COMPASS: COnditioned Manipulation via Precise Aligned Spatial Specification

Anonymous authors
Paper under double-blind review

Abstract

Image editing models are essential for content creation, and instruction-based editors make semantic modifications accessible through natural language. Precise spatial control (i.e., relocating, rotating, or rescaling an object) remains highly unreliable and challenging. We present COMPASS, which conditions a pretrained image editor on rendered 3D bounding-box proxies: we rasterize a proxy box at the source pose and another at the target pose, and paint each face with a fixed color shared across boxes. This allows the spatial control to be represented as local, spatially aligned differences in the pixel domain, which a diffusion transformer can follow without learning cross-position geometry from scratch. COMPASS also supports multi-object edits in a single pass. To demonstrate our approach, we use Qwen-Image-Edit as the base backbone and fine-tune it on a new dataset of 37,000 source/target image pairs, of which 30,000 are used for training and 7,000 are held out as a benchmark with single-object, multi-object, and overlapping-box splits; we further transfer the model zero-shot to the WildDet3D real-image dataset. We compare COMPASS with state-of-the-art editors on this benchmark. COMPASS outperforms existing techniques: (1) on the new dataset, COMPASS achieves a translation error of 0.27 m and a target-mask IoU of 0.765, where competing methods range from 0.31 to 1.19 m and from 0.262 to 0.633; and (2) on the real-image dataset, COMPASS reduces translation error to 4.15 m, nearly half of the best competing method at 7.93 m and under 40% of the worst at 11.83 m.

1 Introduction

Instruction-based image and video editing has matured quickly, driven by applications that require repositioning objects in an existing scene, such as virtual staging (Tukur et al., 2024), previsualization for film and games (Nitsche, 2008), and synthetic data generation to augment 3D or assist robotics training (Lu et al., 2023). Models such as InstructPix2Pix (Brooks et al., 2023), FLUX.1 Kontext (Black Forest Labs et al., 2025), and Qwen-Image-Edit (Wu et al., 2025) follow free-form language to change the content of an image. However, while these models generate plausible images, they rarely produce the requested one. Two fundamental challenges are responsible. First, language is a poor interface for specifying a metric 3D transformation: a user can name a direction or an approximate amount, but rarely knows, let alone can convey, the exact rotation angle, translation vector, or scale factor required. Second, even when the transformation is given explicitly, the backbone model must learn how it maps onto pixel-level change from scratch, a cross-position geometric computation that current architectures acquire slowly and imprecisely.

Prior work either reconstructs the object in 3D and re-renders it (Yenphraphai et al., 2024; Pandey et al., 2024; Sajnani et al., 2025; Chen et al., 2025), paying for exact control with per-image reconstruction and its errors, or conditions a generative model on an abstract pose representation (Michel et al., 2023; Wu et al., 2024; Ruan et al., 2026; Yu et al., 2025), which leaves the second challenge unaddressed (Sec. 2).

All prior approaches assume the 3D transformation must live either entirely in an explicit scene or in an abstract parameter vector. We argue for a representation in between – an oriented 3D bounding box. Given a rigid transform, we rasterize proxy boxes at the source and target pose from the fixed viewpoint of the source camera, and paint the six faces of each box with fixed, predefined colors,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

so the same color denotes the same face in both the source and target proxies. Thus, transformation is given explicitly without requiring per-instance computation: the box positions tell the model *where* the object is and should go; the face colors, which travel rigidly with the box, tell it *how* the object is oriented in each state and which visible surfaces will be exposed or occluded after the edit. Translation, rotation, and rescaling thereby become local appearance differences between two spatially aligned images, a signal that a diffusion transformer can process directly without learning new geometric computation.

Bounding boxes are trivial to obtain, from a user placing a proxy actor in a 3D scene editor (Blender Online Community, 2018), a real-time engine (Epic Games, 2024), or a promptable monocular 3D detector (Huang et al., 2026) applied to an in-the-wild photograph. The target box is derived from the source box rather than specified from scratch: the user may drag, rotate, or scale the source proxy in the same editor, or directly enter the translation, yaw change, and scale factor numbers.

A critical challenge of the bounding box representation is that, when several objects have similar sizes or their boxes overlap in the image, a naive backbone often fails to infer the correct pairing of each object, so it collapses, moving the wrong object, blending two into one, or duplicating an object at multiple plausible destinations. We term this failure *correspondence hallucination*. We address this challenge with two small learned embeddings on the box tokens: (1) a *role* embedding marks each box as source or target, and (2) an *object-ID* embedding shared between the source and target boxes of the same object. Since the pairing is given rather than inferred, multi-object editing reduces to concatenating box pairs with distinct IDs; and because IDs are assigned in depth order, they also specify which object is in front when two target boxes overlap. The embeddings can be trained jointly with the adapters and do not require a change in the backbone.

To train and evaluate COMPASS, we require paired images of the same scene showing one or more objects at a known source and target pose each, captured from a fixed camera. No existing dataset provides this. We therefore build a data generation pipeline in Unreal Engine 5 (Epic Games, 2024) that populates photorealistic city scenes with native City Sample props and license-filtered Objaverse 1.0 assets (Deitke et al., 2023), normalizes assets to a common metric size, samples rigid transformations with per-object collision and visibility checks, and renders source and target frames with lighting that responds physically to the edit. The same pipeline yields both training data and a held-out benchmark with controlled single-object, multi-object, and overlapping-box splits. We render 37,000 1024×1024 source–target pairs, with 7,000 pairs reserved for evaluation. The test split is disjoint from training in both objects and city regions.

In our experiments, COMPASS finetunes the Qwen-Image-Edit backbone using this new dataset, and we evaluate its performance using both the new held-out benchmark, and also applied zero-shot to in-the-wild photographs from WildDet3D (Huang et al., 2026) using the dataset's annotated 3D boxes as source boxes. We show that COMPASS places edits at the requested position and orientation while preserving object identity and the surrounding scene, outperforming both explicit-3D and learned-conditioning baselines on pose fidelity and image quality. For single object transformation, COMPASS achieves a translation error of 0.27 m and a target-mask IoU of 0.765, compared to 0.31–1.19 m and 0.262–0.633 for prior methods and alternative conditionings. In the multi-object and overlapping settings, the improvements are even larger. On the multi-object split, COMPASS places 75.8% of objects in their own target box, against at most 56.1% for competing methods. On the overlapping-box split, COMPASS reduces translation error from 0.41 m to 0.28 m, within 3 cm of what the evaluation detector reports on ground-truth images, while raising the fraction of objects placed in their own target box from 66.5% to 70.2% and cutting merge failures from 1.7% to 1.2%.

Contributions.

- **3D bounding boxes as a pixel-domain editing representation.** These boxes serve as local, spatially aligned appearance cues that a pretrained diffusion transformer can follow directly, without learning cross-position geometry from scratch or reconstructing the object.
- **Role and object-ID embeddings for correspondence.** We identify correspondence hallucination as the central failure mode of box-based conditioning, and resolve it with two lightweight embeddings, trained jointly with the adapters, without changing the backbone.
- **A photorealistic paired dataset, benchmark, and reusable pipeline.** We build a seed-reproducible Unreal Engine 5 pipeline, re-parameterizable to new assets, scenes, and edit

types, that produces the first photorealistic paired dataset with exact 3D transformations, per-object 3D boxes, and masks, together with a held-out benchmark spanning single-object, multi-object, and overlapping-box splits.

