EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

Text-Guided Interactive Scene Synthesis with Scene Prior Guidance

Shaoheng Fang[†1] and Haitao Yang[1] and Raymond Mooney[1] and Qixing Huang[‡1]

[1]University of Texas at Austin

Abstract

3D scene synthesis using natural language instructions has become a popular direction in computer graphics, with significant progress made by data-driven generative models recently. However, previous methods have mainly focused on one-time scene generation, lacking the interactive capability to generate, update, or correct scenes according to user instructions. To overcome this limitation, this paper focuses on text-guided interactive scene synthesis. First, we introduce the SceneMod dataset, which comprises 168k paired scenes with textual descriptions of the modifications. To support the interactive scene synthesis task, we propose a two-stage diffusion generative model that integrates scene-prior guidance into the denoising process to explicitly enforce physical constraints and foster more realistic scenes. Experimental results demonstrate that our approach outperforms baseline methods in text-guided scene synthesis tasks. Our system expands the scope of data-driven scene synthesis tasks and provides a novel, more flexible tool for users and designers in 3D scene generation. Code and dataset are available at `https://github.com/bshfang/SceneMod`.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Computer graphics;*** *Natural language processing;* • ***Computer systems organization*** $\rightarrow$ ***Neural networks;***

1. Introduction

3D scene synthesis based on deep learning is an important task with numerous applications for 3D content generation, from interior design to video gaming and virtual reality. It demands the generated scenes to be realistic, high-fidelity, and functionally accurate [PPL*24]. Integrating instruction to generate scenes that conform to users' instructions and envisioned concepts is also vital. For example, in a generation system, a user might want to either increase or decrease the number of table lamps in the bedroom or wish to reconfigure the overall furniture layout and direction. Also, users may prefer a sequential, step-by-step object generation process, initiating with the generation of primary items such as a double bed for the bedroom or a dining table and chairs for the dining room, subsequently expanding towards a fully realized scene. Therefore, an ideal scene synthesis system should possess the capability to interactively update its generated results in alignment with user commands rather than merely producing a complete scene in a single effort.

This paper focuses on the interactive, text-guided scene synthesis task. From the user interaction perspective, guidance via free-form language emerges as the most flexible approach, being highly semantic, unconstrained, and a very intuitive means of conveying ideas. In the domain of 3D scene generation, textual description guidance has also become increasingly popular, offering a user-friendly and flexible interface for creation. Many data-driven generative methods have explored text-guided generation [WYN21, PKS*21, TNM*23, LXJ*23, LM24]. However, all these methodologies focus on one-off scene generation, lacking the capability for interactive updates and modifications to scenes.

To obtain interactive controllability in a generative system, the first challenge is to link language with possible modifications to 3D scenes. While numerous works have studied text-guided image editing [MMM*24], descriptions of possible modifications in 3D scenes encompass more subtle spatial relationships and patterns. The textual instruction can not only delineate the position change of objects within the scene but also contain the relative positioning and geometrical relationships among elements in the scene. Moreover, no datasets exist that associate pairs of 3D scenes with textual instructions to update from the source scene to the target scene. Current 3D indoor scene datasets merely provide independent indoor scenes without paired scenes that could facilitate the learning of scene modifications. Consequently, we introduce the SceneMod dataset, which comprises 168k pairs of scenes with textual modifiers, to support training an interactive scene synthesis system.

Based on the SceneMod dataset, we developed a unified two-stage graph diffusion model for various interactive scene synthesis tasks. On the one hand, the diffusion model has demonstrated robust capabilities across various generative tasks [YZS*23] and

[†] e-mail: bshfang@utexas.edu
[‡] e-mail: huangqx@cs.utexas.edu

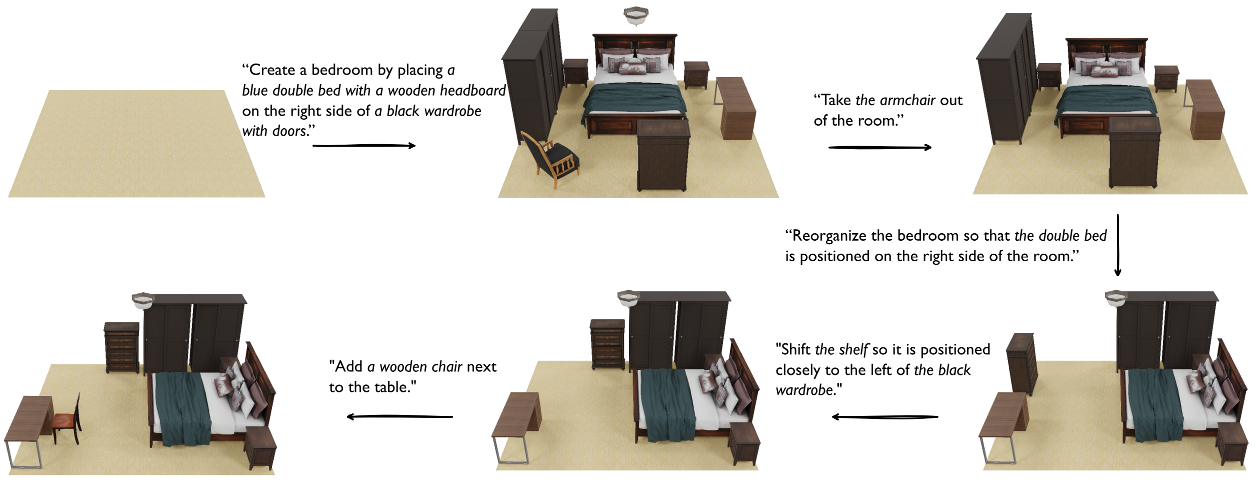

"Create a bedroom by placing *a blue double bed with a wooden headboard* on the right side of *a black wardrobe with doors*."
"Take *the armchair* out of the room."
"Reorganize the bedroom so that *the double bed* is positioned on the right side of the room."
"Shift *the shelf* so it is positioned closely to the left of *the black wardrobe*."
"Add *a wooden chair* next to the table."

Figure 1: We introduce a text-guided interactive scene synthesis system that supports progressive generation and modification during scene synthesis. The examples in the figures are samples from our proposed dataset.

also excels in the domain of scene synthesis [TNM*23, LM24]. On the other hand, the scene graph data structure serves as an effective and informative data representation, explicitly encapsulating the structural and geometrical relationships between objects. Especially for text-guided scene synthesis tasks, scene graphs provide a clear grounding of objects and actions described in the text, thereby enhancing the model's ability to link textual guidance with scene modifications.

There are still numerous challenges associated with the data-driven learning of scene synthesis. First, in terms of data, obtaining reliable training data is a significant hurdle. The data must be manually crafted by professionals, which is both time-consuming and costly, constraining the scale of the dataset. Some incorrect data with incorrect penetration or unrealistic placements in the current training dataset is another issue [LGWM22], leading to the same errors reproduced in the generative models' outputs. Second, regarding model learning, scene design involves a variety of generic geometric patterns and constraints, such as ensuring that generated objects do not overlap or penetrate each other. Neural network models are not easily trained to learn these patterns and constraints explicitly. To address these issues, we define a scene-prior loss based on geometric constraints or those learned from training data and combine them to construct more reliable training datasets and generate more feasible results. We propose a scene-prior guidance diffusion model, which guides the random denoising process of graph diffusion to a more realistic and correct generation.

We evaluate our system on the 3D-FRONT [FCG*21] dataset and our proposed SceneMod dataset. Experimental results show that our approach can generate realistic, high-fidelity indoor scenes interactively according to user guidance. Our method outperforms baseline approaches both qualitatively and quantitatively and surpasses baselines in a perceptual study. The design of the two-stage graph diffusion model and scene-prior guidance are validated through the ablation studies.

In summary, our contributions are:

- We introduce the text-guided interactive scene synthesis system, enabling progressive generation and update of indoor 3D scenes.
- We introduce the SceneMod dataset that associates textual description with modification on 3D scenes that supports the training of the text-guided interactive scene synthesis system.
- We propose a two-stage graph diffusion model with scene-prior guidance that supports various scene synthesis tasks.

2. Related Work

2.1. Indoor 3D Scene Synthesis

Early works in scene synthesis primarily addressed the optimization problems based on prior knowledge, either from hand-crafted rules and constraints [MSL*11, YYW*12] or from hand-craft modeling of structural patterns in indoor scenes [YYT*11, FRS*12, KCKK12, LCK*14, CLW*14]. These methods are good at ensuring constraints and capturing low-order patterns but cannot generate novel and diverse results. More recent data-driven approaches directly learn indoor scene arrangements from large-scale datasets, utilizing various machine learning techniques, including convolutional networks [RWL19, WSCR18], Variational Auto-Encoders (VAEs) [PZR20, YZY*21], Generative Adversarial Networks (GANs) [ZYM*20, YGZT21], transformers [WYN21, PKS*21, LGWM22], and diffusion methods [TNM*23, WDP*23, LM24]. Additionally, scene graphs are commonly used to represent indoor scenes in scene synthesis due to their ability to provide rich structural and geometrical information [LPX*19, WLW*19, ZWK19, HHT*20, WDNT20, DMNT21, ZÖW*24, LM24].

Given that indoor scene design often conforms to specific constraints or priors, some data-driven methods have been developed to integrate these scene priors into the model's training or inference. [YZY*21] employs an optimization stage to refine object arrangements generated by a variation auto-encoder model based on learned relative attribute priors. [LGWM22] incorporates an ergonomic loss, crafted by experts, into the training of a transformer-based model. Different from those methods, we use a diffusion model to generate the scene graph of indoor scenes and guide the sampling process with scene priors to satisfy various priors inherent in indoor scene design explicitly.

2.2. Interactive 3D Scene Synthesis

In 3D indoor scene synthesis, allowing users to interactively participate in the generation process enhances the flexibility and control of the output, thereby aligning more closely with user expectations. There are various kinds of methods for integrating user feedback into the synthesis process. [MSL*11] generates a series of scenes based on interior design guidelines and lets users choose from them. Tools like SceneSuggest [SCA17] and SceneDirector [ZTL*23] provide interactive interfaces where users can select objects to generate or modify through mouse clicks. Based on user-selected query point location, [YYT15, ZWK19] suggest appropriate object categories and models to detail the scene. Meanwhile, [LZWT20, DMNT21, ZÖW*24] employ a scene graph as the scene representation, enabling users to modify the scene through explicit manipulation of the scene graph. [CESM17] and [MPF*18] utilize natural language descriptions to guide scene generation and editing, offering a more flexible and user-friendly guidance method. In this paper, we aim to introduce a general text-guided interactive system capable of generating and modifying 3D scenes based on free-form natural language instructions in an end-to-end manner.

2.3. Language-driven 3D Scene Synthesis

Traditional approaches [CSM14, CMS*15, CESM17, MPF*18] extract information about object categories and spatial relationships from language descriptions through sentence parsing and then apply optimization techniques to arrange objects within a scene. More recently, methods use cross-attention mechanisms [VSP*17] to integrate text descriptions as conditions for models, enhancing the proficiency of text-guided generation through data-driven training. To obtain training data with text descriptions, [WYN21, PKS*21, TNM*23, LM24] automatically generates textual descriptions based on the object categories and spatial relationships present in scenes. Additionally, [LXJ*23, LM24] extract visual features of 3D objects and incorporate visual information into descriptions, enabling text to control scene style and object appearance. With the advent of large language models (LLMs), some LLM-based methods have been designed for text-guided scene synthesis [FZF*24, WLSF23, YSW*24, AKGH*24]. These methods exploit the textual generative capabilities of LLMs to produce scene layouts in CSS-like formats [FZF*24] or domain-specific languages [YSW*24, AKGH*24]. Their advantage is the ability to generate open-universe scenes without the need for additional training. However, they are constrained by LLMs' limited understanding of high-order semantics [YBL*23, CXK*24], such as spatial relationships in 3D scenes, which hampers their potential to execute subtle instructions. Despite recent advances, text-guided scene synthesis has mainly focused on one-time scene generation, lacking interactive editing and progressive generation capabilities.

