

FoR-SALE: Frame of Reference-guided Spatial Adjustment in LLM-based Diffusion Editing

Tanawan Premisri & Parisa Kordjamshidi

Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA
{premsrit, kordjams}@msu.edu

Abstract

Current text-to-image generation models, even state-of-the-art models, exhibit a significant performance gap when spatial expressions are described from non-camera perspectives. To address this limitation, we propose **Frame of Reference-guided Spatial Adjustment in LLM-based Diffusion Editing (FoR-SALE)**, an extension of the Self-correcting LLM-controlled Diffusion (SLD). FoR-SALE first evaluates the alignment between a given text and an initially generated image, and then refines the image based on the expressed *FoR* in the spatial description. It employs vision modules to extract the spatial configuration of the generated image and simultaneously maps the spatial expression to a corresponding camera perspective. This unified perspective enables direct evaluation of alignment between language and vision. When misalignment is detected, the required editing operations are generated and applied. FoR-SALE introduces novel latent-space operations to adjust the facing direction and depth of generated images. We evaluate FoR-SALE on three benchmarks designed to assess spatial understanding with FoR: FoR-LMD, FoREST, and FoREST-G, with the latter providing diagnostic subsets covering multi-object scenes, expanded relation types, and naturalized prompts to test the generality of our framework. Our framework improves the performance of SOTA T2I models by up to 7.0 pp using only a single round of correction.

1 Introduction

Spatial understanding refers to the ability to comprehend the locations of objects in space and is fundamental to human cognition and everyday tasks. One key component of spatial understanding is the Frame of Reference (FoR), which defines the perspective from which spatial relations are interpreted. While extensively studied in cognitive linguistics (Mou & McNamara, 2002; Levinson, 2003; Tenbrink, 2011; Coventry et al., 2018), FoRs have received limited attention in AI models, particularly within Multimodal Large Language Models (MLLMs) (Liu et al., 2023; Chen et al., 2024). Recent studies highlight shortcomings in FoR reasoning by MLLMs across multiple tasks, e.g., text-based Question Answering (Premisri & Kordjamshidi, 2025), Visual Question Answering (Zhang et al., 2025b), and especially Text-to-Image (T2I) generation (Wang et al., 2025b). Wang et al. (2025b) and Premisri & Kordjamshidi (2025) show that diffusion models exhibit substantially lower spatial alignment when spatial descriptions are expressed from non-camera perspectives. As illustrated in Figure 1, even SOTA T2I



Figure 1: Examples of images generated by T2I models and the corresponding outputs after one round of correction with FoR-SALE.

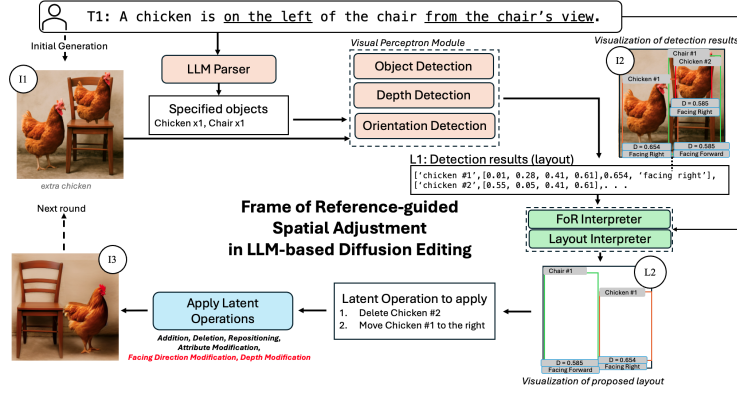


Figure 2: Overview of the FoR-SALE pipeline. It first extracts key objects from the input text using an LLM parser and layout information from the initial image using a Visual Perception Module. The extracted information is passed to the FoR Interpreter and Layout Interpreter to generate a revised layout. A sequence of latent operations is derived by comparing the initial and revised layouts and applied to synthesize an updated image. The resulting image can undergo additional refinement rounds if needed.

models, such as GPT-4o (OpenAI, 2025a) and FLUX.1 (Black Forest Labs, 2024a), struggle to accurately generate images that reflect spatial relations described from non-camera perspectives. To address this issue, we propose the **Frame of Reference-guided Spatial Adjustment in LLM-based Diffusion Editing (FoR-SALE)** framework. Our approach builds upon the Self-correcting LLM-controlled Diffusion (SLD) pipeline (Wu et al., 2024), which uses LLMs as a core reasoning component to validate prompts and generate suggested layouts for editing images via latent-space operations in the diffusion model. However, the original SLD framework does not account for FoR in spatial language, limiting its ability to handle spatial prompts grounded in non-camera perspectives. FoR-SALE extends this paradigm by explicitly modeling FoR and enabling spatial adjustments across diverse perspectives.

Figure 2 illustrates the FoR-SALE pipeline. It begins with standard T2I generation, where an input text (T_1) is passed to a T2I model to produce an image (I_1). Meanwhile, the LLM parser extracts key objects from the input text, which are then passed to the Visual Perception Module to obtain three visual properties: object location, orientation, and depth. These extracted visual properties (I_2) are converted into layout information (L_1). The input expression (T_1) and layout information (L_1) are then fed into the FoR Interpreter, which identifies the FoR and converts the expression into the camera perspective, a unified viewpoint. Subsequently, the Layout Interpreter is employed to generate a suggested layout (L_2) in textual form aligning with the updated expression. The suggested layout is then compared with the visual detection outputs (L_1) to identify mismatches, which are used to formulate self-correction operations, such as adjusting an object’s facing direction or depth. These corrections are applied in the latent space during image synthesis using the Stable Diffusion model. Notably, these operations are generic and can be applied to other diffusion models. Finally, a new image is generated from the modified latent representation, ensuring consistency with the spatial configuration described in the input, particularly with respect to the specified FoR. The resulting image (I_3) can undergo additional refinement rounds if needed.

We evaluate the effectiveness of FoR-SALE on three benchmarks: FoR-LMD, a modification of LMD (Lian et al., 2024) that incorporates perspective cues; FoREST (Premisri & Kordjamshidi, 2025), a benchmark with textual inputs covering diverse FoR cases; and FoREST-G, a diagnostic extension of FoREST that evaluates generalization to multi-object scenes, expanded relation types, and naturalized prompts. FoR-SALE improves images generated by SOTA T2I models, including FLUX.1, Nano-Banana-Pro, and GPT-4o, by up to 7.0 pp in a single correction round and 13.1 pp in three rounds. Results on FoREST-G further demonstrate that FoR-SALE remains effective beyond the original two-object setting, particularly

for intrinsic-FoR cases. Finally, we provide a thorough analysis of the limitations of both T2I models and LLMs in reasoning over layouts from different perspectives. Our contributions¹ can be summarized as follows: (1) We propose the first self-correcting framework that explicitly incorporates frame of reference (FoR) into text-to-image generation. (2) We introduce novel editing operations within a self-correcting framework to handle FoR-aware image generation. (3) We augment an existing benchmark to enable a more comprehensive evaluation of FoR understanding in text-to-image models. Our framework consistently improves FoR-aware spatial alignment, particularly under intrinsic FoR settings.

2 Related Work

Frame of Reference in MLLMs. Multiple benchmarks have been developed to evaluate the spatial understanding of MLLMs across various tasks (Anderson et al., 2018; Mirzaee et al., 2021; Mirzaee & Kordjamshidi, 2022; Shi et al., 2022; Cho et al., 2023). However, most of these benchmarks overlook FoR. Only a few recent benchmarks explicitly address FoR-related reasoning (Liu et al., 2023; Chen et al., 2024; Zhang et al., 2025a; Wang et al., 2025a). For example, Liu et al. (2023) show that training a vision-language model with text that includes FoR information can improve visual question answering (VQA). Wang et al. (2025a) introduce a comprehensive benchmark for spatial VQA that includes FoR examples, although FoR is not its primary focus of evaluation. Three recent studies focus more directly on evaluating FoR understanding in MLLMs. First, Zhang et al. (2025b) assess FoR handling in VQA settings and reveal substantial limitations, especially when reasoning goes beyond the default camera-centric view. Second, Premisri & Kordjamshidi (2025) investigate FoR reasoning in natural language prompts—both ambiguous and unambiguous—and find persistent failures in both question answering and layout generation when the perspective diverges from the camera view. Third, Wang et al. (2025b) evaluate T2I models and find that even SOTA models fail to preserve correct spatial relations when the context is not grounded in the camera perspective and includes 3D information such as orientation and distance. In this work, we extend this line of research by providing a new evaluation of T2I models based on their alignment with FoR-grounded spatial expressions. We also enhance LLM and T2I models’ ability to comprehend varying FoR conditions.

Spatial Alignment in T2I. Several studies have sought to improve the spatial alignment of T2I models with user input. Early approaches introduced predefined spatial constraints—such as depth maps (Zhang et al., 2023; Mo et al., 2024), object layouts (Li et al., 2023), or attention maps (Wang et al., 2024; Pang et al., 2024)—to guide image generation. However, these often require manual configuration or model retraining to interpret the constraints. With advances in LLMs’ spatial reasoning, recent work has leveraged them to automatically generate spatial guidance. For example, Cho et al. (2023) use an LLM to generate layouts that guide diffusion models without additional training. More recent methods employ MLLMs to control 3D spatial arrangements by generating feedback for reinforcement training of diffusion models (Liu et al., 2025), training a T2I model using compositional questions derived from the input (Sun et al., 2025), or producing action plans for sequential editing (Wu et al., 2024; Goswami et al., 2024). While these methods are promising, they do not address spatial FoR. In contrast, we explicitly address this by extending the SLD framework (Wu et al., 2024) to support editing under diverse FoRs.

3 Methodology

FoR-SALE follows the SLD framework (Wu et al., 2024), which consists of two main components: (1) LLM-driven visual perception and (2) LLM-controlled Diffusion. We further adapt these two components to enable finer-grained perception and layout interpretation for FoR recognition and image correction.

¹Code and data are available at <https://github.com/HLR/FoR-SALE>.

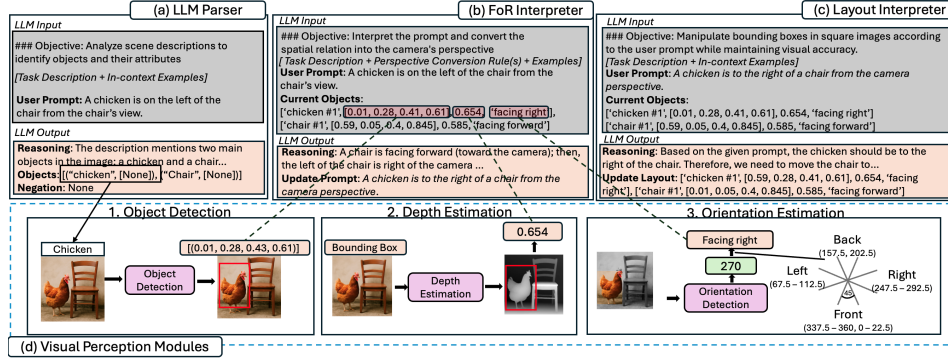


Figure 3: Example inputs and outputs from the LLM Parser, FoR Interpreter, Layout Interpreter, and Visual Perception Module. The LLM Parser output guides the Visual Perception Module in extracting object-specific information, including bounding boxes, orientation, and depth. This information is then passed to the FoR Interpreter, which converts the spatial expression to the camera perspective. The Layout Interpreter subsequently generates a suggested spatial layout based on the converted expression.

3.1 LLM-driven Visual Perception Module

The process begins with standard T2I generation, in which a textual input is used to create an image. FoR-SALE then proceeds to extract key information from the spatial expression via an LLM Parser and from the generated image via a visual perception module.

1) LLM Parser. In this first step, we prompt an LLM with instructions and in-context examples to extract a list of key object mentions and their attributes from the input text, denoted as L . For example, given the spatial expression *A red chicken is on the left of a chair from the chair's view*, the output of the LLM is $L = [(\text{"chicken"}, [\text{"red"}]), (\text{"chair"}, [\text{None}])]$ where "red" is the attribute associated with the chicken, and "None" indicates that no specific attribute is mentioned for the chair. All prompt specifications are provided in Appendix K.

2) Visual Perception Module. The obtained list L is fed into the visual perception module with open-vocabulary object detection. In our FoR-SALE, we add new visual perception components to deal with FoR. These include depth estimation and orientation detection. Figure 3 (d) illustrates this module. The open-vocabulary object detector receives the information in L in the following format "image of a/an [attribute] [object name]" and outputs bounding boxes, denoted as B . The outputs are represented in the following list format, $((\text{attribute}) (\text{object name}) (\text{\#object ID}), [x, y, w, h])$ where (x, y) indicates the coordinates of the upper-left corner of the bounding box from 0.0 to 1.0, w is its width, and h is its height. The object ID is a serial number assigned uniquely to each detected object. Next, the depth estimation model is used to predict the depth map, denoted as D . To extract object-specific depth values (D_i), a segmentation mask is obtained using the bounding boxes from B , and the average pixel depth within each region is computed using the following equation: $D_i = \frac{1}{|R_i|} \sum_{j \in R_i} d_j$, where i is the index of the object, R_i is the mask region of the object, and d_j is the depth at pixel j . The value of D_i ranges from 0 to 1. Finally, an orientation detection model is applied to the object segmentation to obtain the object's orientation angle. This angle is then converted into a facing direction, denoted as f_i . There are eight facing-direction categories: $\text{orientation} = \{\text{ForwardLeft}, \text{Left}, \text{BackwardLeft}, \text{Back}, \text{BackwardRight}, \text{Right}, \text{ForwardRight}, \text{Front}\}$. Each category spans a 45-degree range, starting from 22.5° to 67.5° for ForwardLeft, and continuing in 45° intervals for the remaining orientation labels. We collect this visual information for each object and obtain a new information list in a new format, denoted $V_L = \{((\text{attribute}) (\text{object name}) (\text{\#object ID}), [x, y, w, h], D_i, f_i)\}$. An example of representation is shown in Figure 3.

3.2 LLM-Controlled Diffusion

After obtaining visual information (V_L), two additional modules are employed to analyze and modify the image: LLM Interpreters and Image Correction.

1) LLM Interpreters. This module analyzes V_L together with the input text T and proposes a revised layout, denoted as $\tilde{V}_L$ in the same format. The original SLD framework employs an LLM for layout interpretation. However, in FoR-SALE, we incorporate an additional LLM, namely the FoR Interpreter. Figure 3 (b) and (c) illustrate these two LLMs.