- **State-of-the-art performance, with zero-shot transfer to real images.** COMPASS outperforms instruction-based, explicit-3D, and learned-conditioning baselines on every metric of our benchmark, with the largest margins in multi-object and overlapping settings, and transfers zero-shot to real photographs with annotated 3D boxes.

2 Related Work

Instruction-based and spatially controlled editing. Instruction-following editors (Brooks et al., 2023; Black Forest Labs et al., 2025; Wu et al., 2025) handle semantic and stylistic edits well but have no channel for continuous geometry: "move it a little to the left" is interpreted rather than executed. Spatial control methods add layout signals: ControlNet (Zhang et al., 2023) and T2I-Adapter (Mou et al., 2024) condition on aligned depth or edge maps, GLIGEN (Li et al., 2023) grounds objects with 2D boxes, and drag-based methods (Pan et al., 2023; Shi et al., 2024) move handle points on the image plane. All specify *where* an object appears, not how it is oriented. COMPASS keeps the insight that dense, aligned conditioning is what diffusion models read best and extends it to full 3D pose via oriented proxy boxes with fixed per-face colors.

3D-aware editing with explicit geometry. One family reconstructs the object before editing it: Image Sculpting (Yenphraphai et al., 2024) edits a textured mesh, Diffusion Handles (Pandey et al., 2024) lifts activations onto estimated depth, GeoDiffuser (Sajnani et al., 2025) warps attention maps by a user-given transform, and BlenderFusion (Chen et al., 2025) manipulates segmented 3D entities in Blender; 3D-aware GANs (Chan et al., 2022) and NeRF editing (Haque et al., 2023) follow the same reconstruct–edit–render template. Control is exact, but it depends on single-image depth or shape estimation, often per-image optimization, and fails on thin, transparent, or unusual objects. COMPASS never reconstructs: its only 3D input is an oriented box.

3D-aware editing with learned pose conditioning. A second family supplies pose as a learned condition: language descriptions (Michel et al., 2023), pose parameters from a 3D box (Wu et al., 2024), relative camera pose (Ruan et al., 2026; Liu et al., 2023), or 2D source/target masks in an image-to-video model (Yu et al., 2025), which exposes no rotation and reports nearby objects vanishing. These avoid reconstruction, but pose reaches the network as abstract parameters or position-only masks, so the pixel-level effect of a rotation must be learned and multi-object binding rests on implicit pairing.

Two concurrent works use the same per-face colored boxes but stop short of multi-object editing: Thinking in Boxes (Bhat et al., 2026) anchors boxes to a depth-aligned floor for viewpoint changes but reports identity outputs when several boxes share a similar size; BoxCtrl (Wang et al., 2026) fine-tunes FLUX.1 Kontext on single-object renders with online RL for sim-to-real transfer but edits objects one at a time. COMPASS targets exactly this regime: role and object-ID embeddings bind each source–target pair, so several objects, even with intersecting or similar-sized boxes, are moved in a single pass (Sec. 3). Our face coloring is related to NOCS (Wang et al., 2019), which colors surface points by canonical coordinates for pose estimation; we assign one fixed color per cuboid face, conveying orientation without knowing the object's shape. Oriented 3D boxes are the standard output of monocular 3D detectors (Brazil et al., 2023; Huang et al., 2026), so any detector or a user dragging a cuboid can produce our input.

3 Method

3.1 Problem definition

Let $I_{\mathrm{src}} \in \mathbb{R}^{H\times W\times 3}$ be a source image and let the object to be edited be given by an oriented 3D bounding box expressed in the camera frame of I_{src}. We adopt the gravity-aligned parameterization standard in outdoor 3D detection,

$$\mathcal{B}_{\mathrm{src}} = (\mathbf{t},\ \theta,\ \mathbf{s}), \qquad \mathbf{t}\in\mathbb{R}^3,\ \ \theta\in[0,2\pi),\ \ \mathbf{s}\in\mathbb{R}^3_{>0}, \tag{1}$$

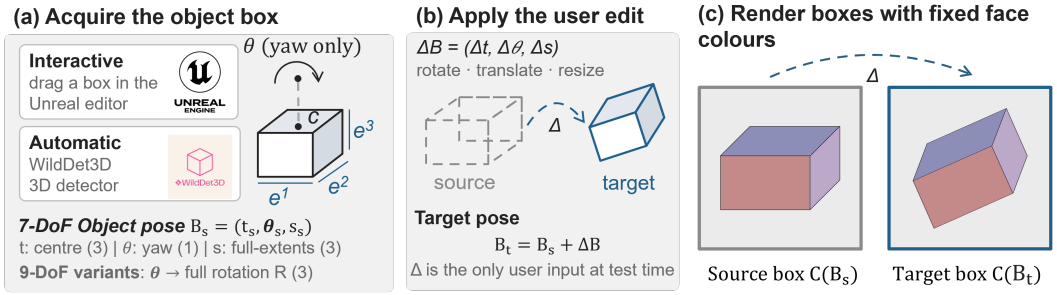

Figure 1: Three steps to perform box-to-image conditioning.

with center $\mathbf{t}$, yaw θ around the vertical axis, and extents $\mathbf{s}$, i.e. a 7-DoF box whose eight corners are

$$\mathbf{p}_k = \mathbf{t} + R_z(\theta)\left(\boldsymbol{\sigma}_k \odot \tfrac{1}{2}\mathbf{s}\right), \qquad \boldsymbol{\sigma}_k \in \{-1,+1\}^3. \tag{2}$$

An edit is a tuple $\Delta = (\Delta\mathbf{t}, \Delta\theta, \Delta\sigma) \in \mathbb{R}^3 \times [-\pi,\pi) \times \mathbb{R}_{>0}$ consisting of a translation, a change of yaw, and an isotropic rescaling, and it defines the target box

$$\mathcal{B}_{\text{tgt}} = \left(\mathbf{t} + \Delta\mathbf{t},\ \theta + \Delta\theta,\ \Delta\sigma\,\mathbf{s}\right). \tag{3}$$

Given $(I_{\text{src}}, \mathcal{B}_{\text{src}}, \Delta)$, the task is to synthesize an image I_{tgt} in which the object occupies $\mathcal{B}_{\text{tgt}}$ while its identity and all pixels not affected by the edit are preserved. The specification is metric and complete, so unlike text- or drag-based editing, where the intended placement is underdetermined and any plausible outcome is acceptable, correctness here is measured by whether the object lands on $\mathcal{B}_{\text{tgt}}$. Our experiments restrict Δ to planar translation (Sec. 4); however, the formulation itself places no such restriction, and Δ may specify any 3D translation.

3.2 Boxes as an Editing Representation

Editing an object in 3D amounts to three operations, *translating* it to a new location, *rotating* it to a new orientation, and *rescaling* it. We specify all three with a single primitive, an oriented bounding box whose faces carry fixed canonical colors. The process is shown in Fig. 1.

Rendering a box into the image plane. We render a box $\mathcal{B}$ into a *box image* $C(\mathcal{B}) \in \mathbb{R}^{H\times W\times 3}$ using the pinhole camera of the source image. The corners of Eq. equation 2 define six faces. A face is drawn if it faces the camera, $\langle \mathbf{n}_j,\ \mathbf{d}_j\rangle > \tau$ with $\mathbf{n}_j$ its outward normal, $\mathbf{d}_j$ the unit vector from its center to the camera and τ a small grazing threshold. Visible faces are sorted by depth and painted back to front, and the image is rasterized with supersampling and downsampled to the target resolution. Because the six colors are shared by every box, a rotation appears as a change in which colored faces are exposed and a translation as a displacement of the same colored polygons. The box image carries scale and pose only, and nothing of the object's appearance.

