Thinking in Boxes:
3D Editing in Real Images Made Easy

Pradhaan S Bhat[1*] **Naveen Chandra R**[1*] **Rishubh Parihar**[1] **Vaibhav Vavilala**[2]
R. Venkatesh Babu[1] **D.A. Forsyth**[3] **Anand Bhattad**[4]
[1]Indian Institute of Science [2]Apple [3]UIUC [4]Johns Hopkins University

Project Page: `https://thinking-in-boxes.github.io/`

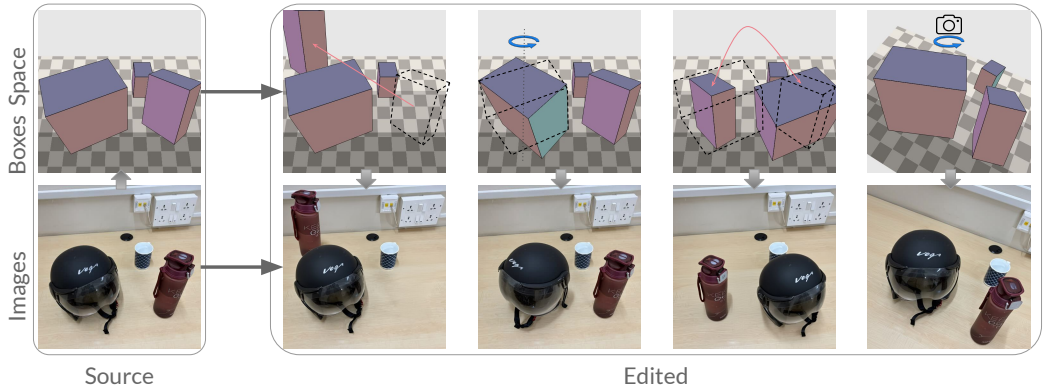

Figure 1: **Thinking in boxes.** Given a single source image (bottom left), the user fits 3D boxes around objects of interest, anchored on a depth-aligned floor (top left). Editing the boxes drives the corresponding edit in the image: from left to right, translation, rotation, combined translation and rotation, and a camera viewpoint change. The bottles' and helmets' appearance is preserved across all four edits, including regions of the object not visible in the source — note the back of the bottle and side view of the helmet revealed under rotation. Note that all qualitative figures are real images.

Abstract

Text and 2D-conditioning interfaces provide weak, ambiguous control over spatial transformations in image editing – particularly under large object motions and camera changes. Prior work has used 3D primitives such as boxes, but only as loose conditioning signals indicating approximate object location rather than specifying the transformation. We instead use 3D boxes as structured specifications: the user provides the input and output boxes of the edit, casting editing as a well-posed geometry problem. This "thinking in boxes" interface, where each box face is color-coded to convey 3D orientation, gives precise control over translation, rotation, scaling, and viewpoint changes in real images while preserving scene and object identity, and recovering previously unseen object regions. To ground transformations in scene appearance, we introduce a depth-aligned planar floor as a global reference frame, shaded with depth-aware cues. Conditioned on this structure, an image generator produces consistent results under large transformations. Trained in two stages – on synthetic multi-object scenes and a small set of real-world videos from Objectron – the system generalizes to complex, in-the-wild real images. Our method operates directly on real photographs and substantially outperforms recent state-of-the-art methods on large 3D edits.

*Equal Contribution

Preprint.

1 Introduction

We present a system that lets a user manipulate a simple, abstracted representation of the objects in an image, and turns those manipulations into edits of the image itself. The user specifies an edit by placing a 3D box around an object in the input image and a second box where they want the object to end up; a learned procedure maps this pair of boxes into the internal representation of a generator, producing the edited image. The same procedure handles translation, rotation, occlusion, and viewpoint changes and is learned once for all manipulations – not retrained or optimized per image.

We call this interface "**thinking in boxes**" (Fig. 1): the box pair implicitly defines the desired translation, rotation, and scale in 3D, while its projection tells the generator which regions become newly visible and how the object's silhouette should change under perspective. To ground edits in scene appearance, we further introduce a depth-aligned planar floor, shaded with depth-aware cues to provide relative spatial positioning of objects and background.

Our method is trained in two stages: first on synthetic edit sequences with multiple objects per scene, then on a small set of real-world scenes from Objectron [2]. It generalizes to complex, real-world images, significantly outperforming state-of-the-art methods on large spatial edits and successfully recovering previously occluded object regions never seen in the input. We find the depth-aligned floor to be extremely important: it provides the global reference frame that lets the same procedure handle object motion and camera motion within a single coherent representation, rather than treating them as separate problems or using specialized video models.

Large-scale text-conditioned image generators [33, 63] have become a powerful prior for image editing, controlled through instruction-based prompts [34] or 2D signals like bounding boxes [72] and drag points [52]. These interfaces are coarse for spatial edits. Text prompts cannot specify how far an object should move [36] and a 2D box cannot disambiguate between translation, rotation, and camera motion. For spatial edits – moving objects through 3D space – text and 2D interfaces access only a fraction of the degrees of freedom the task requires.

Several recent methods aim to fill this gap, each with significant tradeoffs in the choice of input representation. One line uses per-object latent representations [64, 18] that work well in restricted domains – driving scenes, household objects – but transfer poorly to in-the-wild photos. A second line uses depth as the 3D representation, lifting diffusion activations or attention maps to 3D via the depth proxy [50, 42, 5]; this supports 3D edits on real images without retraining, but the depth-only representation is brittle under large transformations and significant disocclusion, and the methods require per-image inversion or optimization. A third line uses detailed 3D primitives – meshes or convex blocks [60, 14] – applies the edit in image or depth space, and uses diffusion as a final cleanup pass; this paradigm has so far been shown on generated images rather than real photographs [60]. We propose a different choice. Boxes at the object level and a depth-aligned floor at the scene level forms a minimal representation – the user only places and moves boxes – and structurally complete: boxes encode object pose and visibility, the floor encodes the scene geometry and disambiguates the object from the camera motion. Given that representation, a generator trained once on the source–target pair renders large 3D edits, including substantial rotations and disocclusions, on in-the-wild photographs.

2 Related Work

3D Control in Diffusion Models. Recently, several methods [1, 43, 5] have been proposed to enable 3D control in text-to-image generation. LooseControl [5] leverages the depth of 3D bounding boxes and trains a ControlNet [73] to condition for 3D-aware generation. Build-A-Scene [17] iteratively applies this process to perform interactive 3D Layout Control. Another major direction of work is to condition the text-to-image diffusion model on individual 3D properties, such as camera control [12] or object orientations [43, 30, 38]. These methods encode the 3D representation into a text embedding space to condition the generator. More recently, SeeThrough3D [1] demonstrated occlusion-aware 3D layout control by designing a translucent 3D box layout for text-to-image generation. Generative blocks world [60] represent scene elements as 3D convex primitives and enable control during generation by geometrically manipulating primitives. Both SeeThrough3D and Generative blocks world are limited to controlling layouts of generated images and do not work with real images.

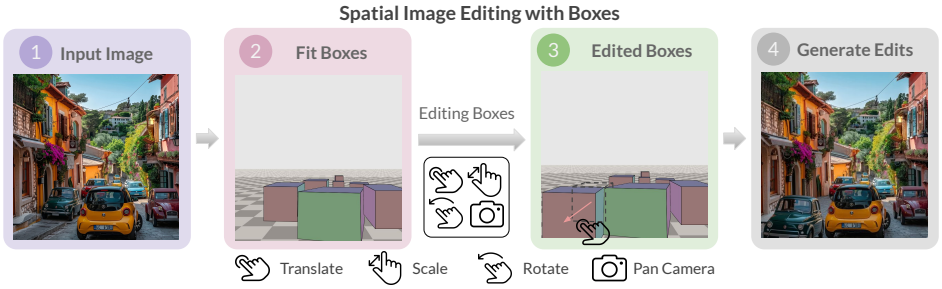

Figure 2: **Editing Pipeline.** (1) Users provide a real source image and (2) fit 3D boxes to the objects within the scene using a point-and-click interface. (3) The boxes can be manipulated in 3D space, allowing for scaling, rotation, translation, and camera moves. Both the source and target box layouts are projected into 2D and serve, alongside the source image, as inputs to an image editing model [34]. (4) The model generates an edit that respects the underlying scene geometry and follows the user's layout. In this example, the car on the left is moved forward with the rest of the scene preserved.

Image Editing with Diffusion Models. Recent large text-to-image diffusion models have enabled high-quality image editing [24, 6] thanks to their rich image generation priors. Inference time methods involve inverting real images into the latent space using diffusion inversion [55, 29, 40], and then editing is performed by manipulating the cross-attention maps [24, 7, 70, 4] during denoising. More recently, training-based methods fine-tune a pretrained diffusion model for instruction-based image editing [6, 56, 57, 76, 63], scene relighting [35], material control [51, 13], or scene composition [10, 68]. For fine-grained editing, continuous control has been explored to smoothly control individual semantic attributes [22, 28, 44]. Another approach aims to personalize the generation given one or a few images of a subject and enable generation in different compositions with text prompts [21, 49, 32, 31, 70, 23].

Geometric Editing in Diffusion models. Existing methods for geometric image editing rely on text [66, 37] or drag based control [53, 41, 77, 27] or 3D representations of the objects in the scene, either through meshes [11, 71] or built with depth maps and object masks [42, 79, 50, 45, 9, 75]. However, these representations are not user-friendly, require external models [8, 69, 47, 65] and offer limited 3D control. Furthermore, these methods involve a hole-filling procedure via an inpainting model to fill the holes due to the displacement of objects which can result in inconsistent texture and lighting in the scene. Another line of work leverage supervised learning on video data [14, 3] to perform image editing, however, these methods often require a chain of intermediate models [59, 8, 48, 67] to obtain clean supervisory data which is computationally expensive. To overcome all these issues, we design a representation that encodes 3D layouts in a user-friendly manner. Coarse representations such as 3D boxes and convex primitives serve as an effective representation to model object geometry and enable control over 3D scene layout [1, 64, 18, 60]. Motivated by this, we associate pixels of objects in the scene with an editing-friendly cuboid representation for any scene.