- **FoR Interpreter.** Based on findings that MLLMs demonstrate stronger performance when reasoning about spatial expressions described from a camera perspective (Zhang et al., 2025b; Premisri & Kordjamshidi, 2025; Wang et al., 2025b), we hypothesize that converting spatial expressions to a camera perspective can enhance FoR reasoning. The FoR Interpreter takes the spatial text T and visual information from the generated image V_L as input, and outputs a rewritten spatial expression from the camera perspective, denoted as T' . If no spatial relation is present, the model returns the unchanged input. We provide in-context information to guide the FoR Interpreter in performing this perspective conversion. In particular, we include spatial perspective conversion rules. A total of 32 rules are manually defined, one for each combination of the eight facing directions considered in the Visual Perception Module and four spatial relations (front, back, left, right). For example, *if the object is facing left, the left side of the object corresponds to the front direction from the camera perspective*. These rules cover directional spatial relations that are most strongly affected by FoR interpretation, as well as the eight possible facing directions. This set of spatial relations is based on qualitative directional relations, which are a closed set, making the formalization feasible (Kordjamshidi et al., 2010). An example of the input and output is shown in Figure 3(b), and all rules are included in Appendix I. To examine whether these rules can be automatically generated, we further evaluate their extensibility through an additional experiment that uses an LLM to generate rules automatically. The results and discussion are provided in RQ3 of Section 4.5.

- **Layout Interpreter.** After obtaining the spatial expression, T' , that follows the camera perspective, the second LLM uses T' and V_L as input to analyze the layout. The Layout Interpreter LLM is prompted with manually crafted in-context examples to analyze whether the current layout aligns with the provided T' . If misalignment is detected, the LLM is instructed to propose a revised layout $\tilde{V}_L$ that satisfies the spatial description. An example of the input and output is shown in Figure 3(c).

2) Image Correction. In this step, we compare the current layout V_L with the proposed layout $\tilde{V}_L$ using an exact-matching process to detect misalignment. If there is any misalignment between the two layouts, we create a sequence of editing operations to modify the image and align it with $\tilde{V}_L$. The original SLD framework includes four editing operations: Addition, Deletion, Reposition, and Attribute Modification. Our framework extends this set by introducing **two new operations for handling FoR**: Facing Direction Modification and Depth Modification. Before applying any operation, backward diffusion (Ho et al., 2020) is performed on the initial image to obtain its latent representation, which serves as the basis for all subsequent editing actions. After all editing actions are applied, Stable Diffusion is called to synthesize the final image.

- **Addition.** The operation first generates the object within the designated bounding box using the base Stable Diffusion model, after which its segment is extracted using SAM (Kirillov et al., 2023). Next, a backward diffusion process is applied to the generated object segment to obtain an object-specific latent representation. This representation is then merged into the original image’s latent space to complete the composition.

- **Deletion.** The process first segments the object using SAM within its bounding box. The latent representation of the segmented region is then removed and replaced with Gaussian noise, enabling reconstruction of the object region during the final diffusion step.

- **Reposition.** To preserve the object’s aspect ratio, this step begins by shifting and resizing the object from its original bounding box to the new bounding box. After repositioning,

SAM is used to segment the object. Then, a backward diffusion process is used to obtain the latent representation. This new representation is then integrated into the image latent space at the updated location. To remove the original object, we replace the corresponding latent region segmented via SAM with Gaussian noise prior to the final diffusion step.

- **Attribute Modification.** The process begins by employing SAM to segment the object region within its bounding box. An attribute modification diffusion model, e.g., DiffEdit (Couairon et al., 2023), is then called with a new prompt to modify the object’s attribute within the defined region. After the attribute is edited, a backward diffusion process is performed to extract the corresponding latent representation. This updated latent is then integrated into the image latent space to complete the modification.
- **Facing Direction Modification.** This new operation begins by using SAM to segment the object’s region. Then, DiffEdit is invoked with a prompt specifying the desired facing direction to generate an image of the object in that orientation. Next, the base diffusion model is used to perform a backward diffusion process for obtaining the latent representation. Finally, this latent is integrated into the image latent space to complete the modification.
- **Depth Modification.** This new operation begins by synthesizing the new depth of the given object using the equation, $d_{j'} = \min(1, \max(0, d_j - D_i + D_{i'}))$, where $d_j, d_{j'}$ denote the original and updated depth of pixel j , respectively. D_i represents the current average depth of object i defined in Section 3.1, and $D_{i'}$ is the new target depth proposed by the Layout Interpreter. Next, we shift and resize the synthesized depth map of this object to the target bounding box. A diffusion model is then called with ControlNet (Zhang et al., 2023) to generate an object with the specified depth. After generating a new object, the segmentation and backward diffusion are performed to obtain the latent representation. Finally, this latent representation is integrated into the image latent space to complete the modification.

4 Experiments

4.1 Datasets

FoR-LMD is a synthetic dataset designed to assess several skills, including spatial understanding. We extend the LMD benchmark by adding perspective cues to the input template to incorporate FoR information. The modified prompt template is: $(obj_1) (R_1) (ref_1) \text{ and } (obj_2) (R_2) (ref_2)$, where ref_1 and ref_2 specify the reference perspective, camera view (relative) or object-centric view (intrinsic). To focus on perspective-sensitive relations, we restrict R_1, R_2 to {left, right, front, back}. This results in 500 samples with explicit perspective.

FoREST (Premshri & Kordjamshidi, 2025) is a synthetic benchmark designed to evaluate FoR understanding with explicit FoR annotations (C-Split). We sample 500 spatial expressions to match the size of FoR-LMD. Each prompt explicitly specifies the spatial perspective and the facing direction of the reference object, which are not provided in FoR-LMD.

FoREST-G extends FoREST with three subsets: Complex, Relation-Augmented, and Language-Variation. These subsets are designed to evaluate generalization beyond the original two-object setting and fixed prompt templates. The Complex subset increases the number of objects and the complexity of spatial relations. The Relation-Augmented subset adds vertical and diagonal relations, while the Language-Variation subset uses an LLM to rewrite the original spatial descriptions while preserving the target spatial constraints. Further details on the construction of these subsets are provided in Appendix B.

4.2 Evaluation Method

We adopt the evaluation scheme proposed by Wang et al. (2025b), which has been shown to align with human judgment. To evaluate generated images, we use the Visual Perception module (Section 3.1) to extract bounding boxes, depth, and object orientations. We first verify that the number of detected objects matches the textual specification; in the evaluated benchmarks, exactly one instance of each object must be present. If this condition

is not satisfied, the image is marked as incorrect. We then compare detected orientations with annotated labels; any misalignment results in the image being marked as incorrect. Spatial relations are evaluated using the provided FoR annotations. When the FoR is non-camera-centric (intrinsic), spatial relations are converted to the camera perspective using the detected orientation of the reference object. Finally, spatial correctness is assessed using predefined geometric specifications of spatial relations under the camera-perspective assumption (Huang et al., 2023; Cho et al., 2023; Wang et al., 2025b). To validate our automatic evaluation, we conducted a human study on 200 randomly sampled images. The resulting Cohen’s kappa score of 0.791 indicates strong agreement between human judgments and the automatic evaluation. Details of the study are provided in Appendix J.

4.3 Baseline Models

For baseline comparison, we select seven T2I models: Stable Diffusion (SD) 1.5 (Rombach et al., 2022), SD 2.1 (Rombach et al., 2022), SD 3.5-Large (Stability AI, 2024), GLIGEN (Li et al., 2023), FLUX.1 (Black Forest Labs, 2024a), Nano-Banana-Pro (Google DeepMind, 2025), and GPT-4o (gpt-image-1) (OpenAI, 2025b). Given our focus on recent models, results for older baselines, including SD 1.5, SD 2.1, SD 3.5-Large, and GLIGEN, are presented in the Appendix. We also include a prompt-engineering approach for extracting a cognitive map (Yang et al., 2025) as an additional baseline with GPT-4o. For comparison with LLM-based editing frameworks, we include SLD and GraPE (Goswami et al., 2024), both self-correcting pipelines that achieve strong performance using GPT-4o as the LLM interpreter. SLD serves as the baseline framework for FoR-SALE, whereas GraPE uses an MLLM to determine editing actions and invokes SOTA image-editing models for modification. We also compare FoR-SALE with a strong image-editing model, Qwen-Image-Edit (Wu et al., 2025). All experiments were conducted on two A6000 GPUs, totaling approximately 450 GPU-hours. Further implementation details are provided in Appendix D.

4.4 FoR-SALE Implementation Details

We select Qwen3-32B-Reasoning (Qwen Team, 2025) for all LLM components in the FoR-SALE pipeline. For the Visual Perception module, we employ OWLv2 (Minderer et al., 2023) for object detection, DPT (Ranftl et al., 2021) for depth estimation, and OrientAnything (Wang et al., 2025c) for orientation detection. We use SD 1.5 as the base diffusion model to create objects and as the final step to denoise the composed latent space. The same visual processing tools are used in evaluation. To verify that our improvements are not tied to the performance of specific visual processing tools, we additionally report experimental results using alternative ones. The results demonstrate consistent improvements from the FoR-SALE framework, regardless of the choice of vision tools. This outcome is expected, as the visual tools are limited to basic tasks (e.g., object detection), while the LLM carries out the reasoning. We report results using the alternative tools in Appendix F. A detailed execution-time analysis of each component is presented in Appendix C. Our results show that FoR-SALE introduces only a **runtime overhead of approximately 45%** relative to the SLD baseline. This is a reasonable trade-off for enabling spatial reasoning from various perspectives. More implementation details are provided in Appendix A.

4.5 Results and Discussion

RQ1. Can the SOTA T2I models follow the FoR expressed in the text?

As shown in Table 1, the best-performing model, GPT-4o, achieves only 52.20% accuracy, **highlighting the difficulty of T2I generation, even with only two objects in a spatial relation**. While GPT-4o performs well on *relative* FoR in FoR-LMD (94.76%), its accuracy drops sharply to 24.35% on *intrinsic* FoR, revealing a substantial performance gap. A similar trend is observed in a more recent model (Nano-Banana-Pro), which also shows a dramatic drop in intrinsic FoR cases. This finding is consistent with prior work (Premisri & Kordjamshidi, 2025; Wang et al., 2025b), which highlights the challenges of reasoning about FoR in non-camera perspectives. Despite their low performance on FoR-LMD, GPT-4o and Nano-Banana-Pro achieve reasonable results on the FoREST benchmark, which explicitly

Method	FoR-LMD			FoREST			Overall $\uparrow$
	Rel. $\uparrow$	Int. $\uparrow$	Avg. $\uparrow$	Rel. $\uparrow$	Int. $\uparrow$	Avg. $\uparrow$	
FLUX.1	58.95	25.83	41.00	18.18	15.56	17.00	29.00
+ Qwen-Image-Edit	68.12	26.57	45.60	33.45	24.44	29.40	37.50
+ Qwen-Image-Edit (2 $\times$)	68.99	26.57	46.00	34.90	24.00	30.00	38.00
+ 1-round GraPE	54.15	18.08	34.60	17.45	11.56	14.80	24.70
+ 1-round SLD	63.32	25.09	42.60	24.72	12.00	19.00	30.80
+ 1-round FoR-SALE	65.07	27.67	44.80	25.09	22.22	23.80	34.30
+ 2-round FoR-SALE	67.68	28.04	46.20	30.18	29.78	30.00	38.10
+ 3-round FoR-SALE	69.43	25.84	45.80	32.72	31.11	32.00	38.90
Nano-Banana-Pro	73.36	7.38	37.60	52.00	27.55	41.00	39.30
+ Qwen-Image-Edit	75.10	9.59	39.60	49.09	26.67	39.00	39.30
+ 1-round GraPE	73.36	6.64	37.20	48.72	25.33	38.20	37.70
+ 1-round SLD	81.22	13.28	44.40	53.81	24.44	40.60	42.50
+ 1-round FoR-SALE	84.72	14.02	46.40	60.72	28.44	46.20	46.30
+ 2-round FoR-SALE	89.52	21.77	52.80	62.18	29.33	47.40	50.10
+ 3-round FoR-SALE	91.70	26.58	56.40	61.45	32.44	48.40	52.40
GPT-4o (gpt-image-1)	94.76	24.35	56.60	57.81	35.56	47.80	52.20
+ Qwen-Image-Edit	94.32	22.88	55.60	61.45	32.00	48.20	51.90
+ 1-round GraPE	93.89	19.56	53.60	55.64	30.22	44.20	48.90
+ 1-round SLD	89.08	21.40	52.40	43.27	23.56	34.40	43.40
+ 1-round FoR-SALE	93.01	35.42	61.80	54.18	37.33	46.60	54.20
+ 2-round FoR-SALE	93.01	34.32	61.20	48.73	39.11	44.40	52.80
+ 3-round FoR-SALE	91.26	38.37	62.60	53.81	42.22	48.60	55.60

Table 1: Accuracy of generated images across various baselines. Relative (Rel.) denotes camera-perspective spatial expressions, while intrinsic (Int.) denotes spatial expressions from another object’s perspective.

encodes facing direction in the input. This is likely due to their 3D spatial training. In contrast, diffusion models such as FLUX.1 fail to effectively leverage 3D spatial information (e.g., orientation) and struggle under both relative and intrinsic FoR settings in FoREST. Other diffusion models exhibit behavior similar to that of FLUX.1. Detailed results for these additional baselines are presented in Appendix E.1. In addition to the above baselines, we apply a spatial prompting strategy to extract a cognitive map (Yang et al., 2025) using GPT-4o. We find that the cognitive map improves GPT-4o’s object-counting accuracy but provides only limited benefits for FoR spatial reasoning; details are reported in Appendix E.3.

RQ2. How effective is FoR-SALE in editing images to follow the FoR expressed in text?

To answer this question, we compare FoR-SALE with two self-correcting frameworks, SLD and GraPE, and one SOTA image-editing model, Qwen-Image-Edit, in Table 1. FoR-SALE generally outperforms both self-correcting frameworks in both relative and intrinsic FoR settings. Nevertheless, Qwen-Image-Edit performs better in relative cases when the initial image is generated by FLUX.1. We attribute this to its strong spatial reasoning ability, which better processes camera-perspective descriptions and significantly improves weaker base models such as FLUX.1. However, Qwen-Image-Edit struggles to refine images generated by the stronger baselines, GPT-4o and Nano-Banana-Pro, especially under the considerably more challenging intrinsic-FoR setting. In contrast, FoR-SALE achieves consistent improvements in intrinsic FoR settings, up to 11.1 percentage points (pp) after one round and 19.2 pp after three rounds, and continues to improve with additional rounds. On the other hand, applying additional rounds using Qwen-Image-Edit yields little to no further improvements. Figure 4 shows the error types in images edited by SLD and FoR-SALE when using initial images from FLUX.1. FoR-SALE shows clear improvements in left and right relations, which can often be corrected through 2D spatial adjustments. These gains are expected when the Layout Interpreter accurately infers the FoR, highlighting the positive impact of the FoR Interpreter. FoR-SALE also reduces many orientation errors; however, correcting 3D

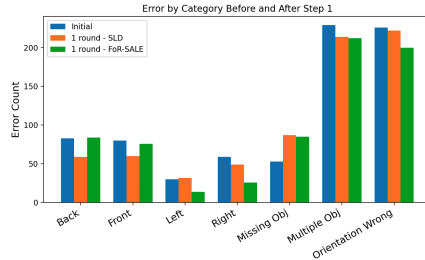


Figure 4: Error analysis of images generated by FLUX.1 (blue), SLD (orange), and FoR-SALE (green).