Conditioning. The editor maps $(I_{\text{src}}, C_{\text{src}}, C_{\text{tgt}})$ to I_{tgt}; the edit is never passed explicitly but implied by the pair of boxes, keeping all conditioning in the image domain. At test time the source box comes from a 3D detector or interactive placement, and the target box from Eq. equation 3.

3.3 Image Editor

Our editor is a dual-stream diffusion transformer (Wu et al., 2025). One stream carries image tokens, the other text tokens, and the two exchange information through joint attention in every block. All conditioning enters through the image stream: the source photograph and the box images are encoded by the frozen VAE into latents, patchified into tokens, and concatenated with the noisy target latent into a single sequence, $[\,\mathbf{z}_{\text{noise}} \mid \mathbf{z}_{\text{src}} \mid \mathbf{z}_{C_{\text{src}}} \mid \mathbf{z}_{C_{\text{tgt}}}\,]$. The text stream receives a fixed instruction that only names the task; it carries no information about the edit itself. The output is the denoised target latent, which the VAE decodes into the edited image I_{tgt} (Fig. 2). All images are processed at 512×512 (Appendix A.2). We adapt the pretrained model by placing rank-32 LoRA adapters (Hu et al., 2022) on the attention projections of the image stream and keep everything else frozen, so the model learns to read the box images without changing how it renders photographs.

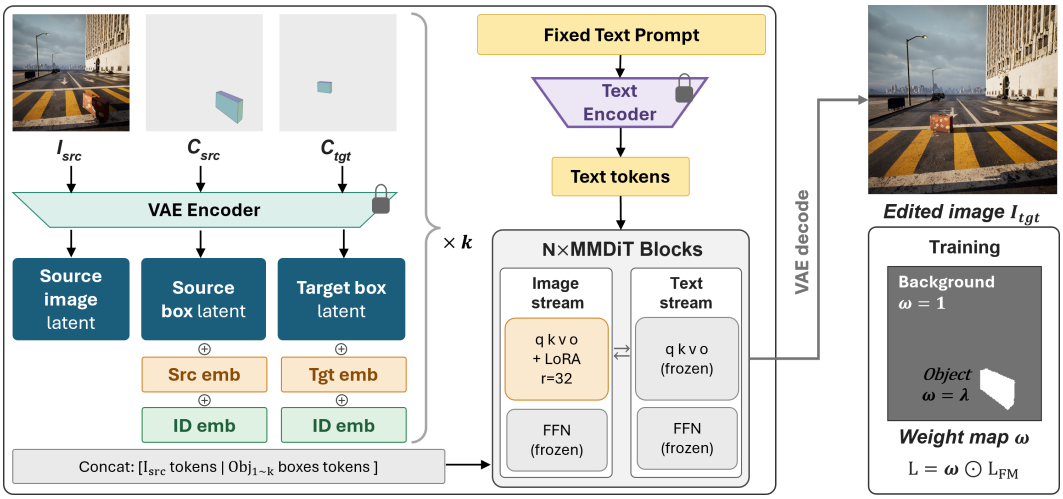

Figure 2: Overview of COMPASS

Role and object-ID embeddings. Because the source and target boxes are rendered with the same six colors into images of the same size, nothing in the tokens themselves says which map is the "before" and which the "after", and with several objects nothing says which target box belongs to which source box. We resolve both at the token level with two small learned tables. A *role* embedding $\mathbf{e}^{\text{role}}_r$, $r \in \{\text{src}, \text{tgt}\}$, is added to every token of the corresponding box map, and an *object-ID* embedding $\mathbf{e}^{\text{id}}_k$, $k \in \{1, \ldots, K_{\max}\}$, is added to all box tokens of object k:

$$\mathbf{z}_{k,r} \leftarrow \mathbf{z}_{k,r} + \mathbf{e}^{\text{role}}_r + \mathbf{e}^{\text{id}}_k. \tag{4}$$

The pairing between a source box and its target box is thus stated in the input rather than inferred from geometry. Editing k objects amounts to appending k box pairs with distinct IDs to the sequence, $[\,\mathbf{z}_{\text{noise}} \mid \mathbf{z}_{\text{src}} \mid \mathbf{z}_{\text{obj}_1} \mid \cdots \mid \mathbf{z}_{\text{obj}_k}\,]$, up to $K_{\max}$ objects per pass ($K_{\max} = 3$ in our experiments). IDs are not arbitrary: we assign them in order of increasing depth of the source box center, so $k = 1$ is the object nearest the camera. The ID therefore also carries the depth ordering, and when two target boxes overlap in the image it specifies which object should occlude the other.

Object-weighted flow-matching objective. We train with rectified flow matching (Lipman et al., 2023). Let $\mathbf{x}_1$ be the VAE latent of the target image, $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\mathbf{x}_\tau = \tau\,\mathbf{x}_1 + (1-\tau)\,\mathbf{x}_0$, with τ drawn from the resolution-dependent timestep schedule of the backbone (Wu et al., 2025), which we leave unchanged. The network v_θ predicts the velocity $\mathbf{x}_1 - \mathbf{x}_0$ from the noisy latent and the conditioning tokens $\mathbf{c}$ described above. Because the edited object occupies a small fraction of the frame, a uniform loss lets the trivial solution, copying the source, dominate early training. We therefore weight the loss with a spatial map ω:

$$\mathcal{L} = \mathbb{E}_{\tau,\mathbf{x}_0}\Big[\,\big\|\omega \odot \big(v_\theta(\mathbf{x}_\tau, \tau, \mathbf{c}) - (\mathbf{x}_1 - \mathbf{x}_0)\big)\big\|_2^2\Big], \qquad \omega(\mathbf{p}) = \begin{cases} \lambda & \mathbf{p} \in \Omega, \\ 1 & \text{otherwise}, \end{cases} \tag{5}$$

where Ω is the object's mask in the target image, downsampled to latent resolution, and $\lambda > 1$ ($\lambda = 3$ in all experiments). The unit weight on the background is deliberate: it keeps scene preservation in the objective rather than delegating it to a separate loss or to inference-time blending. An overview of COMPASS is shown in Fig. 2.

4 Dataset Generation

Assets. Backgrounds come from the *City Sample* project (Epic Games, 2022), Epic Games' public release of a roughly 4 km × 4 km photorealistic city with buildings, roads, vehicles, street furniture, and vegetation under fully dynamic lighting, providing varied and realistic street-level scenes. Edited objects come from two sources in equal proportion: native City Sample props, and assets from Objaverse 1.0 (Deitke et al., 2023), a collection of over 800K artist-created 3D models. Because Objaverse licenses vary per asset, we apply our own license filter and retain only assets whose

terms permit redistribution and derivative use. In both sources we sample uniformly over asset groups (meshes for City Sample, LVIS categories for Objaverse) rather than instances to avoid over-representing common classes. City Sample props keep their artist-authored metric size; Objaverse meshes, whose native units are arbitrary, are normalized so that the longest side of every asset is 1 m before the per-sample edit is applied, which keeps the full asset pool within the pipeline's camera-distance and pixel-size window at the cost of some objects appearing at an unrealistic scale (Appendix A.1).