3 Method

We frame image editing as a transformation between two states of a structured scene representation (Fig. 2). Given a source image I_{src}, the user lifts it into a set of 3D boxes, edits the boxes directly, and the resulting box pair conditions an image generator that produces the edited image.

3D object boxes. We represent each object of interest as a 3D box B_i with position, orientation, and scale. To convey 3D orientation through a 2D conditioning signal, we assign a fixed color to each face: red (front), green (back), pink (right), cyan (left), blue (top), and yellow (bottom). The visible face colors in the projected box encode which side of the object faces the camera, independent of object identity or appearance. This color code is canonical across all scenes and is the only mechanism by which the generator reads object orientation.

Scene layout. The input to our method – a set of boxes in two configurations – has to tell the generator unambiguously what edit to perform. The mapping from the source configuration to the target configuration must have a single interpretation. Boxes alone do not satisfy this: a leftward shift of every box can equally indicate that the objects moved left or that the camera panned right, and the generator has no way to choose. We resolve the ambiguity by anchoring the boxes in a

shared coordinate frame – a depth-aligned planar floor, rendered as a checkerboard with depth-aware shading. The floor moves with the camera and stays fixed under object motion, so any change in the relative configuration of boxes and floor uniquely identifies what moved. It also provides a global reference for contact and shadow. Both boxes and the floor form the *3D scene layout* L_{src}, which we project to 2D as the spatial conditioning image used by the generator

Fitting boxes to images. The user fits boxes to objects through a point-and-click interface (Fig. 2). The floor is estimated automatically from the image, so the user only specifies object boxes – they never author the floor explicitly. To reduce manual effort further, off-the-shelf 3D box detectors [26] produce an initial set of boxes that the user refines.

Editing boxes. The user manipulates the boxes in 3D to specify the desired edit. We support four operations, all expressed as direct manipulation of the same boxes. *Translation* moves a box to a new position while preserving its orientation and scale. *Scaling* resizes a box in place. *Rotation* re-orients a box so that a different colored face is visible to the camera – e.g., turning a chair from frontal to side view rotates the red face out and the pink or cyan face in. *Camera moves* re-orient all boxes and the floor jointly with respect to a new viewpoint. The result is a target layout L_{tgt}, projected to 2D as the target spatial conditioning image. All four edit types are expressed in the same box language; the generator does not need separate handling for translation, rotation, or camera change.

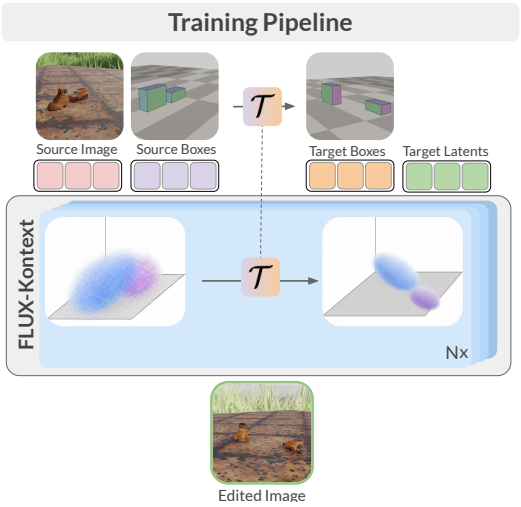

Figure 3: **Training pipeline.** A training example provides a source image, a noisy target image, and the source and target box projections that specify the transformation $\mathcal{T}$ between them. All four are encoded by the VAE and concatenated spatially. Joint attention inside FLUX-Kontext applies $\mathcal{T}$ to the image latents – mapping the source latent to the target latent under box-pair conditioning. LoRA layers in the attention matrices enable box-conditioned editing while preserving the base model's prior.

Editing images. We build upon FLUX-Kontext [34, 19] image editor that operates on multimodal token streams. The image $Isrc$, projected source layout $Lsrc$ and projected target layout L_{tgt} are first encoded into latent tokens using the VAE and then concatenated along the spatial dimension [76, 1]. All three streams have the same positional encoding, so there is an alignment between image region and the box that covers it. Internally to MMDiT, each stream attends to the others. Image tokens attend to source layout tokens to associate appearances with geometry and to target layout tokens to spatialize appearance in the output. We then decode a single stream of tokens as output representing the edited image.

Training. We fine-tune FLUX-Kontext with LoRA [25] layers injected into the attention matrices, leaving the rest of the model frozen. This adapts the base editor to consume the new box-conditioning streams while preserving its strong generative prior.

4 Dataset

For our training, we first create a synthetic dataset of paired views, with same objects existing in two separate 3D positions, and a pair of views that share these objects. Each pair of views is completely calibrated; the transformation from one view's boxes to the other's is provided as supervision.

Scene construction. Each scene contains two objects placed on a planar floor under a sampled HDRI. Objects are drawn from the Objaverse [15] pool, normalized to a common size, and randomly oriented on the ground under a non-overlap constraint. We render the scene from two camera viewpoints. Between the two views, the objects undergo a controlled perturbation – a fresh rotation, a uniform rescaling, and a translation – so that the second view contains the same objects in a transformed configuration. The pair therefore supervises both *what edit was made* (the box-pair transformation) and *what the resulting image should look like* (the second view). Two objects per scene is the smallest setting that exercises inter-object occlusion and contextual edits without overwhelming training cost.

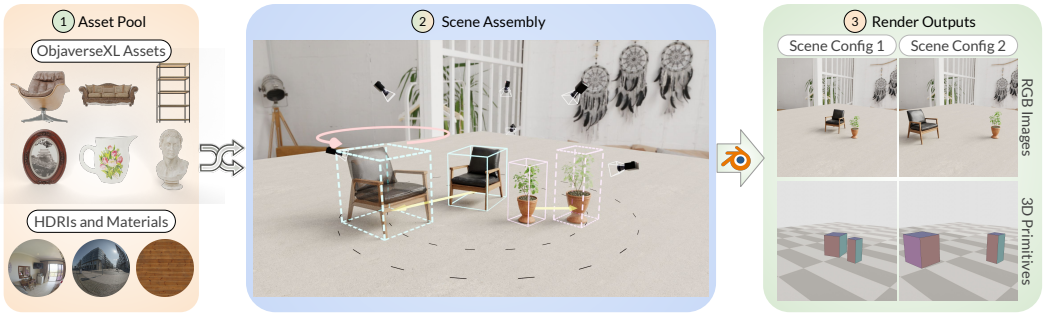

Figure 4: **Dataset rendering pipeline. (1)** 3D assets are drawn from a subset of Objaverse-XL [15]; HDRIs and floor textures are taken from 3D-Fixer [71]. **(2)** Each scene places [N] scale-normalized objects on a textured floor under a sampled HDRI, with collision-aware placement. We then perturb the objects in rotation $[0, 2\pi]$, scale $[0.5, 1.5]$, and position to produce a second scene configuration. **(3)** Blender Cycles renders both configurations, producing paired RGB images and color-coded 3D box visualizations along with camera intrinsics, extrinsics, and per-object material parameters.

Renders and metadata. For each view we render a photorealistic RGB image at 512×512 and a corresponding oriented-box overlay, where the visible faces of each object's 3D box are color-coded by their canonical orientation. These two form the input–conditioning pair used during training.

Scale. The dataset comprises 110,000 scenes (220,000 views) drawn from 10,143 unique 3D objects, paired with 1,154 HDRIs and 5,760 floor materials. Additional 10,000 image pairs are taken from the Objectron [2] dataset for finetuning. Additional details are in the appendix.

5 Experiments

Implementation details: We use Flux-Kontext [34] as our base image editing model. We initially train for 50K steps on a synthetic rendered dataset and then train for another 10K steps on a hybrid dataset of 10K real samples obtained from objectron and 10K synthetic set. Additional implementation details can be found in the appendix.

Evaluation Dataset We evaluate on two settings: *object editing* (objects are moved while camera is fixed) and *camera editing* (camera moves while objects stay fixed). Each setting is tested on both synthetic and real data.

Synthetic. Both settings use synthetic test sets rendered with the same pipeline as our training data (Sec. 4), held out from training. For object-editing pairs, we apply random translations and rotations to objects; for camera-editing pairs, we apply random viewpoint changes with objects fixed.

Real. For object editing, we use WildDet-3D [26], which provides 3D bounding box annotations on in-the-wild images. We construct test pairs by applying random translation and rotation perturbations to each labeled box, constraining the perturbed box to remain within the image frame. For camera editing, we use Objectron [2] samples held out from training, whose video sequences naturally provide viewpoint changes; we sample frame pairs from each video and use the provided per-frame 3D bounding boxes to construct the source and target layouts.

Baselines: For object manipulation (Fig 8), we compare against SAM-3D [11], 3D-Fixer [71], and SpatialEdit [66], along with three training-free baselines: GeoDiffuser [50], Diffusion Handles [42], and FreeFine [79]. For camera editing, we compare against SpatialEdit [66], SEVA [78], and Qwen-Camera-Control [63]. We additionally compare qualitatively against GBW [60] in Fig 9, which operates only on generated images, on the failure cases reported in their paper.

