeted geometric variation.

Our main contributions are as follows:

- A "one-plan-to-many-layouts" method that generates multiple physically consistent 3D scenes from a single prompt.
- A force-directed layout technique that translates semantic rules into forces and torques within a series of 2D physics simulations.
- A scalable validation loop combining deterministic checks for physical consistency with a Vision-Language Model (VLM)-based plausibility assessment.

[1] The authors are with the Institute for Factory Automation and Production Systems (FAPS) at Friedrich-Alexander-Universität Erlangen-Nürnberg. (corresponding author e-mail: christopher.may@faps.fau.de).

II. Related Work

The automated generation of 3D indoor environments is a critical enabler for scaling up the training and testing of robotic agents. The field has evolved from procedural techniques to learning-based methods, and, most recently, to pipelines driven by Large Language Models [6]. Our work advances this approach by introducing a force-directed placement mechanism that does not seek a single optimal layout, but instead efficiently samples a distribution of physically feasible, simulation-ready scenes from a single plan.

A. From Rigid Procedures to Flexible LLM Planners

The need for large-scale, varied 3D environments was first addressed by procedural generation. ProcTHOR [2] demonstrated that generating thousands of interactive scenes from hand-crafted rules significantly improves performance on embodied AI benchmarks [7], while Infinigen Indoors [8] showed that procedural pipelines can also achieve photorealistic fidelity. Together, these systems establish the value of scalable, simulator-ready data but remain constrained by fixed rule sets that lack the flexibility to interpret novel open-vocabulary text prompts.

Subsequent data-driven methods increased realism by learning layout distributions directly from 3D scene datasets, but still operate under fixed, structured inputs rather than free-form text. For example, House-GAN++ uses a graph-constrained generative adversarial network to refine floor plans conditioned on a bubble diagram [9]. ATISS employs an autoregressive Transformer to generate object sets conditioned on room type and a floor plan image, while DiffuScene introduces denoising diffusion for scene synthesis conditioned on scene graphs. [10], [11]. Despite advances in layout generation, these models lack native open-vocabulary text control.

To overcome this rigidity, the current state-of-the-art has shifted semantic interpretation to LLMs, which excel at generating declarative scene specifications from free-form text. This "LLM-as-planner" pattern is validated by several systems. Open-Universe and LayoutGPT prompt an LLM to generate a program in a domain-specific language (DSL) that defines objects and their qualitative spatial relations [4], [12]. Similarly, AnyHome converts textual narratives into structured graphs, while SceneCraft employs an LLM agent to author Blender-executable Python code [5], [13]. Although these methods effectively leverage LLMs for high-level planning, the plans they produce are purely symbolic, creating the critical challenge of turning the declarative "what" into a physically plausible "where."

B. Layout Realization: Optimization as a Corrective Step

Because LLMs excel at semantic but not spatial reasoning, their generated plans are often geometrically incoherent, containing collisions or illogical placements. The dominant approach to realizing these plans is corrective, treating the LLM's output as a set of soft constraints to be resolved by an optimization process into a single, geometrically consistent solution. This strategy is implemented in various ways. LayoutVLM refines object poses by minimizing a combined objective to enforce semantic alignment and collision avoidance [3]. Open-Universe solves its DSL program as a constraint satisfaction problem with a gradient-based optimizer [4]. Similarly, Scenethesis uses an agentic framework in which an LLM's coarse plan is iteratively refined by vision and optimization modules [14]. While these methods can produce high-quality single scenes, their architectural reliance on finding one energy minimum makes them unsuited for the data generation needs of robotics.

Diffusion-based models such as DiffuScene and PhyScene offer an inherently generative alternative. By learning the underlying data distribution of a training set, they can produce a diverse distribution of outputs by initiating the reverse denoising process from different random noise vectors. However, generating this diversity requires multiple independent and computationally expensive runs [11], [15].

Our work diverges from both corrective optimization and global sampling paradigms. Building on generative approaches to robotic simulation environments [16], we employ a physics simulation as the core generative engine. Our "one-plan-to-many-layouts" architecture translates an LLM's declarative rules into a system of forces, which are resolved by searching for stable equilibria rather than a single global optimum. This approach makes our method fundamentally geared towards diverse data generation required for robotics.

III. Method

Our method is designed to transform a single, open-ended text prompt into a set of physically valid and simulation-ready 3D indoor scenes. The process is structured as a multi-stage workflow that first translates the prompt into a declarative rule set, then uses a force-directed method to generate a valid layout and finally validates the result for quality (Fig. 2).