3. Text-guided Scene Modification Dataset

To facilitate the training of our interactive scene modification model, we introduce the new SceneMod dataset, which comprises 168k samples in the form of (scene A, scene B, text description) triplet. Scene B is derived from scene A through specific manipulations, with the text description providing natural language guidance on the modifications.

We develop an automated sample generation pipeline to produce feasible pairs of scene modifications and human-like natural language descriptions. The pipeline overview is illustrated in Figure 2.

3.1. Scene Pairs Generation and Selection

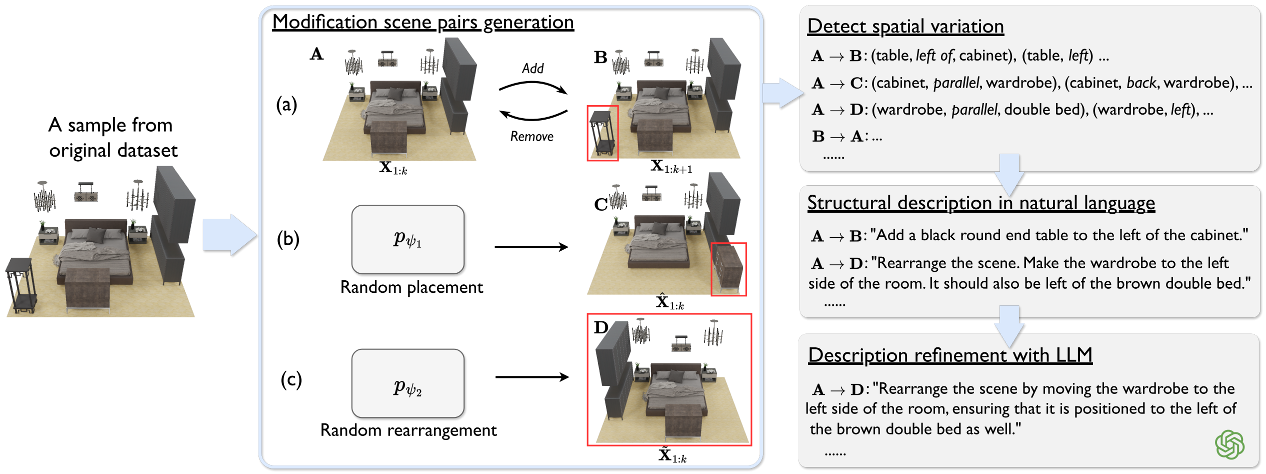
The SceneMod dataset encompasses two types of modification patterns: i) Object-level modification, which pertains to operations on individual objects. This includes placing an object in a scene, removing an object, or moving an object. ii) Scene-level modification, which involves alterations to the entire scene. It requires a rearrangement of the positions of objects within the scene. By integrating these two levels of modifications, users can easily refine or adjust the generated scenes in the interactive scene synthesis system.

In this section, we denote the 3D-front dataset as $\{\mathbf{X}^j_{1:n_j}\}_{j=1}^{N_d}$, where N_d is the sample number of 3D-front dataset and n_j is the object number of scene $\mathbf{X}^j$. Note that $\mathbf{X}^j$ can be viewed as the set of nodes of the scene graph in Section 4.1, where a node representation is $x^j_i = [\mathbf{t}^j_i, \mathbf{s}^j_i, \theta^j_i, c^j_i, \mathbf{f}^j_i], i = 1, \cdots, n_j$. For brevity, we will omit the superscript j.

Object-level scene modification pairs Pairs related to remove and add operations can be directly sourced from the original scenes in the 3D-front dataset. From a well-annotated scene $\mathbf{X}$, we randomly subsample $k+1$ objects to form the pairs ($\mathbf{X}_{1:k}$, $\mathbf{X}_{1:k+1}$) or ($\mathbf{X}_{1:k+1}$, $\mathbf{X}_{1:k}$), where the only change between the two scenes is the addition or removal of an object. To create scene pairs demonstrating object movement, we train a generative model $p_{\psi_1}(\hat{x}_k|\mathbf{X_{1:k-1}}, c_k, \mathbf{f}_k)$, which is capable of suggesting possible locations for a given object model. The scene with a new object placement can be formatted as $\hat{\mathbf{X}}_{1:k} = [\mathbf{X}_{1:k-1}, \hat{x}_k]$. Then, we can form a pair ($\mathbf{X}_{1:k}$, $\hat{\mathbf{X}}_{1:k}$) or ($\hat{\mathbf{X}}_{1:k}$, $\mathbf{X}_{1:k}$) along with the original scene, wherein the position of an object changes.

Scene-level scene modification pairs While interactive operations on individual objects are somewhat limited, we also want to support abstract and high-level commands for whole-scene modification. To acquire such data pairs, we train a scene rearrangement generative model $p_{\psi_2}(\tilde{\mathbf{X}}_{1:k}|\mathbf{X}_{1:k}) = p_{\psi_2}(\tilde{\mathbf{X}}_{1:k}|\{c_i, \mathbf{f}_i\}_{1:k})$, which can provide complete scene arrangement suggestions with the condition of an object set $\{c_i, \mathbf{f}_i\}_{1:k}$ from the origin scene. The newly generated layout and the original scene together constitute a scene modification pair ($\mathbf{X}_{1:k}$, $\tilde{\mathbf{X}}_{1:k}$).

Splits We employ the same splits for the 3D-front dataset as used in [PKS*21], and perform the scene pairs generation and selection in each split. Additionally, we use half of the data from each split subset to train the two generative models p_{ψ_1} and p_{ψ_2}, and the remaining half to generate results for constructing scene modification pairs. More details about the training of p_{ψ_1} and p_{ψ_2} and the scene pairs selection process are provided in the supplementary material.

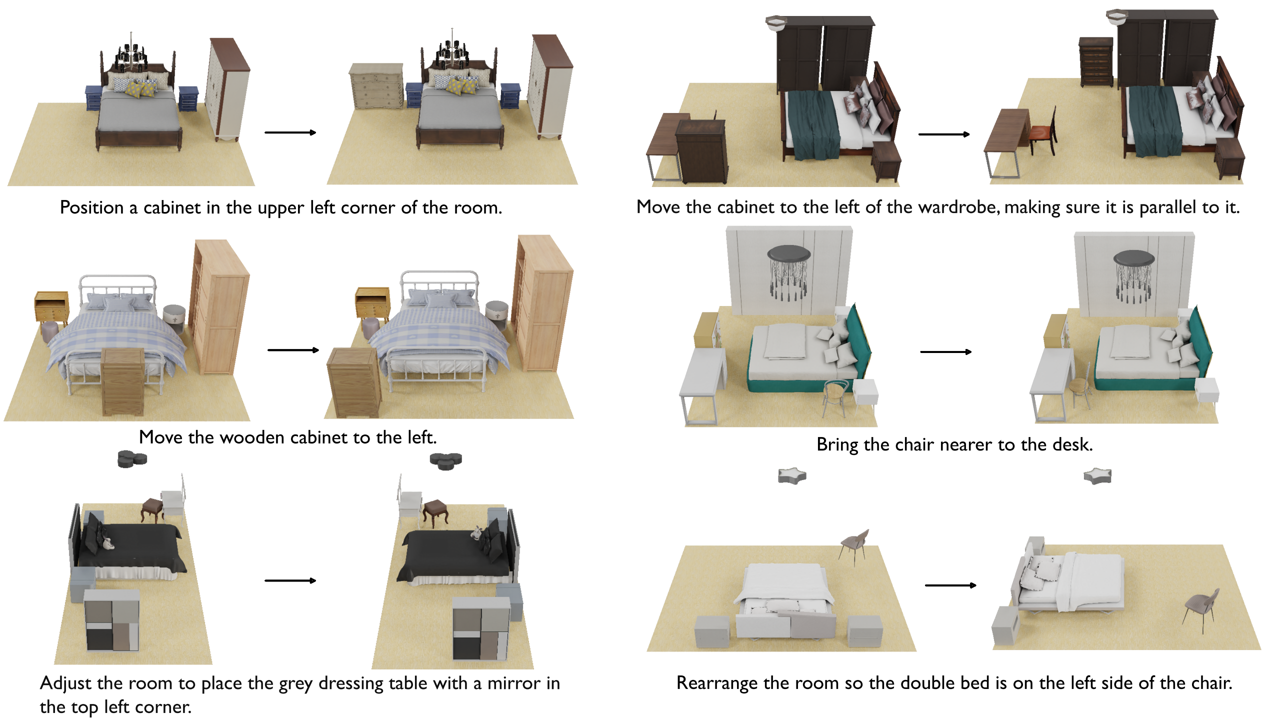


$\mathbf{A} \to \mathbf{B}$: (table, *left of*, cabinet), (table, *left*) ...
$\mathbf{A} \to \mathbf{C}$: (cabinet, *parallel*, wardrobe), (cabinet, *back*, wardrobe), ...
$\mathbf{A} \to \mathbf{D}$: (wardrobe, *parallel*, double bed), (wardrobe, *left*), ...
$\mathbf{B} \to \mathbf{A}$: ...
......

$\mathbf{A} \to \mathbf{B}$: "Add a black round end table to the left of the cabinet."
$\mathbf{A} \to \mathbf{D}$: "Rearrange the scene. Make the wardrobe to the left side of the room. It should also be left of the brown double bed."
......

$\mathbf{A} \to \mathbf{D}$: "Rearrange the scene by moving the wardrobe to the left side of the room, ensuring that it is positioned to the left of the brown double bed as well."
......

Figure 2: The automatic pipeline for scene modification data generation.



Position a cabinet in the upper left corner of the room.

Move the cabinet to the left of the wardrobe, making sure it is parallel to it.

Move the wooden cabinet to the left.

Bring the chair nearer to the desk.

Adjust the room to place the grey dressing table with a mirror in the top left corner.

Rearrange the room so the double bed is on the left side of the chair.

Figure 3: Examples in the SceneMod dataset.

3.2. Automatic Description Generation

Spatial variation detection The generation of textual instructions begins by detecting the low-level spatial variations between pairs of scenes. These spatial variants describe changes of objects within the scene. The variation pattern for an object primarily includes two aspects: 1) changes in its relative relationships with other objects, which can be considered as changes in the edge attributes in the corresponding scene graph representations; 2) alterations in its own position, such as location and orientation. For instance, in Figure 2, a table is added to scene A to generate scene B. The variation pattern would include changes in relative relationships, such as *(table, left of, cabinet)*, indicating that the 'table is left of the cabinet,' or patterns of its absolute position alterations, such as *(table, left)*, where the newly added table is positioned on the left side of the room and near the left wall. Please refer to the supplementary for more details.

Description generation and refinement After detecting all spatial variation patterns, we randomly select 0-2 patterns for each pair

	Object-level			Scene-level	Overall
	Add	Remove	Move	Rearrange	
Bedroom	24,147	2,462	41,094	16,022	83,725
Living/Dining	26,055	3,149	40,814	14,183	84,201

Table 1: Data statistics for the SceneMod dataset. It shows the number of samples for each modification type and room type.

of scene modifications and generate a structural description in natural language according to the type of modification. Taking Figure 2(a) as an example, the modification in pair (A, B) involves placing a table in the scene. Suppose we sample *(table, left of, cabinet)* as the variation pattern to describe. A rule-based method can generate the corresponding description: *'Add a black round end table to the left of the cabinet.'* Note that we utilize object captions annotated by [LM24], which include descriptions of visual information about the objects beyond just their category. This allows our dataset to support interaction instructions incorporating descriptions of an object's visual information. Given that instructions generated by rule-based methods can be rigid, lack diversity, and may not align with natural human instructions, we use a large language model [AAA*23] to paraphrase the generated description, making it more human-like.