Pipeline. Data generation is fully automated and driven by a single random seed, so the entire dataset can be regenerated bit-for-bit or extended simply by drawing more seeds. For each pair we place one, two, or three objects on street-level surfaces, position an eye-level camera at a distance scaled to the objects' size, and sample an independent edit for every object, each consisting of a planar translation, a yaw rotation, and a uniform scale, with the vertical position set so that the rescaled object rests on its support surface. An edit is accepted only if the new pose rests on flat ground, is collision-free with the scene and with the other objects, and remains largely visible from the camera. We render the source view, apply all edits to the same objects in-engine, allow the lighting to re-converge, and render the target view. Temporal anti-aliasing is disabled so the two renders differ only by the edits and their lighting effects. Each pair consists of 1024×1024 source and target images, the corresponding proxy-box images used as conditioning (Sec. 3), pixel-exact per-object masks, and a JSON record with the camera pose, the 3D boxes of every object before and after the edit, and the edit parameters.

Splits. We render 37,000 pairs in total and hold out 7,000 for evaluation. The test split is disjoint from training in both objects and city regions. Test pairs are grouped by the correspondence difficulty they pose. *Single* contains pairs with one object. Pairs with two or three objects are assigned to *Overlap* if the maximum pairwise 2D IoU between the projected source boxes exceeds 0.15, and to *Multi* otherwise; both are selected post hoc from the rendered pairs rather than constructed by hand.

5 Experiments

5.1 Experiment Setup

Backbone and Datasets. Our backbone and dataset setup is discussed in Sec. 3 and 4, respectively. In addition to the new dataset, we also build a test set from WildDet3D (Huang et al., 2026) of 474 single-object samples, whose in-the-wild photographs carry oriented 3D bounding boxes. We take the annotated box as $\mathcal{B}_{\mathrm{src}}$, and sample edits from the training distribution on the object's support plane.

Evaluation metrics. We assess edits along four axes. ***(1) Pose fidelity*** tests whether the object actually lands where requested: we run an off-the-shelf 3D detector (Huang et al., 2026) on the output, match the detection to $\mathcal{B}_{\mathrm{tgt}}$, and report translation, yaw (modulo symmetry), and scale error, plus mask IoU against the ground-truth target mask. Pose error is undefined when the object cannot be found, so we exclude misses from these averages and report the miss rate separately as its own signal; on real data we also report the *removal rate*, the fraction of samples where the detector no longer finds the object at its source pose, which indicates whether an edit was attempted at all. ***(2) Identity preservation*** (DINOv2 similarity between the source object crop and the target-box crop) and ***(3) background preservation*** (LPIPS outside the dilated union of both boxes, and cycle LPIPS after applying the inverse edit and comparing to I_{src}) together capture whether the edit is confined to the intended object: a method could otherwise place the object correctly while regenerating its appearance or corrupting the rest of the scene. Neither background metric requires a ground-truth target, so both remain available on real images. ***(4) Multi-object correspondence***, evaluated on the multi and overlap splits, tests whether each edit is bound to the right object: a detection occupying a target box is matched to the source objects by DINOv2 identity, and we report the fraction placed correctly (Corr) along with swap, merge, and vanish rates (Appendix A.4). Generated pairs additionally admit FID against the ground-truth targets for overall perceptual quality.

Table 1: **Results on the generated single-object test split.** †: given the ground-truth transform. *: trained on our data with the same backbone and adapter budget.

	Pose fidelity				Identity	Background		Quality
Method	Trans. (m)↓	Yaw (°)↓	Scale (%)↓	IoU↑	DINO↑	LPIPS↓	Cycle↓	FID↓
Instruction-based								
Qwen-Image-Edit	0.84	23.4	39.3	0.270	0.263	0.23	0.33	5.6
FLUX.1 Kontext	0.85	25.3	<u>23.5</u>	0.262	0.217	0.17	<u>0.17</u>	7.9
Explicit 3D geometry†								
Diffusion Handles	0.88	35.8	30.0	0.349	0.193	0.58	0.64	17.1
GeoDiffuser	0.66	28.3	24.9	0.335	0.205	0.39	0.46	4.2
Learned pose conditioning								
OBJect-3DIT	1.19	<u>21.5</u>	43.7	0.323	0.203	0.34	0.35	6.6
Magic Fixup†	0.40	23.5	27.2	0.540	0.288	0.36	0.43	<u>4.0</u>
*Same backbone, other conditioning**								
2D box masks	<u>0.31</u>	28.9	25.8	0.621	0.347	0.19	0.28	4.3
Pose params	0.74	24.1	23.9	<u>0.633</u>	<u>0.371</u>	<u>0.14</u>	0.30	5.7
COMPASS (ours)	**0.27**	**18.0**	**21.3**	**0.765**	**0.493**	**0.10**	**0.12**	**3.1**

Baselines. We compare against three groups (Sec. 2). *Instruction-based*: unmodified Qwen-Image-Edit, isolating our adapter's contribution, and FLUX.1 Kontext (Black Forest Labs et al., 2025), both given a text description of the edit. *Explicit-geometry*: Diffusion Handles (Pandey et al., 2024) and GeoDiffuser (Sajnani et al., 2025), both given the ground-truth transform. *Learned pose-conditioned*: OBJect-3DIT (Michel et al., 2023), given a language description of the transformation, and Magic Fixup (Alzayer et al., 2025), given the source crop pasted at the ground-truth target box. We additionally train two same-backbone, same-data ablations isolating the conditioning representation: *2D box masks* in place of $C_{\mathrm{src}}, C_{\mathrm{tgt}}$ with axis-aligned 2D masks, and *pose parameters*, injecting the 7-DoF target box as a numeric embedding in the text stream.

5.2 Results

We report results on the generated test split (single, multi, overlap), the real-image set built from WildDet3D, and controlled ablations on the conditioning representation. In all tables, **bold** marks the best and <u>underline</u> the second-best result among the compared methods. Unless stated otherwise, learned baselines and ablations are trained on our data with the same backbone, adapter budget, and schedule (Appendix A.3).

Single-object editing on generated dataset. Table 1 compares against the baseline groups above. The two instruction-based editors move the object in roughly the same direction but land far from the target box, with large yaw errors. The explicit-geometry methods receive the exact transform, yet they have the two worst background LPIPS scores in the table, and Diffusion Handles also has the worst yaw error. Magic Fixup is the strongest prior method on translation (0.40 m) and IoU (0.540), but its yaw error (23.5°) is no better than the instruction-based editors, since the pasted 2D crop shows the object from its original viewpoint and carries no signal about the change in orientation.

The two same-backbone variants separate the two things the conditioning must convey. 2D box masks nearly match our translation error but have the highest yaw error of any learned method: an axis-aligned box says where the object goes but nothing about how it is turned. Pose parameters fail the other way, with identity and background preservation close to ours but translation error nearly tripling: a 7-DoF vector in the text stream is read as a soft hint rather than a spatial target. COMPASS is best on every column, with the largest margins on IoU and DINO — the two metrics that couple pose and identity, and where a pixel-aligned proxy of the target object should help most. Fig. 3 shows qualitative examples.