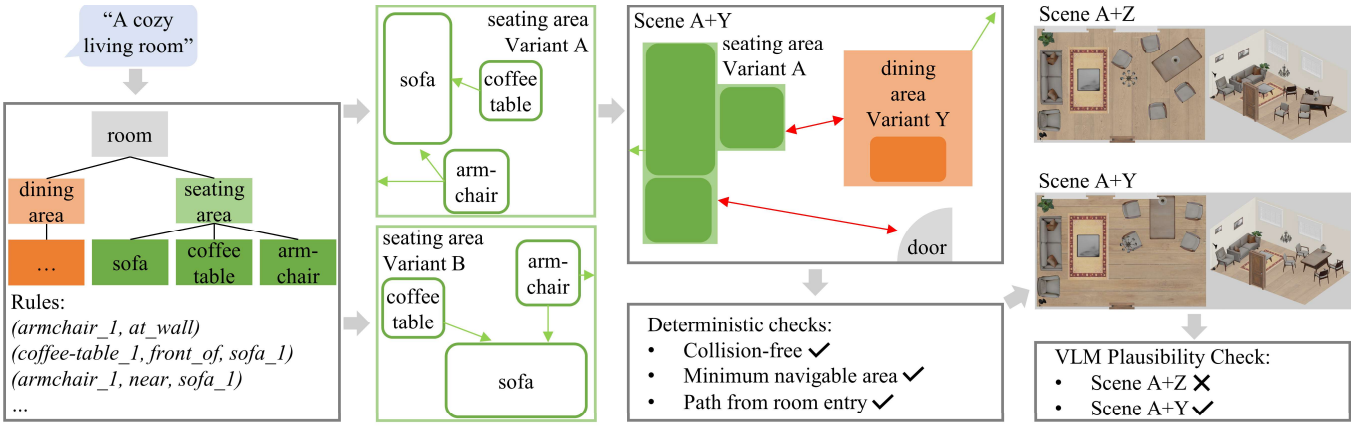
Formally, the Pipeline G can be described as a composition of three functions

$$G = V \circ P \circ L, \tag{1}$$

where L generates the structured scene specification, P performs the force-directed placement, and V handles the final validation and export. Each component is detailed in the following sections, with implementation details at the end of the chapter.

A. LLM-based Generation of Declarative Scene Specifications (L)

The first stage of our pipeline converts a free-form text prompt into a structured, machine-readable specification. We use a Large Language Model (LLM) for this task, leveraging its commonsense reasoning to interpret open-ended language without the manually crafted rules that limit procedural methods. Crucially, the LLM produces a declarative set of constraints, not a single deterministic layout. This decoupling of semantic intent from concrete coordinates enables our downstream force-directed method to generate a diverse range of scenes from a single plan, avoiding the spatial reasoning limitations of LLMs [4], [17], following the declarative, language-to-structure pattern common in recent scene generators [4], [5].



(a) Declarative Scene Specification (b) Force-Directed Placement & Variation (c) 3D Assembly & Validation

Figure 2. Overview of our "one-plan-to-many-layouts" approach. (a) An LLM generates a single declarative plan with a scene graph and rules. (b) A force-directed simulation then creates multiple local layout variants for each functional area and combines them to form different room configurations, which are filtered using deterministic checks for physical validity. (c) Resulting layouts are assembled into 3D scenes and assessed for semantic plausibility by a VLM, where Scene A+Z fails due to implausible chair placement.

To ensure a reliable and structured output, we use a multi-step prompting approach that decomposes the complex specification task into a sequence of managed steps. This process sequentially elicits the room's properties and functional areas; a comprehensive list of objects with their corresponding placement rules (e.g., *at_wall*) and the pairwise spatial relations between them (e.g., *next_to*), selected from a predefined vocabulary; and a hierarchical scene graph. As a form of scene augmentation to better match real-world statistics [18], the LLM is also prompted to include secondary, decorative items. A final step prompts the LLM to validate its own output for logical consistency before passing the completed specification to the placement stage.

B. Force-Directed Placement and Variation (P)

The foundation of our pipeline is a novel force-directed placement method that translates the discrete vocabulary of rules generated by the LLM into a continuous system of forces and torques. The primary challenge is that linguistic rules are often qualitative and can conflict (e.g., positioning an object near a window while also avoiding proximity to a wall). Our approach frames the layout task as an energy minimization problem, a philosophy shared by other contemporary generative systems, although our approach to resolving constraints is distinct [4], [19]. This follows classic potential-field formulations and force-directed relaxation to low-energy states, a technique that is rooted in graph drawing [20], [21]. We solve the problem using a 2D physics simulation that translates abstract rules into forces and torques, which allows for the graceful handling of competing constraints and enables the emergence of multiple, stable configurations.

1) Grounding Scene Specifications with 3D Assets

Before initiating the force-directed placement, the pipeline grounds the abstract object list by selecting a corresponding 3D model for each item from a pre-existing asset database. It searches the database using the natural language description of the desired object and by cosine-similarity weighting of the embedded descriptions.

The dimensions provided by the LLM are not used to create simple primitives, but rather to isotropically scale the chosen 3D model. The 2D bounding box of this scaled model serves as the physical proxy within the simulation, ensuring that the layout directly accounts for the geometry of the final assets.