3.3. Dataset Statistics

Following previous data-driven scene synthesis works [PKS*21, TNM*23, LM24], we focus on three types of indoor rooms from the 3D-front dataset to construct the SceneMod dataset, including bedrooms, dining rooms, and living rooms. Due to the overlap in this partition, where approximately 60 percent of the samples in the dining room and living room types are identical, we have merged these two room types, considering 'bedroom' and 'living room & dining room' as the two types in our dataset. In total, the SceneMod dataset comprises about 168k samples, each including a pair of scenes with a textual description of the modifications between them. The split for training, validation, and testing is approximately 8:1:1. Detailed data statistics can be found in Table 1.

4. Method

We propose a two-stage scene-prior-guided graph diffusion model for interactive scene generation. In this section, we introduce the scene graph representation and task formulation (Section 4.1), two-stage graph diffusion model (Section 4.2), scene-prior definition (Section 4.3) and the scene-prior guidance denoising process (Section 4.4). An overview of our method is shown in Figure 4.

4.1. Problem Setup

Scene graph representation A 3D scene graph, represented as $\mathbf{G} = (\mathbf{X}, \mathbf{E})$, is a structural representation where $\mathbf{X} = \{\mathbf{x}_i | i \in \{1, \cdots, n\}\}$ denotes the set of object nodes, $\mathbf{E} = \{\mathbf{e}_{ij} | i \in \{1, \cdots, n\}, j \in \{1, \cdots, n\}\}$ denotes the set of directed edges from $\mathbf{x}_i$ to $\mathbf{x}_j$.

Each node representation $\mathbf{x}_i = [\mathbf{t}_i, \mathbf{s}_i, \theta_i, c_i, \mathbf{f}_i]$ contains all information of an object. $\mathbf{t}_i \in \mathbb{R}^3$, $\mathbf{s}_i \in \mathbb{R}^3$ and $\theta_i \in [-\pi, \pi)$ are the location, size and orientation of the object. For modeling, the orientation θ_i is represented using its sine and cosine values. $c_i \in \{1, \cdots, N_c\}$ is the category of the object, where N_c is number of all categories in the dataset. $\mathbf{f}_i \in \mathbb{R}^f$ denotes the model feature of the object. To effectively represent both the 3D information and the visual semantic features of the object, we utilize OpenShape [LSK*24] to extract features from the textured object model following [LM24]. Then, we also train an autoencoder [HMP*17] on the extracted features to reduce its dimensionality to $f = 32$.

Edge representation encodes the relative relationships $\mathbf{e}_{ij} = [r_{ij}, o_{ij}, p_{ij}, \mathbf{d}_{ij}]$ between two objects, including spatial relationship $r_{ij} \in \{1, \cdots, N_r\}$ such as 'left of' or 'above', orientation relationship $o_{ij} \in \{1, \cdots, N_o\}$ such as 'vertical' or 'parallel', and other abstract relative relationship $p_{ij} \in \{1, \cdots, N_p\}$ such as 'facing' and 'symmetric'. Additionally, $\mathbf{d}_{ij} = \mathbf{t}_i - \mathbf{t}_j$ is the relative position between the two objects. Please refer to the supplementary material for more details.

Interactive scene modification The interactive 3D scene generative model is formularized as $p_\Phi(\mathbf{G}^k | \mathbf{G}^{k-1}, \mathbf{c})$, where k is the step number in the interactive generation process, $\mathbf{G}^0$ could be an empty scene or any partial scene. $\mathbf{c} = (\mathbf{l}, \mathbf{b})$ is the condition, where $\mathbf{l}$ denotes the natural language description and $\mathbf{b}$ denotes the floorplan boundary of the 3D scene. For simplicity, here we represent the floorplan of an indoor room as its four boundaries $\mathbf{b} = [x_{min}, x_{max}, y_{min}, y_{max}]$.

We apply a coarse-to-fine paradigm and decouple the generative process into two stages. The first stage updates the semantic information in the scene graph, denoted as $\mathbf{G}_S = (\mathbf{X}_S, \mathbf{E}_S)$. The node semantic feature $\mathbf{X}_S$ consists of node categories c_i and object model features $\mathbf{f}_i$, and the edge semantic feature $\mathbf{E}_S$ contains all relative relationship semantics r_{ij}, o_{ij} and p_{ij}. After we have updated all semantic information of the scene graph, the second stage generates the exact layout of the 3D scene, denoted as $\mathbf{G}_L = (\mathbf{X}_L, \mathbf{E}_L)$. $\mathbf{X}_L$ is defined by the specific arrangement of every object, including location $\mathbf{t}_i$, size $\mathbf{s}_i$, and orientation θ_i. $\mathbf{E}_L$ include the relative position $\mathbf{d}_{ij}$ between two nodes.

This two-stage generative model can be formulated as

$$p_\Phi(\mathbf{G}^k | \mathbf{G}^{k-1}, \mathbf{c}) = p_{\Phi_1}(\mathbf{G}_S^k | \mathbf{G}_S^{k-1}, \mathbf{c}) p_{\Phi_2}(\mathbf{G}_L^k | \mathbf{G}_S^k, \mathbf{G}^{k-1}, \mathbf{c}) \quad (1)$$

4.2. Scene Graph Diffusion

We build our text-guided interactive scene generation pipeline based on diffusion models. The two stages of the generation process $p_{\Phi_1}(\mathbf{G}_S^k | \mathbf{G}^{k-1}, \mathbf{c})$ and $p_{\Phi_2}(\mathbf{G}_L^k | \mathbf{G}_S^k, \mathbf{G}^{k-1}, \mathbf{c})$ share the same diffusion model design. To simplify notations, here we will omit the subscript of scene graph $\mathbf{G}^k$. The optional condition graphs $\mathbf{G}_S^k$ and $\mathbf{G}^{k-1}$ will also be omitted and viewed as a part of the condition $\mathbf{c}$ for brevity.

Diffusion process. The forward diffusion process progressively adds noise to the graph following the Markov chain until its distribution is close to the latent distribution $q(\mathbf{G}_{1:T} | \mathbf{G}_0) = \prod_{t=1}^{T} q(\mathbf{G}_t | \mathbf{G}_{t-1})$. For our scene graph diffusion, Gaussian noises are added to node and edge features independently according to a

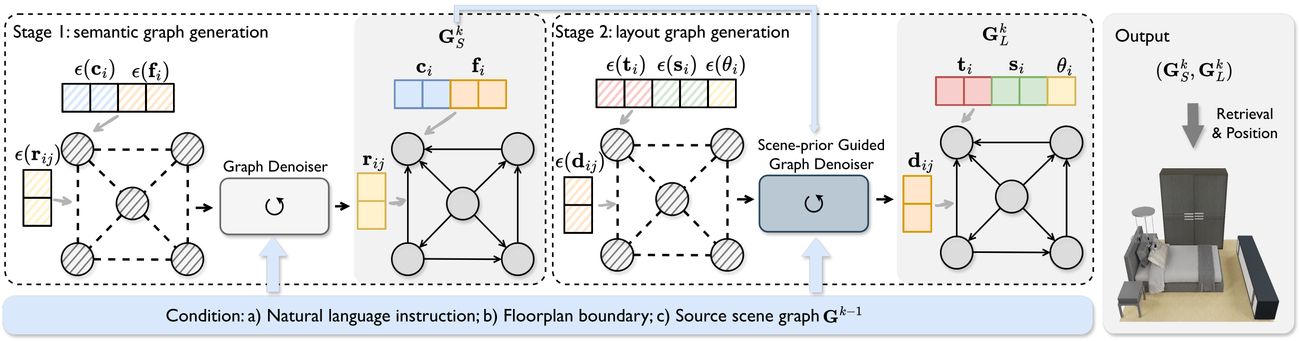

Stage 1: semantic graph generation
$\epsilon(\mathbf{c}_i)$ $\epsilon(\mathbf{f}_i)$
$\epsilon(\mathbf{r}_{ij})$
Graph Denoiser
$\mathbf{G}_S^k$
$\mathbf{c}_i$ $\mathbf{f}_i$
$\mathbf{r}_{ij}$
Stage 2: layout graph generation
$\epsilon(\mathbf{t}_i)$ $\epsilon(\mathbf{s}_i)$ $\epsilon(\theta_i)$
$\epsilon(\mathbf{d}_{ij})$
Scene-prior Guided Graph Denoiser
$\mathbf{G}_L^k$
$\mathbf{t}_i$ $\mathbf{s}_i$ θ_i
$\mathbf{d}_{ij}$
Output
$(\mathbf{G}_S^k, \mathbf{G}_L^k)$
Retrieval & Position
Condition: a) Natural language instruction; b) Floorplan boundary; c) Source scene graph $\mathbf{G}^{k-1}$

Figure 4: Overview of the two-stage scene graph diffusion model.

variance schedule $(\beta_t)_{t=1}^T$.

$$q(\mathbf{X}_t|\mathbf{X}_0) = \mathcal{N}(\mathbf{X}_t; \sqrt{\bar{\alpha}_t}\mathbf{X}_0, (1-\bar{\alpha}_t)\mathbf{I})$$
$$q(\mathbf{E}_t|\mathbf{E}_0) = \mathcal{N}(\mathbf{E}_t; \sqrt{\bar{\alpha}_t}\mathbf{E}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2}$$

where $\alpha_t = 1-\beta_t$, $\bar{\alpha}_t = \prod_{s=0}^T \alpha_s$.

Denoising process. To reverse the diffusion process, the denoising process is also a Markov Chain that predicts and removes the noise gradually, starting from a standard multivariate Gaussian distribution $\mathcal{N}(0,\mathbf{I})$. The denoising process is parameterized by a learned Gaussian transition process, denoted as $p_\theta(\mathbf{G}_{t-1}|\mathbf{G}_t)$. The denoising steps for node and edge are defined as

$$p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t) = \mathcal{N}(\mathbf{X}_{t-1}; \mu_{\theta,X}(\mathbf{G}_t, t, \mathbf{c}), \Sigma_{\theta,\mathbf{X}})$$
$$p_\theta(\mathbf{E}_{t-1}|\mathbf{E}_t) = \mathcal{N}(\mathbf{E}_{t-1}; \mu_{\theta,E}(\mathbf{G}_t, t, \mathbf{c}), \Sigma_{\theta,\mathbf{E}}) \tag{3}$$

where μ_θ and Σ_θ are two models that are trained to predict the mean and variance of the denoising process respectively. For simplicity, Σ_θ is fixed to a constant that is determined by time t following [HJA20]. $\mu_{\theta,X}$ and $\mu_{\theta,E}$ denote the node part and edge part of the graph output predicted by a graph neural network μ_θ.