Multi-object and overlapping edits. Because every metric here is derived from a detector, the detector caps achievable scores: even the ground-truth targets yield only 89.0%/88.0% correspon-

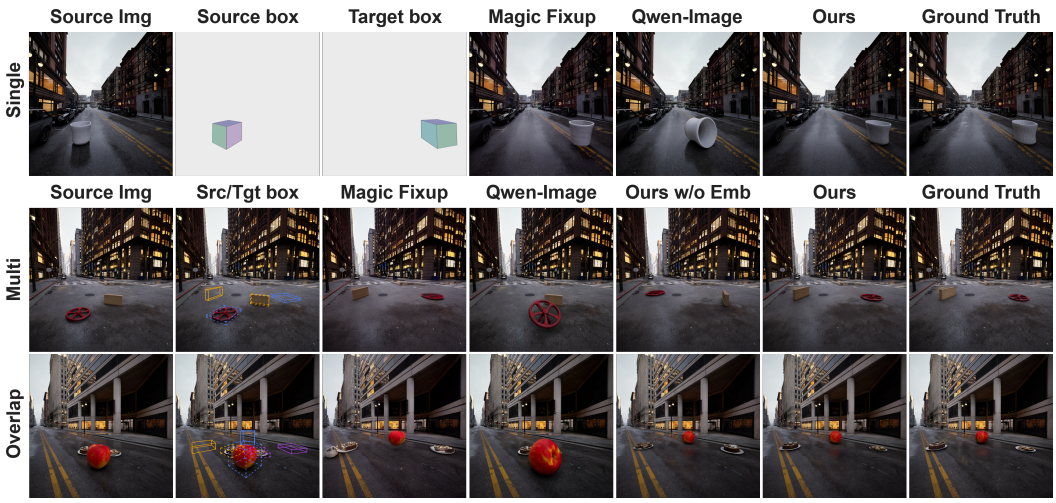

Figure 3: Qualitative comparison on generated dataset: single-obj., multi-obj., and overlap splits.

Table 2: **Results on the generated multi-object and overlap test splits** (split definitions in Sec. 4). Corr.: fraction of objects placed in their own target box; Swap/Merge/Vanish: failure rates (%); protocol in Appendix A.4. Trans./Yaw are computed over Corr objects only. GT ref.: the same detector run on ground-truth target images, which bounds the achievable scores; excluded from the ranking together with the w/o IDs ablation. †: given the ground-truth transform. *: same backbone, other conditioning.

Split	Method	Trans. (m)↓	Yaw (°)↓	Corr. (%)↑	Swap↓	Merge↓	Vanish↓
Multi	GT ref.	0.32	16.7	89.0	1.8	1.0	9.2
	Magic Fixup†	0.53	24.7	<u>56.1</u>	17.2	2.2	<u>26.8</u>
	Pose params*	<u>0.52</u>	<u>22.7</u>	55.2	<u>13.9</u>	<u>1.9</u>	30.9
	Ours w/o IDs	0.48	20.1	65.9	14.3	1.6	19.8
	Ours	**0.42**	**18.3**	**75.8**	**11.6**	**1.0**	**12.6**
Overlap	GT ref.	0.25	14.9	88.0	3.4	1.3	8.6
	Magic Fixup†	0.54	23.2	45.9	25.1	2.6	29.0
	Pose params*	<u>0.41</u>	<u>20.9</u>	<u>66.5</u>	<u>19.0</u>	<u>1.7</u>	<u>14.5</u>
	Ours w/o IDs	0.39	18.8	63.7	17.1	1.4	19.2
	Ours	**0.28**	**17.4**	**70.2**	**15.7**	**1.2**	**14.2**

dence and nonzero merge/vanish rates on multi/overlap, despite no object actually merging or disappearing. Table 2 reports this *detector reference* (GT ref.) alongside each method's pose fidelity, correspondence accuracy, and failure rates, so all methods should be read relative to it rather than to 100%.

Magic Fixup, whose pasted crops carry no notion of which target belongs to which source, assigns only 45.9% of overlap objects to the correct target. The pose-parameter variant trained on our data performs comparably on the multi split and better on overlap, but the binding between the box's appearance and its pose vector is implicit: its swap rate rises from 13.9% to 19.0% once boxes intersect. COMPASS approaches the detector reference on merge and vanish rates, so its remaining gap to the reference (13.2 and 17.8 points) is mostly swaps — a correspondence error, not a synthesis one. Removing the role and object-ID embeddings ("w/o IDs") costs 9.9 and 6.5 points of correspondence on multi and overlap, with the loss on overlap concentrated in swaps and vanishes, confirming that the binding is supplied by the embeddings rather than by the pixel-domain conditioning itself.

Real images. Table 3 reports results for editing real images. Absolute errors are far larger than on generated data because WildDet3D scenes span tens of metres and the evaluation detector is correspondingly noisier, so again the numbers should be read relative to each other. COMPASS



Figure 4: Qualitative comparison on real images.

Table 3: **Real-image evaluation on the WildDet3D-derived set.** Metrics that require a ground-truth target are omitted. Pose metrics are computed over the n samples in which the detector re-identifies the edited object; "Miss" is the fraction where it does not. †: given the ground-truth transform. *: same backbone, other conditioning.

	Pose fidelity			Removal	Identity	Background LPIPS		Detection	
Method	Trans. (m)↓	Yaw (°)↓	Scale (%)↓	rate↑	DINO↑	Bg↓	Cycle↓	n	Miss (%)↓
Qwen-Image-Edit	9.39	38.0	51.0	44.3	**0.352**	**0.23**	0.34	106	77.6
GeoDiffuser†	11.83	41.9	54.4	37.3	0.264	0.31	0.37	86	81.9
Magic Fixup†	7.93	31.5	42.9	42.4	0.263	0.31	0.38	150	68.4
2D box masks*	8.16	40.6	44.1	40.7	0.292	0.25	0.33	212	55.3
Pose params*	10.31	28.3	39.8	44.2	0.301	0.24	0.31	99	79.1
COMPASS (ours)	**4.15**	**25.6**	**33.4**	**54.0**	0.310	**0.23**	**0.29**	**277**	**41.6**

roughly halves the translation error of the strongest baseline (4.15 vs. 7.93 m for Magic Fixup), lowers yaw error to 25.6° (from 28.3°), and is the only method that re-identifies more than half of the edited objects at the target (miss rate 41.6% against 55.3% for the next best). The removal rate shows that 54.0% of objects left their original location, against 44.3% for the frozen backbone, which tends to leave the object in place — consistent with its higher DINO score, and reflecting an editor that changes little rather than one that preserves identity through a real move. Cycle LPIPS of COMPASS is higher than on generated data, reflecting harder invertibility on real images, but remains the lowest in the table.

6 Conclusion

We presented COMPASS, which conditions image editing on rendered 3D bounding-box proxies rather than language or an abstract pose vector, expressing rigid transformations as local appearance differences that a backbone can follow directly. The biggest challenge introduced by this representation is correspondence hallucination, which we resolve using two lightweight role and object-ID embeddings, turning multi-object and overlapping edits into a single forward pass in which depth-ordered IDs also fix which object occludes which. To train and evaluate the model, we built a new dataset that consists of photorealistic paired images with exact transformations, per-object boxes, and masks, and a held-out benchmark with single-object, multi-object, and overlapping-box splits. Trained on this dataset, COMPASS outperforms instruction-based, explicit-3D, and pose-conditioned baselines on every metric of the benchmark and transfers zero-shot to real photographs from WildDet3D. Our results mark a substantial leap over prior work on precise 3D spatial editing. In future work, we plan to improve lighting interaction under strong directional light and texture synthesis on newly exposed surfaces.