Layout Interpreter	LLM-Layout Accuracy			Image Accuracy		
	Rel. ↑	Int. ↑	Avg. ↑	Rel. ↑	Int. ↑	Avg. ↑
o3	99.40	79.03	89.30	69.24	30.64	50.10
o4-mini	99.20	64.52	82.00	74.40	29.44	52.10
Qwen3-32B	98.21	45.97	72.30	73.61	21.77	47.90
Qwen3-32B + FoR Interpreter (No Rules)	95.23	54.03	74.80	69.84	24.80	47.50
Qwen3-32B + FoR Interpreter (Partial Rules)	93.25	81.65	87.50	70.63	39.52	55.20
Qwen3-32B + FoR Interpreter (Generated Rules)	94.25	82.05	88.20	70.63	38.40	54.60
Qwen3-32B + FoR Interpreter (Full Rules)	93.85	84.48	89.20	71.82	36.29	54.20
Qwen3-32B + Routing + FoR Interpreter	98.20	85.68	92.00	71.60	39.00	55.70

Table 2: Accuracy of suggested layouts and edited images under different Layout Interpreter settings, using initial images generated by GPT-4o (gpt-image-1). Rel. and Int. denote relative and intrinsic FoR cases, respectively.

aspects such as depth and facing direction remains challenging, with high error rates in these categories. Performance on front-back relations shows limited improvement and sometimes worsens relative to SLD, further underscoring the difficulty of 3D editing. We suspect that SLD’s apparent improvement in front/back errors does not lead to an overall performance increase, as it introduces new errors due to a lack of depth information. To evaluate this hypothesis, we provide a detailed analysis in Appendix G, comparing errors in front and back relations. The analysis reveals that SLD’s front/back errors are reduced due to the generation of extra objects, which are later counted as multiple-object errors. We also observe that multiple-object and missing-object errors remain high for both models, highlighting a limitation in current editing frameworks. Finally, by sampling failure cases and manually categorizing each error, we find that the majority of mistakes arise from incorrect orientation generation, failures in the final diffusion stage to synthesize the target object, and shortcomings of the Visual Perception Modules in detection. Further details are provided in Appendix H.

RQ3. How accurately do LLMs perform layout editing?

To address this question, we conduct an ablation study on the LLMs used for the Layout Interpreter, evaluating two SOTA reasoning models: o3 and o4-mini (OpenAI, 2025c). We also examine three FoR Interpreter settings: (1) No-Rules, where no perspective-conversion rules are provided; (2) Partial-Rules, which include only rules related to facing directions present in the input or detection results; and (3) Full-Rules, which include all manually crafted rules. We report both LLM-generated layout accuracy and final edited-image accuracy using the evaluation protocol in Section 4.2. Table 2 presents the results. LLM-generated layout accuracy is consistently higher than final-image accuracy, indicating that executing layout-guided edits remains challenging. For Qwen3, the FoR Interpreter substantially improves intrinsic-FoR layout accuracy from 45.97% to 84.48% with Full-Rules, while Partial-Rules already achieves 81.65%. This shows that the **predefined FoR Interpreter rules effectively support intrinsic-FoR reasoning, even when the rule set is incomplete**. With the FoR Interpreter, Qwen3 also outperforms o3 on the more challenging intrinsic-FoR cases. However, applying the FoR Interpreter to all prompts slightly reduces Qwen3’s relative-FoR accuracy, since relative-FoR prompts do not require perspective conversion. This motivates a routing mechanism that invokes the FoR Interpreter only for intrinsic-FoR cases. Routing improves average layout accuracy from 89.20% to 92.00% and increases relative-FoR accuracy from 93.85% to 98.20%, while maintaining intrinsic-FoR accuracy comparable to that of the Full-Rules setting. More details about the routing experiment are presented in Appendix E.4.

We further evaluate whether FoR Interpreter rules can be generated automatically by an LLM. The Generated-Rules setting achieves 88.20% average layout accuracy, close to the 89.20% achieved with Full-Rules, suggesting that the perspective-conversion rule set can be automatically extended from a small number of examples. We observe a similar trend in the FoREST-G generalization test, where LLM-generated rules remain effective in settings with additional objects and relation types. Experimental details and additional results on FoREST-G are provided in Appendix E.5. Finally, although o3 produces highly accurate

Subset	Model	Rel.	Int.	Avg.
Complex	FLUX.1	8.25	8.25	8.25
	Qwen-Image-Edit	16.42	10.79	12.67
	FoR-SALE	14.33	12.88	13.37
Rel.-Aug.	FLUX.1	14.80	10.29	12.17
	Qwen-Image-Edit	26.00	16.57	20.50
	FoR-SALE	21.20	21.43	21.33
Lang.-Var.	FLUX.1	42.72	18.04	30.75
	Qwen-Image-Edit	54.37	23.20	39.25
	FoR-SALE	48.54	29.90	39.50

(a) Diagnostic subsets of FoREST-G.

Method	Rel.	Int.	Avg.
FLUX.1	38.57	20.70	29.00
+ SLD	42.26	19.15	30.80
+ FoR-SALE	43.25	25.20	34.30
- Facing Direction Modification	40.67	22.17	31.50
- Depth Modification	42.65	25.20	34.00
+ FoR-SALE (2 rounds)	47.22	28.83	38.10
- Depth Modification	45.83	26.21	36.10

(b) Ablation of editing operations.

Table 3: Additional diagnostic and ablation results. (a) Image-level accuracy on the diagnostic generalization subsets of FoREST-G. FoR-SALE achieves the highest intrinsic-FoR accuracy across all three subsets. (b) Ablation results using FLUX.1-generated images, demonstrating the contributions of facing-direction modification and depth modification.

layouts, it yields lower final image accuracy than some alternatives, likely because its layouts require more editing actions; we analyze this further in Appendix E.

RQ4. How do the new editing actions help FoR-SALE?

To answer this question, we ablate facing-direction and depth modification using FLUX.1-generated images. As shown in Table 3(b), removing facing-direction modification reduces accuracy by 2.8 pp after one round, whereas removing depth modification reduces it by 0.3 pp after one round and approximately 2 pp after two rounds. This shows that facing-direction modification yields a larger immediate gain, whereas depth modification becomes more influential by the second correction round. We further evaluate a stronger depth-editing backbone by replacing ControlNet with FLUX.1-Fill (Black Forest Labs, 2024b). This backbone improves FoR-SALE’s performance, although depth editing remains imperfect, suggesting that more reliable 3D-aware methods could yield further gains. Details of this experiment are provided in Appendix G.3. Together, these results confirm the importance of both editing operations for spatial alignment.

RQ5. How does FoR-SALE generalize beyond simple two-object settings?

To address this question, we evaluate FoR-SALE on FoREST-G, which contains three subsets targeting different aspects of generalization: compositional scene structure (Complex), broader relation coverage (Relation-Augmented), and natural language variation (Language-Variation). We compare FoR-SALE with the SOTA image-editing model Qwen-Image-Edit in Table 3(a). The results show that performance drops substantially on the Complex and Relation-Augmented subsets, reflecting the increased difficulty of handling more complex spatial constraints. Nevertheless, FoR-SALE follows the same trend as on the main benchmarks, achieving the best intrinsic-FoR accuracy across all subsets and the best average accuracy in each subset. This suggests that FoR-SALE remains effective under harder scene structures, expanded relation types, and naturalized prompts, especially for intrinsic-FoR cases, which are the primary challenge in FoR-aware T2I generation.

5 Conclusion

Given the limitations of current text-to-image (T2I) models in handling spatial relations across diverse frames of reference (FoR), we propose FoR-SALE—Frame of Reference-guided Spatial Adjustment in LLM-based Diffusion Editing—to address this challenge. Our framework extends the Self-correcting LLM-controlled Diffusion approach by introducing three key components: a comprehensive Visual Perception Module, a dedicated FoR Interpreter, and two new latent editing actions. FoR-SALE can be applied to various T2I models and effectively improves the spatial alignment of images initially generated by those models, increasing accuracy by up to 7.0 pp after one correction round and 13.1 pp after three rounds. Using GPT-4o as the base generator, our method achieves SOTA performance on spatial expressions involving FoRs, particularly for intrinsic FoRs, which are especially challenging. These results demonstrate the robustness of our proposed framework for FoR-aware spatial reasoning.

Acknowledgments

This project is partially supported by the Office of Naval Research (ONR) grant N00014-23-1-2417. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research. We thank anonymous reviewers for their constructive feedback, which greatly helped us improve this manuscript.

References

- Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- Reiner Birkel, Diana Wofk, and Matthias Müller. MiDaS v3.1 – a model zoo for robust monocular relative depth estimation, 2023. URL <https://arxiv.org/abs/2307.14460>.
- Black Forest Labs. FLUX.1 [dev], 2024a. URL <https://huggingface.co/black-forest-labs/FLUX.1-dev>. Model card.
- Black Forest Labs. FLUX.1 Fill [dev], 2024b. URL <https://huggingface.co/black-forest-labs/FLUX.1-Fill-dev>. Model card.
- Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14455–14465, 2024.
- Jaemin Cho, Abhay Zala, and Mohit Bansal. Visual programming for step-by-step text-to-image generation and evaluation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 6048–6069, 2023.
- Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. DiffEdit: Diffusion-based semantic image editing with mask guidance. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=3lge0p5o-M->.
- Kenny R. Coventry, Elena Andonova, Thora Tenbrink, Harmen B. Gudde, and Paul E. Engelhardt. Cued by what we see and hear: Spatial reference frame use in language. *Frontiers in Psychology*, Volume 9 - 2018, 2018. ISSN 1664-1078. doi: 10.3389/fpsyg.2018.01287. URL <https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2018.01287>.
- Google DeepMind. Gemini 3 Pro Image model card, November 2025. URL <https://deepmind.google/models/model-cards/gemini-3-pro-image/>.
- Ashish Goswami, Satyam Kumar Modi, Santhosh Rishi Deshineni, Harman Singh, Prathosh A. P, and Parag Singla. GraPE: A generate-plan-edit framework for compositional t2i synthesis, 2024. URL <https://arxiv.org/abs/2412.06089>.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 6840–6851, 2020. URL <https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf>.
- Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747, 2023.

- Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, 2023.
- Parisa Kordjamshidi, Martijn Van Otterlo, and Marie-Francine Moens. Spatial role labeling: Task definition and annotation scheme. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias (eds.), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA). URL <https://aclanthology.org/L10-1584/>.
- Stephen C. Levinson. *Space in Language and Cognition: Explorations in Cognitive Diversity*. Language Culture and Cognition. Cambridge University Press, 2003.
- Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22511–22521, 2023.
- Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. LLM-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=hFALpTb4fR>. Featured Certification.
- Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651, 2023. doi: 10.1162/tacl.a.00566. URL <https://aclanthology.org/2023.tacl-1.37/>.
- Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying dino with grounded pre-training for open-set object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. doi: 10.1007/978-3-031-72970-6.3.
- Zheyuan Liu, Munan Ning, Qihui Zhang, Shuo Yang, Zhongrui Wang, Yiwei Yang, Xianzhe Xu, Yibing Song, Weihua Chen, Fan Wang, and Li Yuan. Cot-lized diffusion: Let’s reinforce t2i generation step-by-step. In *Advances in Neural Information Processing Systems*, volume 38, 2025.
- Matthias Minderer, Alexey A. Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. In *Advances in Neural Information Processing Systems*, 2023.
- Roshanak Mirzaee and Parisa Kordjamshidi. Transfer learning with synthetic corpora for spatial role labeling and reasoning. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6148–6165, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.413. URL <https://aclanthology.org/2022.emnlp-main.413/>.
- Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjamshidi. SPARTQA: A textual question answering benchmark for spatial reasoning. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4582–4598, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.364. URL <https://aclanthology.org/2021.naacl-main.364/>.
- Sicheng Mo, Fangzhou Mu, Kuan Heng Lin, Yanli Liu, Bochen Guan, Yin Li, and Bolei Zhou. Freecontrol: Training-free spatial control of any text-to-image diffusion model with any condition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7465–7475, June 2024.

- Weimin Mou and Timothy P McNamara. Intrinsic frames of reference in spatial memory. *J Exp Psychol Learn Mem Cogn*, 28(1):162–170, January 2002.
- OpenAI. Addendum to gpt-4o system card: Native image generation. Technical Report Native_Image_Generation_System_Card, OpenAI, San Francisco, CA, March 2025a. Available at: https://cdn.openai.com/11998be9-5319-4302-bfbf-1167e093f1fb/Native_Image_Generation_System_Card.pdf.
- OpenAI. Introducing our latest image generation model in the API, April 2025b. URL <https://openai.com/index/image-generation-api/>. Published April 23, 2025.
- OpenAI. OpenAI o3 and o4-mini System Card. Technical report, OpenAI, April 2025c. URL <https://openai.com/index/o3-o4-mini-system-card/>.
- Lianyu Pang, Jian Yin, Baoquan Zhao, Feize Wu, Fu Lee Wang, Qing Li, and Xudong Mao. AttnDreamBooth: Towards text-aligned personalized text-to-image generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=4bINoegDcm>.
- Tanawan Premisri and Parisa Kordjamshidi. FoREST: Frame of reference evaluation in spatial reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 34977–35003. Association for Computational Linguistics, 2025. URL <https://aclanthology.org/2025.emnlp-main.1772/>.
- Qwen Team. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.
- René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12179–12188, 2021. doi: 10.1109/ICCV48922.2021.01196.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.
- Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11321–11329, Jun. 2022. doi: 10.1609/aaai.v36i10.21383. URL <https://ojs.aaai.org/index.php/AAAI/article/view/21383>.
- Stability AI. Stable diffusion 3.5 large. <https://huggingface.co/stabilityai/stable-diffusion-3.5-large>, 2024. Multimodal Diffusion Transformer (MMDiT) text-to-image model with 8.1 billion parameters; released under Stability AI Community License.
- Jiao Sun, Deqing Fu, Yushi Hu, Su Wang, Royi Rassin, Da-Cheng Juan, Dana Alon, Charles Herrmann, Sjoerd Van Steenkiste, Ranjay Krishna, and Cyrus Rashtchian. DreamSync: Aligning text-to-image generation with image understanding feedback. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5920–5945, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.304. URL <https://aclanthology.org/2025.naacl-long.304/>.
- Thora Tenbrink. Reference frames of space and time in language. *Journal of Pragmatics*, 43(3):704–722, 2011. ISSN 0378-2166. doi: <https://doi.org/10.1016/j.pragma.2010.06.020>. URL <https://www.sciencedirect.com/science/article/pii/S037821661000192X>. The Language of Space and Time.
- Ruichen Wang, Zekang Chen, Chen Chen, Jian Ma, Haonan Lu, and Xiaodong Lin. Compositional text-to-image synthesis with attention map control of diffusion models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(6):5544–5552, 2024. doi: 10.1609/aaai.v38i6.28364.