2) Force-Directed Layout Simulation

Our force-directed method translates the LLM's declarative rules into a 2D multi-body dynamic simulation. In this system, each object is a rigid body, and each rule defines a corresponding force or torque that influences it. This approach allows the system to gracefully resolve competing constraints as it settles into a stable, low-energy state.

The net force $\vec{F}_i$ and torque τ_i acting on each object o_i are the sum of these components:

Rule and Relation Forces ($\vec{F}_r$, $\vec{F}_{rel}$): These forces enact the LLM-generated semantic plan. Unary rules (e.g., *at_wall*) create attractive forces pulling an object toward a geometric feature of the room. Binary relations (e.g., *next_to*) generate forces between pairs of objects to satisfy a spatial relationship.

Physical Constraint Forces ($\vec{F}_{rep}$, $\vec{F}_{bound}$): These forces ensure physical plausibility. A strong, short-range logarithmic repulsive force ($\vec{F}_{rep}$) prevents collisions between all objects. An inverse-square potential from the room boundaries ($\vec{F}_{bound}$) keeps objects within the defined space.

Torques (τ_r, τ_{rel}, τ_{align}): Torques orient objects according to the semantic plan and environmental convention. Rule- and relation-based torques align objects relative to room features or other objects. A general alignment torque τ_{align} promotes settling at cardinal angles (0°, 90°, etc.), reflecting typical arrangements in man-made environments [22].

These components are summed to determine the net force and torque on each object at each timestep as:

$$\vec{F}_i = \sum_{r \in R_i} \vec{F}_r(o_i) + \sum_{(o_j, rel) \in B_i} \vec{F}_{rel}(o_i, o_j) + \sum_{o_j \in O, j \neq i} \vec{F}_{rep}(o_i, o_j) + \vec{F}_{bound}(o_i) \quad (2)$$

$$\tau_i = \sum_{r \in R_i} \tau_r(o_i) + \sum_{(o_j, rel) \in B_i} \tau_{rel}(o_i, o_j) + \tau_{align}(o_i) \quad (3)$$

To ensure robust convergence, we implement a simulated annealing schedule where a global temperature parameter, T, scales all semantic forces and torques [23]. This schedule allows for large-scale adjustments in the initial high-energy state, followed by fine-grained settling as the forces are dampened. The temperature is annealed from an initial value of $T = 1$ toward zero as a function of the simulation step, t, according to

$$T(t) = \frac{1}{1 + k \cdot t}, \quad (4)$$

where k is a cooling-rate constant (empirically set to 0.02). The simulation terminates when the system converges to a stable, low-energy state, indicated by object positions and rotations changing less than a predefined threshold over several timesteps. This force-directed simulation serves as the core engine for our hierarchical placement strategy, which is detailed in the next section.

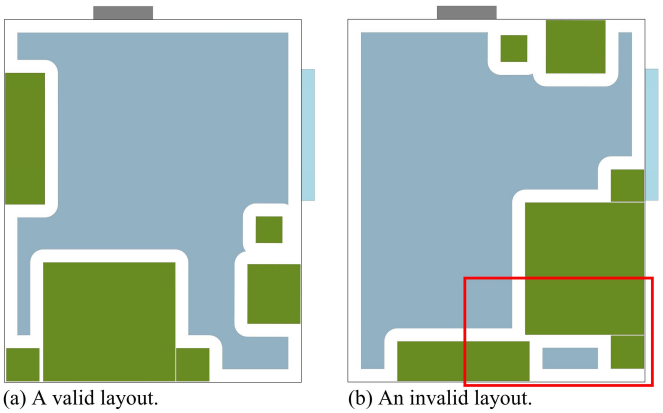
3) Hierarchical 2D Layout Generation

Our method generates a large and diverse set of scenes from a single prompt by decomposing the global layout problem into a sequence of smaller, localized simulations. This hierarchical strategy serves two purposes: it makes the complex placement problem computationally tractable, and more importantly, it creates variation by first generating multiple valid layouts for local functional areas and then combining them into complete room configurations. We adopt this process, a proven approach to scene synthesis [22], [24], in three stages:

1. Local Variant Generation: We generate multiple layouts for each functional area identified by the LLM (e.g., a "Sleeping Area") using a hierarchical simulation. This hierarchy is dictated by the scene graph, where strong dependencies (e.g., *on_top*) create parent-child bonds, while weaker relations (e.g., *facing*) act as constraints between independent peers. The simulation begins with initialization: to ensure diversity, parent objects are placed randomly, while child objects are placed near their parents to accelerate convergence. Finally, parent objects are simulated to a stable state before their children are introduced and settled.

2. Combinatorial Assembly: Next, once a set of valid layout variants for each functional area has been generated, we perform a combinatorial assembly. Each area layout is abstracted as a single bounding polygon with its own placement rules derived from the LLM's description. A final high-level simulation then arranges these area-level bounding boxes within the room.