Also, instead of directly predicting the denoised graph $\mathbf{G}_{t-1}$, we follow the common technique [HJA20] that predicts the noises ϵ_X and ϵ_E added on the graph. Then $\mu_{\theta,X}$ can be formulated as

$$\mu_{\theta,\mathbf{X}}(\mathbf{G}_t, t, \mathbf{c}) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{X}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_{\theta,\mathbf{X}}(\mathbf{G}_t, t, \mathbf{c}))$$
$$\mu_{\theta,\mathbf{E}}(\mathbf{G}_t, t, \mathbf{c}) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{E}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_{\theta,\mathbf{E}}(\mathbf{G}_t, t, \mathbf{c})) \tag{4}$$

The model's optimization objective can be expressed as:

$$\mathcal{L} := \mathbb{E}_{t,\mathbf{G}_0,\epsilon}\left[\|\epsilon_{\mathbf{X}} - \epsilon_{\theta,\mathbf{X}}(\mathbf{G}_t, t, \mathbf{c})\|^2 + \|\epsilon_{\mathbf{E}} - \epsilon_{\theta,\mathbf{E}}(\mathbf{G}_t, t, \mathbf{c})\|^2\right] \tag{5}$$

Therefore, by training the network $\epsilon_\theta(\mathbf{G}_t, t, \mathbf{c})$, we can denoise the graph of standard noise step by step and finally obtain a clean scene graph based on multiple conditions, including natural language instructions, floorplan boundaries or previous scene graph.

Model Architecture The backbone of our model is based on a general graph transformer model [DB20]. Before each layer of the graph neural network, all conditioning information, including the floorplan boundary and node and edge embeddings of the source scene, are fused to the node and edge features. A frozen CLIP text encoder [RKH*21] is utilized to extract textual features, and an additional cross-attention module [VSP*17] is injected into the graph neural network to integrate the textual guidance features.

ALGORITHM 1: Scene-prior guided diffusion sampling

```
Input: Denoiser $\mu_\theta$, condition $\mathbf{c}$, scene-prior cost function
    $\mathcal{S}$, guidance threshold $m$.
$\mathbf{X}_T, \mathbf{E}_T \leftarrow$ sample from $\mathcal{N}(0,\mathbf{I})$
$G_T \leftarrow (\mathbf{X}_T, \mathbf{E}_T)$
for $t = T, \ldots, 1$ do
    $(\mu_\mathbf{X}, \mu_\mathbf{E}), \Sigma \leftarrow \mu_\theta(G_t, t, \mathbf{c}), \Sigma_\theta(t)$
    if $t < m$ then
        $\mu_\mathbf{X} \leftarrow \mu_\mathbf{X} + s\Sigma\nabla_{\mathbf{X}_t}\mathcal{S}(G_t)$
    end
    $\mathbf{X}_{t-1}, \mathbf{E}_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu_\mathbf{X}, \Sigma), \mathcal{N}(\mu_\mathbf{E}, \Sigma)$
    $G_{t-1} \leftarrow (\mathbf{X}_{t-1}, \mathbf{E}_{t-1})$
end
return $G_0$
```

4.3. Scene Priors Definition

We have defined a series of scene prior cost functions to explicitly reflect the authenticity, physical feasibility, and uncertainty of object arrangement in a 3D indoor scene.

Constraints. First, the arrangement of a 3D scene must adhere to some constraints. It should prevent physical inaccuracies, such as objects intersecting spatially or being placed outside the floor plan boundary. Following [YZY*21], we consider the constraints of an indoor scene within the 2D plane from the top-down view. In 3D indoor scenes, it is possible for objects to overlap in a top-down view, such as ceiling lamps and beds or tables and chairs. We employ an indicator $\mathbf{I}_{c_i,c_j}$ to determine whether overlapping between two categories of objects is permissible and exclude these pairs from the cost function calculation. The cost function for scene constraints is defined as

$$\begin{aligned}
\mathcal{L}_{overlap} &= \sum_{x_i,x_j\in\mathbf{X}} \mathbf{I}_{c_i,c_j}\cdot \mathrm{IOU}_{2D}(x_i,x_j) \\
\mathcal{L}_{bound} &= \sum_{x_i\in\mathbf{X}} \|\mathrm{relu}(b_{min}-a^i_{min})\|^2 + \\
&\quad \|\mathrm{relu}(a^i_{max}-b_{max})\|^2 \\
\mathcal{L}_{constraint} &= \mathcal{L}_{overlap} + \mathcal{L}_{bound}
\end{aligned} \tag{6}$$

		Object-level				Scene-level			Overall
		Acc (↑)	Err$_t$(↓)	Err$_a$(↓)	iRecall (↑)	Err$_t$(↓)	Err$_a$(↓)	iRecall (↑)	iRecall (↑)
Bedroom	DiffuScene	96.48	26.50	11.32	73.02	132.94	69.36	63.54	71.74
	InstructScene*	98.30	24.64	9.32	76.90	107.72	64.95	69.79	75.94
	Ours	**98.30**	**18.05**	**6.74**	**82.07**	**104.86**	**57.33**	**71.88**	**80.70**
Living/Dining	DiffuScene	88.99	23.39	8.94	57.35	228.05	74.12	41.30	55.47
	InstructScene*	96.49	20.20	7.21	63.69	204.32	75.81	47.83	61.83
	Ours	**96.49**	**17.23**	**5.41**	**69.74**	**192.06**	**73.55**	**54.38**	**67.94**

Table 2: Quantitive evaluations for text-guided scene modification tasks.

where $b_{min} = (x_{min,y_{min}})$ and $b_{max} = (x_{max}, y_{max})$ are the bottom-left and top-right corners of the floor plan boundary and a^i_{min} and a^i_{max} are the bounding box's bottom-left and top-right corners of each object node x_i.

Arrangement uncertainty. In indoor scene design, layout arrangements often exhibit specific patterns, including the relative attributes between objects or the absolute attributes of the objects themselves. For example, TV stands are typically arranged opposite sofas, and wardrobes are always placed against a wall. To extract these patterns, following [YZY*21], we use the generalized Gaussian mixture models to learn parametric prior distributions for both absolute and relative attributes. Then, we can define a layout arrangement uncertainty cost function based on the prior distributions.

$$\mathcal{L}_{uncertainty} = \sum_{x_i \in \mathbf{X}} -\log M_{\mu_{c_i}}(x_i) + \sum_{e_{ij} \in \mathbf{E}} -\log M_{\mu_{(c_i,c_j)}}(e_{ij}) \tag{7}$$

Here M denotes the Gaussian mixture models and μ_{c_i} and $\mu_{(c_i,c_j)}$ are hyperparameters of the mixture models that are learned from the training datasets. Please refer to the supplementary material for details.

4.4. Scene Prior Guidance

During model training, the models implicitly learn specific patterns through a data-driven approach. However, the generated scenes cannot guarantee the satisfaction of constraint conditions, and the scene's arrangement uncertainty is not explicitly incorporated into the model. Therefore, we introduce scene-prior guidance during the denoising sampling process. The cost function is formulated as follows

$$\mathcal{S}(\mathbf{G}) = \lambda_1 \mathcal{L}_{constraint} + \lambda_2 \mathcal{L}_{uncertainty} \tag{8}$$

where λ_1 and λ_2 are the balancing parameters. The scene-prior guided diffusion sampling is shown in Algorithm 1. Scene prior guidance is applied only during the latter part of the sampling process when the noise level is relatively low, controlled by the guidance threshold m. This guidance can also be considered a refinement mechanism during the diffusion sampling process, steering the scene toward a more accurate and realistic result.

5. Experimental Results

5.1. Experimental settings

We conduct experiments on two text-guided scene synthesis tasks: text-guided scene modification and text-guided scene generation. The text-guided scene modification experiments are conducted on the proposed SceneMod dataset, which is derived from 3D-Front [FCG*21] and includes two room types: bedrooms and a combined category of living rooms and dining rooms. The text-guided scene generation experiments are conducted on 3D-Front. We follow the same data split for training and evaluation as specified by [PKS*21] and conduct experiments for both bedroom and living room types separately.

Baselines We compare our results with three state-of-the-art methods: (a) Atiss [PKS*21], a transformer-based method that predicts 3D objects in sequence. Due to its autoregressive generation approach, transformer-based methods are not well-suited for scene modification tasks. While they can perform complete scene rearrangement by resampling from learned distributions, they offer limited control over modifying specific objects within an existing scene, such as adding, removing, or repositioning individual objects. Therefore, we only compare it in the context of text-guided scene generation. (b) Diffuscene [TNM*23], a diffusion-based method that generates scenes as an unordered set of objects. For the text-guided scene modification task, we use the original scene as an additional condition for the diffusion model. To integrate the original scene, we apply the same method used for fusing the source room node condition in our graph diffusion method. Specifically, the source room information is encoded and then concatenated with the node features before being fed into the denoising network. (c) InstructScene [LM24], which also employs a two-stage generation process. In the first stage, it generates a semantic graph driven by text instructions, and the second stage is a general layout decoder that decodes the scene from the predicted semantic graph. To support modification tasks, we use the same semantic graph generation model (first-stage generation) as our method.

Evaluation Metrics For the text-guided scene generation task, we follow previous work [LM24] and report the instruction recall (iRecall) to assess the accuracy of scene generation based on the instructions. Moreover, we report the Fréchet Inception Distance (FID) score [HRU*17] and Scene Completion Accuracy (SCA) score to measure the plausibility and diversity of the generated scenes based on their rendered images. To ensure a fair comparison, we adopt the same rendering method as [LM24].

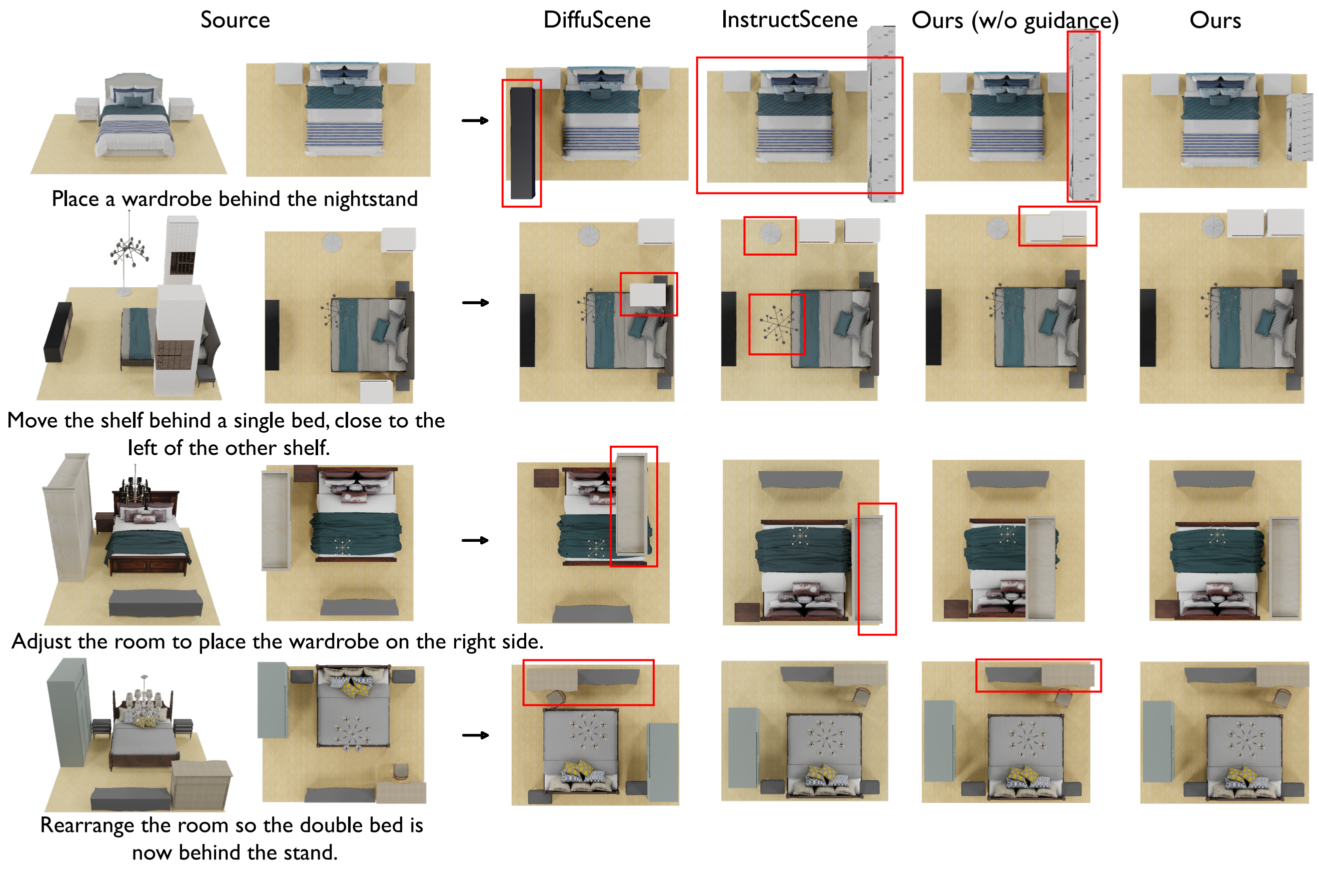

Source
DiffuScene
InstructScene
Ours (w/o guidance)
Ours
Place a wardrobe behind the nightstand
Move the shelf behind a single bed, close to the left of the other shelf.
Adjust the room to place the wardrobe on the right side.
Rearrange the room so the double bed is now behind the stand.