- Xingrui Wang, Wufei Ma, Tiezheng Zhang, Celso M de Melo, Jieneng Chen, and Alan Yuille. Spatial457: A diagnostic benchmark for 6d spatial reasoning of large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24669–24679, 2025a.
- Zehan Wang, Jiayang Xu, Ziang Zhang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Genspace: Benchmarking spatially-aware image generation. In *Advances in Neural Information Processing Systems*, 2025b. Datasets and Benchmarks Track.
- Zehan Wang, Ziang Zhang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Orient anything: Learning robust object orientation estimation from rendering 3D models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 65556–65574, 2025c.
- Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-Image technical report, 2025. URL <https://arxiv.org/abs/2508.02324>.
- Tsung-Han Wu, Long Lian, Joseph E. Gonzalez, Boyi Li, and Trevor Darrell. Self-correcting llm-controlled diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6327–6336, June 2024.
- Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10632–10643, 2025. doi: 10.1109/CVPR52734.2025.00994.
- Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3836–3847, 2023.
- Yue Zhang, Zhiyang Xu, Ying Shen, Parisa Kordjamshidi, and Lifu Huang. SPARTUN3D: Situated spatial understanding of 3D world in large language model. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL <https://openreview.net/forum?id=FGmkSL8NR0>.
- Zheyuan Zhang, Fengyuan Hu, Jayjun Lee, Freda Shi, Parisa Kordjamshidi, Joyce Chai, and Ziqiao Ma. Do vision-language models represent space and how? evaluating spatial frame of reference under ambiguities. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL <https://openreview.net/forum?id=84pDoCD41H>.

A FoR-SALE Implementation Details

The random seed is set to 78 in all experiments to ensure reproducibility.

A.1 LLM Parser

For the implementation of the LLM Parser, we employ Qwen3-32B with reasoning generation (thinking tokens) disabled to enable faster inference, given the simplicity of the task. The temperature is set to 0 for reproducible results, and the maximum token limit is 8196. Listing 3 in Section K provides the complete prompt and examples used for this LLM Parser.

A.2 FoR Interpreter

We select Qwen3-32B with reasoning generation (thinking tokens) enabled for the FoR Interpreter, as this component requires reasoning over the provided rules. To ensure reproducibility, the temperature is set to 0, and the maximum token limit is 8196. Listing 4 in Section K presents the complete prompt and examples used for the FoR Interpreter.

A.3 Layout Interpreter

Similar to the FoR Interpreter, we use Qwen3-32B with reasoning generation (thinking tokens) enabled for this LLM component. For the ablation study, we also evaluate two additional LLMs via the OpenAI API: o3 (model name: o3-2025-04-16) and o4-mini, both from OpenAI. To ensure reproducibility, the temperature is set to 0, and the maximum token limit is 8196. This configuration is applied consistently across all LLMs used in the Layout Interpreter. The prompt for this Layout Interpreter is in Listing 5 in Section K.

A.4 Visual Perception Module

For the implementation of the Visual Perception Module, we employ three components, including object detection, depth estimation, and orientation detection, as mentioned in the main paper. For open-vocabulary object detection, we use OWLv2, with the model ID *google/owlv2-base-patch16-ensemble*. For depth estimation, we select DPT, using the model ID *Intel/dpt-large*. Finally, for orientation detection, we employ OrientAnything, with ViT-Large as the base model. The model weights are loaded from the checkpoint *croplargeEX2/dino_weight.pt*, as provided in the official GitHub repository.

A.5 Evaluation Functions

There are six evaluation functions used to assess the generated image. The visual details are represented in the following format: ((attribute) (object name) (#object ID), $[x, y, w, h]$, D_i , f_i) where (x, y) indicates the coordinates of the upper-left corner of the bounding box from 0.0 to 1.0, w is its width, h is its height, D_i is depth from 0.0 to 1.0 where 1.0 indicates the nearest position to the camera, and f_i is the facing direction label. Each comparison involves two objects, denoted as obj_1 and obj_2 . Before performing the comparison, we compute the center of each object’s bounding box, denoted by (c_x, c_y) , where $c_x = x + w/2$ and $c_y = y + h/2$. The procedure for each comparison is described below.

- **Left.** We determine whether the center of obj_1 is to the left of obj_2 by checking whether c_x of obj_1 is less than c_x of obj_2 . The condition is defined as follows:

$$c_x^{obj_1} < c_x^{obj_2}$$

- **Right.** We determine whether the center of obj_1 is to the right of obj_2 by checking whether c_x of obj_1 is greater than c_x of obj_2 . The condition is defined as follows:

$$c_x^{obj_1} > c_x^{obj_2}$$

- **Above.** We determine whether the center of obj_1 is above obj_2 by checking whether c_y of obj_1 is less than c_y of obj_2 . The condition is defined as follows:

$$c_y^{obj_1} < c_y^{obj_2}$$

- **Below.** We determine whether the center of obj_1 is below obj_2 by checking whether c_y of obj_1 is greater than c_y of obj_2 . The condition is defined as follows:

$$c_y^{obj_1} > c_y^{obj_2}$$

- **Front.** We determine whether obj_1 is in front of obj_2 by comparing D_1 (depth of obj_1) with D_2 (depth of obj_2). The condition is defined as follows:

$$D_1 > D_2$$

- **Back.** We determine whether obj_1 is behind obj_2 using the following inequality:

$$D_1 < D_2$$

FoR-SALE Component	Time (s)
LLM Parser	21.64
Object Detection	0.94
Depth Estimation	1.23
Orientation Detection	1.60
FoR Interpreter	31.84
Layout Interpreter	45.52
Latent Operations	17.91
Final Image Generation	7.99

(a) FoR-SALE runtime breakdown.

Method	Time (s)
FoR-SALE	128.67
SLD	88.52
Qwen-Image-Edit	145.81

(b) End-to-end runtime.

Table 4: Runtime analysis. (a) Execution time of each FoR-SALE component. (b) Average end-to-end runtime of editing methods under the same setting.

B FoREST-G Construction

To construct **FoREST-G**, an extension of FoREST (Premisri & Kordjamshidi, 2025), we follow the prompt templates from the *Clear* portion of FoREST, where the frame of reference is explicitly specified for each spatial relation. We then construct three diagnostic subsets—**Complex**, **Relation-Augmented**, and **Language-Variation**—to evaluate generalization beyond the original two-object setting. Each subset contains 500 spatial expressions, resulting in 1,500 additional examples in total.

The Complex subset introduces chain and star relation topologies to represent relations among multiple objects. Chain structures connect objects sequentially, requiring multi-hop reasoning, while star structures organize multiple relations around a shared reference object. This subset evaluates whether FoR-SALE generalizes beyond isolated pairwise reasoning while retaining the original relation set: {left, right, front, back}. We limit each scene to at most four objects.

The Relation-Augmented subset extends the original relation vocabulary beyond {left, right, front, back} by introducing diagonal relations (e.g., front-left and back-right) and vertical relations (above and below). To isolate the effect of the expanded relation types, this subset retains the original two-object setting.

The Language-Variation subset follows the original FoREST (Premisri & Kordjamshidi, 2025) construction but uses an LLM to rewrite the generated contexts into more natural and diverse language while preserving the underlying spatial constraints. This subset evaluates generalization beyond template-based spatial descriptions. The prompt used for language rewriting is provided in Listing 7.

C Time Efficiency Analysis

We report the execution time of each component in Table 4, averaged over one round of FoR-SALE applied to an initial baseline SLD pipeline. Note that the baseline SLD in this experiment uses Qwen3-32B as the LLM-Interpreter to ensure a fair comparison with FoR-SALE. The overhead is primarily attributable to three LLM-based components. The LLM Parser, which extracts objects from the input, is invoked only in the first round and therefore introduces no additional cost in subsequent editing rounds. The Layout Interpreter is the most time-consuming component, as it requires the LLM to reason about and propose a spatial layout. This cost is comparable to that of the original SLD framework, which similarly relies on both a parser and the Layout Interpreter. Overall, FoR-SALE introduces a runtime overhead of approximately 45% relative to SLD, with the FoR Interpreter accounting for most of the additional cost. This trade-off is justified by the ability to handle spatial relations from non-camera perspectives, a challenging yet largely overlooked aspect in prior work, and yields a notable accuracy improvement even after a single round of FoR-SALE under our evaluation benchmarks.

D Baseline Model Parameters

D.1 Stable Diffusion (SD)

For baselines using SD1.5 and SD2.1, we set the number of inference steps to 50, while keeping all other parameters at their default values. The model ID for SD1.5 is *sd-legacy/stable-diffusion-v1-5*, and for SD2.1, it is *stabilityai/stable-diffusion-2-1*. The baseline using SD3.5-Large employs the model ID *stabilityai/stable-diffusion-3.5-large*, with the number of inference steps set to 30; all other parameters remain unchanged.

D.2 GLIGEN

We use Qwen3 to generate the initial layout for the GLIGEN baseline. The prompt used for layout generation is shown in Listing 2. For the GLIGEN model, we use the model ID *masterful/gligen-1-4-generation-text-box*. We also provide facing direction information when generating images with GLIGEN by augmenting the object names with the corresponding facing directions extracted from the layout generated by Qwen3. The number of inference steps is set to 50, while all other parameters remain unchanged.

D.3 FLUX.1

For generating images with the FLUX.1 baseline, we employ the pipeline with model ID *black-forest-labs/FLUX.1-dev*. The guidance scale is set to 3.5, following the recommended value. The image resolution is 1024×1024, and the number of inference steps is set to 50. Other parameters are set to the default values.

D.4 GPT-4o (gpt-image-1)

We utilize the OpenAI API to generate images for the GPT-4o baseline, employing the model ID *gpt-image-1*. The background setting is set to auto, and the image resolution is configured to 1024×1024. All other parameters are left at their default values. The observed API cost per generated image in our experiments was approximately \$0.01 – \$0.02.

D.5 Nano-Banana-Pro

We use the Gemini API to generate images with Nano-Banana-Pro, using the model ID *gemini-3-pro-image-preview*. We set only the image aspect ratio to 1:1, while keeping all other parameters at their default values.

D.6 Qwen-Image-Edit

For image editing with Qwen-Image-Edit, we employ the diffusion pipeline with the model ID *Qwen/Qwen-Image-Edit*. We use the following prompt to guide the editing process to match the spatial expression: “*Edit the given image to match the following description: T,*” where *T* denotes the input spatial expression. The number of inference steps is set to 30, following the model’s recommendation. All other parameters are kept at their default values.

E Additional Results on Text-to-Image (T2I) baselines

E.1 Additional Results for Early T2I Models

We provide additional results for early T2I models, including SD1.5, SD2.1, and GLIGEN, using layouts generated by Qwen3, as shown in Table 5. All models perform significantly worse than the SOTA baselines discussed in the main results—particularly SD1.5 and SD2.1, which achieve less than 10% accuracy. While GLIGEN shows more acceptable performance on the FoR-LMD benchmark, it performs poorly when orientation requirements are introduced, as in the context of the FoREST benchmark. GLIGEN’s accuracy drops

Method	FoR-LMD			FoREST			Overall Avg.
	Relative	Intrinsic	Average	Relative	Intrinsic	Average	
SD 1.5	12.66	11.80	12.20	7.63	4.00	6.00	9.10
SD 2.1	13.97	10.33	12.00	5.09	7.11	6.00	9.00
Qwen3 + GLIGEN	58.52	21.40	38.40	2.54	1.33	2.00	20.20
SD 3.5-Large	63.75	24.72	42.60	18.11	11.11	15.00	28.80
+ 1-round GraPE	55.46	16.97	34.60	14.91	7.56	11.60	23.10
+ 1-round SLD	61.57	19.56	38.80	22.55	11.55	17.60	28.20
+ 1-round FoR-SALE	61.14	26.56	42.40	24.00	16.00	20.40	31.40
+ 2-round FoR-SALE	67.25	26.94	45.40	28.00	22.22	25.40	35.40
+ 3-round FoR-SALE	70.31	29.52	48.20	28.00	22.22	25.40	36.80
GPT-4o	94.76	24.35	56.60	57.81	35.56	47.80	52.20
+ Cognitive Map	98.23	30.36	62.00	56.00	37.33	47.60	54.80
+ 1-round FoR-SALE	93.01	35.42	61.80	54.18	37.33	46.60	54.20

Table 5: Accuracy of images generated by additional baselines and FoR-SALE. Relative denotes camera-based spatial expression; Intrinsic uses another object’s perspective.

Error Type	Missing Object	Over-Generated Objects	Spatial Relations
Cognitive Map	7	38	173
FoR-SALE	10	75	177

Table 6: Number of errors in each category obtained using the cognitive map and FoR-SALE using GPT-4o as the base generator.

to just 2%, indicating a lack of understanding of object-level attributes—especially facing direction—even when this information is explicitly provided during generation. For SD3.5-Large, the model exhibits behavior similar to FLUX.1 in Table 1, showing strong performance on relative FoR; however, it performs poorly on intrinsic FoR and when processing 3D spatial information.

