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: $p_{\psi_1}(\hat{\mathbf{x}}_k|\mathbf{X_{1:k-1}}, c_k, \mathbf{f}_k)$ 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 $\mathbf{e}_{ij} = [r_{ij}, o_{ij}, p_{ij}, \mathbf{d}_{ij}]$. 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

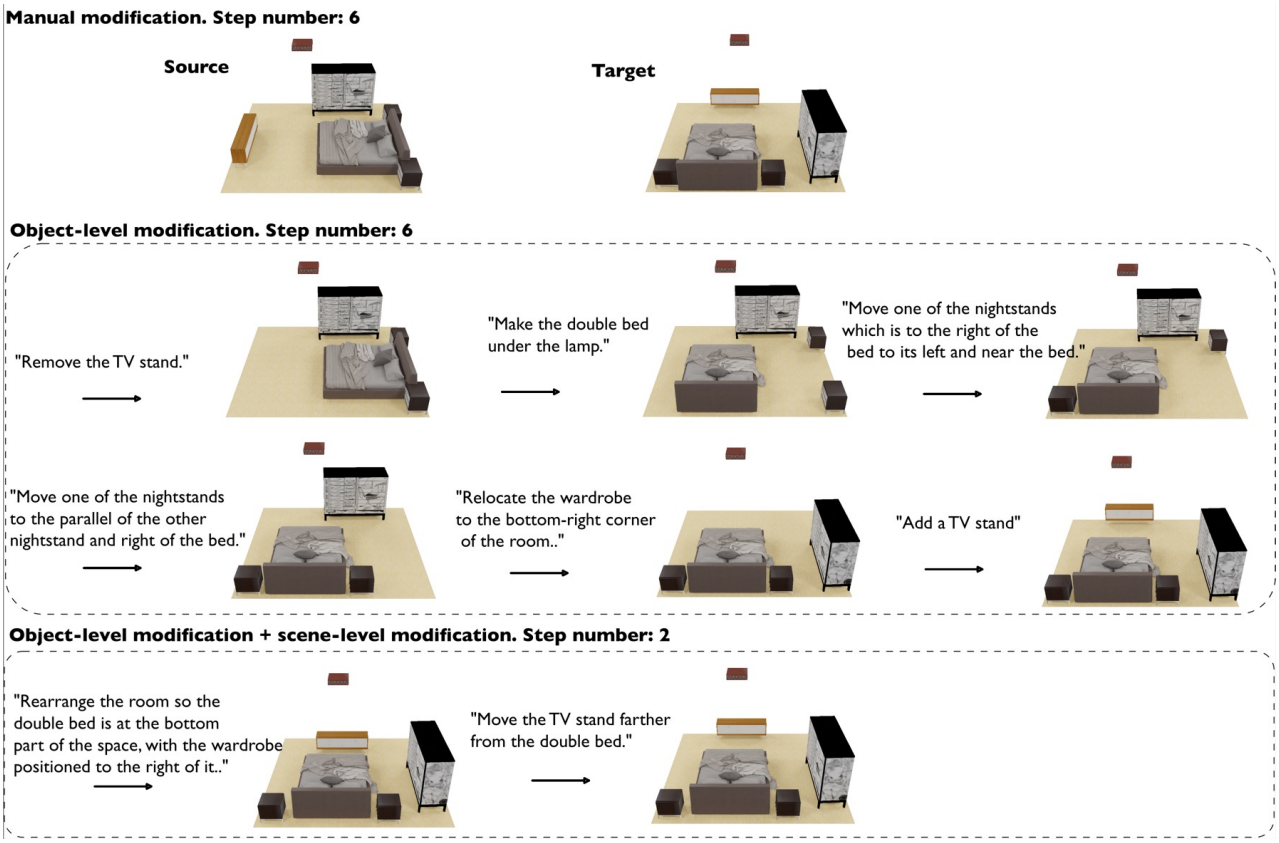

Figure 9: Interactive generation example.

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.