AI USE STATEMENT

In this work, we did not use generative AI tools for any task in the required-disclosure category: we did not use them to generate synthetic datasets, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean or reformat datasets, or interpret results. The remaining required-disclosure tasks (developing theoretical models or conceptual frameworks, formulating mathematical claims, providing ingredients for or writing proofs, translation, and qualitative or thematic data analysis) do not apply to this work. Additionally, we used generative AI tools to edit the manuscript for readability, i.e., to polish grammar, wording, and sentence flow of text written by the authors. No other recommended disclosure tasks (e.g., drafting sections, generating code or figures, literature search, or reference formatting) were performed with generative AI. We have reviewed all AI-assisted edits: every suggested change was checked by the authors against the original text and the reported results to ensure no technical claim, number, or citation was altered. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure that our results are reproducible. All experiments, including data generation, training, and evaluation, were run with a fixed random seed of 0. Complete implementation details are provided in the appendix: Appendix A.3 lists all model and optimization hyperparameters, including the LoRA configuration, learning rate schedule, batch size, and number of training steps. Together, these details are sufficient to reproduce all results in the paper.

REFERENCES

Hadi Alzayer, Zhihao Xia, Xuaner (Cecilia) Zhang, Eli Shechtman, Jia-Bin Huang, and Michael Gharbi. Magic fixup: Streamlining photo editing by watching dynamic videos. *ACM Transactions on Graphics*, 44(5), July 2025. ISSN 0730-0301. doi: 10.1145/3750722. URL https://doi.org/10.1145/3750722.

Pradhaan S Bhat, Naveen Chandra R, Rishubh Parihar, Vaibhav Vavilala, R. Venkatesh Babu, D. A. Forsyth, and Anand Bhattad. Thinking in boxes: 3D editing in real images made easy. *arXiv preprint arXiv:2606.20556*, 2026.

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.

Blender Online Community. *Blender – a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3D: A large benchmark and model for 3D object detection in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13154–13164, 2023. arXiv:2207.10660.

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 (CVPR)*, pp. 18392–18402, 2023. arXiv:2211.09800.

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16123–16133, 2022. arXiv:2112.07945.

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.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13142–13153, 2023. arXiv:2212.08051.

Epic Games. City sample. Unreal Engine 5 sample project, 2022. URL `https://www.fab.com/listings/4898e707-7855-404b-af0e-a505ee690e68`. Accessed 2026-09.

Epic Games. Unreal engine, 2024. URL `https://www.unrealengine.com`. Version 5.

Ayaan Haque, Matthew Tancik, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19740–19750, 2023. arXiv:2303.12789.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

Weikai Huang, Jieyu Zhang, Sijun Li, Taoyang Jia, Jiafei Duan, Yunqian Cheng, Jaemin Cho, Matthew Wallingford, Rustin Soraki, Chris Dongjoo Kim, Shuo Liu, Donovan Clay, Taira Anderson, Winson Han, Ali Farhadi, Bharath Hariharan, Zhongzheng Ren, and Ranjay Krishna. WildDet3D: Scaling promptable 3D detection in the wild. *arXiv preprint arXiv:2604.08626*, 2026.

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. arXiv:2301.07093.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.02747.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3D object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9298–9309, 2023. arXiv:2303.11328.

Yingzhou Lu, Lulu Chen, Yuanyuan Zhang, Minjie Shen, Huazheng Wang, Xiao Wang, Capucine van Rechem, Tianfan Fu, and Wenqi Wei. Machine learning for synthetic data generation: A review. *arXiv preprint arXiv:2302.04062*, 2023.

Oscar Michel, Anand Bhattad, Eli VanderBilt, Ranjay Krishna, Aniruddha Kembhavi, and Tanmay Gupta. OBJECT 3DIT: Language-guided 3D-aware image editing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2307.11073.

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2I-Adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 4296–4304, 2024. doi: 10.1609/aaai.v38i5.28226. arXiv:2302.08453.

Michael Nitsche. Experiments in the use of game technology for pre-visualization. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, pp. 160–165, 2008. doi: 10.1145/1496984.1497011.

Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your GAN: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. doi: 10.1145/3588432.3591500. arXiv:2305.10973.

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 (CVPR)*, pp. 7695–7704, 2024. arXiv:2312.02190.

Penghui Ruan, Bojia Zi, Xianbiao Qi, Youze Huang, Rong Xiao, Pichao Wang, Jiannong Cao, and Yuhui Shi. Ctrl&Shift: High-quality geometry-aware object manipulation in visual generation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2602.11440.

Rahul Sajnani, Jeroen Vanbaar, Jie Min, Kapil Katyal, and Srinath Sridhar. GeoDiffuser: Geometry-based image editing with diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025. arXiv:2404.14403.

Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent Y. F. 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 (CVPR)*, pp. 8839–8849, 2024. arXiv:2306.14435.

Muhammad Tukur, Yehia Boraey, Sara Jashari, Alberto Jaspe Villanueva, Uzair Shah, Mahmood Al-Zubaidi, Giovanni Pintore, Enrico Gobbetti, Jens Schneider, Marco Agus, and Noora Fetais. Virtual staging technologies for the metaverse. In *2024 2nd International Conference on Intelligent Metaverse Technologies & Applications (iMETA)*, pp. 206–213, 2024. doi: 10.1109/iMETA62882.2024.10807986.

Feifei Wang, Shiyuan Yang, Xiaoyu Li, and Jing Liao. BoxCtrl: 3D-aware visual prompting for geometric image editing. *arXiv preprint arXiv:2606.23270*, 2026.

He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2642–2651, 2019. arXiv:1901.02970.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-Image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.09292.

Jiraphon Yenphraphai, Xichen Pan, Sainan Liu, Daniele Panozzo, and Saining Xie. Image sculpting: Precise object editing with 3D geometry control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4241–4251, 2024. arXiv:2401.01702.

Xin Yu, Tianyu Wang, Soo Ye Kim, Paul Guerrero, Xi Chen, Qing Liu, Zhe Lin, and Xiaojuan Qi. ObjectMover: Generative object movement with video prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17682–17691, 2025. arXiv:2503.08037.

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. arXiv:2302.05543.

A Appendix

A.1 Data Generation

Scene and assets. We render training pairs in Unreal Engine 5 using the *City Sample* project (Epic Games, 2022), a photorealistic urban environment with Lumen global illumination, Nanite geometry, and dense street-level props. Edited objects come from two sources, each selected for half of the samples. The first is the set of native City Sample props. We enumerate every instanced static mesh in the level, exclude meshes whose names identify them as world geometry (buildings, roads, sidewalks, walls, curbs, roofs, decals, water, sky), keep vehicles, and retain instances whose scaled bounding box has a longest side in $[50, 1200]$ cm. This yields street furniture, signage, vehicles, and vegetation with a fixed, artist-authored metric size. The second is Objaverse 1.0 (Deitke et al., 2023), restricted to assets with LVIS category labels and licenses that permit redistribution and derivative use. Objaverse meshes have arbitrary native units, so every asset carries a precomputed normalization scale that brings its longest side to 100 cm (see below); the size tests below are applied to the mesh *after* this scale. We discard Objaverse assets whose materials render incorrectly in UE5 (translucent sections, `KHR_materials_unlit`, missing base color, or a large fraction of black pixels), assets flagged as scene-like or with a far pivot, and assets with a very flat aspect ratio; exact counts are given below. To avoid the sampling being dominated by frequent meshes (the level contains more than $20{,}000$ instances of a single bollard mesh), we group City Sample instances by mesh and Objaverse assets by LVIS category and draw a group uniformly at random before drawing an instance within it. Objaverse assets are *streamed*: only the meshes being edited in the current pair are resident in the level at a time, so the background never contains other Objaverse objects and the asset pool is not limited by GPU memory.