E.2 Image Generation Errors across Different Baselines

Figure 5 illustrates the error distribution for images generated by SD3.5-Large, FLUX.1, and GPT-4o. We observe notable differences among these models. Note that, while SD3.5-Large and FLUX.1 are diffusion-based T2I models, GPT-4o is a unified generative model trained on multimodal input-output tasks. GPT-4o exhibits significantly fewer missing or additional key objects, indicating stronger object grounding and a more accurate object count. It also shows lower error rates in front/back relations and orientation, suggesting improved

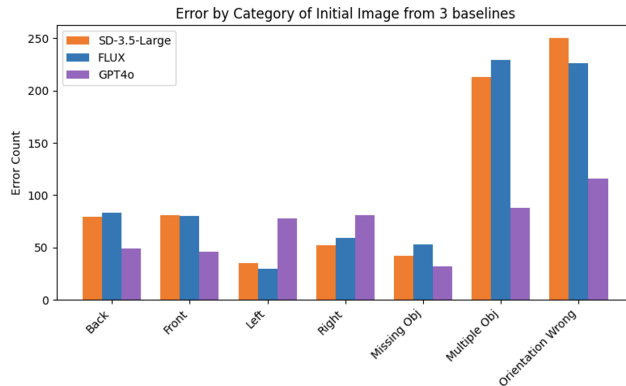


Figure 5: Error analysis of images generated by SD 3.5-Large, FLUX.1, and GPT-4o.

Layout Interpreter	Add	Remove	Attr.	Reposition	Facing	Depth	Depth + Rep.	# Actions
o3	3.60	10.63	0.00	49.82	15.95	10.45	9.55	1110
o4-mini	4.98	11.92	0.00	39.85	20.64	18.98	3.75	906
Qwen3	3.66	10.47	0.00	36.65	16.86	25.65	6.70	955
+ FoR-I (No Rules)	3.81	9.18	0.00	42.47	13.40	26.91	4.23	970
+ FoR-I (Partial Rules)	3.33	8.22	0.00	41.78	10.76	31.12	4.79	1022
+ FoR-I (Full Rules)	3.40	6.90	0.00	43.25	11.95	29.74	4.86	1029

Table 7: Distribution of editing actions required for FoR-LMD and FoREST using GPT-4o-generated initial images under different Layout Interpreter settings. FoR-I denotes FoR Interpreter. Attr., Facing, Depth, and Rep. denote Attribute Modification, Facing Direction Modification, Depth Modification, and Reposition, respectively.

performance in handling 3D spatial configurations, including depth and facing direction. However, GPT-4o performs worse on left/right relations compared to the diffusion-based models. We anticipate that this may be attributed to challenges in perspective conversion, as evidenced by GPT-4o’s high performance on relative FoRs in the FoR-LMD benchmark (94.76%), which requires only camera-centric understanding, contrasted with its significantly lower accuracy on intrinsic FoRs, as reported in the main results. These findings suggest a trade-off in GPT-4o’s spatial performance—namely, strong handling of camera-centric spatial expressions, but limited generalization to non-camera perspectives in text-to-image tasks.

E.3 Cognitive Map

We provide additional results using cognitive-map prompting, a recent spatial-reasoning approach (Yang et al., 2025). We use prompt engineering to extract a cognitive map before generating the image with GPT-4o, rather than applying one round of FoR-SALE. We provide the results in Table 5. We observe that the model achieves a significant gain on FoR-LMD, especially on the relative FoR. This is expected, as the cognitive map should improve 2D spatial relations. We further compared the errors in FoR-LMD based on the output of our automatic evaluator to investigate the source of this improvement in Table 6. We found that the cognitive map helps the model predict a more accurate count of objects, which contributes to improved FoR-LMD in both relative and intrinsic FoR. However, cognitive-map prompting remains weaker on intrinsic FoR-LMD than FoR-SALE (30.36% versus 35.42%), indicating that improved object counting does not resolve intrinsic-FoR reasoning. This is where our proposed framework provides the most value.

E.4 FoR Interpreter Routing

As discussed in Section 4.5, the FoR Interpreter can slightly hurt performance on spatial expressions with relative FoR. To mitigate this issue, we introduce a routing mechanism that determines whether the FoR Interpreter should be invoked based on FoR classification.

In this experiment, we use an LLM to classify whether the prompt expresses an intrinsic or relative spatial relation. If the prompt contains an intrinsic spatial relation, the pipeline invokes the FoR Interpreter before the Layout Interpreter. Otherwise, for relative/camera-centric relations, the pipeline bypasses the FoR Interpreter and directly invokes the Layout Interpreter. This prevents unnecessary FoR conversion for prompts that are already expressed in a relative/camera-centric frame. The prompt used for this classification is provided in Listing 8.

Based on the results in Table 8, the routing mechanism improves average accuracy from 89.20 to 92.00 over Qwen3 + FoR Interpreter. It also recovers the relative-FoR performance drop caused by unnecessary FoR conversion, increasing relative accuracy from 93.85 to 98.20. Intrinsic-FoR performance remains comparable to the non-routed FoR Interpreter setting as expected, since intrinsic prompts are still handled by the FoR Interpreter.

Evaluation	Model	Relative $\uparrow$	Intrinsic $\uparrow$	Avg. $\uparrow$
FoR-LMD & FoREST	o3	99.40	79.03	89.30
	Qwen3-32B	98.21	45.97	72.30
	Qwen3-32B + FoR Interpreter (32 rules)	93.85	84.48	89.20
	Qwen3-32B + FoR Interpreter (Generated rules)	94.25	82.05	88.20
	Qwen3-32B + Routing + FoR Interpreter	98.20	85.68	92.00
Complex (FoREST-G)	o3	97.59	74.43	85.67
	Qwen3-32B	88.65	56.63	72.17
	Qwen3-32B + FoR Interpreter (32 rules)	82.47	76.70	79.50
Rel.-Aug. (FoREST-G)	o3	97.28	75.16	86.00
	Qwen3-32B	87.76	50.33	68.67
	Qwen3-32B + FoR Interpreter (32 rules)	73.47	75.82	74.67
	Qwen3-32B + FoR Interpreter (Generated rules)	78.23	79.08	78.67
Lang.-Var.(FoREST-G)	o3	100.00	86.60	93.50
	Qwen3-32B	97.09	52.58	75.50
	Qwen3-32B + FoR Interpreter (32 rules)	92.23	92.78	92.50

Table 8: Layout accuracy under different FoR Interpreter settings on the original and diagnostic evaluation subsets.

E.5 FoR Interpreter Additional Experiment

LLM as Rule Generator. To examine the scalability of the handcrafted FoR rules presented in Appendix I, we conduct additional experiments in which an LLM generates perspective-conversion rules from few-shot examples. We consider two settings.

First, we provide only four example rules and prompt an LLM, o3 (OpenAI, 2025c), to generate the remaining rules for the benchmark relations and facing directions used in FoR-LMD and FoREST. This setting tests whether the LLM can recover the original rule space from partial examples. As shown in Tables 2 and 8, the generated rules provide sufficient information for the FoR Interpreter, achieving performance close to the manually crafted rules in the original setting (FoR-LMD and FoREST). This suggests that the rules can be generated and used effectively.

Second, we provide all 32 original rules and ask the LLM to generate rules for the new relations in the Relation-Augmented subset of FoREST-G. This setting tests whether the FoR Interpreter can extend beyond the original left, right, front, and back relations. We then augment the FoR Interpreter with the generated rules and evaluate its performance on this subset. As shown in Table 8, the 32-rule FoR Interpreter improves intrinsic accuracy from 50.33 to 75.82 over vanilla Qwen3, showing that explicit FoR conversion remains useful beyond the original benchmark setting. Adding LLM-generated rules further improves intrinsic accuracy to 79.08 and average accuracy to 78.67, yielding an average gain of 4.0 pp over the 32-rule variant. This suggests that the rule set can be extended beyond the original relations, and that rules generated from the handcrafted examples remain useful for more complex relation types.

FoREST-G. As shown in Table 8, we evaluate the FoR Interpreter on FoREST-G using GPT-4o-generated initial images. Across all FoREST-G subsets, which test generalization to complex scenes, augmented relation types, and language variation, incorporating the FoR Interpreter consistently improves Qwen3, especially on intrinsic-FoR cases. In several settings, Qwen3 with the FoR Interpreter approaches or matches the performance of the stronger o3 model. These results highlight the importance of explicit FoR conversion in helping LLMs interpret spatial relations from different frames of reference.

Method	FoR-LMD			FoREST		
	Rel. $\uparrow$	Int. $\uparrow$	Avg. $\uparrow$	Rel. $\uparrow$	Int. $\uparrow$	Avg. $\uparrow$
FLUX.1	62.44	33.21	46.60	19.27	12.00	16.00
Qwen-Image-Edit	72.05	35.42	52.20	32.00	20.89	27.00
Qwen-Image-Edit (2 $\times$)	71.61	35.05	51.80	32.36	21.78	27.60
+ FoR-SALE (1 round)	66.81	41.69	53.20	23.63	21.78	22.80
+ FoR-SALE (2 rounds)	69.43	43.91	55.60	28.36	24.00	26.40
+ FoR-SALE (3 rounds)	71.61	43.91	56.20	31.27	28.89	30.20

Table 9: Accuracy of generated images with an alternative perception stack. We replace OWLv2 and DPT with Grounding-DINO and MiDaS, respectively. The improvement trend remains consistent, especially for intrinsic-FoR cases.

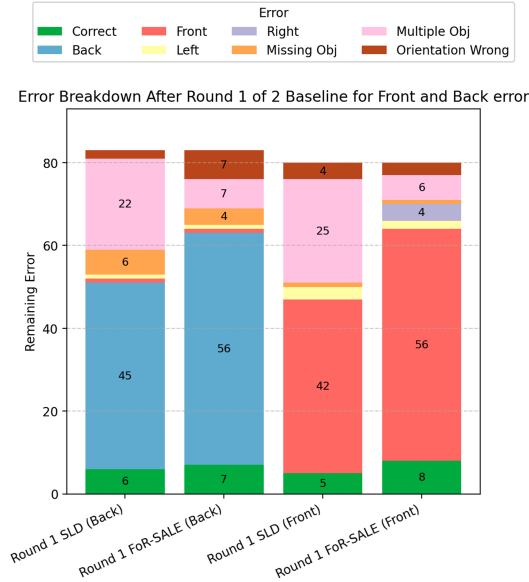


Figure 6: Error breakdown after one round of editing initial images from FLUX.1 using SLD and FoR-SALE on front and back relation errors.

F Visual Perception Modules

In this section, we present additional experiments with alternative visual perception stacks to ensure that the observed improvements are not tied to the specific perception modules used in FoR-SALE or in the evaluation protocol. Specifically, we replace the object detector with Grounding-DINO (Liu et al., 2024) and the depth estimator with MiDaS 3.0 (Birkel et al., 2023). We then re-evaluate the same images generated by FLUX.1 and one-round FoR-SALE, using the initial FLUX.1 images as input. The results are reported in Table 9. We observe that both the original evaluation stack in Table 1 and the alternative stack in Table 9 show similar improvements of 5.3 and 6.7 pp, respectively, over the initial FLUX.1 images, a difference of only 1.4 pp between the two evaluation settings. These findings indicate that FoR-SALE consistently improves performance even when different visual perception modules are used.

G Analysis of the FoR-SALE Framework

G.1 Additional Quantitative Error Analysis of FoR-SALE

In this section, we further analyze the errors observed after applying one round of FoR-SALE to the initial images generated by FLUX.1. We compare SLD and FoR-SALE in editing images containing front/back spatial relation errors in Figure 6. We observe that while SLD attempts to correct the front/back relation, it often introduces multiple instances of the target objects instead of editing the original ones. This behavior results in a lower front/back error after one round of editing, but it comes at the cost of generating additional object-related errors. We attribute this limitation to SLD’s lack of depth awareness, which leads to incorrect editing operations. In contrast, FoR-SALE, which incorporates depth information, achieves slightly better correction on front/back errors without introducing new object duplication or misalignment. Importantly, FoR-SALE avoids introducing new error types, making it more robust for subsequent editing rounds.

G.2 Detailed Analysis of the Effects of Different Layout Interpreters and Editing Actions

We report the distribution of editing actions required for images generated by GPT-4o when using different Layout Interpreters in Table 7. We observe that o3 requires significantly more editing actions compared to other models, with repositioning accounting for 59.37% of all actions (repositioning and depth modification with repositioning). This suggests that o3 often generates layouts where the object is repositioned, likely indicating that it is proposing an entirely new scene layout rather than minimally adjusting the original. This behavior may explain the performance drop observed when using o3-generated layouts, as reported in the main results. It also highlights a limitation of the FoR-SALE framework: the difficulty in handling cases that require multiple or complex repositioning actions. These findings suggest that future work may explore improved strategies for accurately moving objects—or even fully regenerating images—when layout revisions are extensive.

G.3 Alternative Depth Editing Tools

To investigate whether the depth editing error is specific to ControlNet rather than the FoR reasoning component, we conduct an additional experiment replacing ControlNet with FLUX.1-Fill-Dev as a stronger recent editing backbone on the FoREST benchmark.

According to Table 10, the results show that FLUX.1-Fill-Dev improves final image-level accuracy for both relative and intrinsic FoR cases. Relative accuracy increases from 25.09% to 31.27%, and intrinsic accuracy increases from 22.22% to 28.44%. The error analysis also shows reductions in key execution-level failure modes: missing-object errors decrease from 8.80% to 8.20%, and orientation errors decrease from 40.00% to 33.60%. However, not all failure modes improve. 3D incorrectness slightly increases from 7.00% to 7.40%, and extra-object errors increase from 17.60% to 18.00%. This suggests that FLUX.1-Fill-Dev better handles orientation editing, but can introduce other generation errors, especially object-count inconsistency.

Nevertheless, the absolute accuracy remains limited, reflecting the difficulty of the evaluated dataset rather than a failure of the FoR reasoning module. The task requires editors to jointly modify orientation, preserve plausible depth, and satisfy the target FoR relation. The low FoREST performance of Qwen-Image-Edit, despite being a strong recent editor, further suggests that these failures stem from general limitations of current diffusion/editing backbones on orientation-sensitive spatial editing rather than from FoR-SALE itself.

Overall, this experiment shows that FoR-SALE is modular and can benefit from stronger editing backbones, as reflected by improved relative/intrinsic accuracy and reduced orientation errors with FLUX.1-Fill-Dev. At the same time, fully reliable image-level FoR correction remains constrained by the capabilities of current editing models.

Model	Rel. $\uparrow$	Int. $\uparrow$	3D Inc. $\downarrow$	Missing $\downarrow$	Extra $\downarrow$	Orient. Err. $\downarrow$
FLUX.1	18.18	15.56	—	—	—	—
FoR-SALE	25.09	22.22	7.00%	8.80%	17.60%	40.00%
FoR-SALE + FLUX.1-Fill-Dev	31.27	28.44	7.40%	8.20%	18.00%	33.60%

Table 10: Results with FLUX.1-Fill-Dev as a stronger fill-based editing backbone compared with the original baseline. 3D Inc. denotes incorrect 3D relations, Missing denotes missing-object errors, Extra denotes extra-object errors, and Orient. Err. denotes orientation errors.