Figure 5: Qualitative results for text-guided scene modification. The results show that our method outperforms the baseline models, and scene prior guidance can act as a refinement of the generated results.

For the novel scene modification tasks, we use four kinds of metrics to evaluate model performance: instruction recall (iRecall), object selection accuracy (Acc), location error (Err_t), and angle error (Err_a). iRecall evaluates whether the variation patterns described in the textual descriptions are accurately implemented in the modified scenes. Object selection accuracy (Acc) measures the accuracy of the object category for the add and remove operation in object-level modifications. Err metrics measure discrepancies between corresponding objects in predicted and ground truth scenes. Err_t is defined by the distance between locations, and Err_a is defined by the intersection angle difference. Assume $\mathbf{X}^{pred} = \left\{\mathbf{x}_i^{pred} | i \in \{1, \cdots, n\}\right\}$ is the generated scene after modification and $\mathbf{X}^{gt}$ is the ground truth scene.

$$\begin{aligned} \text{Err}_t &= \frac{1}{n}\sum_{i=1}^{n} \|t_i^{pred} - t_i^{gt}\|^2 \\ \text{Err}_a &= \frac{1}{n}\sum_{i=1}^{n} \text{intersec}(\theta_i^{pred}, \theta_i^{gt}) \end{aligned} \tag{9}$$

Implementation details We train the models on various types of indoor rooms for text-guided scene generation and scene modification tasks, respectively. Please refer to the supplementary material for more implementation details.

5.2. Quantitative results

Text-guided scene modification Table 2 presents a comprehensive comparison between our proposed methods and other baselines for text-guided scene modification. Our method outperforms the two baselines in both object-level and scene-level modification tasks across both room types, showing improvements of up to 10% compared to previous methods. Note that for InstructScene* here, to support modification tasks, we use the same semantic graph generation model (first-stage generation) as our method. InstructScene's performance is unsatisfactory because it uses a general unconditional layout decoder, which cannot handle modifications involving absolute position changes of objects. Also, it is unable to maintain the positions of other objects in the scene during object-level modifications. Our method consistently outperforms others in all settings, clearly demonstrating that our method is more suitable for interactive scene synthesis tasks.

Text-guided scene generation Table 3 shows the quantitative results for the text-guided scene generation tasks. Our results are comparable to the state-of-the-art method and outperform all other methods in the bedroom type for both FID and SCA metrics. Our

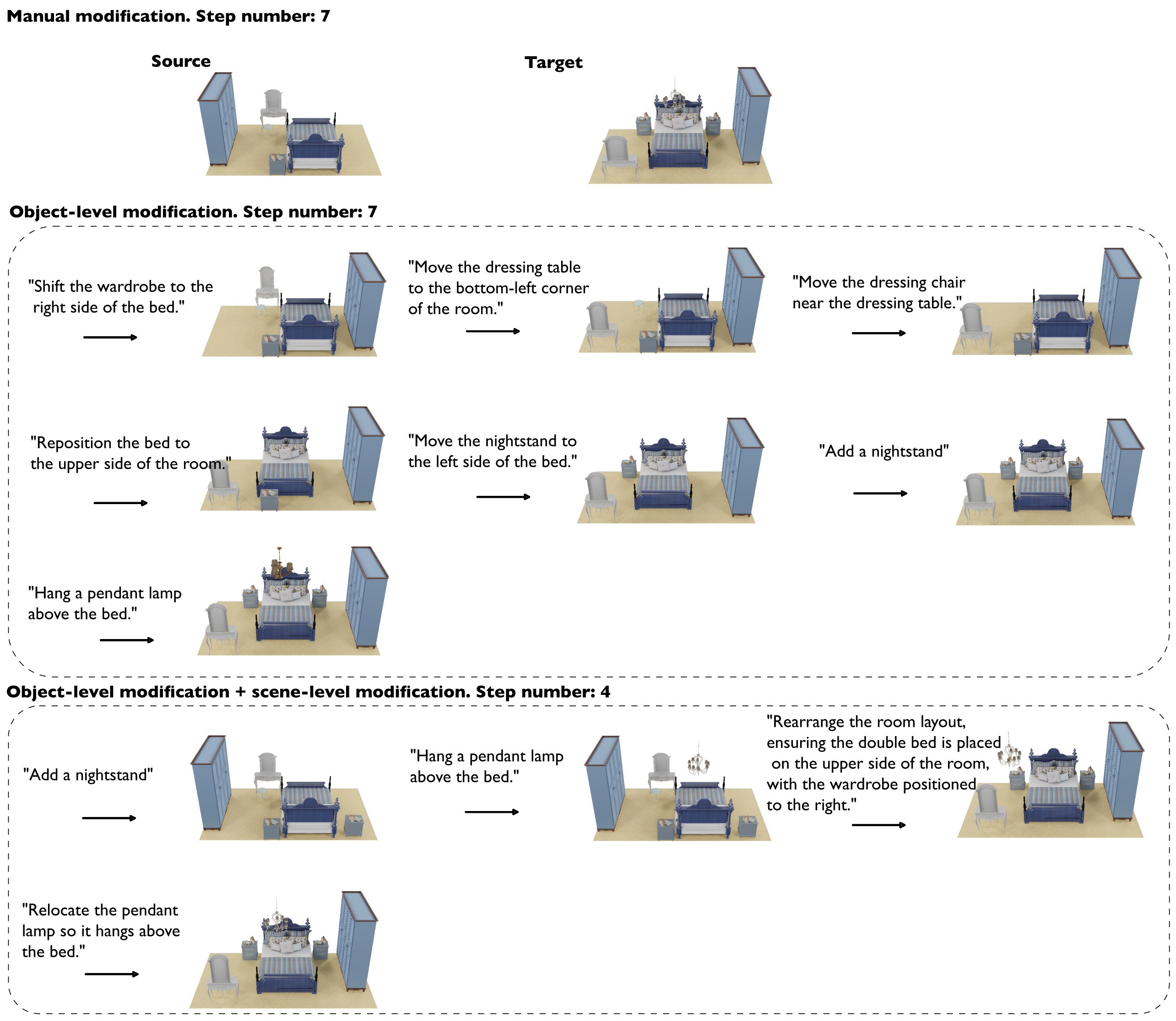

Manual modification. Step number: 7
Source
Target
Object-level modification. Step number: 7
"Shift the wardrobe to the right side of the bed."
"Move the dressing table to the bottom-left corner of the room."
"Move the dressing chair near the dressing table."
"Reposition the bed to the upper side of the room."
"Move the nightstand to the left side of the bed."
"Add a nightstand"
"Hang a pendant lamp above the bed."
Object-level modification + scene-level modification. Step number: 4
"Add a nightstand"
"Hang a pendant lamp above the bed."
"Rearrange the room layout, ensuring the double bed is placed on the upper side of the room, with the wardrobe positioned to the right."
"Relocate the pendant lamp so it hangs above the bed."

Figure 6: Interactive generation example. Given the current source scene and a target scene, the objective is to interactively update the source scene towards the target scene in a minimum number of steps.

results are slightly inferior to InstructScene, which employs a discrete diffusion method for generating the semantic graph, potentially making it more suitable for generating the one-hot semantic categories. However, it is important to note that the discrete diffusion method used in InstructScene can be directly applied to our framework. Results in Table 2 show that when using the same semantic graph generation model, our method outperforms InstructScene. To maintain uniformity and simplicity, we just use the same diffusion model in both generation stages of our method.

5.3. Ablation studies

Scene prior guidance Table 4 presents the results of the ablation study on the selection of scene prior guidance. The experiments are conducted on scene modification tasks for the bedroom type. We report metrics for both physical plausibility and modification error. Overlap and Out-of-bound represent the percentage of issues where a modified object overlaps with other objects or exceeds the boundary. Err is the average location error Err_t for all object-level and scene-level modification samples. Using constraint prior significantly enhances the physical plausibility of the generated scenes, halving the percentage of unreasonable arrangements while keeping the error metric nearly constant. Furthermore, incorporating arrangement uncertainty prior allows us to further improve modification accuracy while maintaining high levels of geometric and physical correctness.

Two-stage generation Table 5 presents the results of the ablation study on different generation stages. In the one-stage approach, all scene parameters are generated by the diffusion model,

		Text-guided generation		
		iRecall (↑)	FID (↓)	SCA(%)
Bedroom	ATISS	48.13	119.73	59.17
	DiffuScene	56.43	123.09	60.49
	InstructScene	**73.64**	114.78	56.02
	Ours	72.84	**113.46**	**55.57**
Living	ATISS	29.50	117.67	69.38
	DiffuScene	31.15	122.20	72.92
	InstructScene	**56.81**	**110.39**	**65.42**
	Ours	53.19	112.08	66.15

Table 3: Quantitive evaluations for text-guided scene generation tasks.

	Overlap	Out-of-bound	Err
w/o guidance	16.38	16.32	30.00
w/ constraint	**7.20**	11.64	30.16
w/ uncertainty	13.94	21.86	29.97
w/ full guidance	7.42	**11.64**	**29.71**

Table 4: Ablation for scene prior guidance

which is trained under the same training settings with diffusion steps $T = 100$. The model is tested on the bedroom object-level modification task. The one-stage approach shows a notable drop in performance, with lower accuracy in following modification instructions and reduced spatial precision. The two-stage method addresses these issues by first generating a coarse layout and then refining object placement, leading to better alignment with instructions and improved geometric accuracy. These results highlight the advantages of the two-stage design in producing more precise and reliable scene modifications.

5.4. Qualitative results

Qualitative results Qualitative comparisons, as shown in Figure 5, demonstrate that the scene prior guidance acts as a refinement to the generated results. Also, our method outperforms the baseline models. InstructScene [LM24] uses a general unconditional layout decoder, so it cannot maintain the positions of other objects in the scene during object-level modifications.