5.1 Qualitative Results

We present qualitative results on both real and in-the-wild scenes in Fig. 5 and Fig. 7. Our method closely adheres to the targeted layout changes and produces realistic edits across a wide range of inputs. Despite being trained predominantly on synthetic data rendered in Blender and fine-tuned on only a small amount of real-world data, our model generalizes effectively to real scenarios, preserving overall scene integrity as well as the fine-grained textures and details of both the edited object and its surroundings. As shown in row 3 of Fig. 7, the model maintains coherent scene structure even when

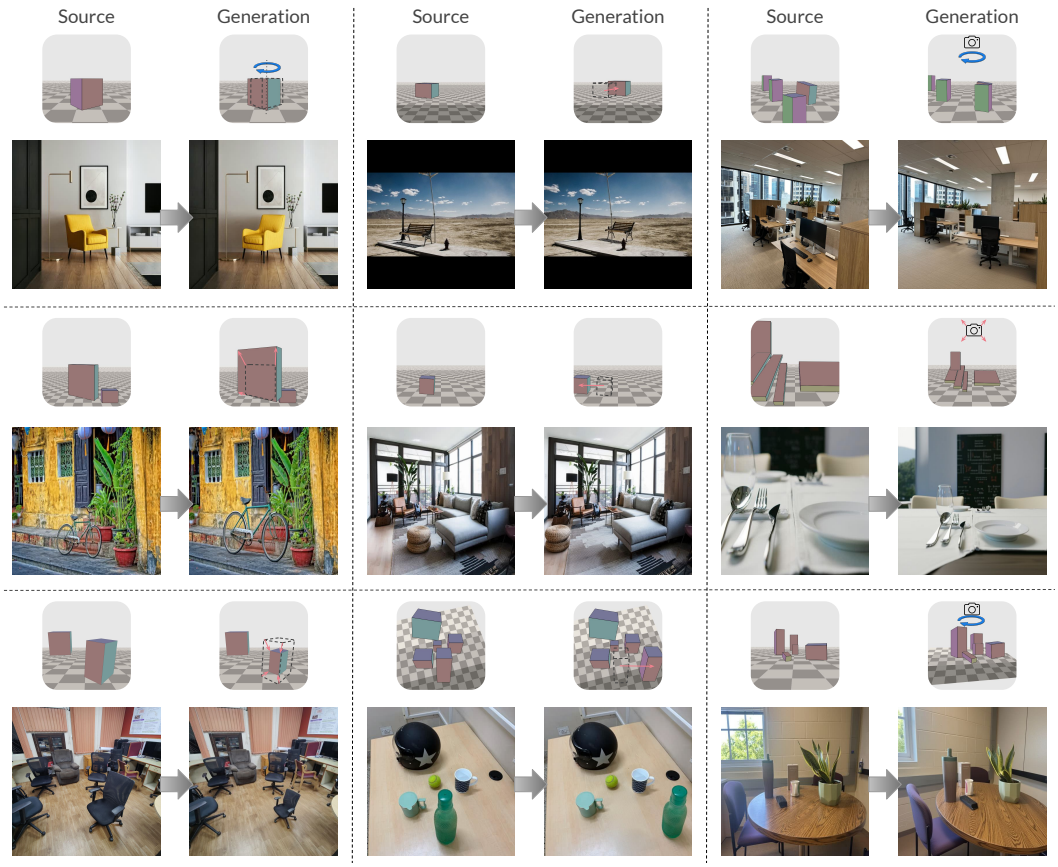

Figure 5: **In-the-wild results.** Nine real-image edits across diverse indoor and outdoor scenes (offices, dining rooms, street scenes, desert, living rooms). Each panel shows the source image with its fitted boxes (top-left inset), the user's edited boxes (top-right inset), and the resulting generation. Edits include translation, rotation, scaling, and camera viewpoint changes; the same model handles all four operations. Scene context, lighting, and occluded regions are preserved consistently across edits, with no per-image optimization or fine-tuning. Notice the back of the chair in bottom right figure.

multiple edits are applied simultaneously. Row 1 of the same figure further demonstrates that the model accurately follows edit instructions on a sketch-style input, despite this domain being entirely out of distribution.

5.2 User Study

We conducted an A/B user study with 49 participants, each asked to select the preferred output from randomly sampled generations produced by our method and three baselines: FreeFine [79], SAM3D [11], and SpatialEdit [66]. We evaluate a) object preservation, b) background preservation, and c) layout following. Results highlight high preference rates for our method in all evaluation categories (see Fig.6)

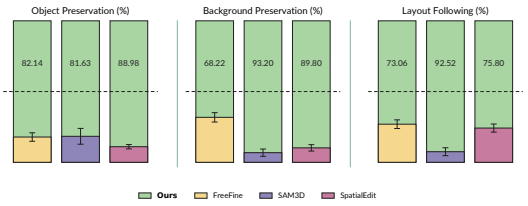

Figure 6: **User Study.** Each bar represents user preference rate for our method compared to baselines, categorized by specific evaluation criteria.

5.3 Metrics

Following FreeFine [79], we evaluate each baseline's edited output I_e against the ground-truth target I_{gt} using eight metrics that span image quality, region-localized consistency, and edit fidelity. For image quality, we report PSNR $\uparrow$, SSIM [62] $\uparrow$, LPIPS [74] $\downarrow$, and DreamSim [20] $\downarrow$. For region-localized consistency, we measure subject and background consistency as masked DINOv3-ViT-B/16 [54] feature similarity computed under the foreground mask and its complement, respectively. For edit fidelity, we report four metrics: **Warp Error**, the masked L_1 difference between I_e and

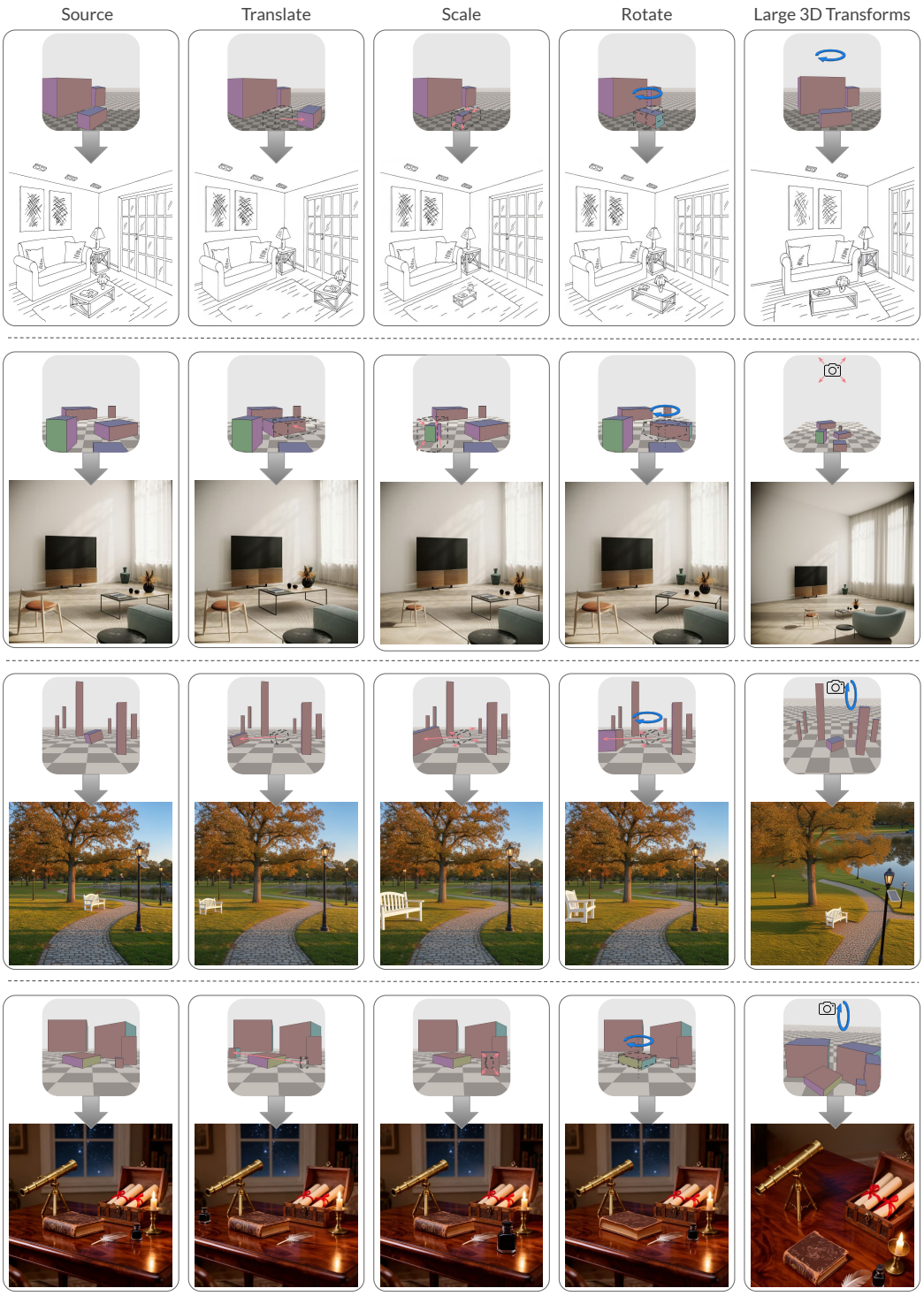

Figure 7: **Scene editing across operations, scenes, and rendering styles.** Each row shows a source image (column 1) edited under four operations: translation, scaling, rotation, and a large combined transformation (columns 2–5). The same trained model handles all operations across all scenes – including the line drawing in row 1, despite training only on photographic data. The model preserves rendering style under the edit (row 1 stays a line drawing) and respects scene-level effects that prior methods typically destroy: shadow and lighting consistency under viewpoint change in the autumn park (row 3), and fire and metallic reflections on polished surfaces in the table with book scenes (row 4). Best viewed on screen and zoom in to see all edits clearly.

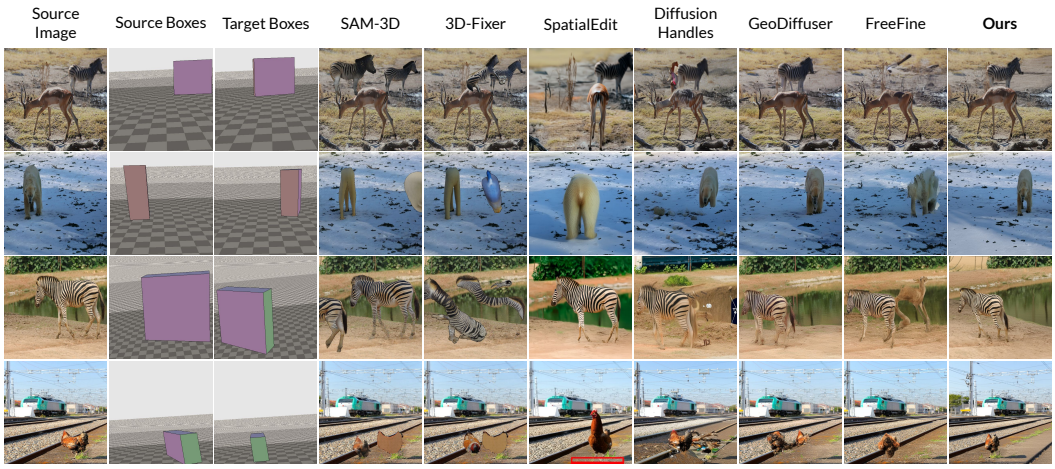

Figure 8: **Qualitative comparison on real images.** For each example, the user specifies an edit by editing 3D boxes (columns 2–3), which we render to 2D as the target conditioning input for our method and convert to the appropriate input format for each baseline (mask, depth, control points). The examples shown are deliberately hard: every subject is a non-rigid animal (zebras, impala, polar bear, chicken), and the edits include rotations and disocclusions. These cases break the assumptions of methods that fit 3D meshes or warp pixels via depth, since deformable objects do not admit clean rigid transformations in image space. Baselines [11, 71, 66, 42, 50, 79] accordingly fail to apply the edit, introduce identity drift, or destroy scene context. Our method (rightmost column) handles the same edits cleanly, recovering object regions not visible in the input without any hallucinations.

a reference target; **Mean Distance**, the pixel error between DIFT [58] semantic correspondences computed from source-to-I_e and source-to-reference; **IoU**, the intersection-over-union between the SAM3 [8] mask of the generated object and the target bounding box; and **Angular Error**, between target and generated orientations.