3. Floor Plan Instantiation and Validation: Finally, we test all valid combinations of the area variants against a set of hard physical and functional constraints. A floor plan is considered valid only if it meets all of the following criteria: (i) all objects are collision-free and fully within the room boundaries; (ii) a minimum percentage of the total floor area



(a) A valid layout. (b) An invalid layout.

Figure 3. An example of our deterministic validation filter, which discards invalid layouts. Both bedroom layouts were generated from the same declarative plan and contain among others a bed with nightstands, and a wardrobe. While layout (a) is valid, layout (b) is discarded because it fails the functional accessibility check: the placement of the wardrobe renders the nightstand inaccessible (red box). This flaw is revealed by the navigable area analysis (blue), which visualizes the floor space available to an agent.

remains navigable for an agent of a given radius; and (iii) a path exists from a room entry point to each primary object, ensuring functional accessibility. For instance, a physically stable layout may still be functionally invalid if it renders a key object inaccessible, a failure case this check is designed to catch (Fig. 3). Configurations that fail any of these checks are discarded. This deterministic filtering stage is critical for efficiency, as it ensures that only physically and functionally sound layouts are passed to the more computationally expensive 3D assembly and VLM validation stages.

4) 3D Scene Assembly and Decoration Placement

The final 3D scene is constructed by assembling validated 2D layouts and then placing all remaining objects on their respective surfaces using the same 2D physics solver. The process begins by instantiating the 3D models for all ground-based objects at the final poses determined by the validated 2D floor plan. With this 3D foundation in place, the scene is completed by placing the remaining decorative, surface-bound, and wall-mounted objects. Rather than using distinct 3D placement rules, we reuse our core 2D force-directed method by projecting 3D surfaces into separate 2D simulation planes.

Surface-Relative Placement: For objects with *on_top* relationships (e.g., a lamp on a nightstand), a secondary 2D physics simulation is performed on the top surface of the parent object. The bounding boxes of other objects already on that surface (or overhanging it) act as constraints, ensuring a valid and plausible placement.

Wall and Ceiling Placement: To place objects such as paintings or ceiling lights, each wall and ceiling are computationally "unfolded" into separate 2D planes. The 2D footprints of any nearby floor objects are projected onto these planes as obstacle zones. A final set of 2D simulations is then run on these surfaces to place the remaining objects according to their rules (e.g., *on_wall_mid*, *above bed_1*).

This consistent use of a 2D physics solver across multiple surfaces – the floor, object tops, walls, and ceiling – is a core component of our method, allowing for a unified and robust approach to generating complete and detailed 3D scenes.

C. Validation and Export (V)

Our final stage is a validation loop that iteratively generates scenes until a target number of high-quality outputs is produced. This process uses a two-stage filter to ensure efficiency. Each candidate 3D scene first undergoes the series of deterministic physical and functional checks described previously. Layouts failing these inexpensive checks are immediately discarded, preventing computational resources from being spent on physically non-viable scenes. Scenes that pass this initial filter are rendered into 2D images and assessed by a VLM for semantic plausibility, including overall functionality, logical consistency, and realism. This use of a multi-modal model to judge qualitative aspects of a scene is related to prior work focused specifically on layout validation [3].

If the VLM accepts the scene, it is added to the final dataset and exported to a standard simulator format. If the VLM rejects it, the scene is discarded, and the pipeline loops to generate a new layout variant from the original semantic plan. This entire process repeats until the desired number of valid scenes has been collected or a predefined maximum number of attempts is reached. This hybrid approach ensures high throughput by reserving the computationally expensive VLM assessment for only the most promising, physically-sound candidates.

D. Implementation Details

The entire pipeline was implemented in Python, building upon multiple established libraries for its core components. All natural language processing, structured data generation, and visual validation tasks were performed using OpenAI's GPT-4o model (gpt-4o-2024-08-06) via its public API. Although prior work has demonstrated the effectiveness of fine-tuning models for specific layout tasks [3], this approach can introduce significant overhead and reduce reproducibility. For this reason, we deliberately used the off-the-shelf pretrained model, relying on structured prompting to ensure consistent and reliable outputs for both the scene specification (L) and validation (V) stages. Code, the full rule-to-force specification, and sample outputs are available on GitHub[1].

The force-directed placement method (P) was built using the Box2D library. The translation of semantic rules into forces and torques was parameterized empirically. We did not tune parameters for each individual rule; instead, our focus was on balancing the relative weights between the different categories of forces. Strong, short-range repulsive potentials were given the highest priority to guarantee collision-free layouts. These were followed by moderate-strength attractive forces for hard placement constraints (e.g., *at_wall*) and weaker forces for soft relational constraints (e.g., *near*). This hierarchical weighting strategy proved to be effective in producing stable and logically coherent scenes. Final scene visualization and rendering were automated using Blender, chosen for its comprehensive Python API, which allowed for seamless integration into our pipeline. Our method is asset-agnostic, focusing entirely on the generation of physically and semantically valid layouts. For the results presented in this paper, furniture models were generated using AI-based tools, while secondary decorative and architectural elements were sourced from public 3D asset repositories. The selection and generation of high-fidelity 3D models were considered outside the scope of this work.