Interactive generation In Figure 6, we present an interactive modification example using our proposed model. Given a current source scene and a target scene we aim to achieve, our model progressively updates the source scene to match the target scene. If a user manually operated the modification process, the operation steps would correspond to the number of objects in the scene. We demonstrate two modification sequences: one using only object-level modification operations and another combining both object-level and scene-level modifications. Our interactive approach allows scene generation and modification to be easily performed using text descriptions, even with only object-level modification operations, thereby eliminating manual operations. Furthermore, it demonstrates that the combined use of scene-level and object-level modification operations can significantly enhance the efficiency of scene editing. The entire scene can undergo a complex update in just two steps, showcasing the effectiveness and rationality of our operation setting that supports multi-scale modification. Please refer to the supplementary material for more quantitative comparisons and applications.

	Acc (↑)	Err_t(↓)	Err_a(↓)	iRecall (↑)
one-stage	95.71	26.19	10.70	74.96
two-stage	98.30	18.05	6.74	82.07

Table 5: Ablation for generation stages

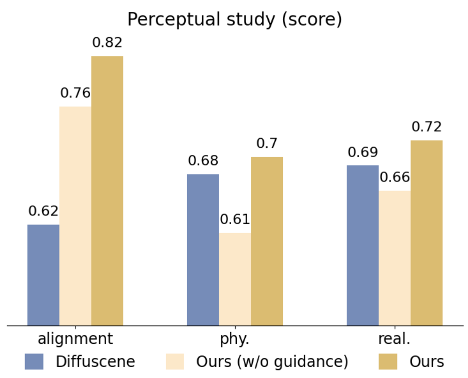


Figure 7: Perceptual study results. Scores on text alignment, physical correctness, and realism.

5.5. Perceptual study

We conduct perceptual studies with 6 graduate participants to evaluate the quality of the modification results against [TNM*23] and the effectiveness of the scene-prior guidance design. We randomly sampled 100 examples from the SceneMod test split, including object-level and scene-level modification samples across various types of modifications. Models' performance is measured according to three different aspects: 1) Text alignment. ("Does the generated scene align with the modification described in the text content?"), 2) physical correctness ("Is the arrangement of the objects correct and physically plausible?"), 3) functionality and realism. ("Is the generated scene realistic, and does it adhere to the room's function?"). Participants are shown one result from one method at a time and asked to rate each result as Yes (1), No (0), or Not Sure (0.5). The average scores are displayed in Fig. 7. Then, participants are shown side-by-side results from three models with the same source scene and text instruction and asked to determine 1) which follows instructions better and 2) looks more realistic. Fig. 8 depicts the results. Fig. 7 and Fig. 8 show that our full model can generate more realistic and physically plausible scene modifications according to the text descriptions. Additionally, the scene-prior guidance's design enhances the generated results quality in all aspects.

6. Conclusions

We introduce a text-guided interactive scene synthesis system that combines data-driven learning and scene-prior guidance, supporting progressive generation and modification. To the best of our

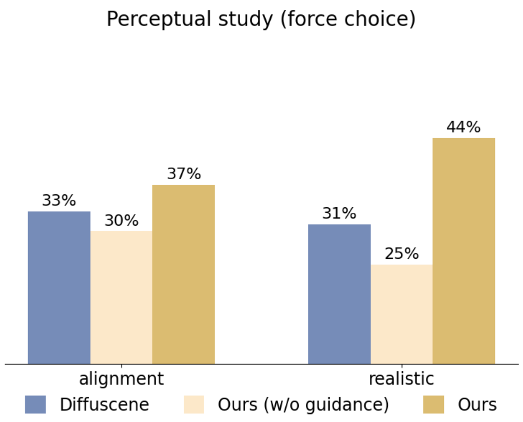


Figure 8: Perceptual study results. Force choice on text alignment and realism.

knowledge, we are the first to integrate text-guided interaction in the data-driven scene synthesis process. We propose a large-scale dataset, SceneMod, to support the training of interactive scene modification. This dataset includes numerous viable scene modification pairs and corresponding natural language instructions. Qualitative and quantitative results demonstrate that the proposed method is effective for interactive scene synthesis and that scene-prior guidance makes the generated results more plausible and realistic.

Acknowledgements

This work was supported by Damo Academy through Damo Academy Innovative Research Program.

References

[AAA*23] Achiam J., Adler S., Agarwal S., Ahmad L., Akkaya I., Aleman F. L., Almeida D., Altenschmidt J., Altman S., Anadkat S., et al.: Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023). 5

[AKGH*24] Aguina-Kang R., Gumin M., Han D. H., Morris S., Yoo S. J., Ganeshan A., Jones R. K., Wei Q. A., Fu K., Ritchie D.: Open-universe indoor scene generation using llm program synthesis and uncurated object databases. *arXiv preprint arXiv:2403.09675* (2024). 3

[CESM17] Chang A. X., Eric M., Savva M., Manning C. D.: Sceneseer: 3d scene design with natural language. *arXiv preprint arXiv:1703.00050* (2017). 3

[CLW*14] Chen K., Lai Y.-K., Wu Y.-X., Martin R., Hu S.-M.: Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM Transactions on Graphics 33*, 6 (2014). 2

[CMS*15] Chang A., Monroe W., Savva M., Potts C., Manning C. D.: Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289* (2015). 3

[CSM14] Chang A., Savva M., Manning C. D.: Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 2028–2038. 3

[CXK*24] Chen B., Xu Z., Kirmani S., Ichter B., Driess D., Florence P., Sadigh D., Guibas L., Xia F.: Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. *arXiv preprint arXiv:2401.12168* (2024). 3

[DB20] Dwivedi V. P., Bresson X.: A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699* (2020). 6

[DMNT21] Dhamo H., Manhardt F., Navab N., Tombari F.: Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 16352–16361. 2, 3

[FCG*21] Fu H., Cai B., Gao L., Zhang L., Li J. W. C., Xun Z., Sun C., Jia R., Zhao B., Zhang H.: 3d-front: 3d furnished rooms with layouts and semantics, 2021. arXiv:2011.09127. 2, 7

[FRS*12] Fisher M., Ritchie D., Savva M., Funkhouser T., Hanrahan P.: Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 1–11. 2

[FZF*24] Feng W., Zhu W., Fu T.-j., Jampani V., Akula A., He X., Basu S., Wang X. E., Wang W. Y.: Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems 36* (2024). 3

[HHT*20] Hu R., Huang Z., Tang Y., Van Kaick O., Zhang H., Huang H.: Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 118–1. 2

[HJA20] Ho J., Jain A., Abbeel P.: Denoising diffusion probabilistic models. *Advances in neural information processing systems 33* (2020), 6840–6851. 6, 14

[HMP*17] Higgins I., Matthey L., Pal A., Burgess C. P., Glorot X., Botvinick M. M., Mohamed S., Lerchner A.: beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster) 3* (2017). 5

[HRU*17] Heusel M., Ramsauer H., Unterthiner T., Nessler B., Hochreiter S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems 30* (2017). 7

[KCKK12] Kalogerakis E., Chaudhuri S., Koller D., Koltun V.: A probabilistic model for component-based shape synthesis. *Acm Transactions on Graphics (TOG) 31*, 4 (2012), 1–11. 2

[LCK*14] Liu T., Chaudhuri S., Kim V. G., Huang Q., Mitra N. J., Funkhouser T.: Creating consistent scene graphs using a probabilistic grammar. *ACM Transactions on Graphics (TOG) 33*, 6 (2014), 1–12. 2

[LGWM22] Leimer K., Guerrero P., Weiss T., Musialski P.: Layoutenhancer: Generating good indoor layouts from imperfect data. In *SIGGRAPH Asia 2022 Conference Papers* (2022), pp. 1–8. 2

[LM24] Lin C., Mu Y.: Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. *arXiv preprint arXiv:2402.04717* (2024). 1, 2, 3, 5, 7, 10, 13

[LPX*19] Li M., Patil A. G., Xu K., Chaudhuri S., Khan O., Shamir A., Tu C., Chen B., Cohen-Or D., Zhang H.: Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG) 38*, 2 (2019), 1–16. 2

[LSK*24] Liu M., Shi R., Kuang K., Zhu Y., Li X., Han S., Cai H., Porikli F., Su H.: Openshape: Scaling up 3d shape representation towards open-world understanding. *Advances in Neural Information Processing Systems 36* (2024). 5

[LXJ*23] Liu J., Xiong W., Jones I., Nie Y., Gupta A., Oğuz B.: Clip-layout: Style-consistent indoor scene synthesis with semantic furniture embedding. *arXiv preprint arXiv:2303.03565* (2023). 1, 3

[LZWT20] Luo A., Zhang Z., Wu J., Tenenbaum J. B.: End-to-end optimization of scene layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 3754–3763. 3

[MMM*24] Melnik A., Miasayedzenkau M., Makaravets D., Pirshtuk D., Akbulut E., Holzmann D., Renusch T., Reichert G., Ritter H.: Face generation and editing with stylegan: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024). 1

[MPF*18] Ma R., Patil A. G., Fisher M., Li M., Pirk S., Hua B.-S., Yeung S.-K., Tong X., Guibas L., Zhang H.: Language-driven synthesis of 3d scenes from scene databases. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–16. 3

[MSL*11] Merrell P., Schkufza E., Li Z., Agrawala M., Koltun V.: Interactive furniture layout using interior design guidelines. *ACM transactions on graphics (TOG) 30*, 4 (2011), 1–10. 2, 3

[PKS*21] Paschalidou D., Kar A., Shugrina M., Kreis K., Geiger A., Fidler S.: Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems 34* (2021), 12013–12026. 1, 2, 3, 5, 7

[PPL*24] Patil A. G., Patil S. G., Li M., Fisher M., Savva M., Zhang H.: Advances in data-driven analysis and synthesis of 3d indoor scenes. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e14927. 1

[PZR20] Purkait P., Zach C., Reid I.: Sg-vae: Scene grammar variational autoencoder to generate new indoor scenes. In *European Conference on Computer Vision* (2020), Springer, pp. 155–171. 2

[RKH*21] Radford A., Kim J. W., Hallacy C., Ramesh A., Goh G., Agarwal S., Sastry G., Askell A., Mishkin P., Clark J., et al.: Learning transferable visual models from natural language supervision. In *International conference on machine learning* (2021), PMLR, pp. 8748–8763. 6

[RWL19] Ritchie D., Wang K., Lin Y.-a.: Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 6182–6190. 2

[SCA17] Savva M., Chang A. X., Agrawala M.: Scenesuggest: Context-driven 3d scene design. *arXiv preprint arXiv:1703.00061* (2017). 3

[TNM*23] Tang J., Nie Y., Markhasin L., Dai A., Thies J., Niessner M.: Diffuscene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis. *arXiv preprint arXiv:2303.14207* (2023). 1, 2, 3, 5, 7, 10

[VSP*17] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I.: Attention is all you need. *Advances in neural information processing systems 30* (2017). 3, 6

[WDNT20] Wald J., Dhamo H., Navab N., Tombari F.: Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 3961–3970. 2

[WDP*23] Wei Q. A., Ding S., Park J. J., Sajnani R., Poulenard A., Sridhar S., Guibas L.: Lego-net: Learning regular rearrangements of objects in rooms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 19037–19047. 2

[WLSF23] Wen Z., Liu Z., Sridhar S., Fu R.: Anyhome: Open-vocabulary generation of structured and textured 3d homes. *arXiv preprint arXiv:2312.06644* (2023). 3

[WLW*19] Wang K., Lin Y.-A., Weissmann B., Savva M., Chang A. X., Ritchie D.: Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–15. 2

[WSCR18] Wang K., Savva M., Chang A. X., Ritchie D.: Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–14. 2

[WYN21] Wang X., Yeshwanth C., Niessner M.: Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)* (2021), IEEE, pp. 106–115. 1, 2, 3

[YBL*23] Yamada Y., Bao Y., Lampinen A. K., Kasai J., Yildirim I.: Evaluating spatial understanding of large language models. *arXiv preprint arXiv:2310.14540* (2023). 3