5.4 Object Control in Real-World Scenes

We report quantitative metrics against both 3D editing and image editing baselines. Results are shown in Tab. 1 and Tab. 2. Our method significantly outperforms prior work, including SAM3D [11], 3D-Fixer [71], SpatialEdit [66], DiffusionHandles [42], GeoDiffuser [50], and FreeFine [79]. The edits are temporally consistent across views and geometrically faithful to the target transformation. Our method also respects the intended placement of 3D objects in space. On real object editing (Tab. 1), our method ranks first or second on every metric. When second, it remains close to the top baseline. When first, it leads by a clear margin particularly in mean distance error and angular error. On synthetic object editing (Tab. 2), our method outperforms all baselines across every metric. Qualitative comparisons are provided in Fig. 8. Additional results are discussed in the appendix.

Figure 9: **GBW's reported failure cases.** This is an example GBW [60] flags as difficult for their method. Our box-space conditioning handles it, while their depth-and-warp convex primitive pipeline does not. Source (left), GBW (middle), ours (right).

Table 1: Quantitative comparison on *object editing* using WildDet-3D [26] data.

Method	Consistency (DINO)		Edit fidelity			
	Subject ↑	Background ↑	Warp Error ↓	Mean Distance ↓	IoU ↑	Angular Error ↓
SAM3D [11]	0.632	0.956	0.175	79.087	0.356	99.239
3D-Fixer [71]	0.570	0.942	0.165	74.294	0.361	108.762
SpatialEdit [66]	0.581	0.884	0.234	107.438	**0.399**	98.171
DiffusionHandles [42]	0.631	0.924	0.154	<u>32.634</u>	0.369	95.659
GeoDiffuser [50]	**0.646**	0.953	0.143	35.488	0.373	<u>94.824</u>
FreeFine [79]	0.623	**0.956**	<u>0.143</u>	42.125	0.382	98.705
Ours	<u>0.637</u>	<u>0.955</u>	**0.136**	**25.910**	<u>0.382</u>	**92.051**

Table 2: Quantitative comparison on *object editing* using synthetic test set.

Method	Image quality				Consistency (DINO)		Edit fidelity			
	PSNR ↑	SSIM ↑	LPIPS ↓	DreamSim ↓	Subject ↑	Background ↑	Warp Error ↓	Mean Distance ↓	IoU ↑	Angular Error ↓
SAM3D [11]	20.415	0.810	0.208	0.195	0.769	0.931	0.173	44.557	0.374	123.838
3D-Fixer [71]	20.822	0.812	0.203	0.207	0.724	0.929	0.170	47.110	0.368	126.193
SpatialEdit [66]	12.194	0.433	0.601	0.391	0.608	0.792	0.262	100.287	0.274	128.287
DiffusionHandles [42]	18.717	0.693	0.366	0.347	0.651	0.828	0.146	55.535	0.357	125.289
GeoDiffuser [50]	20.654	0.740	0.275	0.282	0.668	0.876	0.146	57.451	0.365	124.215
FreeFine [79]	20.852	0.777	0.226	0.222	0.698	0.905	0.145	55.997	0.359	122.606
Ours	**23.686**	**0.821**	**0.130**	**0.092**	**0.879**	**0.964**	**0.101**	**21.392**	**0.534**	**76.739**

5.5 Ablations

We ablate design choices of our box conditioning: (1) the **full setup (ours)**, in which objects are placed on a checkered-pattern floor and the conditioning boxes are rendered from different viewpoints with directional color-coded faces; (2) **no floor**, where the checkered floor is removed and only the boxes are rendered, leaving the model without an explicit spatial reference for the scene layout; and (3) **uniform box color**, where all faces of the conditioning box share a single color, removing the directional cue that encodes object orientation.

All three variants are trained on the same $10K$ pairs with identical object placements and lighting; only the floor and box appearance differ. Each model is evaluated on 500 test samples, with quantitative results reported in Table 3

Table 3: Ablation on real-set camera edits.

Variant	Subj.↑	Bg.↑	Warp↓	MeanDist↓	IoU↑	Angular.↓
nofloor	0.562	0.829	0.174	47.667	0.381	**102.667**
samecolor	0.568	**0.882**	0.169	57.618	**0.399**	105.472
original	**0.574**	0.870	**0.153**	**42.616**	0.389	103.771

Effect of the checkered floor. Without the checkered floor, the model loses a key cue for localizing objects relative to their surroundings, leading to noticeably worse position preservation. We attribute this to the floor pattern providing a consistent global reference frame that anchors the model's sense of spatial layout.

Effect of directional box coloring. Replacing the color-coded box faces with a uniform color degrades orientation accuracy: the model can still localize the object, but tends to misread its facing direction. This confirms that the per-face coloring is the primary signal through which orientation information is injected.

Visualizations demonstrating effectiveness of providing both the checkered floor and the direction specific box colors are shown in the appendix. Together, these ablations show that the floor pattern and directional box coloring contribute complementary signals, the global position and local orientation respectively, and that both are necessary to achieve the full performance of our method.

6 Conclusion and Limitations

We propose an effective and intuitive primitive based image editing framework that enables 3D aware scene edits such as object translation, rotation and camera control. Given an input image, we fit 3D box primitives on individual objects and place the primitives on a floor with checkered pattern. This primitive based representation can be edited easily by dragging or scaling the primitive boxes. Training an image editing model on the source image, its primitives and target primitives results in precise spatial control of the scene objects. To train this model, we generate a synthetic dataset where 3D assets are placed in a rendering engine and captured from different viewpoints in various configurations. Though trained on synthetic dataset, our method works exceptionally well for real world image editing and performs accurate 3D aware edits. This suggests that the diffusion editing models inherently have object aware representations that can be exposed as a simple user interface with small fine-tuning. This opens up new direction for understanding representations inside the foundation image editing models and how to expose them for downstream tasks.

Limitation. When objects share similar scales, their bounding boxes become indistinguishable. This creates prompt ambiguity, as the final state could represent several different transformations. Consequently, the model fails in these cases, producing only the identity transformation (see Fig. 10). Moreover, a common limitation of current 3D-editing frameworks is the inability to perfectly isolate modifications. The editing process frequently introduces unintended artifacts and minor pixel-level alterations to the background, failing to maintain strict background consistency. Addressing these unintended background alterations is a highly valuable direction for future research.

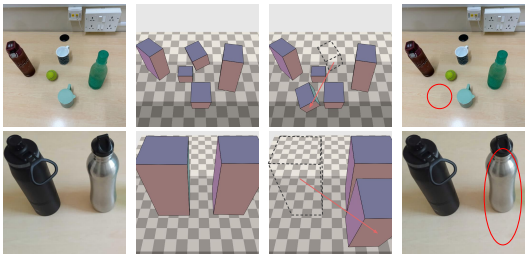

Figure 10: **Limitation:** Our model fails when multiple objects could plausibly satisfy the target layout. In both examples, either object could be moved to match the target boxes (small objects, top row; large objects, bottom row), and the model refuses to commit to one – producing an identity transformation instead.

Acknowledgement. This work is supported by PMRF by Govt. of India.

References

[1] Vaibhav Agrawal, Rishubh Parihar, Pradhaan Bhat, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. Seethrough3d: Occlusion aware 3d control in text-to-image generation. In *CVPR*, 2026.

[2] Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7822–7831, 2021.

[3] Hadi Alzayer, Zhihao Xia, Xuaner Zhang, Eli Shechtman, Jia-Bin Huang, and Michael Gharbi. Magic fixup: Streamlining photo editing by watching dynamic videos. *ACM Transactions on Graphics*, 44(5):1–25, 2025.

[4] Omri Avrahami, Or Patashnik, Ohad Fried, Egor Nemchinov, Kfir Aberman, Dani Lischinski, and Daniel Cohen-Or. Stable flow: Vital layers for training-free image editing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7877–7888, 2025.

[5] Shariq Farooq Bhat, Niloy Mitra, and Peter Wonka. Loosecontrol: Lifting controlnet for generalized depth conditioning. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[6] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18392–18402, 2023.

[7] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22560–22570, 2023.

[8] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

[9] Jiacheng Chen, Ramin Mehran, Xuhui Jia, Saining Xie, and Sanghyun Woo. Blenderfusion: 3d-grounded visual editing and generative compositing. *arXiv preprint arXiv:2506.17450*, 2025.

[10] Xi Chen, Lianghua Huang, Yu Liu, Yujun Shen, Deli Zhao, and Hengshuang Zhao. Anydoor: Zero-shot object-level image customization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6593–6602, 2024.

[11] Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025.

[12] Ta-Ying Cheng, Matheus Gadelha, Thibault Groueix, Matthew Fisher, Radomir Mech, Andrew Markham, and Niki Trigoni. Learning continuous 3d words for text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6753–6762, 2024.

[13] Ta Ying Cheng, Prafull Sharma, Mark Boss, and Varun Jampani. Marble: Material recomposition and blending in clip-space. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13061–13071, 2025.

[14] Yen-Chi Cheng, Krishna Kumar Singh, Jae Shin Yoon, Alexander Schwing, Liang-Yan Gui, Matheus Gadelha, Paul Guerrero, and Nanxuan Zhao. 3d-fixup: Advancing photo editing with 3d priors. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–10, 2025.