IV. Experiments and Results

To validate our method, we conducted a three-stage evaluation designed to test our core contributions. First, we analyzed the efficacy of the force-directed placement mechanism in generating physically and semantically plausible layouts. Second, we measured how scene richness impacts layout diversity and generation efficiency. Finally, we demonstrated the practical utility of the generated scenes by training a Point-Goal Navigation agent and evaluating its performance in a standard robotics simulator.

A. Experimental Setup

To robustly evaluate our pipeline, we generated a diverse dataset starting from the high-level inputs of "bedroom" and "living room". We first prompted GPT-4o to expand these simple room types into a set of unique, descriptive seeds (e.g., "a modern and cozy living room" or "a student's bedroom"). To account for the variable success rate of these generated seeds, we continued this process until we collected more than 500 physically valid scenes for each of the three different conditions of each room type.

We evaluated the pipeline under three conditions:

Ours (Full): Our complete pipeline with the full set of semantic forces plus placement of secondary objects (for results, see Fig. 4).

–Decor: An ablation identical to Full but excluding the placement of secondary objects, designed to isolate the impact of scene richness.

–Relations: A baseline that disables all object-object relational forces, leaving only unary placement rules (e.g., *at_wall*) and physical repulsion. Functionally related objects (e.g., a bed and nightstands) are still simulated together in local groups, isolating the impact of the explicit forces that define their precise relative arrangement.

To ensure fair comparisons, the same initial scene specification from the LLM was used for all three conditions for a given prompt. The ablations were applied by either removing secondary objects (–Decor) and by zeroing the weights of relational forces (–Relations). This approach yielded a final dataset of 3,680 scenes across the three conditions, sourced from 64 unique descriptive prompts (31 for living room and 33 for bedroom).

B. Metrics

We evaluated our method using metrics for scene diversity, pipeline efficiency, and downstream task performance. We calculated per-room-type metrics with 95% confidence intervals obtained from 10,000 bootstrap resamples, sampling prompts with replacement within each type. We report an overall weighted average to summarize performance across the full dataset. Agreement is quantified using Cohen's κ [25] for two raters and Fleiss' κ [26] for multiple annotators. When we use qualitative labels, we follow Landis & Koch [27];

[1] https://github.com/FAU-FAPS/OnePromptManyRooms

(a) One Plan → Two Layouts

(b) One initial Prompt → Six Rooms

Figure 4. Our pipeline generates diversity from two distinct sources: physical placement and semantic generation. (a) Placement-level diversity: Two distinct layouts generated from a single, detailed semantic plan, demonstrating how the force-directed simulation finds multiple stable equilibria. (b) Semantic-level diversity: Six different living rooms generated from the single high-level prompt "living room", demonstrating the LLM's ability to first create varied semantic plans, with one resulting layout shown for each.

we also report raw agreement.

Scene diversity is assessed from three perspectives. To make our diversity metrics robust against the LLM's linguistic variations (e.g., “sofa” vs. “couch”), we first canonicalized all object labels. We use the all-MiniLM-L6-v2 [28] model to reassign labels to one of 15 canonical classes based on cosine similarity (threshold ≥0.6).

Geometric Dissimilarity measures the physical difference between two scenes. It is calculated as the optimal matching cost between the objects in two scenes, where the cost function considers differences in object position, size, and yaw. We use the Hungarian algorithm for this category-wise assignment and apply a fixed penalty for objects present in one scene but not the other.

Topological Dissimilarity quantifies differences in the navigable free space available to an agent. We compute this as the Jaccard distance between the 2D occupancy masks of two scenes, where obstacles are inflated by the agent's radius (r = 0.2 m) to accurately reflect navigability.

Relational Dissimilarity captures differences in the functional structure of scenes. It is calculated as the Jaccard distance between the sets of relational triples (e.g., *(lamp, on_top, nightstand)*) present in two scene graphs. This metric assesses the abstract semantic graph, allowing us to distinguish changes in functional relationships from changes in the concrete geometric layout.

Pipeline efficiency is measured by **Validity** (the percentage of layouts passing deterministic checks), **Plausibility** (the VLM acceptance rate), and **Throughput** (plausible scenes generated per hour on a regular desktop PC). Finally, for the downstream navigation task, we report the standard metrics of **Success Rate (SR)** and **Success weighted by Path Length (SPL)** [29].

C. Efficacy Analysis of Force-Directed Placement

Relational forces are essential for plausibility. Disabling them reduced the VLM acceptance rate from 87.17% to 78.16% and also lowered physical validity (Table I). These drops indicate that semantic relations act as a structural prior,

TABLE I. Relational forces are critical for semantic plausibility.