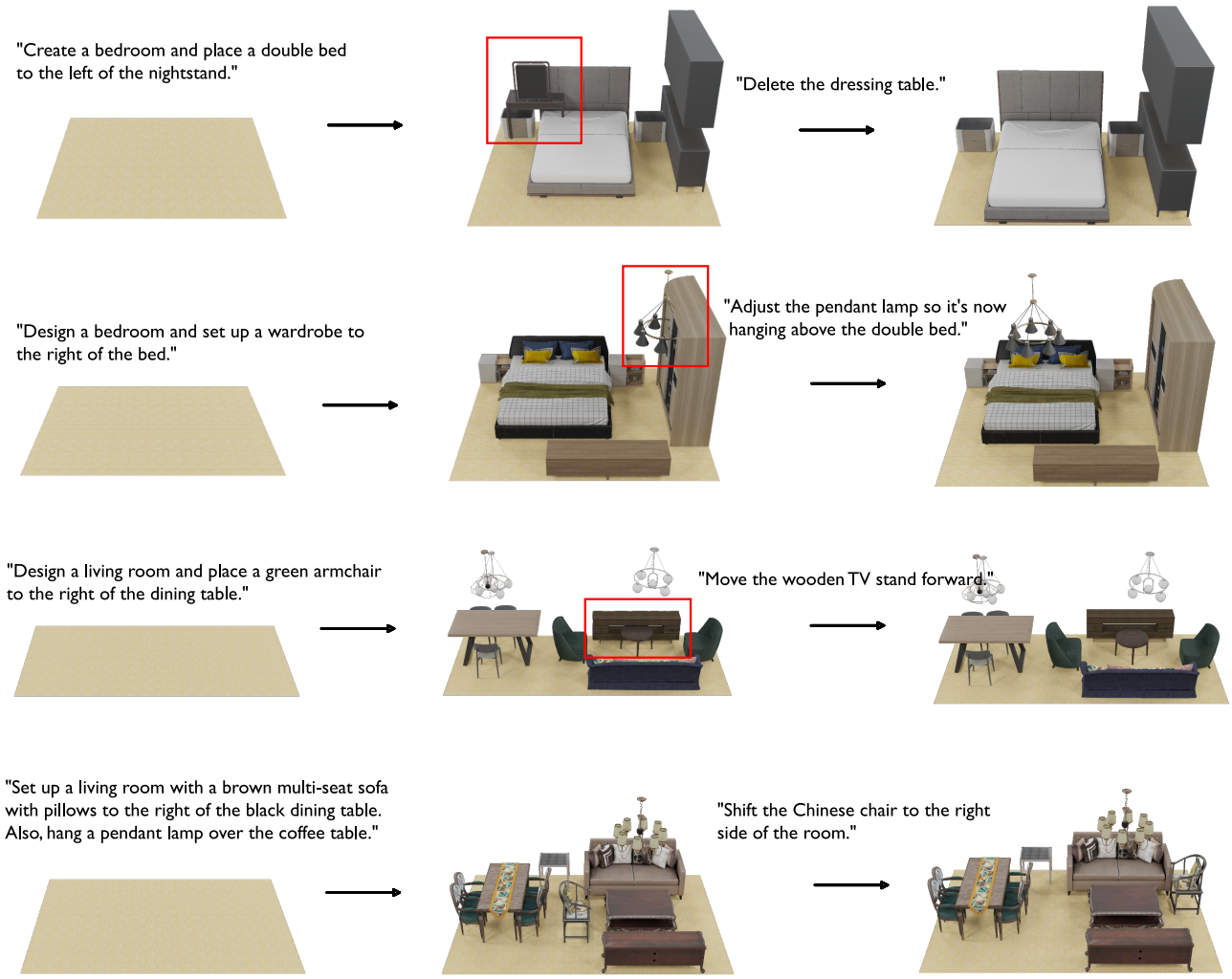

Figure 10: Scene correction application. Our interactive scene synthesis design can easily fix the errors in scene generation through text guidance.