[15] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023.

[16] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Knauer, Klaus H. Strobl, Matthias Humt, and Rudolph Triebel. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 8(82):4901, 2023.

[17] Abdelrahman Eldesokey and Peter Wonka. Build-a-scene: Interactive 3d layout control for diffusion-based image generation. *arXiv preprint arXiv:2408.14819*, 2024.

[18] Alejandro Escontrela, Shrinu Kushagra, Sjoerd van Steenkiste, Yulia Rubanova, Aleksander Holynski, Kelsey Allen, Kevin Murphy, and Thomas Kipf. Neural usd: An object-centric framework for iterative editing and control. *arXiv preprint arXiv:2510.23956*, 2025.

[19] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[20] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *arXiv preprint arXiv:2306.09344*, 2023.

[21] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations*.

[22] Rohit Gandikota, Joanna Materzyńska, Tingrui Zhou, Antonio Torralba, and David Bau. Concept sliders: Lora adaptors for precise control in diffusion models. In *European Conference on Computer Vision*, pages 172–188. Springer, 2024.

[23] Daniel Garibi, Shahar Yadin, Roni Paiss, Omer Tov, Shiran Zada, Ariel Ephrat, Tomer Michaeli, Inbar Mosseri, and Tali Dekel. Tokenverse: Versatile multi-concept personalization in token modulation space. *ACM Transactions On Graphics (TOG)*, 44(4):1–11, 2025.

[24] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. 2022.

[25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[26] Weikai Huang, Jieyu Zhang, Sijun Li, Taoyang Jia, Jiafei Duan, Yunqian Cheng, Jaemin Cho, Mattew Wallingford, Rustin Soraki, Chris Dongjoo Kim, et al. Wilddet3d: Scaling promptable 3d detection in the wild. *arXiv preprint arXiv:2604.08626*, 2026.

[27] Ziqi Jiang, Zhen Wang, and Long Chen. Clipdrag: Combining text-based and drag-based instructions for image editing. In *International Conference on Learning Representations*, volume 2025, pages 5237–5253, 2025.

[28] Ronen Kamenetsky, Sara Dorfman, Daniel Garibi, Roni Paiss, Or Patashnik, and Daniel Cohen-Or. Saedit: Token-level control for continuous image editing via sparse autoencoder. *arXiv preprint arXiv:2510.05081*, 2025.

[29] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6007–6017, 2023.

[30] Nupur Kumari, Grace Su, Richard Zhang, Taesung Park, Eli Shechtman, and Jun-Yan Zhu. Customizing text-to-image diffusion with object viewpoint control. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–13, 2024.

[31] Nupur Kumari, Xi Yin, Jun-Yan Zhu, Ishan Misra, and Samaneh Azadi. Generating multi-image synthetic data for text-to-image customization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16524–16534, 2025.

[32] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1931–1941, 2023.

[33] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[34] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

[35] Nadav Magar, Amir Hertz, Eric Tabellion, Yael Pritch, Alex Rav-Acha, Ariel Shamir, and Yedid Hoshen. Lightlab: Controlling light sources in images with diffusion models. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–11, 2025.

[36] Oscar Michel, Anand Bhattad, Eli VanderBilt, Ranjay Krishna, Aniruddha Kembhavi, and Tanmay Gupta. Object 3dit: Language-guided 3d-aware image editing. *Advances in Neural Information Processing Systems*, 36:3497–3516, 2023.

[37] Oscar Michel, Anand Bhattad, Eli VanderBilt, Ranjay Krishna, Aniruddha Kembhavi, and Tanmay Gupta. Object 3dit: Language-guided 3d-aware image editing. *Advances in Neural Information Processing Systems*, 36:3497–3516, 2023.

[38] Yunhong Min, Daehyeon Choi, Kyeongmin Yeo, Jihyun Lee, and Minhyuk Sung. Origen: Zero-shot 3d orientation grounding in text-to-image generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[39] Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. *arXiv preprint arXiv:2306.06101*, 2023.

[40] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6038–6047, 2023.

[41] Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models. In *International Conference on Learning Representations*, volume 2024, pages 31620–31631, 2024.

[42] Karran Pandey, Paul Guerrero, Matheus Gadelha, Yannick Hold-Geoffroy, Karan Singh, and Niloy J Mitra. Diffusion handles enabling 3d edits for diffusion models by lifting activations to 3d. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7695–7704, 2024.

[43] Rishubh Parihar, Vaibhav Agrawal, Sachidanand VS, and Venkatesh Babu Radhakrishnan. Compass control: Multi object orientation control for text-to-image generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2791–2801, 2025.

[44] Rishubh Parihar, Or Patashnik, Daniil Ostashev, R Venkatesh Babu, Daniel Cohen-Or, and Kuan-Chieh Wang. Kontinuous kontext: Continuous strength control for instruction-based image editing. *arXiv preprint arXiv:2510.08532*, 2025.

[45] Rishubh Parihar, Sachidanand VS, and R Venkatesh Babu. Zero-shot depth aware image editing with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15748–15759, 2025.

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[47] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10106–10116, 2024.

[48] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

[49] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.

[50] Rahul Sajnani, Jeroen Vanbaar, Jie Min, Kapil D Katyal, and Srinath Sridhar. Geodiffuser: Geometry-based image editing with diffusion models. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pages 472–482, 2025.

[51] Prafull Sharma, Varun Jampani, Yuanzhen Li, Xuhui Jia, Dmitry Lagun, Fredo Durand, Bill Freeman, and Mark Matthews. Alchemist: Parametric control of material properties with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24130–24141, 2024.

[52] Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8839–8849, 2024.

[53] Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8839–8849, 2024.

[54] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[55] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[56] Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. Ominicontrol: Minimal and universal control for diffusion transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14940–14950, 2025.

[57] Zhenxiong Tan, Qiaochu Xue, Xingyi Yang, Songhua Liu, and Xinchao Wang. Ominicontrol2: Efficient conditioning for diffusion transformers. *arXiv preprint arXiv:2503.08280*, 2025.

[58] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *Advances in neural information processing systems*, 36:1363–1389, 2023.

[59] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[60] Vaibhav Vavilala, Seemandhar Jain, Rahul Vasanth, David Forsyth, and Anand Bhattad. Generative blocks world: Moving things around in pictures. In *ICLR*, 2026.

[61] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

[62] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[63] 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.

[64] Ziyi Wu, Yulia Rubanova, Rishabh Kabra, Drew A Hudson, Igor Gilitschenski, Yusuf Aytar, Sjoerd Van Steenkiste, Kelsey R Allen, and Thomas Kipf. Neural assets: 3d-aware multi-object scene synthesis with image diffusion models. *Advances in Neural Information Processing Systems*, 37:76289–76318, 2024.

[65] Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. Native and compact structured latents for 3d generation. *arXiv preprint arXiv:2512.14692*, 2025.

[66] Yicheng Xiao, Wenhu Zhang, Lin Song, Yukang Chen, Wenbo Li, Nan Jiang, Tianhe Ren, Haokun Lin, Wei Huang, Haoyang Huang, et al. Spatialedit: Benchmarking fine-grained image spatial editing. *arXiv preprint arXiv:2604.04911*, 2026.

[67] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[68] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18381–18391, 2023.

[69] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10371–10381, 2024.

[70] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

[71] Ze-Xin Yin, Liu Liu, Xinjie Wang, Wei Sui, Zhizhong Su, Jian Yang, and Jin Xie. 3d-fixer: Coarse-to-fine in-place completion for 3d scenes from a single image. *arXiv preprint arXiv:2604.04406*, 2026.

[72] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[73] 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*, pages 3836–3847, 2023.

[74] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[75] Shuo Zhang, Wenzhuo Wu, Huayu Zhang, Jiarong Cheng, Xianghao Zang, Chao Ban, Hao Sun, Zhongjiang He, Tianwei Cao, Kongming Liang, et al. Geometric image editing via effects-sensitive in-context inpainting with diffusion transformers. *arXiv preprint arXiv:2602.08388*, 2026.

[76] Yuxuan Zhang, Yirui Yuan, Yiren Song, Haofan Wang, and Jiaming Liu. Easycontrol: Adding efficient and flexible control for diffusion transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19513–19524, 2025.

[77] Zewei Zhang, Huan Liu, Jun Chen, and Xiangyu Xu. Gooddrag: Towards good practices for drag editing with diffusion models. In *International Conference on Learning Representations*, volume 2025, pages 13785–13808, 2025.

[78] Jensen Zhou, Hang Gao, Vikram Voleti, Aaryaman Vasishta, Chun-Han Yao, Mark Boss, Philip Torr, Christian Rupprecht, and Varun Jampani. Stable virtual camera: Generative view synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12405–12414, 2025.

[79] Hanshen Zhu, Zhen Zhu, Kaile Zhang, Yiming Gong, Yuliang Liu, and Xiang Bai. Training-free geometric image editing on diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19130–19140, 2025.

Supplemental Materials

Table of Contents

A Additional results

A.1 Qualitative results

We provide further results in Fig.11 and 13, showcasing robust 3D scene editing across multiple styles, perspectives and operations.