Condition	**Validity (%)**	**Plausibility (VLM, %)**	**Plausibility (human, %)**
Ours (Full)	35.93 [31.42, 40.57]	87.17 [81.62, 92.02]	75.33 [70.33, 80.33]
–Relations	33.40 [28.20, 38.86]	78.16 [71.09, 84.61]	55.33 [49.67, 61.00]

steering the simulation toward functionally accessible layouts before the VLM’s final filter.

The VLM showed moderate agreement with the human majority-vote label (Cohen’s κ = 0.49; 80.5% raw agreement) on 200 randomly sampled scenes (100 per condition). Inter-rater reliability among the three annotators was fair (Fleiss’ κ = 0.22). Agreement with the VLM was numerically higher for Full than –Relations (κ = 0.58 vs. 0.41). Together with the overall agreement, this supports using the VLM as a scalable proxy.

D. Impact of Scene Richness and Diversity

We compare the Ours (Full) condition to a –Decor ablation that removes secondary objects, and a Random baseline that places objects without semantic guidance but subject to the same physical validity checks.

The –Decor ablation reveals a trade-off between scene complexity and throughput. Removing secondary objects increased the number of plausible scenes generated per hour from 160.79 to 216.40. This change also slightly increased geometric and topological dissimilarity (Table II), suggesting the decorative elements act as spatial constraints that guide the simulation toward more consistent layouts. A Random baseline shows our force-directed method's essential role, as it ensured plausible layouts with an 87.17% VLM acceptance rate compared to 15.86% for Random. This low yield proves the inefficiency of naive generation, confirming our force-directed simulation is a crucial filter for producing high-quality, scalable data.

To quantify variation across prompts in the Full condition, we compared the first accepted scene from each of our 64 unique descriptions. The mean geometric dissimilarity was

TABLE II. Secondary objects reduce generation throughput and slightly decrease layout diversity. Both configurations produce significant intra-prompt variation, substantially more structured than a Random baseline.

Condition	Validity (%)	Plausibility (%)	Throughput (scenes / h)	Geometric Dissimilarity (Intra)	Topological Dissimilarity (Intra)
Ours (Full)	35.93 [31.42, 40.57]	87.17 [81.62, 92.02]	160.79 [148.52, 172.14]	0.14 [0.13, 0.14]	0.55 [0.54, 0.55]
–Decor	36.09 [31.57, 40.83]	81.59 [75.28, 87.41]	216.40 [198.01, 234.35]	0.15 [0.14, 0.16]	0.57 [0.57, 0.58]
Random	N/A	15.86 [10.96, 19.76]	N/A	0.16 [0.16, 0.17]	0.59 [0.58, 0.59]

0.99 [0.97, 1.00], far exceeding the 0.14 intra-prompt variation (Table II) and confirming that different prompts yield structurally distinct scenes. Relational dissimilarity was also high (0.81 [0.80, 0.82]), while topological dissimilarity remained comparable (0.89 [0.89, 0.90] vs. 0.55 [0.54, 0.55]). This suggests that while prompts alter the scene's functional structure, the navigable complexity remains consistent.

E. Downstream Evaluation: Point-Goal Navigation in Habitat

An agent trained on our generated scenes for a Point-Goal Navigation (PointNav) task successfully transfers to scanned real-world environments within simulation, validating our pipeline's utility. Using a standard PPO agent [30] with RGB-D input in Habitat [1] without further tuning, the policy achieved an SR of 0.84 and an SPL of 0.62 on nine unseen rooms from the Replica dataset [31] (Table III). It also generalized well to 465 held-out scenes generated from 13 unseen prompts, achieving an SR of 0.89. An agent trained on sparser scenes without decorative objects (–Decor) exhibited slightly improved transfer performance (SR 0.85, SPL 0.70), suggesting decorative density is a meaningful factor when transferring policies to photorealistic scanned environments. While a baseline trained on the large-scale HM3D dataset [32] performs best (SR 0.93), our results establish our generative pipeline as a viable alternative for producing effective training data without requiring real-world scans.

V. Discussion

We presented a “one-plan-to-many-layouts” method that decouples semantic planning from geometric realization to efficiently generate a distribution of physically consistent 3D scenes. Our core contribution is using a force-directed simulation as a generative engine that converges to multiple distinct, stable equilibria from a single declarative plan. We validate this approach for robotics by showing that a navigation agent trained exclusively on our generated scenes achieves competitive performance on photorealistic 3D reconstructions of real environments (Replica [31]) within simulation (Table III). This establishes that our pipeline produces data with the controlled diversity and physical plausibility required for robust policy learning.

A key finding is that the agent trained on sparser scenes (–Decor) showed slightly better transfer performance,

TABLE III. A navigation agent trained exclusively on our generated scenes achieves a Success Rate (SR) and Success Weighted by Path Length (SPL) on real-world scans (Replica) in simulation that is competitive with an agent trained on the large-scale HM3D dataset.