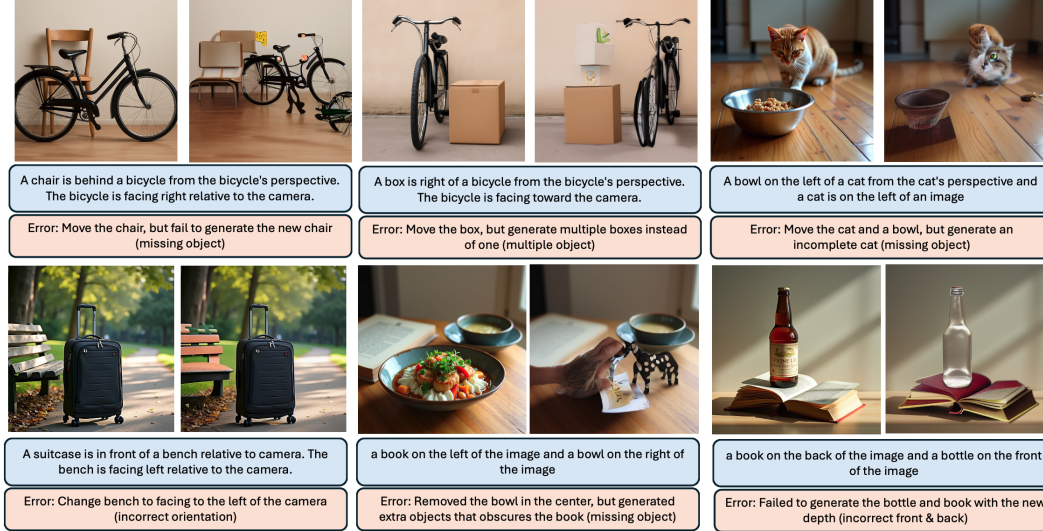


Figure 7: Examples of editing errors using FoR-SALE. The blue box indicates the input spatial expression, while the orange box explains the editing action and the underlying reason for the error.

H Failure Case Observation

We present examples of FoR-SALE editing failures in Figure 7. The most common errors include multiple instances of key objects, incorrect orientation, and missing objects, as also reflected in the main paper’s quantitative results. We attribute these failures primarily to challenges in object removal and regeneration, which can lead to either the unintended deletion of key objects or the generation of extraneous ones—ultimately making the intended objects undetectable in the final image. Additionally, we believe that modifying orientation and depth remains difficult for current diffusion models, which limits the effectiveness of FoR-SALE in correcting these types of spatial errors.

H.1 Percentage of FoR-SALE failure cases

Observation Setting. To identify the sources of FoR-SALE generation errors, we sample 60 images (10% of failure cases in round 1) from the incorrect cases produced by applying one round of FoR-SALE to initial images generated by FLUX.1. We categorize the errors into six types:

- 1. Removing object failure (E1).** FoR-SALE fails to remove an object from the image completely.
- 2. Incorrect orientation generation (E2).** The diffusion model in FoR-SALE generates objects with incorrect orientations.
- 3. Incorrect depth generation (E3).** The diffusion model with ControlNet in FoR-SALE generates images with incorrect depth.

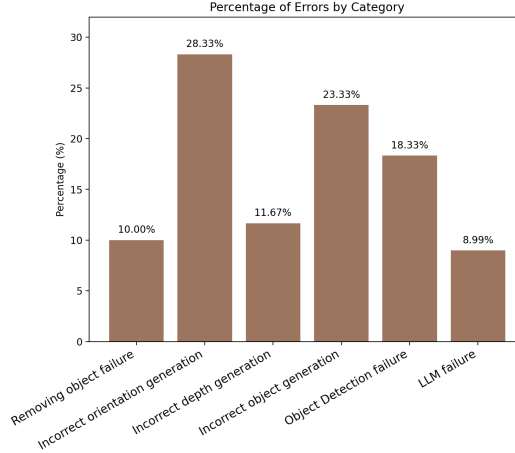


Figure 8: Percentage of each error category among 60 sampled failure cases after one round of FoR-SALE applied to FLUX.1-generated images.

Baseline	Soft-Error	Hard-Error
SD 3.5-Large	56.25	73.20
FLUX.1	67.10	78.71
GPT-4o (gpt-image-1)	49.47	72.35

Table 11: DiffEdit orientation-editing error rates after one round of FoR-SALE across three base generators.

4. Incorrect object generation (E4). The final Stable Diffusion model in FoR-SALE fails to synthesize the correct object image from the edited latent space.

5. Object detection failure (E5). The visual perception modules fail to detect correct object properties or spatial coordinates.

6. LLM failure (E6). The LLM-controlled editing suggests an incorrect image layout, leading to erroneous edits.

Results. We report the manually checked cases in Figure 8. The bar chart illustrates the percentage of each error that occurs in the incorrect sample. We observe three major errors that contributed to the failure of FoR-SALE. The first error (23.33% of all errors) is generation failure, which occurs when multiple compositions of the latent space overlap at the same pixel location, leading to poor-quality object images (example in the upper-left picture in Figure 7). This issue is closely related to the second source of error, object detection failure (18.33% of all errors), which arises when two objects are positioned too closely. In such cases, the detector may fail to capture all objects or focus on incorrect regions of the image due to flawed generation. The third and most significant source of error is incorrect orientation generation (28.33% of all errors). As noted by Wang et al. (2025b), even SOTA T2I models struggle to produce correct object orientations, highlighting a fundamental challenge that requires stronger 3D awareness in diffusion models. A similar challenge is observed for depth generation, where limited progress has been made in depth-editing actions. To further analyze DiffEdit’s orientation editing, we examine images from three base generators that underwent only orientation editing during one round of FoR-SALE. We report the error percentage of DiffEdit orientation edits in Table 11. The soft-error metric allows more orientation labels to be considered correct (e.g., treating “left” as correct for backward-left and forward-left, and vice versa), whereas the hard-error metric requires the generated orientation to match the expected label exactly.

I Perspective Conversion Rules

In this section, we present all manually crafted perspective-conversion rules used in the FoR Interpreter and their corresponding camera-perspective mappings. The rules are categorized by the facing direction of the reference object. Each facing direction is associated with exactly four conversion rules, corresponding to the four spatial relations considered in this work, i.e., left, right, front, and back.

1. Facing toward the camera.
 - (a) Left. If the object is facing toward the camera (front), then the left side of the object is on the right from the camera perspective.
 - (b) Right. If the object is facing toward the camera (front), then the right side of the object is on the left from the camera perspective.
 - (c) Front. If the object is facing toward the camera (front), then the front side of the object is in the front direction from the camera perspective.
 - (d) Back. If the object is facing toward the camera (front), then the back side of the object is in the back direction from the camera perspective.
2. Facing forward-left.
 - (a) Left. If the object is facing forward-left, then the left side of the object is on the right from the camera perspective.
 - (b) Right. If the object is facing forward-left, then the right side of the object is on the left from the camera perspective.
 - (c) Front. If the object is facing forward-left, then the front side of the object is in the front direction from the camera perspective.
 - (d) Back. If the object is facing forward-left, then the back side of the object is in the back direction from the camera perspective.
3. Facing left.
 - (a) Left. If the object is facing left, then the left side of the object is in the front direction from the camera perspective.
 - (b) Right. If the object is facing left, then the right side of the object is in the back direction from the camera perspective.
 - (c) Front. If the object is facing left, then the front side of the object is on the left from the camera perspective.
 - (d) Back. If the object is facing left, then the back side of the object is on the right from the camera perspective.
4. Facing backward-left.
 - (a) Left. If the object is facing backward-left, then the left side of the object is on the left from the camera perspective.
 - (b) Right. If the object is facing backward-left, then the right side of the object is on the right from the camera perspective.
 - (c) Front. If the object is facing backward-left, then the front side of the object is in the back direction from the camera perspective.
 - (d) Back. If the object is facing backward-left, then the back side of the object is in the front direction from the camera perspective.
5. Facing away from the camera.
 - (a) Left. If the object is facing away from the camera (back), then the left side of the object is on the left from the camera perspective.
 - (b) Right. If the object is facing away from the camera (back), then the right side of the object is on the right from the camera perspective.
 - (c) Front. If the object is facing away from the camera (back), then the front side of the object is in the back direction from the camera perspective.
 - (d) Back. If the object is facing away from the camera (back), then the back side of the object is in the front direction from the camera perspective.

6. Facing backward-right.
 - (a) Left. If the object is facing backward-right, then the left side of the object is on the left from the camera perspective.
 - (b) Right. If the object is facing backward-right, then the right side of the object is on the right from the camera perspective.
 - (c) Front. If the object is facing backward-right, then the front side of the object is in the back direction from the camera perspective.
 - (d) Back. If the object is facing backward-right, then the back side of the object is in the front direction from the camera perspective.
7. Facing right.
 - (a) Left. If the object is facing right, then the left side of the object is in the back direction from the camera perspective.
 - (b) Right. If the object is facing right, then the right side of the object is in the front direction from the camera perspective.
 - (c) Front. If the object is facing right, then the front side of the object is on the right from the camera perspective.
 - (d) Back. If the object is facing right, then the back side of the object is on the left from the camera perspective.
8. Facing forward-right.
 - (a) Left. If the object is facing forward-right, then the left side of the object is on the right from the camera perspective.
 - (b) Right. If the object is facing forward-right, then the right side of the object is on the left from the camera perspective.
 - (c) Front. If the object is facing forward-right, then the front side of the object is in the front direction from the camera perspective.
 - (d) Back. If the object is facing forward-right, then the back side of the object is in the back direction from the camera perspective.

J Human Alignment Score

Listing 1: Instruction for collecting human results on evaluating generated images.

```

"""
Instruction:
You will be presented with an image generated by a generative model along
with its corresponding input prompt. Your task is to determine
whether the generated image aligns with the prompt. If the image is
aligned, assign a score of 1; otherwise, assign a score of 0.
"""

```

J.1 Experimental setting

We provide a set of 200 images—sampled from the initial FLUX.1 generations and the first round of FoR-SALE—to a human annotator, compensated as a research assistant. This selection ensures diversity in the generated outputs and maintains independence from our specific generation framework. The annotator follows the instructions in Listing 1 to judge whether each image aligns with the corresponding input description. Each sample is labeled as either aligned (1) or not aligned (0), and we collect these binary scores for all images.

K Prompt Specifications

In this section, we provide the prompts used by the LLM components throughout the entire experiment. The listings show the static prompt templates. Task-specific inputs are appended programmatically at runtime. For example, `{conversion_rules}` and `{remaining_examples}` denote dynamically inserted content; the 32 manually defined rules are provided in Appendix I.

Listing 2: Prompt for generating a layout for GLIGEN.

```

Your task is to generate the bounding boxes of objects mentioned in the
caption, along with the direction that the objects are facing.
The image is size 512x512.
The bounding box should be in the format of [x, y, width, height], with
each value from 0 to 1.
The direction that the object is facing should be one of these options: [
front, back, left, right].
Please consider the frame of reference of the caption and the direction
of the reference object.
The answer should be in the form:
Reasoning: Explanation
Layout: Layout

The example of layout is [('cat', [0.1, 0.3, 0.5, 0.4], 'right'), ('cow',
[0.6, 0.5, 0.3, 0.4], 'right')]

User prompt: {prompt}
Current Objects: {current_objects}

```

Listing 3: Prompt for LLM Parser.

```

# Your Role: Excellent Parser

## Objective: Analyze scene descriptions to identify objects and their
attributes.

## Process Steps
1. Read the user prompt (scene description).
2. Identify all objects mentioned with quantities.
3. Extract attributes of each object (color, size, material, etc.).
4. Ignore facing attribute (facing to left, facing to right, facing
forward).
5. If the description mentions objects that shouldn't be in the image,
take note of the negation part.
6. Explain your understanding (reasoning) and then format your result (
answer/negation) as shown in the examples.
7. Importance of Extracting Attributes: Attributes provide specific
details about the objects. This helps differentiate between similar
objects and gives a clearer understanding of the scene.

## Examples

- Example 1
  User prompt: A brown horse is beneath a black dog. Another orange cat
  is beneath a brown horse.
  Reasoning: The description talks about three objects: a brown horse, a
  black dog, and an orange cat. We report the color attribute
  thoroughly. No specified negation terms. No background is mentioned
  and thus fill in the default one.
  Objects: [('horse', ['brown']), ('dog', ['black']), ('cat', ['orange'])
  ]
  Background: A realistic image
  Negation:

- Example 2
  User prompt: There's a white car and a yellow airplane in a garage.
  They're in front of two dogs and behind a cat. The car is small.
  Another yellow car is outside the garage.
  Reasoning: The scene has two cars, one airplane, two dogs, and a cat.
  The car and airplane have colors. The first car also has a size. No
  specified negation terms. The background is a garage.
  Objects: [('car', ['white and small', 'yellow']), ('airplane', ['yellow
  ']), ('dog', [None, None]), ('cat', [None])]
  Background: A realistic image in a garage

```

Negation:

- Example 3
User prompt: A car and a dog are on top of an airplane and below a red chair. There's another dog sitting on the mentioned chair.
Reasoning: Five object instances are described: one car, one airplane, two dogs, and one red chair. No specified negation terms. No background is mentioned, and thus fill in the default one.
Objects: [('car', [None]), ('airplane', [None]), ('dog', [None, None]), ('chair', ['red'])]
Background: A realistic image
Negation:
- Example 4
User prompt: An oil painting at the beach of a blue bicycle to the left of a bench and to the right of a palm tree with five seagulls in the sky.
Reasoning: Here, there are five seagulls, one blue bicycle, one palm tree, and one bench. No specified negation terms. The background is an oil painting at the beach.
Objects: [('bicycle', ['blue']), ('palm tree', [None]), ('seagull', [None, None, None, None, None]), ('bench', [None])]
Background: An oil painting at the beach
Negation:
- Example 5
User prompt: An animated-style image of a scene without backpacks.
Reasoning: The description clearly states no backpacks, so this must be acknowledged. The user provides the negative prompt of backpacks. The background is an animated-style image.
Objects: [('backpacks', [None])]
Background: An animated-style image
Negation: backpacks
- Example 6
User Prompt: Make the dog a sleeping dog and remove all shadows in an image of a grassland.
Reasoning: The user prompt specifies a sleeping dog and requests that all shadows be removed. The background is a realistic image of a grassland.
Objects: [('dog', ['sleeping']), ('shadow', [None])]
Background: A realistic image of a grassland
Negation: shadows
- Example 7
User Prompt: A fire hydrant is behind a cat relative to the observer. The cat is facing away from the observer.
Reasoning: Two objects are described: one fire hydrant and one cat. The facing direction is ignored as instructed. No specified negation terms. No background is mentioned, and thus fill in the default one.
Objects: [('fire hydrant', [None]), ('cat', [None])]
Background: A realistic image
Negation:

Your Current Task: Follow the steps closely and accurately identify objects based on the given prompt. Ensure adherence to the above output format.