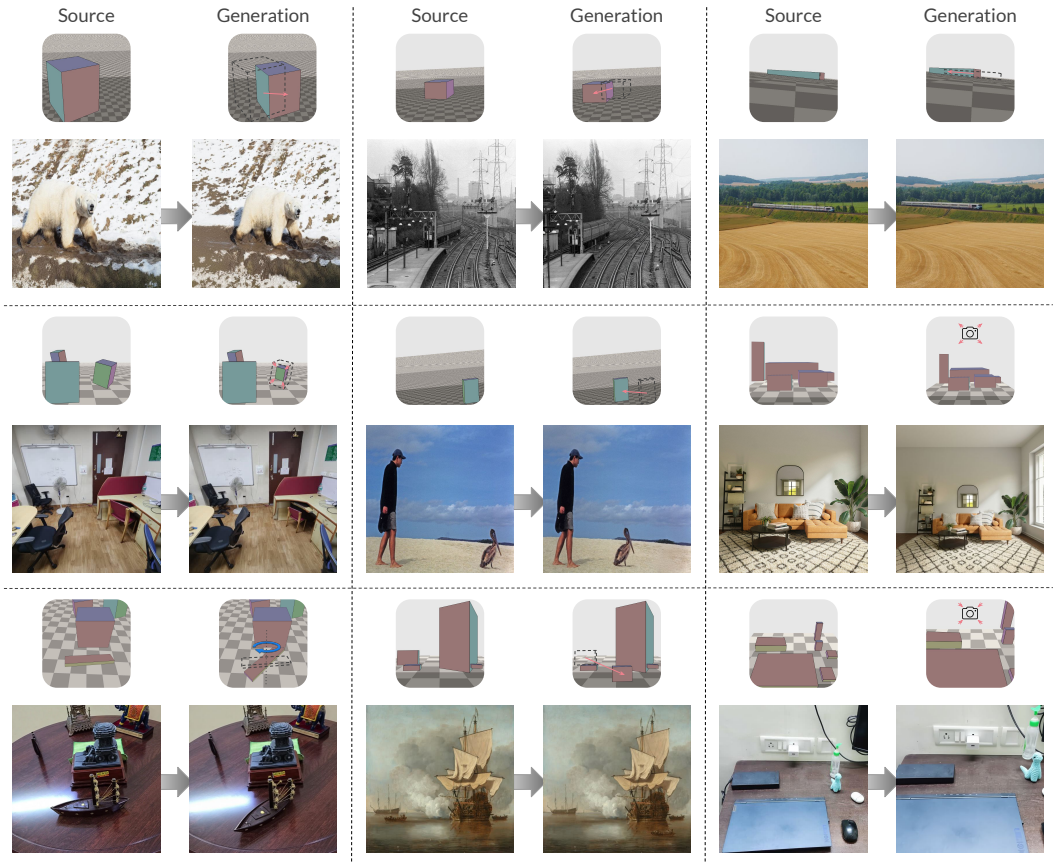

Figure 11: **In-the-wild 3D editing:** We demonstrate the robustness of our method by showcasing additional 3D edits across real world scenes having different lighting, viewpoints and styles. Our method although trained on one or two object scenes, generalizes to complex multi object layouts in real images. Notably we demonstrate all 3D edit operations in this figure from object-centric translation, scaling and rotation to changing camera viewpoints with a single checkpointing, demonstrating the strength of our box conditioning.

As shown in Fig.12, our model also maintains high-fidelity details even when applying small, incremental shifts to object positions.

Figure 12: **Smooth Interpolation**: We demonstrate smooth interpolation between the source image and target layout. We overlay the source object position in terms of 2D bounding box on each image to clearly distinguish the incremental layout adjustments and this suggests that the model has internally learnt how to adjusts objects smoothly through box transformations.

Additionally, we compare our model's results with leading proprietary generative models on 3D edits with real-world scenes, As seen in Fig.14, they struggle to perform precise 3D edits, primarily due to their limitation in terms of text-based conditioning. Our method is able to perform precise 3D edits on these scenes that are otherwise challenging for proprietary models.

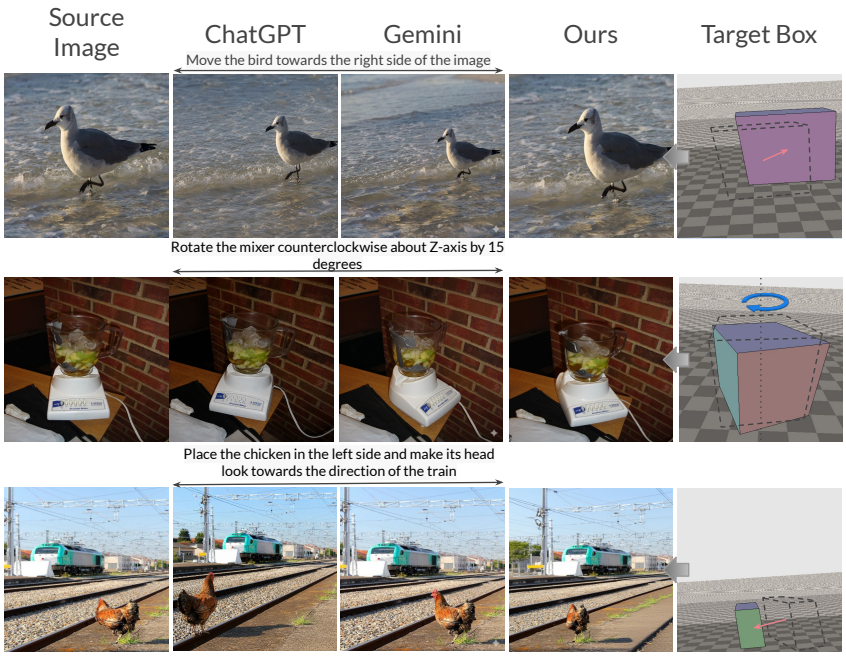

Figure 14: **Comparison with SOTA Models.** We evaluate our method against leading proprietary image generative models, Gemini and ChatGPT, on 3D editing tasks within real-world scenes. Since these models primarily rely on text-based conditioning, which lacks the spatial granularity necessary for 3D manipulation, they often fail to execute the precise 3D edits required by users.

A.2 Quantitative results

In the camera edit benchmark (Tab. 5), our method outperforms all baselines by a significant margin. It consistently preserves scene structure across edits. On the synthetic camera edit benchmark (Tab. 4), our method again surpasses all baselines. The only exception is background preservation, where we rank a close second to SpatialEdit [66].

Table 4: Quantitative comparison on *camera editing* using synthetic test set.

Method	Reconstruction quality				Consistency (DINO)		Edit fidelity			
	PSNR ↑	SSIM ↑	LPIPS ↓	DreamSim ↓	Subject ↑	Background ↑	Warp Error ↓	Mean Distance ↓	IoU ↑	Angular Error ↓
SEVA [78]	13.726	0.460	0.554	0.397	0.571	0.783	0.210	118.374	0.207	118.039
SpatialEdit [66]	12.271	0.420	0.622	0.345	0.642	**0.892**	0.228	123.629	0.199	118.236
Qwen-Camera-LoRA	13.280	0.432	0.645	0.480	0.526	0.683	0.203	137.366	0.220	117.583
Ours	**17.449**	**0.575**	**0.351**	**0.203**	**0.649**	0.857	**0.130**	**48.298**	**0.239**	**112.358**

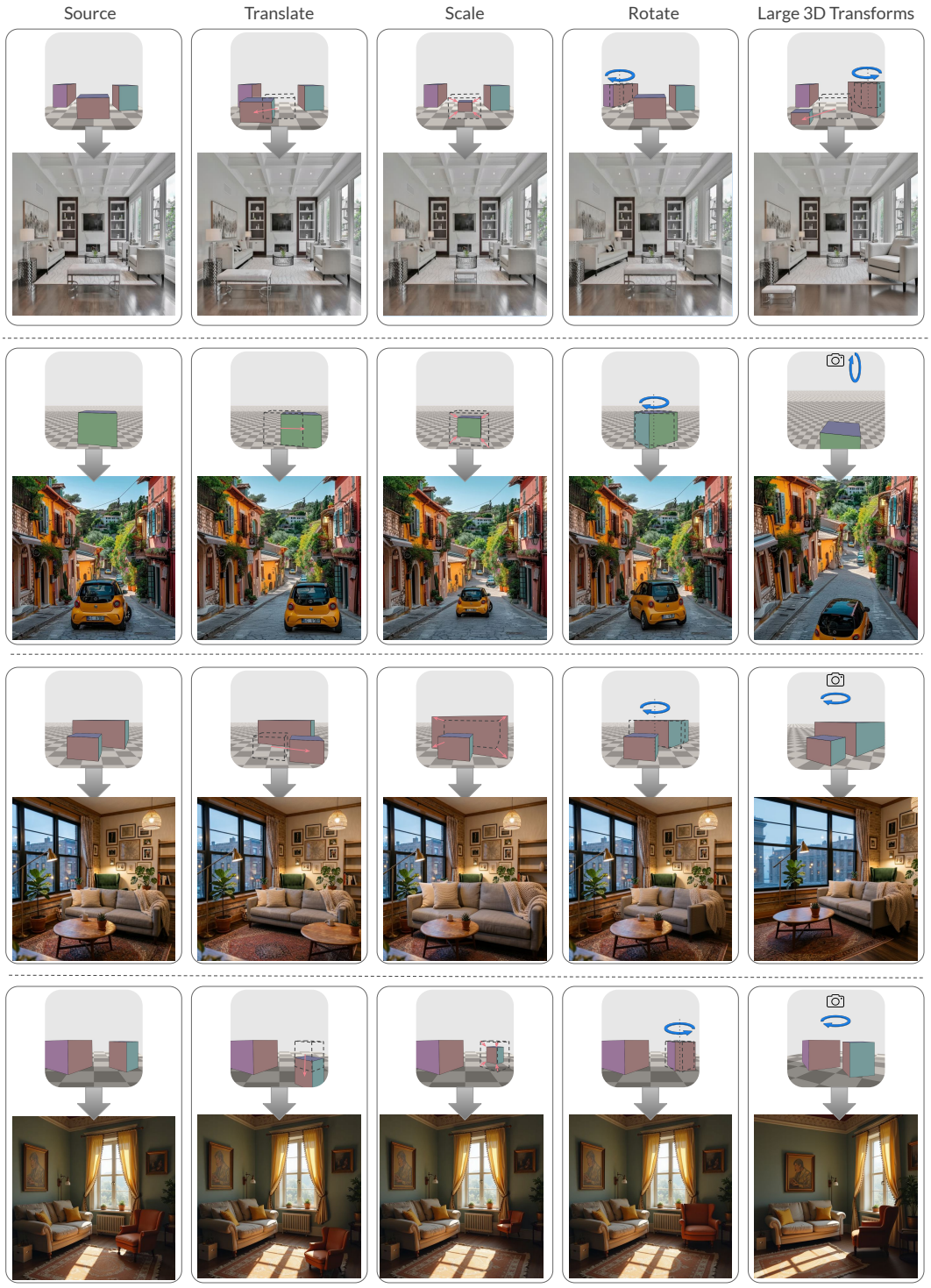

Figure 13: **Rethinking a scene:** This figure illustrates our model's ability to manipulate 3D scene layouts using primitive box representations. We demonstrate precise control over object location, orientation, and scale, enabling complex 3D transformations such as extreme viewing angles that require the seamless reorganization of multiple scene elements. Ultimately, our method serves as a robust tool for performing diverse, high-fidelity 3D edits on real-world scenes.