Training Dataset	Test Dataset	SR	SPL
Ours (Full)	Generated (Held-out)	0.89	0.72
	Replica	0.84	0.62
–Decor	Generated (Held-out)	0.87	0.75
	Replica	0.85	0.70
HM3D	Generated (Held-out)	0.89	0.70
	Replica	0.93	0.78

revealing a task-dependent trade-off likely driven by appearance and topology shifts between our assets and Replica. For geometry-dominated tasks like PointNav, prioritizing layout diversity over object density improves transfer [33], while semantics-dependent tasks would likely benefit from richer scenes [34], [35].

Our key architectural distinction is framing layout generation as generative sampling rather than corrective optimization. Corrective methods aim to find a single, high-quality layout from an LLM's potentially flawed plan [3]. In contrast, our force-directed simulation is designed to settle into multiple distinct, valid low-energy states from a single set of rules. This focus on distribution coverage directly addresses the need in robotics for large, varied datasets that enable agents to generalize beyond their training environments.

The pipeline's scalability stems from coupling a physics-based prior with learned validation. Relying solely on a VLM to filter randomly generated layouts is computationally infeasible, as confirmed by our low baseline acceptance rate (Table II). Instead, our force-directed method guides the generation process toward a region of plausible layouts. Our ablation study supports this; enabling semantic relational forces significantly improves both the final VLM plausibility and the initial rate of physical validity (Table I). The physics-based prior thus efficiently generates sound candidates, reserving the VLM for a final semantic check.

Although our approach is effective, its implementation highlights clear avenues for future work. The visual quality of our scenes is tied to our current asset pipeline, which was not the focus of this study. Integrating on-demand, generative 3D asset models is a natural next step. More critically, while our method handles wall and surface placement via 2D projections, it cannot resolve complex 3D interactions such as overhangs or placement on irregular surfaces. Extending to multi-room floorplans with articulated objects is equally essential. Additionally, cross-family VLM validation would reduce the potential self-referential bias of using GPT-4o for both planning and assessment.

VI. Conclusion

We address the data generation bottleneck for robotic learning by introducing a "one-plan-to-many-layouts" method that uses a force-directed physics simulation as a generative engine to efficiently decouple semantic planning from geometric realization. A navigation agent trained exclusively

on our generated data generalizes effectively to photorealistic 3D reconstructions of real environments, validating that our approach produces the controlled diversity required for robust policy learning. By demonstrating physics-based simulation as a generative engine rather than a corrective tool, this work establishes a scalable method for producing the diverse environments essential for advancing embodied AI.

ACKNOWLEDGMENT

The authors used GitHub Copilot with Anthropic Claude models to assist in writing code for this publication.

REFERENCES

[1] M. Savva *et al.*, "Habitat: A Platform for Embodied AI Research," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 9338–9346. doi: 10.1109/ICCV.2019.00943.

[2] M. Deitke *et al.*, "ProcTHOR: large-scale embodied AI using procedural generation," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, in NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.

[3] F.-Y. Sun *et al.*, "LayoutVLM: Differentiable Optimization of 3D Layout via Vision-Language Models," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2025, pp. 29469–29478. doi: 10.1109/CVPR52734.2025.02744.

[4] R. Aguina-Kang *et al.*, "Open-Universe Indoor Scene Generation using LLM Program Synthesis and Uncurated Object Databases," 2024, arXiv. doi: 10.48550/arXiv.2403.09675.

[5] R. Fu, Z. Wen, Z. Liu, and S. Sridhar, "AnyHome: Open-Vocabulary Generation of Structured and Textured 3D Homes," in *Computer Vision – ECCV 2024*, Cham: Springer Nature Switzerland, 2025, pp. 52–70. doi: 10.1007/978-3-031-72933-1_4.

[6] B. Wen, H. Xie, Z. Chen, F. Hong, and Z. Liu, "3D Scene Generation: A Survey," 2025, *arXiv*. doi: 10.48550/arxiv.2505.05474.

[7] E. Kolve *et al.*, "AI2-THOR: An Interactive 3D Environment for Visual AI," 2017, *arXiv*. doi: 10.48550/arXiv.1712.05474.

[8] A. Raistrick *et al.*, "Infinigen Indoors: Photorealistic Indoor Scenes using Procedural Generation," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 21783–21794. doi: 10.1109/CVPR52733.2024.02058.

[9] N. Nauata, S. Hosseini, K.-H. Chang, H. Chu, C.-Y. Cheng, and Y. Furukawa, "House-GAN++: Generative Adversarial Layout Refinement Network towards Intelligent Computational Agent for Professional Architects," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13627–13636. doi: 10.1109/CVPR46437.2021.01342.