User prompt: {prompt}
Objects: {current_objects}
Background: {background}
Negation: {negation}

Listing 4: Prompt for FoR Interpreter.

```

# Your Role: Expert on spatial relations in multiple perspectives

## Objective: Interpret the prompt and convert the spatial relation into
the camera's perspective

## Image and Object Specification
1. Image Coordinates: Define square images with top-left at [0, 0] and
bottom-right at [1, 1].
2. Four pieces of information are given for each object in order: object
name, bounding box, depth, and facing direction.
3. Object Format: (object, box, depth, facing direction)
4. Box Format: [Top-left x, Top-left y, Width, Height]
5. Depth: Define depth of the object from furthest at 0 and nearest at 1.
6. Facing Direction: An orientation of the object relative to the camera,
which can be None, left, forward-left, backward-left, right, forward
-right, backward-right, front (facing forward or facing toward), or
back (facing backward or facing away).

## Key Guidelines
1. Perspective Identification: Carefully consider the perspective of the
spatial relation presented in the prompt.
2. Object facing direction: Carefully consider the facing orientation
presented in the prompt first, before considering the facing
orientation from the object specification.
3. Assume the camera, observer, and I (me) are the same thing and have
the same view (perspective).
4. Look at the example closely to see how the conversion needs to be made
.

<RULES>
{conversion_rules}
</RULES>

## Process Steps
1. Read and understand the user prompt (scene description).
2. Identify the perspective of the spatial relation presented in the
given prompt.
3. Check whether the facing direction is provided in the prompt.
4. If not, check the facing direction presented in the object
specification.
5. Explain your understanding (reasoning) and then convert the
perspective into the camera's perspective.
6. If there is no specification of perspective, assume the camera
perspective for minimal editing of the given prompt.
7. Do not modify any other part of the prompt except for spatial relation
(s).
8. Do not update the object, only modify the prompt.

## Examples
- Example 1
  User prompt: a backpack on the right of a car from the car's
perspective and a car on the left
  Current Objects: [('backpack #1', [0.302, 0.293, 0.335, 0.194], 0.63,
None), ('car #1', [0.027, 0.324, 0.246, 0.160], 0.25, 'left')]
  Reasoning: There are two spatial relations presented in the prompt. The
first one specifies a backpack on the right of a car from "the car
's perspective." There is no specific facing direction of the car
presented in the prompt. Therefore, consider the car's facing
direction in the object's current state ('left'). The car is facing
to the left of the photo. Therefore, the right of the car from the
car's perspective corresponds to behind the car from the camera's
perspective. The second spatial relation is that the car is on the

```

- left. This does not specify the perspective. Therefore, assume a camera perspective for this one and make no update to the second spatial relation.
- Updated prompt: a backpack behind a car from the camera's perspective and a car on the left
- Example 2

User prompt: a cat is on the left and the cup is on the right of the cat from the cat's view

Current Objects: [('cat #1', [0.169, 0.563, 0.323, 0.291], 0.901, 'right'), ('cup #1', [0.59, 0.186, 0.408, 0.814], 0.732, None)]

Reasoning: There are two spatial relations presented in the prompt. The first spatial relation is a cat on the left. The prompt does not specify the perspective. Therefore, assume a camera perspective for this one and make no update. The second one specifies that the cup is on the right of the cat from "the cat's view." There is no specific facing direction of the cat presented in the prompt. Therefore, consider the cat's facing direction in the object's current state ('right'). The cat is facing to the right of the photo. Therefore, the right of the cat from the cat's perspective corresponds to in front of the cat from the camera's perspective.

Updated prompt: a cat is on the left and the cup is in front of the cat from the camera's view
 - Example 3

User prompt: A cow is in front of a sheep from the camera angle. The sheep is facing right relative to the camera.

Current Objects: [('cow #1', [0.354, 0.365, 0.285, 0.385], 0.41, None), ('sheep #1', [0.608, 0.120, 0.285, 0.200], 0.82, 'right')]

Reasoning: There is only one spatial relation presented in the prompt. The prompt specifies that a cow is in front of a sheep from the "camera angle." This spatial relation is from the camera's perspective. Therefore, there is no need for change.

Updated prompt: A cow is in front of a sheep from the camera angle. The sheep is facing right relative to the camera.
 - Example 4

User prompt: A fire hydrant is back of a sheep from the sheep's perspective. The sheep is facing away from the camera.

Current Objects: [('fire hydrant #1', [0.113, 0.365, 0.251, 0.251], 0.64, None), ('sheep #1', [0.608, 0.120, 0.251, 0.251], 0.52, 'back')]

Reasoning: There is only one spatial relation presented in the prompt. The prompt specifies that a fire hydrant is behind a sheep from "the sheep's perspective." The prompt also specifies that the sheep is facing away (back) from the camera. Therefore, the back of the sheep corresponds to the front direction of the camera. The updated spatial relation is that the fire hydrant is in front of the sheep from the camera's perspective.

Updated prompt: A fire hydrant is in front of a sheep from the camera's perspective. The sheep is facing away from the camera.
 - Example 5

User prompt: A deer is to the left of a car from the car's perspective. The car is facing away from the camera.

Current Objects: [('deer #1', [0.454, 0.165, 0.285, 0.385], 0.42, None), ('car #1', [0.608, 0.620, 0.285, 0.200], 0.83, 'back')]

Reasoning: There is only one spatial relation presented in the prompt. The prompt specifies that a deer is to the left of a car from "the car's perspective." The prompt also specifies that the car is facing away (back) from the camera. Therefore, the left side of the car corresponds to the left direction of the camera. The updated spatial relation is that the deer is to the left of the car from the camera's perspective.

- Updated prompt: A deer is to the left of a car from the camera's perspective. The car is facing away from the camera.
- Example 6

User prompt: A cow is to the right of a horse from the horse's perspective. The horse is facing toward relative to the camera.

Current Objects: [('cow #1', [0.113, 0.365, 0.352, 0.352], 0.83, None), ('horse #1', [0.608, 0.120, 0.352, 0.352], 0.25, 'front')]

Reasoning: There is only one spatial relation presented in the prompt. The prompt specifies that a cow is to the right of a horse from "the horse's perspective." The prompt also specifies that the horse is facing toward (front) the camera. Therefore, the right side of the horse corresponds to the left direction of the camera. The updated spatial relation is that the cow is to the left of the horse from the camera's perspective.

Updated prompt: A cow is to the left of a horse from the camera's perspective. The horse is facing toward relative to the camera.
 - Example 7

User prompt: A deer is in front of a sheep from the sheep's perspective. The sheep is facing toward relative to the camera.

Current Objects: [('deer #1', [0.454, 0.365, 0.285, 0.385], 0.64, None), ('sheep #1', [0.608, 0.120, 0.285, 0.200], 0.32, 'front')]

Reasoning: There is only one spatial relation presented in the prompt. The prompt specifies that a deer is in front of a sheep from "the sheep's perspective." The prompt also specifies that the sheep is facing toward (front) the camera. Therefore, the front of the sheep corresponds to the front direction of the camera. The updated spatial relation is that the deer is in front of the sheep from the camera's perspective.

Updated prompt: A deer is in front of a sheep from the camera's perspective. The sheep is facing toward relative to the camera.
 - Example 8

User prompt: A deer is in front of a dog from the dog's perspective. The dog is facing right relative to the camera.

Current Objects: [('deer #1', [0.186, 0.592, 0.449, 0.408], 0.45, 'front'), ('dog #1', [0.376, 0.194, 0.624, 0.502], 0.53, 'right')]

Reasoning: There is only one spatial relation presented in the prompt. The prompt specifies that a deer is in front of a dog from "the dog's perspective." The prompt also specifies that the dog is facing to the right relative to the camera. Therefore, the front of the dog corresponds to the right direction of the camera. The updated spatial relation is that the deer is to the right of the dog from the camera's perspective.

Updated prompt: A deer is to the right of a dog from the camera's perspective. The dog is facing right relative to the camera.
 - Example 9

User prompt: A deer is to the right of a car from the car's perspective. The car is facing away from the camera.

Current Objects: [('deer #1', [0.454, 0.165, 0.285, 0.385], 0.42, None), ('car #1', [0.608, 0.620, 0.285, 0.200], 0.83, 'back')]

Reasoning: There is only one spatial relation presented in the prompt. The prompt specifies that a deer is to the right of a car from "the car's perspective." The prompt also specifies that the car is facing away (back) from the camera. Therefore, the right side of the car corresponds to the right direction of the camera. The updated spatial relation is that the deer is to the right of the car from the camera's perspective.

Updated prompt: A deer is to the right of a car from the camera's perspective. The car is facing away from the camera.

Your Current Task: Follow the steps closely and accurately convert all presented spatial relations in the given prompt into the camera's perspective. Ensure adherence to the above output format.

User prompt: {prompt}

Current Objects: {current_objects}

Listing 5: Prompt for Layout Interpreter.

```
# Your Role: Expert Bounding Box Adjuster

## Objective: Manipulate bounding boxes in square images according to the
user prompt while maintaining visual accuracy.

## Object Specifications and Manipulations
1. Image Coordinates: Define square images with top-left at [0, 0] and
bottom-right at [1, 1].
2. Object Format: (object, box, depth, orientation)
3. Box Format: [Top-left x, Top-left y, Width, Height]
4. Depth: Define the depth of the object, with furthest at 0 and nearest
at 1.
5. Orientation Format: An orientation of the object, which can be None, '
left', 'right', 'front', 'back', 'forward-left', 'forward-right', '
backward-left', or 'backward-right'.
6. Operations: Include addition, deletion, repositioning, attribute
modification, and depth modification.

## Key Guidelines
1. Alignment: Follow the user's prompt, keeping the specified object
count and attributes. Deem it incorrect if the described object lacks
specified attributes.
2. Boundary Adherence: Keep bounding box coordinates within [0, 1].
3. Depth Adherence: Keep each object's depth within [0, 1].
4. Orientation Adherence: An orientation must change depending on the
prompt. If nothing is specified in the prompt, do not change the
orientation of the object.
5. Minimal Modifications: Change bounding boxes or depth only if they do
not match the user's prompt (i.e., do not modify matched objects).
6. Overlap Reduction: Minimize intersections in new boxes and remove the
smallest, least overlapping objects when an object needs to be
removed.
7. Left/Right Relation: Compare the horizontal center of each bounding
box, where center_x = x + width / 2. Object A is left of object B if
center_x(A) < center_x(B), and right of object B if center_x(A) >
center_x(B).
8. Above/Below Relation: Compare the vertical center of each bounding box
, where center_y = y + height / 2. Object A is above object B if
center_y(A) < center_y(B), and below object B if center_y(A) >
center_y(B).
9. Front/Behind Relation: Compare object depth. Object A is in front of
object B if depth(A) > depth(B), and behind object B if depth(A) <
depth(B).

## Process Steps
1. Interpret prompts: Read and understand the user's prompt.
2. Implement Changes: Review and adjust current bounding boxes to meet
user specifications.
3. Explain Adjustments: Justify the reasons behind each alteration and
ensure every adjustment abides by the key guidelines.
4. Output the Result: Present the reasoning first, followed by the
updated objects section, which should include a list of bounding
boxes in Python format.

## Examples
- Example 1
```

- User prompt: A realistic image of landscape scene depicting a green car parking on the left of a blue truck, with a red air balloon and a bird in the sky
- Current Objects: [('green car #1', [0.027, 0.365, 0.275, 0.207], 0.6, None), ('blue truck #1', [0.350, 0.368, 0.272, 0.208], 0.7, None), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176], 0.4, None)]
- Reasoning: The green car is already to the left of the blue truck, and the red air balloon is already present. The prompt also specifies a bird in the sky, but no bird is present in the current objects. Therefore, add a bird in an upper region of the image while keeping all existing matched objects unchanged and ensuring all coordinates and dimensions remain within [0, 1].
- Updated Objects: [('green car #1', [0.027, 0.365, 0.275, 0.207], 0.6, None), ('blue truck #1', [0.350, 0.368, 0.272, 0.208], 0.7, None), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176], 0.4, None), ('bird #1', [0.385, 0.054, 0.186, 0.130], 0.3, None)]
- Example 2
- User prompt: A realistic image of landscape scene depicting a green car parking on the right of a blue truck, with a red air balloon and a bird in the sky
- Current Objects: [('green car #1', [0.027, 0.365, 0.275, 0.207], 0.79, 'left'), ('blue truck #1', [0.350, 0.369, 0.272, 0.208], 0.68, 'right'), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176], 0.15, None)]
- Reasoning: The green car is currently to the left of the blue truck, but the prompt requires the green car to be to the right of the blue truck. Therefore, reposition the car and truck so that their left-right relation matches the prompt. The prompt also specifies a bird in the sky, but no bird is present in the current objects, so add one in an upper image region while minimizing overlap. Keep all other matched objects unchanged.
- Updated Objects: [('green car #1', [0.350, 0.369, 0.275, 0.207], 0.79, 'left'), ('blue truck #1', [0.027, 0.365, 0.272, 0.208], 0.68, 'right'), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176], 0.15, None), ('bird #1', [0.485, 0.054, 0.186, 0.130], 0.15, 'front')]
- Example 3
- User prompt: An oil painting of a pink dolphin jumping on the left of a steam boat on the sea
- Current Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194], 0.76, 'front'), ('pink dolphin #1', [0.027, 0.324, 0.246, 0.160], 0.23, 'left'), ('blue dolphin #1', [0.158, 0.454, 0.376, 0.290], 0.26, 'right')]
- Reasoning: The prompt mentions only one dolphin, but two dolphins are present. The pink dolphin already matches the prompt, so remove the extra blue dolphin while keeping the matched objects unchanged.
- Updated Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194], 0.76, 'front'), ('pink dolphin #1', [0.027, 0.324, 0.246, 0.160], 0.23, 'left')]
- Example 4
- User prompt: An oil painting of a pink dolphin jumping on the left of a steam boat on the sea
- Current Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194], 0.76, 'front'), ('dolphin #1', [0.027, 0.324, 0.246, 0.160], 0.23, 'left')]
- Reasoning: The prompt specifies a pink dolphin, but the current object is a generic dolphin without the required pink attribute. Therefore, modify the dolphin's attribute while keeping its position, depth, and orientation unchanged.
- Updated Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194], 0.76, 'front'), ('pink dolphin #1', [0.027, 0.324, 0.246, 0.160], 0.23, 'left')]

- Example 5

User prompt: a backpack on the right of a car from the car's perspective and a car on the left

Current Objects: [('backpack #1', [0.302, 0.293, 0.335, 0.194], 0.63, None), ('car #1', [0.027, 0.324, 0.246, 0.160], 0.25, 'left')]

Reasoning: The prompt specifies that a backpack is on the right of a car from the car's perspective. There is no specific orientation of the car in the prompt, so use the current car orientation, which is 'left'. When the car faces left relative to the camera, its intrinsic right corresponds to behind the car from the camera's perspective. Therefore, the backpack should have a lower depth than the car. The current backpack depth (0.63) is higher than the car depth (0.25), so the depth relation does not match the prompt. Swap the depths of the car and backpack while keeping their bounding boxes and orientations unchanged.