Table 5: Quantitative comparison on *camera editing* using **Objectron** [2] test set.

Method	Reconstruction quality				Consistency (DINO)		Edit fidelity			
	PSNR ↑	SSIM ↑	LPIPS ↓	DreamSim ↓	Subject ↑	Background ↑	Warp Error ↓	Mean Distance ↓	IoU ↑	Angular Error ↓
SpatialEdit [66]	8.364	0.239	0.801	0.768	0.378	0.552	0.332	116.637	0.208	126.220
Qwen-Camera-LoRA [63]	8.854	0.293	0.807	0.778	0.372	0.532	0.307	134.394	0.166	123.095
SEVA [78]	10.783	0.338	0.655	0.389	0.548	0.795	0.266	104.916	0.206	118.000
Ours	**15.076**	**0.452**	**0.379**	**0.124**	**0.791**	**0.931**	**0.141**	**29.406**	**0.327**	**70.152**

B Dataset Generation: Implementation Details

Generation is split into two decoupled stages. **Stage 1** (composition) lays objects out and synthesizes camera poses, and this scene configuration is saved; no images are produced. **Stage 2** (rendering) reads each scene configuration and produces all photometric and geometric outputs.

Source assets. 3D models come from Objaverse-XL [15] and we use a subset of 10K assets. We use 1K HDRIs and 5.5K PBR floor materials following 3D-Fixer [71].

Stage 1: Composition. Each scene contains $N{=}2$ objects on a planar floor, generated inside BlenderProc [16]. Sampled objects are normalized so their longest side equals 2.5 units, randomly rotated about the vertical axis, and placed using an XY circular-bound proxy with a 0.2 unit collision margin. Two $C{=}2$ camera poses on a lateral orbit at distance $d = (r \cdot 1.35)/\tan(\phi/2)$, where r is the joint bounding-sphere radius and ϕ the narrower field of view.To ensure full visibility, all 8 bounding box vertices of every object must project at least 10 pixels inside the frame. The second view introduces per-object perturbations, including a new azimuth, a uniform rescale between $[0.5, 1.5]$, and a small XY translation, while adhering to the original collision and visibility constraints.

Stage 2: Rendering. Fig.4 showcases our rendering pipeline. Scenes are rendered with Cycles on OPTIX at $512{\times}512$ resolution. Lighting is from the sampled HDRI. Each viewpoint generates two outputs: an RGB rendering and a 3D Box rendering. The latter features a directional 3D bounding box overlay, where the front, back, top, bottom, left, and right faces are color-coded in red, green, blue, yellow, pink, and cyan, respectively. Additionally, the 3D box rendering includes a depth-aligned planar floor rendered with a checkerboard pattern and depth-aware shading cues.

Fig. 15 summarizes the distribution of our training data. Fig. 17 presents representative samples of the HDRI environments and floor materials used during rendering. The category distribution of the 10,143 objects sourced from the SketchFab subset of **ObjaverseXL** [15] is shown in Fig. 16, and additional examples of our synthetic data rendered in Blender are provided in Fig. 18. Together, the wide variety of HDRIs, materials, and object categories yields a diverse training distribution that promotes generalization.

C Implementation details

We employ Flux-Kontext [34] as our foundational image editing model.To integrate our specific conditioning, we train a LoRA adapter of rank 32, applied to the query, key, and value projection matrices across all MMDiT blocks. Throughout the editing process, we utilize an empty text prompt (""), ensuring the model relies exclusively on visual cues derived from our box conditioning.

The training process is executed in two distinct stages:

- **Stage 1:** The model is trained on a synthetic dataset of 100,000 scenes for 50,000 steps. This phase adapts the model's internal representations to execute transformations dictated by the spatial relationship between source and target boxes.
- **Stage 2:** We fine-tune the model for 10,000 steps using a composite dataset of 20,000 scenes. This includes 10,000 samples from the Objectron [2] training set to improve performance on real-world imagery and 10,000 additional synthetic scenes. We retain synthetic data in this stage to ensure the model maintains robust object-level control, as the Objectron dataset focuses primarily on camera-motion-driven changes.

We utilize the Prodigy [39] optimizer with safeguard warmup and bias correction enabled, maintaining a weight decay of 0.01. Training is conducted at a resolution of 512 × 512 with an effective batch size of 16. Our implementation is built on PyTorch [46] and the Hugging Face Diffusers [61] library.

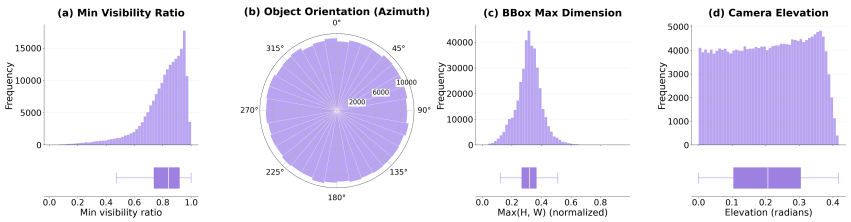

Figure 15: **Statistics of training dataset.** (a) The distribution of the minimum visibility ratio across objects within each scene is left-skewed and highly concentrated near unity (median=0.84). This profile indicates that while the majority of scenes exhibit only mild inter-object occlusion, a substantial long tail extending to 0.01 ensures the inclusion of severe occlusion cases. (b) Object orientations follow a near-uniform distribution over azimuth, avoiding any directional bias in the training data. (c) The maximum 2D bounding-box dimension (normalized by image side) is tightly concentrated around 0.32, reflecting a deliberate scene composition that keeps multiple objects simultaneously visible at moderate scale rather than dominating the frame. (d) Camera elevation is broadly uniform from 0 to roughly 0.4 radians ($\approx 23°$), this favors low-angle viewpoints over bird's-eye configurations, which in turn produce stronger inter-object occlusions and more visually challenging compositions.

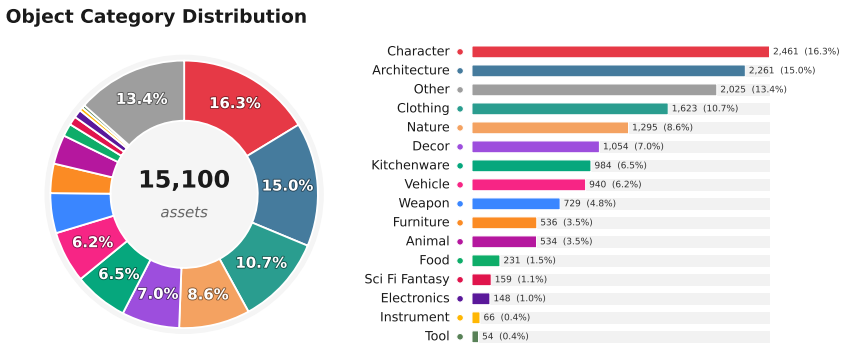

Figure 16: Object category distribution across the 10,143 downloaded assets, derived via multi-label keyword classification on the asset captions. *Left:* donut chart visualizing the relative proportions of each category, with percentages shown for the dominant slices. *Right:* ranked breakdown of all 16 categories with absolute counts and percentages. As assets may match multiple categories (e.g., a knight *character* holding a *weapon*), category counts sum to more than the total number of assets.

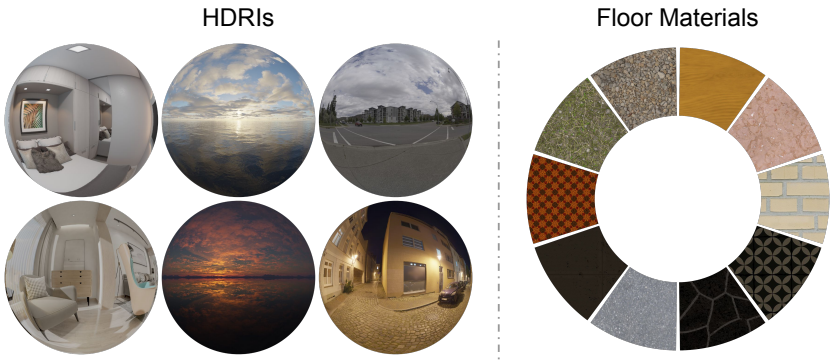

Figure 17: Sample illustrations from our dataset's environmental and material assets. *Left:* Six representative HDRIs spanning indoor (top-left, bottom-left), natural outdoor (top-middle, bottom-middle), and urban outdoor (top-right, bottom-right) environments. *Right:* A radial arrangement of 12 floor material samples drawn from our texture library, illustrating the diversity of surface categories (wood, stone, brick, tile, concrete, metal, etc.).