[10] D. Paschalidou, A. Kar, M. Shugrina, K. Kreis, A. Geiger, and S. Fidler, "ATISS: Autoregressive Transformers for Indoor Scene Synthesis," in *Proc. NeurIPS*, Red Hook, NY: Curran Associates, Inc., 2021, pp. 12013–12026.

[11] J. Tang, Y. Nie, L. Markhasin, A. Dai, J. Thies, and M. Nießner, "DiffuScene: Denoising Diffusion Models for Generative Indoor Scene Synthesis," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 20507–20518. doi: 10.1109/CVPR52733.2024.01938.

[12] W. Feng *et al.*, "Layoutgpt: Compositional visual planning and generation with large language models," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2023.

[13] Z. Hu *et al.*, "SceneCraft: an LLM agent for synthesizing 3D scenes as blender code," in *Proc. ICML*, in ICML'24, vol. 235. Vienna, Austria: JMLR.org, 2024, pp. 19252–19282.

[14] L. Ling *et al.*, "Scenethesis: A Language and Vision Agentic Framework for 3D Scene Generation," 2025, *arXiv*. doi: 10.48550/arXiv.2505.02836.

[15] Y. Yang, B. Jia, P. Zhi, and S. Huang, "PhyScene: Physically Interactable 3D Scene Synthesis for Embodied AI," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 16262–16272. doi: 10.1109/CVPR52733.2024.01539.

[16] C. May, L. Suchy, J. Franke, and S. Reitelshöfer, "Towards Imaginative Robots: A Generative Pipeline for Simulated Environments," *SNE Simul. Notes Eur.*, vol. 35, no. 1, pp. 61–70, 2025, doi: 10.11128/sne.35.tn.10720.

[17] Y. Yamada, Y. Bao, A. K. Lampinen, J. Kasai, and I. Yildirim, "Evaluating Spatial Understanding of Large Language Models," *Trans. Mach. Learn. Res.*, 2024.

[18] M. Khanna *et al.*, "Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 16384–16393. doi: 10.1109/CVPR52733.2024.01550.

[19] N. Gkanatsios, A. Jain, Z. Xian, Y. Zhang, C. Atkeson, and K. Fragkiadaki, "Energy-based Models are Zero-Shot Planners for Compositional Scene Rearrangement," in *Robotics: Science and Systems XIX*, Robotics: Science and Systems Foundation, Jul. 2023. doi: 10.15607/RSS.2023.XIX.030.

[20] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation Proceedings*, Mar. 1985, pp. 500–505. doi: 10.1109/ROBOT.1985.1087247.

[21] P. Eades, "A heuristic for graph drawing," *Congr. Numerantium*, vol. 42, no. 11, pp. 149–160, 1984.

[22] L.-F. Yu, S.-K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. J. Osher, "Make it home: automatic optimization of furniture arrangement," in *ACM SIGGRAPH 2011 papers*, Vancouver British Columbia Canada: ACM, Jul. 2011, pp. 1–12. doi: 10.1145/1964921.1964981.

[23] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proc. IEEE*, vol. 86, no. 11, pp. 2210–2239, Nov. 1998, doi: 10.1109/5.726788.

[24] L. Gao, J.-M. Sun, K. Mo, Y.-K. Lai, L. J. Guibas, and J. Yang, "SceneHGN: Hierarchical Graph Networks for 3D Indoor Scene Generation With Fine-Grained Geometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 7, pp. 8902–8919, Jul. 2023, doi: 10.1109/TPAMI.2023.3237577.

[25] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, Apr. 1960, doi: 10.1177/001316446002000104.

[26] J. L. Fleiss, "Measuring nominal scale agreement among many raters.," *Psychol. Bull.*, vol. 76, no. 5, pp. 378–382, Nov. 1971, doi: 10.1037/h0031619.

[27] J. R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, vol. 33, no. 1, p. 159, Mar. 1977, doi: 10.2307/2529310.

[28] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proc. EMNLP-IJCNLP*, Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3980–3990. doi: 10.18653/v1/D19-1410.

[29] P. Anderson *et al.*, "On Evaluation of Embodied Navigation Agents," 2018, *arXiv*. doi: 10.48550/arXiv.1807.06757.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017, *arXiv*. doi: 10.48550/arxiv.1707.06347.

[31] J. Straub *et al.*, "The Replica Dataset: A Digital Replica of Indoor Spaces," 2019, *arXiv*. doi: 10.48550/arxiv.1906.05797.

[32] S. K. Ramakrishnan *et al.*, "Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[33] E. Wijmans *et al.*, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," in *International Conference on Learning Representations*, 2020.

[34] A. Szot *et al.*, "Habitat 2.0: training home assistants to rearrange their habitat," in *Proc. NeurIPS*, Red Hook, NY, USA: Curran Associates Inc., 2021, pp. 251–266.

[35] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *Proc. NeurIPS*, in NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020, pp. 4247–4258.