[YGZT21] Yang M.-J., Guo Y.-X., Zhou B., Tong X.: Indoor scene generation from a collection of semantic-segmented depth images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 15203–15212. 2

[YSW*24] Yang Y., Sun F.-Y., Weihs L., VanderBilt E., Herrasti A., Han W., Wu J., Haber N., Krishna R., Liu L., et al.: Holodeck: Language guided generation of 3d embodied ai environments. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2024)* (2024), vol. 30, IEEE/CVF, pp. 20–25. 3

[YYT*11] Yu L. F., Yeung S. K., Tang C. K., Terzopoulos D., Chan T. F., Osher S. J.: Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH 2011, v. 30,(4), July 2011, article no. 86 30*, 4 (2011). 2

[YYT15] Yu L.-F., Yeung S.-K., Terzopoulos D.: The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE transactions on visualization and computer graphics 22*, 2 (2015), 1138–1148. 3

[YYW*12] Yeh Y.-T., Yang L., Watson M., Goodman N. D., Hanrahan P.: Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 1–11. 2

[YZS*23] Yang L., Zhang Z., Song Y., Hong S., Xu R., Zhao Y., Zhang W., Cui B., Yang M.-H.: Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys 56*, 4 (2023), 1–39. 1

[YZY*21] Yang H., Zhang Z., Yan S., Huang H., Ma C., Zheng Y., Bajaj C., Huang Q.: Scene synthesis via uncertainty-driven attribute synchronization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 5630–5640. 2, 6, 7, 14

[ZÖW*24] Zhai G., Örnek E. P., Wu S.-C., Di Y., Tombari F., Navab N., Busam B.: Commonscenes: Generating commonsense 3d indoor scenes with scene graphs. *Advances in Neural Information Processing Systems 36* (2024). 2, 3

[ZTL*23] Zhang S.-K., Tam H., Li Y., Ren K.-X., Fu H., Zhang S.-H.: Scenedirector: Interactive scene synthesis by simultaneously editing multiple objects in real-time. *IEEE Transactions on Visualization and Computer Graphics* (2023). 3

[ZWK19] Zhou Y., While Z., Kalogerakis E.: Scenegraphnet: Neural message passing for 3d indoor scene augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 7384–7392. 2, 3

[ZYM*20] Zhang Z., Yang Z., Ma C., Luo L., Huth A., Vouga E., Huang Q.: Deep generative modeling for scene synthesis via hybrid representations. *ACM Transactions on Graphics (TOG) 39*, 2 (2020), 1–21. 2

7. SceneMod dataset

7.1. Scene Pairs Generation and Selection

We generate scene modification pairs through three methods. For modification pairs of add and remove, we sample $k+1$ objects from an original 3D-Front scene sample $\mathbf{X}$ containing n objects. The pairs ($\mathbf{X}_{1:k}$, $\mathbf{X}_{1:k+1}$) and ($\mathbf{X}_{1:k+1}$, $\mathbf{X}_{1:k}$) constitute scene modification pairs for 'add' operation and 'remove' operation, respectively. The object number k of the sub-scene is randomly selected with $k > 3$ and $k > n-4$. Therefore, in our interactive modification task, we consider an initialized scene with a certain number of objects.

Moreover, To obtain object-level movement and scene-level rearrangement modification pairs, we train two generative models: $p_{\psi_1}(\hat{x}_k|\mathbf{X_{1:k-1}}, c_k, \mathbf{f}_k)$ for object placement suggestion and $p_{\psi_2}(\tilde{\mathbf{X}}_{1:k}|\mathbf{X}_{1:k}) = p_{\psi_2}(\tilde{\mathbf{X}}_{1:k}|\{c_i, \mathbf{f}_i\}_{1:k})$ for scene rearrangement suggestion. We can use any generative models for p_{ψ_1} and p_{ψ_2}; here, we use the same two-stage diffusion method as in our experiments, modifying only the input conditions as needed.

To train the generative models p_{ψ_1} and p_{ψ_2}, we use half of the samples from each train/val/test split used for training and the other half for generating pairs. Specifically, for the bedroom category, we use a total of 2041 samples for training and 2000 samples for generating pairs. For the living/dining category, we use 600 samples for training and 575 samples for generating pairs. During training, for each scene in the training set, we also sample a sub-scene $\mathbf{X}_k$ as the training sample. We employ the same training strategy and implementation settings as in the text-guided scene generation experiments in the paper.

After training p_{ψ_1} and p_{ψ_2}, we generate scenes using the remaining half of the data. Since this half includes data from the original train/val/test splits, we can create new dataset splits for SceneMod accordingly. For each sample $\mathbf{X}$, we generate n (n=32) candidate results using the trained models. As these are not manually annotated, the quality of the generated results is not guaranteed. Therefore, we filter out layouts with severe violations of scene constraints, such as significant overlaps and out-of-boundary placements, and then select the results with lower energy according to the scene prior cost function defined in Sec. 4.3, forming the modification pairs for the SceneMod dataset.

7.2. Spatial Variation Detection and Description Generation

The generation of textual instructions starts with detecting low-level spatial variations between pairs of scenes. We first identify the object-level variations between pairs of scenes. Denote $\mathbf{x}_i$ as the object that is modified and selected for description. We primarily consider four types of spatial variation patterns.

i) Changes of relative relationships. A relative relationship can be represented as ($\mathbf{x}_i$, r_{rel}, $\mathbf{x}_j$), where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two objects in the scene and r_{rel} is the relation type between $\mathbf{x}_i$ and $\mathbf{x}_j$. we use the same range of relative relationships as in the edge attributes of the scene graph representations introduced in Section 8.1. Therefore, the calculation and detection of r_{rel} also follow the definition used in scene graph construction.

ii) Changes of absolute position. The absolute position pattern can be denoted as ($\mathbf{x}_i$, r_{abs}), and we consider four types of $r_{abs} \in$ {'left', 'right', 'front', 'back'}, which indicate the part of the room the object is in or its proximity to a specific boundary. Recall the boundary is denoted as $\mathbf{b} = [x_{min}, x_{max}, y_{min}, y_{max}]$, the pattern ($\mathbf{x}_i$, 'left') means
$distance(\mathbf{x}_i, x_{min}) < d$ or $(t_x - x_{min})/(x_{max} - x_{min}) < 1/4$, and the other three patterns have the similar definition.

iii) Changes of relative distance. If the directional relationship between two objects $\mathbf{x}_i$ and $\mathbf{x}_j$ remains the same, but the distance relationship changes, such as changing from 'closely right of' to 'right of', we can describe the change in the distance: e.g. 'Move object A closer to/farther from object B.' This type of variation pattern can be denoted as $(\mathbf{x}_i, \{closer/farther\}, \mathbf{x}_j)$.

iv) Changes of absolute position. If an object's position changes along a single axis (x-axis or y-axis), we can describe the modification directionally. For example, 'Move object A forward/backward.'. The variation pattern can be denoted as $(\mathbf{x}_i, \{left/front/right/back\})$.

As shown in Table 6, after confirming the operation type and detecting the variation patterns, we can generate the corresponding instructions based on specific rules. We select 0-2 variation patterns for each modification pairs to describe the operation. For scene-level modifications, we describe the changes in the entire scene through object-level variations while also providing additional instructions for altering the entire scene (e.g., 'Rearrange the whole scene.').

7.3. Description Refinement

The generated structural description from the identified spatial variation patterns is then refined using the Chatgpt (gpt-3.5-turbo) API. The prompt for the paraphrase is

"Given an instruction generated according to specific structural rules and describes a modification operation to a given scene, please paraphrase it to sound like a person speaks it in a natural, conversational tone. Note that you must not alter the intended actions in the original instruction. There may also exist some directional content, such as 'left' and 'behind'; the meaning of this content must remain unchanged."

8. Method

8.1. Scene Graph Relationships

In our scene graph representation implementation, object node is represented as $\mathbf{x}_i = [\mathbf{t}_i, \mathbf{s}_i, \theta_i, c_i, \mathbf{f}_i]$ and edge attribute is represented as $\mathbf{e}_{ij} = \left[r_{ij}, o_{ij}, p_{ij}, \mathbf{d}_{ij}\right]$. Here, we define three kinds of categorical relative relationships for edge attributes: spatial relationship r, orientation relationship o, and abstract relative relationship p.

For the spatial relationship r_{ij}, we adopt the same definition as in [LM24], which categorizes the relative positions of the center points of objects i and j into 11 classes: 'left of', 'right of', 'in front of', 'behind', 'closely left of', 'closely right of', 'closely in front of', 'closely behind', 'above', 'below', and 'None'.

The orientation relationship o_{ij} considers the directional relationship between two objects, categorized into four classes: parallel, orthogonal, antiparallel, and none. These are defined based on

Operation	Variation pattern	Raw description example generated by rules
Remove	$(\mathbf{x}_i, r_{rel}, \mathbf{x}_j)$	Remove *object i* that is r_{rel} *object j*.
	$(\mathbf{x}_i, r_{abs})$	Delete *object i* that is in the r_{abs} part of the room.
Add	$(\mathbf{x}_i, r_{rel}, \mathbf{x}_j)$	Add *object i* to r_{rel} *object j*.
	$(\mathbf{x}_i, r_{abs})$	Place *object i* in the r_{abs} part of the room.
Move	$(\mathbf{x}_i, r_{rel}, \mathbf{x}_j)$ or $(\mathbf{x}_i, r_{abs}) \rightarrow (\mathbf{x}_i, \hat{r}_{rel}, \mathbf{x}_k)$ or $(\mathbf{x}_i, \hat{r}_{abs})$	Reposition *object i*, which is $\hat{r}_{rel}$ *object j*, to r_{rel} *object k*.
	$\varnothing \rightarrow (\mathbf{x}_i, \hat{r}_{rel}, \mathbf{x}_j)$ or $(\mathbf{x}_i, \hat{r}_{abs})$	Relocate *object i* to r_{rel} *object j*.
	$(\mathbf{x}_i, \{closer/farther\}, \mathbf{x}_j)$	Move *object i* {closer to / farther from} *object j*.
	$(\mathbf{x}_i, \{left/front/right/back\})$	Move *object i* {to its left / forward / $\cdots$ }.

Table 6: All variation patterns and corresponding rule-based description examples for object-level modification operations. In all patterns, $\mathbf{x}_i$ denotes the object that is modified and selected for description.

the angle between the directions of the two objects. If the angle between the objects' directions θ_i and θ_j is less than a threshold (5 degrees in our case), the orientation relationship is classified as parallel. The classifications for orthogonal and antiparallel follow a similar criterion. All other cases are classified as none.

The abstract relative relationship p_{ij} includes other abstract relationships between objects in 3D space, such as symmetric and facing. The symmetric relationship is defined when two objects are parallel, and the angle θ_{ij} is perpendicular to both θ_i and θ_j, where $\theta_{ij} = \arctan(Y_j - Y_i, X_j - X_i)$. This relationship often occurs in scenarios like cabinets placed parallel against a wall or chairs neatly arranged by the side of a table. A facing relationship is defined when θ_{ij} is equal to θ_i, common in situations where a sofa faces a TV stand or chairs are arranged around a table.

All the above categorical relative relationships are generated in the semantic graph from the first stage diffusion model.