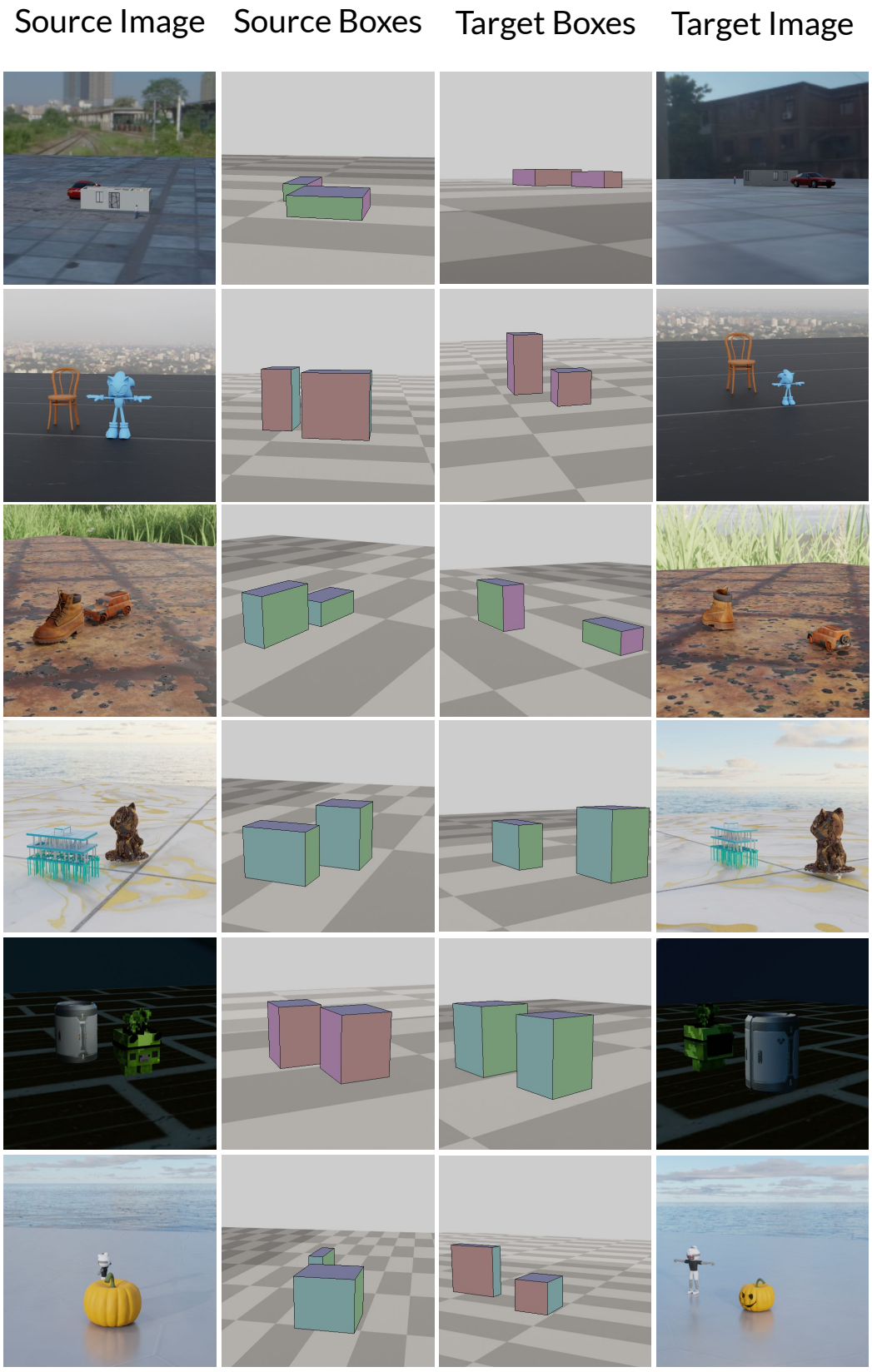

Figure 18: Various ObjaverseXL assets rendered in Blender, which are used in training are visualized in the above figure. The diversity of the lighting and environment condition are visualized.

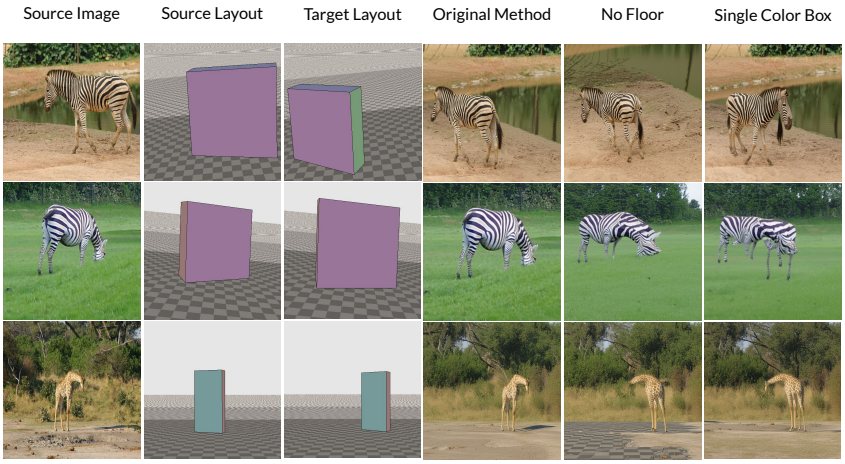

Figure 19: **Ablation visualization.** Adding the checkered floor pattern for ground localization and orientation-specific face colors for the conditioning box yields generations that are substantially more faithful to the edit instructions. Without the floor (*No Floor*), the model fails to preserve the texture and ground contact of the animals, as seen most clearly in the first row. With a uniformly colored box (*Single Color Box*), the model loses the directional cue and frequently flips the animal's orientation to face the opposite direction. These results highlight the importance of encoding both ground reference and orientation information into the spatial conditioning signal.

D Ablation Visualizations

We provide qualitative examples in Fig. 19 to complement the quantitative ablation results reported in the main paper. Each row corresponds to a different test sample, with columns showing the image generated with the original method, the *No Floor* variant, and the *Single Color Box* variant. Without the checkered floor, the model loses its global ground reference and produces outputs in which the animal's texture and contact with the ground are inconsistent with the target layout (most visible in the middle row). Without directional face colors, the model retains correct positioning but frequently flips the animal's facing direction, confirming that the directional coloring is the dominant cue for orientation. A sample of these box renderings are shown in Fig 20 Together, these examples visually reinforce the conclusion that the floor and directional box colors contribute complementary, non-redundant signals.

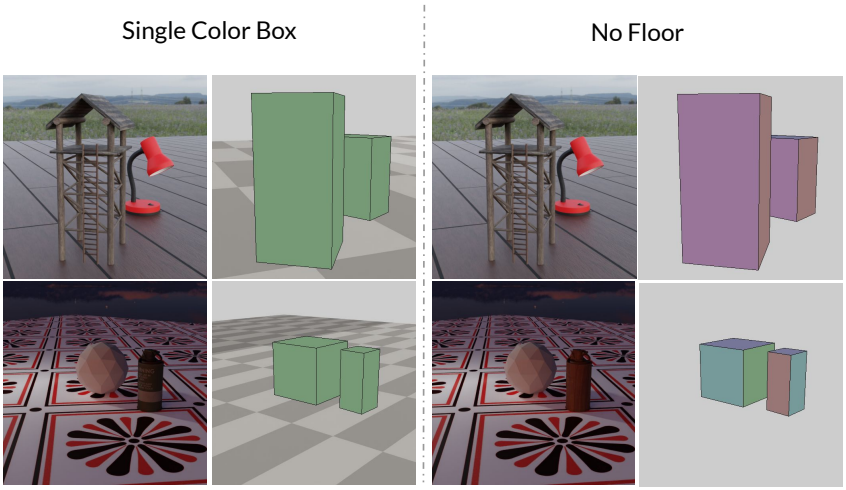

Figure 20: **Ablation Dataset visualization.** Here we show a sample rendering of the assets with the same box colors and without floor.

E User Study Setup

Fig. 21 shows the interface used in our A/B user study. Each question presents the input image, the source and target 3D layouts, and two candidate edits, and participants select the preferred output along one of three criteria: *object preservation* (texture, color, and fine details of the edited object), *background preservation* (consistency of the surrounding scene), and *layout following* (agreement with the target 3D layout). Example questions with reference answers are shown at the beginning of the study to familiarize participants with the evaluation protocol.

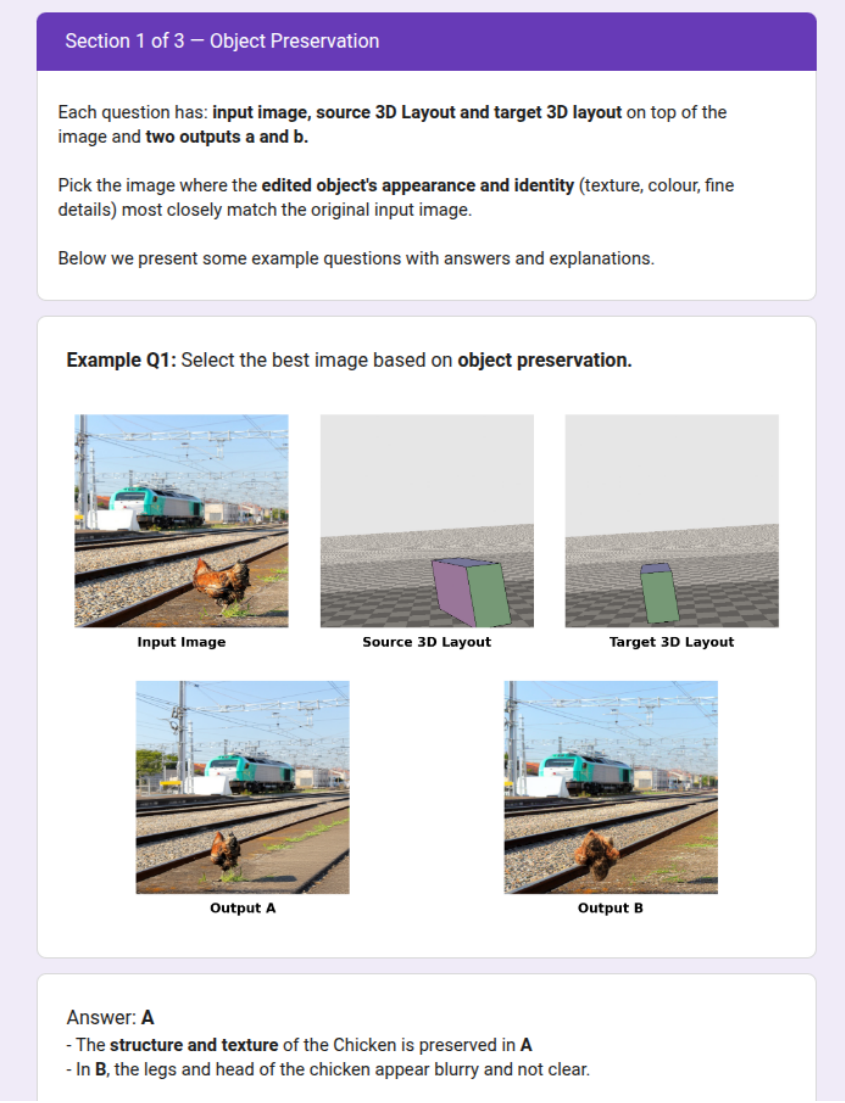

Figure 21: Participants are shown the input image, source and target 3D layouts, and two candidate edited outputs (A and B). They are asked to select the output that best preserves the edited object's appearance and identity (e.g., foreground, background details) with respect to the original input image. Example questions with explanations are provided to familiarize participants with the evaluation protocol.