Objaverse asset preparation. We start from the $21{,}019$ Objaverse 1.0 assets that carry one of $1{,}145$ LVIS category labels and import each into UE5 once. Objaverse meshes come in arbitrary native units (the longest side of the imported meshes ranges from a fraction of a centimetre to 10^8 cm), so every asset is assigned a fixed normalization scale that brings its longest side to exactly 100 cm; the same scale is reused for every sample in which the asset appears, and the sample-level scale factor $\Delta\sigma$ of Sec. 4 is applied on top of it. We deliberately do not restore category-specific real-world sizes: LVIS categories span five orders of magnitude (from peas and thumbtacks to ships and aircraft), and most would either fall outside the camera-distance window of the capture pipeline or be unrecognizable at street level. A uniform metric size keeps every asset inside the same distance and pixel-size window, so the visual diversity of the pool is fully usable, at the cost of some objects appearing at an implausible scale. Assets keep their native orientation; the box is the axis-aligned extent of the mesh in its imported frame, and "front" denotes a fixed face of that frame rather than a semantically identified front. Since the yaw of every placement and every edit is sampled uniformly, no canonical facing is needed for training, and the yaw metric folds category-level symmetries at evaluation time (Appendix A.4).

From these $21{,}019$ assets we keep $15{,}954$ in $1{,}134$ categories after filtering: $18{,}232$ remain after the license whitelist ($635{,}544$ UIDs whose Sketchfab license permits redistribution and derivative use), $18{,}202$ after removing assets whose materials import as fully black or fully transparent, $17{,}342$ after removing scene-like assets and assets with a far pivot, $17{,}029$ after removing assets that render more than 34% black, and $15{,}954$ after requiring an aspect ratio (shortest over longest side) of at least 0.08. Assets with translucent or unlit material sections are dropped at capture time for the rendering reasons given above, leaving roughly $13{,}700$ usable assets, which are then split between train and test by UID.

Placement and camera sampling. For each pair we first draw the number of edited objects $K \in \{1, 2, 3\}$ according to the mixture in Table 4, and for each object decide with probability 0.5 whether it is a City Sample prop or an Objaverse asset. City Sample props are used in place: we pick an instance as described above and search for a camera around it. Objaverse assets are placed on demand: the next asset from the seeded, shuffled pool is placed at a random street-level position within the current split's city region (yaw $\sim \mathcal{U}[0°, 360°)$) and snapped to the ground, with up to 40 positions tried per asset. Ground placement requires that all four footprint corners hit a surface tagged as road, sidewalk, or ground with normal $z \geq 0.7$, that the footprint lies on a single plane (height variation ≤ 30 cm), that the surface is above water level, and that the placed box neither

intersects scene collision nor overlaps any other instance in the scene database, including the objects already placed for the current pair. We then sample an eye-level camera (height $170\,\text{cm}$, pitch $\sim \mathcal{U}[-5°, 5°]$, FOV $90°$) at a distance $d \in [d_{\text{near}}, d_{\text{far}}]$ from the objects, with $d_{\text{near}} = 2.5\,\text{m}$ and $d_{\text{far}} = 0.85\, d_{96}$, where d_{96} is the distance at which the object's longest side would project to exactly $96\,\text{px}$; the factor 0.85 keeps samples away from the acceptance boundary. The camera's azimuth is uniform, and its yaw is aimed at the objects with a random offset of $\pm 25°$ so the targets are not always centered. Cameras inside any object's bounding box or at the edge of the loaded world are rejected. Up to 12 cameras are tried per object before a new object (or, for Objaverse, a new placement) is drawn.

Edit sampling. Given a valid (objects, camera) configuration, we sample an independent rigid edit for every object, consisting of a planar translation with distance $\sim \mathcal{U}[\max(200, 2.2r), 1200]\,\text{cm}$ (where r is the object's footprint radius) in a uniformly random direction, a yaw change $\Delta\theta \sim \mathcal{U}[-90°, 90°]$, and a uniform scale factor $\Delta\sigma \sim \mathcal{U}[0.7, 1.4]$. Each edited object is re-snapped to the ground at its new location under the same ground constraints. An edit is accepted only if both the source and target boxes are fully in frame, project to at least $96 \times 12\,\text{px}$, cover at least 0.8% of the image by silhouette area, have at least 35% of 15 probe points (8 corners, 6 face centers, centroid) unoccluded by other geometry, and do not collide in 3D (boxes shrunk by 10% for the test) with any scene object or with the source and target boxes of the other edited objects (overlap in the 2D projection is allowed; it is what defines the *Overlap* split, Sec. 4). Up to 30 edits are tried per camera; the whole procedure is retried with a new placement until a pair is produced, so every sample index is filled.

Rendering. Each pair yields a source image, a target image, and the corresponding source and target proxy-box images at 1024×1024. To minimize differences between source and target outside the edited region, we disable all temporally-accumulated rendering: anti-aliasing is set to SMAA (a purely spatial method), temporal jitter and temporal upsampling are disabled, motion blur and depth of field are turned off, textures are fully streamed at mip 0, and virtual shadow map caching is disabled so that moving the object does not leave stale shadows. Residual frame-to-frame variation from Lumen's stochastic global illumination cannot be fully eliminated; we account for it in the pixel-level verification described below. After each camera move we wait at least 2 s and 12 frames before capturing; after applying the edit we wait at least 10 s and 92 frames so that Lumen surface caches and shadow maps re-converge at the new object location. World-partition HLOD proxies are hidden to avoid coarse placeholder geometry in the background.

Proxy-box images. Proxy boxes are rendered analytically from the camera parameters at $2\times$ supersampling and downsampled with a Lanczos filter. Each of the six faces has a fixed color (front, back, left, right, top, bottom), back-facing and grazing faces (view-direction dot product ≤ 0.03) are culled, faces smaller than $30\,\text{px}^2$ are dropped, and the remaining faces are painter-sorted by depth and drawn with a dark edge on a uniform light-gray background.

Per-object masks and quality filtering. We additionally record pixel-accurate object masks by enabling UE's custom-depth pass for the target component and reading it from the screenshot's alpha channel, intersected with the projected box region. Pairs are rejected if the true visible area of the object is below 0.3% of the image on either side, or if the object renders as untextured black (mask-interior mean intensity $\leq 8/255$ and contrast against a 7-px surrounding ring ≤ 0.15). Finally, we verify that the edit is actually visible: after both screenshots are written, we compute the per-pixel grayscale difference and require that at least 15% of pixels inside the union of the source and target masks change by more than $12/255$, that the mean difference in this region is at least $5/255$, that the in-region change rate is at least $2\times$ the background change rate, and that changed pixels cover at least 0.3% of the image. Rejected pairs are deleted and resampled.