8.2. Scene Prior Definition

In scene prior guidance, we use the generalized Gaussian mixture models (GMMs) to model the prior distribution of scene arrangement. Following [YZY*21], in $M_{\mu_{c_i}}$ and $M_{\mu_{(c_i,c_j)}}$, we mainly model the translation information of the arrangement and ignore the size and orientation. When modeling $M_{\mu_{c_i}}$, instead of modeling the absolute translation of objects in [YZY*21], we consider the distance to the nearest wall of each object, which is a more informative and general feature in indoor scene arrangement. For the relative position distribution, we model the prior for each dimension of the relative position between two objects separately. A 6-component GMM is utilized to model $M_{\mu_{c_i}}$ for each class c_i and an 8-component GMM is utilized to model $M_{\mu_{(c_i,c_j)}}$ for each class pair (c_i, c_j).

In the calculation of $\mathcal{L}_{overlap}$, we employ an indicator $\mathbf{I}_{c_i,c_j}$ to determine whether overlapping between two categories of objects is permissible and exclude these pairs from the cost function calculation. $\mathbf{I}$ is statistically derived from the dataset. Specifically, if the overlap probability between categories c_i and c_j in a scene exceeds a threshold (0.3 in our implementation), $\mathbf{I}$ is set to 0.

9. Experiment

9.1. Implementation Details

In the denoising network $\epsilon_\theta(\mathbf{G}_t, t, \mathbf{c})$, the noises on each type of attribute in the node representations x_i and edge representations e_{ij} are predicted with a separate decoder after the final layer of graph neural network. Also, the loss for noise prediction on each attribute is calculated separately. When computing the training loss (Eq. 5), we balance the loss calculation for edge attributes by multiplying it by a fixed weight 10.

For the diffusion setting, we use a linear noise schedule following [HJA20], and the number of diffusion steps for the first-stage model is set to $T = 100$, and that for the second-stage model is set to $T = 50$. For our inference-time scene prior guidance, we set the scale $s = 20.0$ and balancing parameters $\lambda_1 = 1.0$ and $\lambda_2 = 0.1$. The guidance is enabled for the last 30% of the denoising steps, where threshold $m = 15$. The input text is encoded using a frozen CLIP-ViT-B/32 model. We employ the AdamW optimization algorithm for training with a learning rate of 0.0002. Scene generation models are trained on a single GPU with a batch size of 128 for 10,000 epochs, and scene modification tasks are trained on eight GPUs with a batch size of 128 for 3,000 epochs.

9.2. Applications

Scene Correction Our interactive scene synthesis design allows users to correct errors using text instructions, fundamentally addressing the problem of invalid scenes in previous scene synthesis methods. Figure 10 shows examples of erroneous scenes generated from text instructions. Our method can easily correct these errors through text guidance.

10. Limitations

Our proposed system has several limitations. Although we support object-level and scene-level modification, our dataset does not account for group-level modification. In indoor design, group-level

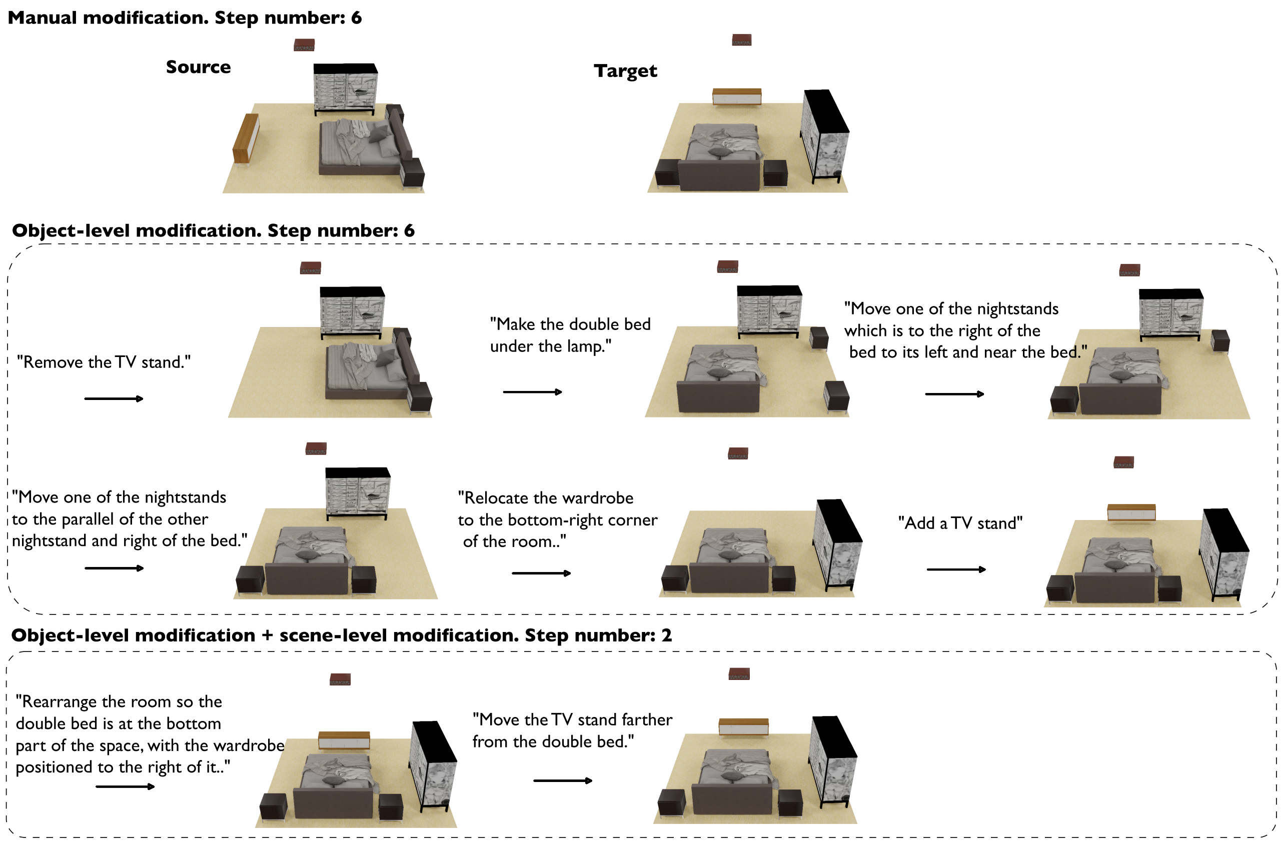

Manual modification. Step number: 6
Source
Target
Object-level modification. Step number: 6
"Remove the TV stand."
"Make the double bed under the lamp."
"Move one of the nightstands which is to the right of the bed to its left and near the bed."
"Move one of the nightstands to the parallel of the other nightstand and right of the bed."
"Relocate the wardrobe to the bottom-right corner of the room.."
"Add a TV stand"
Object-level modification + scene-level modification. Step number: 2
"Rearrange the room so the double bed is at the bottom part of the space, with the wardrobe positioned to the right of it.."
"Move the TV stand farther from the double bed."

Figure 9: Interactive generation example.

modification is helpful for interactive synthesis. For example, tables and chairs are often manipulated together in real-world scenarios. Generating a dataset for group-level modification is challenging and may require significant manual effort, so we have left this interactive operation for future work. Additionally, since the data scales for scene generation and scene modification do not match, we trained separate models for text-guided generation and text-guided modification. A more general model would integrate these two tasks.

Moreover, our work is still limited to data-driven scene synthesis with fixed room types and object categories. While some current LLM-based methods achieve open-vocabulary synthesis, they struggle with precise and detailed interaction. We believe that open-vocabulary interactive scene synthesis represents a promising direction for future research.

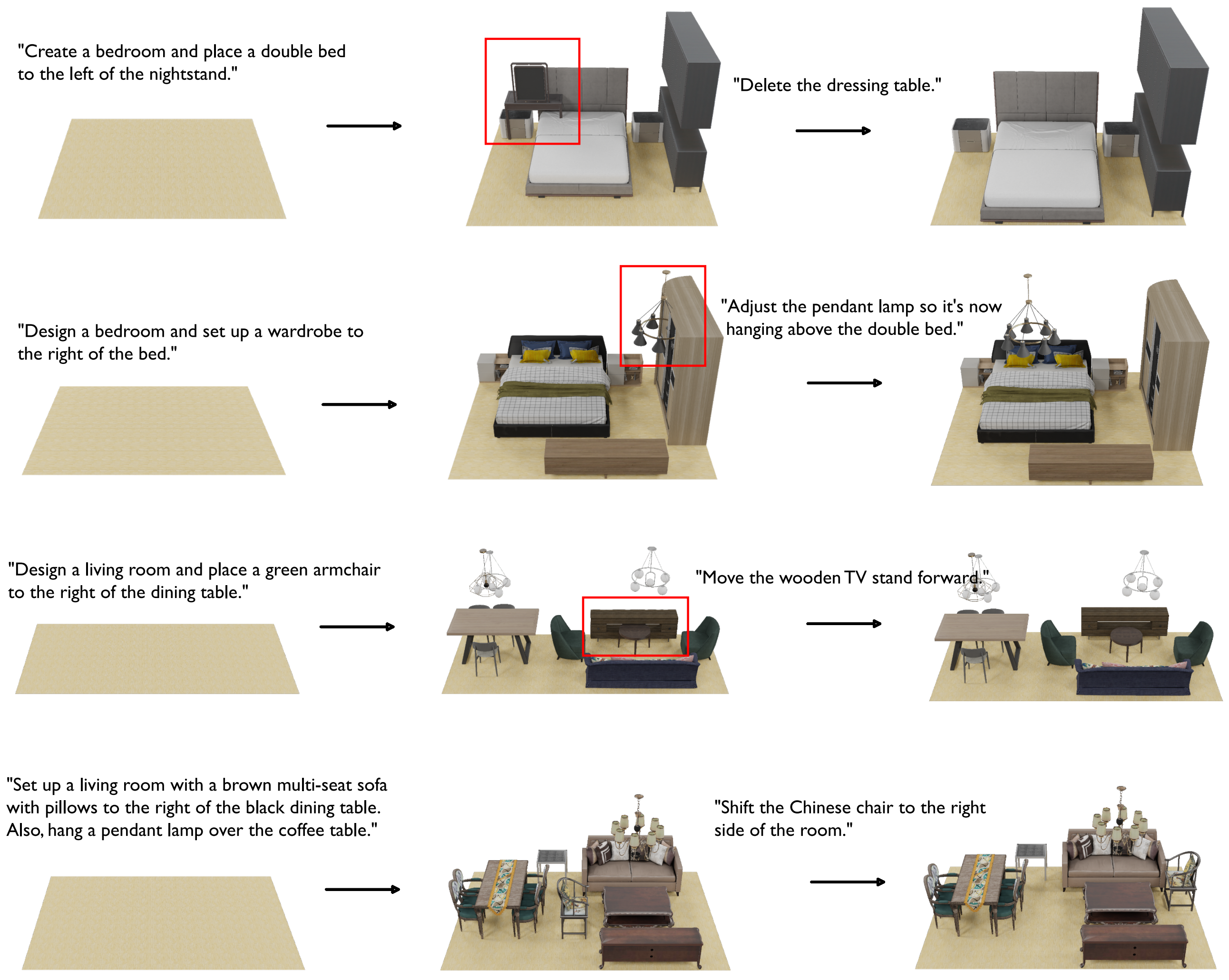

"Create a bedroom and place a double bed to the left of the nightstand."
"Delete the dressing table."
"Design a bedroom and set up a wardrobe to the right of the bed."
"Adjust the pendant lamp so it's now hanging above the double bed."
"Design a living room and place a green armchair to the right of the dining table."
"Move the wooden TV stand forward."
"Set up a living room with a brown multi-seat sofa with pillows to the right of the black dining table. Also, hang a pendant lamp over the coffee table."
"Shift the Chinese chair to the right side of the room."

Figure 10: Scene correction application. Our interactive scene synthesis design can easily fix the errors in scene generation through text guidance.