Updated Objects: [('backpack #1', [0.302, 0.293, 0.335, 0.194], 0.25, None), ('car #1', [0.027, 0.324, 0.246, 0.160], 0.63, 'left')]
- Example 6

User prompt: a cat is on the left and the cup is on the right of the cat from the cat's view

Current Objects: [('cat #1', [0.169, 0.563, 0.323, 0.291], 0.901, 'right'), ('cup #1', [0.59, 0.186, 0.408, 0.814], 0.732, None)]

Reasoning: The prompt specifies that the cat is on the left, which is already satisfied. The prompt also specifies that the cup is on the right of the cat from the cat's view. The prompt does not specify the cat's orientation, so use the current orientation, which is 'right'. When the cat faces right relative to the camera, its intrinsic right corresponds to in front of the cat from the camera's perspective. Therefore, the cup's depth must be greater than the cat's depth. The cat currently has depth 0.901 while the cup has depth 0.732. To make the minimal modification, keep the cat unchanged and increase the cup's depth to a value greater than 0.901.

Updated Objects: [('cat #1', [0.169, 0.563, 0.323, 0.291], 0.901, 'right'), ('cup #1', [0.59, 0.186, 0.408, 0.814], 0.95, None)]
- Example 7

User prompt: A cow is in front of a sheep from the camera angle. The sheep is facing right relative to the camera.

Current Objects: [('cow #1', [0.354, 0.365, 0.285, 0.385], 0.41, None), ('sheep #1', [0.608, 0.120, 0.285, 0.200], 0.82, 'right')]

Reasoning: The prompt specifies that a cow is in front of a sheep from the camera angle. Therefore, the cow's depth should be greater than the sheep's depth. However, the cow depth (0.41) is lower than the sheep depth (0.82), which does not match the prompt. Swap the depths of the cow and sheep while keeping all other properties unchanged.

Updated Objects: [('cow #1', [0.354, 0.365, 0.285, 0.385], 0.82, None), ('sheep #1', [0.608, 0.120, 0.285, 0.200], 0.41, 'right')]
- Example 8

User prompt: A fire hydrant is back of a sheep from the sheep's perspective. The sheep is facing left relative to the camera.

Current Objects: [('fire hydrant #1', [0.113, 0.365, 0.251, 0.251], 0.64, None), ('sheep #1', [0.608, 0.120, 0.251, 0.251], 0.52, 'left')]

Reasoning: The prompt specifies that a fire hydrant is behind a sheep from the sheep's perspective. Since the sheep is facing left relative to the camera, the sheep's intrinsic back corresponds to the camera's right direction. Therefore, the fire hydrant should be to the right of the sheep. The fire hydrant is currently to the left of the sheep, so reposition only the fire hydrant horizontally to the right of the sheep while preserving its size, depth, orientation, and approximate vertical position.

- Updated Objects: [('fire hydrant #1', [0.700, 0.365, 0.251, 0.251], 0.64, None), ('sheep #1', [0.608, 0.120, 0.251, 0.251], 0.52, 'left')]
- Example 9
 User prompt: A cow is to the left of a horse from the horse's perspective. The horse is facing right relative to the camera.
 Current Objects: [('cow #1', [0.113, 0.365, 0.352, 0.352], 0.83, None), ('horse #1', [0.608, 0.120, 0.352, 0.352], 0.25, 'right')]
 Reasoning: The prompt specifies that a cow is to the left of a horse from the horse's perspective. Since the horse is facing right relative to the camera, the horse's intrinsic left corresponds to behind the horse from the camera's perspective. Therefore, the cow's depth should be lower than the horse's depth. However, the cow depth (0.83) is higher than the horse depth (0.25), which does not match the prompt. Swap the depths of the cow and horse while keeping all other properties unchanged.
 Updated Objects: [('cow #1', [0.113, 0.365, 0.352, 0.352], 0.25, None), ('horse #1', [0.608, 0.120, 0.352, 0.352], 0.83, 'right')]
- Example 10
 User prompt: A deer is in front of a car from the car's perspective. The car is facing toward the camera.
 Current Objects: [('deer #1', [0.454, 0.365, 0.285, 0.385], 0.64, None), ('car #1', [0.608, 0.120, 0.285, 0.200], 0.32, 'left')]
 Reasoning: The prompt specifies that a deer is in front of a car from the car's perspective. Since the car is facing toward the camera, the car's intrinsic front corresponds to in front of the car from the camera's perspective. The deer depth (0.64) is higher than the car depth (0.32), which already matches the required spatial relation. However, the current orientation of the car is 'left', while the prompt explicitly specifies that the car is facing toward the camera. Therefore, change only the car's orientation to 'front'.
 Updated Objects: [('deer #1', [0.454, 0.365, 0.285, 0.385], 0.64, None), ('car #1', [0.608, 0.120, 0.285, 0.200], 0.32, 'front')]
- Example 11
 User prompt: A deer is in front of a car from the car's perspective. The car is facing away from the camera.
 Current Objects: [('deer #1', [0.454, 0.165, 0.285, 0.385], 0.42, None), ('car #1', [0.608, 0.620, 0.285, 0.200], 0.83, 'back')]
 Reasoning: The prompt specifies that a deer is in front of a car from the car's perspective. Since the car is facing away from the camera, the car's intrinsic front corresponds to behind the car from the camera's perspective. Therefore, the deer should have a lower depth than the car. The deer depth (0.42) is lower than the car depth (0.83), so the current image already matches the prompt and requires no modification.
 Updated Objects: [('deer #1', [0.454, 0.165, 0.285, 0.385], 0.42, None), ('car #1', [0.608, 0.620, 0.285, 0.200], 0.83, 'back')]
- Example 12
 User prompt: A realistic photo of a scene with a brown bowl on the right and a gray dog on the left
 Current Objects: [('gray dog #1', [0.186, 0.592, 0.449, 0.408], 0.45, 'front'), ('brown bowl #1', [0.376, 0.194, 0.624, 0.502], 0.53, None)]
 Reasoning: The horizontal center of the gray dog is $0.186 + 0.449 / 2 = 0.4105$, while the horizontal center of the brown bowl is $0.376 + 0.624 / 2 = 0.688$. Since the gray dog's center is to the left of the brown bowl's center, the required left-right relation is already satisfied. No modification is needed.

```
Updated Objects: [('gray dog #1', [0.186, 0.592, 0.449, 0.408], 0.45, 'front'), ('brown bowl #1', [0.376, 0.194, 0.624, 0.502], 0.53, None)]
```

Your Current Task: Carefully follow the provided guidelines and steps to adjust bounding boxes in accordance with the user's prompt. Ensure adherence to the above output format.

User prompt: {prompt}
Current Objects: {current_objects}

Listing 6: Prompt for Extract Cognitive Map before Generating Image.

Please also consider performing the cognitive map before generating the image.

Cognitive Map's objective is to identify specific objects from the context, understand the spatial arrangement of the scene, and estimate the center point of each object, assuming the entire scene is represented by a 10x10 grid.

[Rule]

1. Estimate the center location of each instance within the provided categories, assuming the entire scene is represented by a 10x10 grid.
2. If a category contains multiple instances, include all of them.
3. Each object's estimated location should accurately reflect its real position in the scene, preserving the relative spatial relationships among all objects.

Still, you need to generate the real image, not a cognitive map.

Listing 7: Prompt for LLM to humanize the given context.

Instruction:
You rewrite spatial-reasoning prompts to sound more natural and varied while preserving all spatial facts.

Rules:

- Preserve every object, relation, perspective, and facing direction.
- Do not add new objects, remove objects, or change object order.
- Do not change the meaning of relative, intrinsic, camera, image, or object-perspective relations.
- Keep the prompt concise and image-generation friendly.
- Return only valid JSON.

Rewrite the initial prompt into a more natural human-written prompt.

Return this JSON object:

```
{{
  "humanized_prompt": "string",
  "preserved": true,
  "notes": "short explanation of what changed"
}}
```

Initial prompt:
{prompt}

Structured relation info:
{relation_info}

Listing 8: Prompt for classifying frame of reference of given context used in the FoR Interpreter routing experiment.

Your Role: Expert on spatial relation frame-of-reference classification

Objective

```

Interpret the prompt and decide whether the spatial relation is intrinsic
or relative.

## Key Guidelines
1. Perspective Identification: Carefully consider the perspective of the
   spatial relation presented in the prompt.
2. Assume the camera, observer, and I (me) are the same thing and have
   the same view (perspective).
3. If the prompt refers to an object's own perspective, classify it as
   intrinsic.
4. If the prompt refers to the camera, observer, or my perspective,
   classify it as relative.
5. If there is at least one intrinsic spatial relation, the sentence is
   intrinsic.
6. Return only the requested tag format and do not add extra labels.

## Process Steps
1. Read and understand the user prompt (scene description).
2. Identify the perspective of each spatial relation presented in the
   given prompt.
3. Decide whether each relation is intrinsic or relative.
4. If there is at least one intrinsic relation, classify the overall
   prompt as intrinsic; otherwise, classify it as relative.
5. Explain your reasoning and then provide the final category.
6. Do not modify the prompt or the object specification.

## Examples

- Example 1
  User prompt: A fire hydrant is back of a cat relative to observer. The
  cat is facing away from the observer.
  Reasoning: The spatial relation is relative to the observer, so this is
  relative.
  Output: <reasoning> The spatial relation is relative to the observer,
  so this is relative. </reasoning> <answer> relative </answer>

- Example 2
  User prompt: A bird is inside the car and left of the car from the car's
  perspective.
  Reasoning: The left relation is described from the car's own
  perspective, so this is intrinsic.
  Output: <reasoning> The left relation is described from the car's own
  perspective, so this is intrinsic. </reasoning> <answer> intrinsic
  </answer>

- Example 3
  User prompt: A phone is on the left of a tablet from my perspective.
  Reasoning: The relation is from an outside observer's perspective, so
  this is relative.
  Output: <reasoning> The relation is from an outside observer's
  perspective, so this is relative. </reasoning> <answer> relative </
  answer>

Your Current Task: Follow the steps closely and accurately classify the
spatial relation in the given prompt. Return the result in the exact
format: <reasoning> Reasoning </reasoning> <answer> intrinsic/
relative </answer>

User prompt: {prompt}

```

Listing 9: Prompt for generated rules used in FoR Interpreter.

```

# Your Role: Expert Perspective Conversion Rule Generator

## Objective

```

```

Generate deterministic perspective-conversion rules for spatial reasoning

The rules should convert object-centric spatial relations into equivalent
camera-centric spatial relations.

## Perspective Conversion Specifications
1. Camera-centric perspective: The spatial relation is interpreted from
   the image/camera view.
2. Object-centric perspective: The spatial relation is interpreted from
   the relatum/object's own facing direction.
3. Relatum: The reference object whose perspective is used.
4. Target: The object being located relative to the relatum.
5. Conversion rule: A mapping from an object-centric relation and a
   relatum facing direction to a camera-centric relation.

## Key Guidelines
1. Relation Alignment: Follow the given relation set exactly. Do not
   invent new relations.
2. Facing Direction Alignment: Follow the given facing direction set
   exactly. Do not invent new facing directions.
3. Deterministic Mapping: Each relation-facing pair must map to exactly
   one camera-centric relation.
4. Complete Coverage: Generate one rule for every combination of relation
   and facing direction.
5. Extractability: The output must be directly parseable using Python
   with json.loads().
6. Minimal Output: Return only the structured JSON object.
7. Use the example rules below as valid demonstrations of the expected
   mapping style and wording.

## Process Steps
1. Interpret Inputs: Read the given relation set and facing direction set
2. Generate Mappings: For each relation-facing pair, infer the equivalent
   camera-centric relation.
3. Explain Each Rule: Write a concise natural-language rule explaining
   the conversion.
4. Output the Result: Present the final rules in the required JSON format

## Valid Example Rules

The following grouped dictionary is a valid example of perspective-
conversion rules.
Use it as the reference for both mapping behavior and wording style.

Example Relations:
["left", "right", "front", "back"]

Example Facing Directions:
["front", "forward-left", "left", "backward-left", "back", "backward-
right", "right", "forward-right"]

Example Rule Dictionary:
{
  "front": [
    "If the object is facing toward the camera (front), then the left
    side of the object is on the right from the camera
    perspective.",
    "If the object is facing toward the camera (front), then the
    right side of the object is on the left from the camera
    perspective.",
    "If the object is facing toward the camera (front), then the
    front side of the object is in the front direction from the
    camera perspective.",
  ]
}

```

```
        "If the object is facing toward the camera (front), then the back
        side of the object is in the back direction from the camera
        perspective."
    ],
    {remaining_examples}
}

Important:
- This example is provided in grouped dictionary format only as a
  demonstration.
- Your output must NOT use the grouped dictionary format.
- Your output must use the structured JSON schema with the top-level key
  "rules".
- Each string in the example corresponds to one structured rule object.

## Output Format
Return ONLY valid JSON.
Do not include markdown.
Do not include comments.
Do not include trailing commas.
Do not include any explanation outside the JSON object.

The JSON must follow exactly this schema:

{
  "rules": [
    {
      "object_perspective_relation": "<relation>",
      "relatum_facing_direction": "<facing_direction>",
      "camera_perspective_relation": "<converted_relation>",
      "rule_text": "If the object is facing <facing_description>, then
        the <relation> side of the object is <
        converted_relation_description> from the camera perspective."
    }
  ]
}

## Constraints
- Include exactly one rule for every relation-facing_direction pair.
- Use only relations from the provided relation set.
- Use only facing directions from the provided facing direction set.
- The value of "camera_perspective_relation" must also be one of the
  provided relations.
- Do not invent new relation names.
- Do not invent new facing directions.
- Do not omit any relation-facing pair.
- Return JSON only.

## Current Task
Relations:
{relations}

Facing directions:
{facing_directions}

Carefully follow the provided guidelines and steps to generate
deterministic perspective-conversion rules.
Return only the JSON object.
```