Annotations and splits. Every pair is accompanied by a JSON record containing the camera pose and intrinsics, and, for each edited object, the source and target 3D boxes (center, yaw, half-extents) and 2D projections, the exact edit parameters, the asset identifier and category (Objaverse UID, LVIS category, and license for Objaverse assets; asset path and category for City Sample props), per-side mask statistics, and all quality metrics used for filtering. Objects are listed in order of increasing depth of the source box center, which is also the order in which they are assigned to object slots

at training time (Appendix A.2). Train and test splits use disjoint sets of assets from both sources (disjoint Objaverse UIDs and disjoint City Sample instances) and disjoint regions of the city: the map is divided into a checkerboard of square cells, cells are assigned to train or test by parity, and cameras and placements must stay a fixed margin away from cell borders so that a test camera never sees a training cell in its background. Neither the edited objects nor the backgrounds are therefore shared across splits. The capture is fully deterministic given the capture seed, the asset split file, and the level.

Real-image test set: source data. The real-image evaluation of Sec. 5 uses WildDet3D (Huang et al., 2026). We build on its public evaluation split (`InTheWild_v3_val`), which contains $2{,}470$ images with $43{,}867$ annotated boxes; of these, only the $9{,}256$ boxes flagged as having valid 3D annotations carry usable 3D fields, and we discard the rest. No image or box from this split is used for training.

Real-image test set: box canonicalization. WildDet3D boxes are given as a rotation $R \in SO(3)$ and three extents, whereas our model expects the gravity-aligned 7-DoF parameterization of Eq. equation 1. We convert each box as follows. The three columns of R span the object's length, height, and width, respectively; since the camera frame has y pointing down and the vast majority of annotated boxes have their $+y$ column pointing down as well, we take the object's up direction as the negated height column and its support face as the $+y$ face. Although the dataset is described as gravity-aligned, the up directions of boxes within the same image disagree by a median of 5.3° (p90 18.7°), so we do not use per-box up vectors directly. Instead, for every image we estimate a single ground frame: the up vector is the spherical geometric median (Weiszfeld iterations) of the up vectors of all valid boxes in the image, and the zero-yaw direction is the projection of the camera's optical axis onto the resulting ground plane. Each box's yaw is then the azimuth of its length axis after projection onto this plane, and its per-box pitch and roll are dropped. The cost of this projection is small: box corners move by 2.6 cm on average (p90 20 cm), and the median IoU between the original and canonicalized projected silhouettes is 0.946. The dimensions are taken directly from the annotations; no category size prior is applied at any stage.

Real-image test set: source-object filtering. From the $9{,}256$ valid boxes we keep $1{,}544$ candidate source objects. A box is retained if its category is in a whitelist of movable object categories (animals and people are excluded, and synonymous labels are merged), its projected long side is at least 96 px and its short side at least 12 px, its support face is inclined by less than 15° to the estimated ground, its depth lies in $(0, 60]$ m, its full 3D box projects inside the image (tested with our own projection, since the released 2D boxes are clipped), and its silhouette covers between 0.8% and 50% of the image. To limit redundancy we draw at most two source objects per image and cap any single category at 10% of the set. The official truncation and visibility fields are placeholders in this split, so no occlusion-based filtering is applied.

Real-image test set: target sampling. For every source object we sample a target box from the training edit distribution: an isotropic scale factor $\Delta\sigma \sim \mathcal{U}[0.7, 1.4]$, a yaw change $\Delta\theta \sim \mathcal{U}[-90^\circ, 90^\circ]$, and a planar translation in a uniformly random direction within the ground plane with distance $\sim \mathcal{U}[\max(2, 2.2r), 12]$ m, where r is the source footprint radius. After the transform, the target box is re-seated on the plane of the source object's support face, so rescaling grows the object upward from the ground rather than about its center, as in the synthetic data. A target is rejected if it leaves the frame or violates the size and area limits above, if its 2D IoU with any other annotated box exceeds 0.05, if its 3D footprint intersects the source footprint, or if more than half of it lies behind scene surfaces according to a monocular depth map (so that objects are not placed inside walls). Up to 120 targets are tried per source object. Finally, since the target is seated on the plane of the source object's support face, we verify that this plane still corresponds to ground at the target location: samples where the target's bottom face lies more than 0.25 m above the depth-map surface beneath it are discarded. The resulting set contains 474 single-object samples, which is the set reported in Table 3.

A.2 Model Architecture

Backbone. COMPASS is built on Qwen-Image-Edit (Wu et al., 2025), which consists of three components: an MMDiT denoiser, a causal video VAE that we apply to single frames, and a Qwen2.5-VL encoder that produces the text/vision context for the MMDiT's text stream. All three are frozen. In particular, we leave the MMDiT input projection (`img_in`) untouched: our conditioning is injected *before* this projection through zero-initialized additive embeddings (see below), so at initialization the network computes exactly the same function as the pretrained model. All trainable parameters are LoRA adapters on the attention projections of the image stream plus the role and object-ID embeddings described below.

Latent tokenization. Images are processed at 512×512. The VAE encoder (spatial stride 8) maps an image to a $64 \times 64 \times 16$ latent; we use the mode of the posterior rather than a sample. Latents are normalized per channel with the pretrained mean and standard deviation, then packed 2×2 spatially into a 32×32 grid of 64-dimensional tokens, yielding 1024 tokens per image. Each token therefore corresponds to a 16×16 pixel patch; the loss weighting in Appendix A.3 is defined at this granularity.

Sequence construction. Let z_σ denote the noised target latent, x_{src} the source-image latent, and $(c_k^{\text{src}}, c_k^{\text{tgt}})$ the proxy-box renderings of object k before and after the edit, all tokenized as above. For a record with K edited objects ($1 \le K \le K_{\max}$, $K_{\max} = 3$), the image-stream input is the concatenation

$$\mathbf{S} = \left[\, z_\sigma;\ x_{\text{src}};\ c_1^{\text{src}};\ c_1^{\text{tgt}};\ \ldots;\ c_K^{\text{src}};\ c_K^{\text{tgt}} \,\right], \tag{6}$$

i.e., $2 + 2K$ segments of 1024 tokens each. The sequence length therefore depends on K: a single-object record has 4096 image-stream tokens and a three-object record 8192. We do not pad records with fewer than $K_{\max}$ objects; unused slots are simply absent from the sequence. Each segment is registered as a separate frame in the MMDiT's 3D rotary embedding (`img_shapes` $= [(1, 32, 32)] \times (2 + 2K)$), so tokens at the same spatial location in different segments share their spatial RoPE coordinates but receive distinct frame indices. Only the first 1024 output tokens—those aligned with z_σ—are supervised; outputs at the remaining positions are discarded.

Role and object-ID embeddings. The MMDiT has no built-in notion of which segment a token belongs to beyond the frame index. We therefore add the two learned embeddings of Eq. 4 to the 64-dimensional packed tokens of the box segments before `img_in`:

$$\tilde{t} = t + e^{\text{role}}_{r(t)} + e^{\text{id}}_{k(t)}, \tag{7}$$

where $r(t) \in \{\text{src}, \text{tgt}\}$ identifies whether the token belongs to a source- or target-box rendering and $k(t) \in \{1, \ldots, K_{\max}\}$ identifies the object. Neither embedding is added to z_σ or x_{src}; these two segments are identified by their frame index alone. Both tables, $e^{\text{role}} \in \mathbb{R}^{2\times 64}$ and $e^{\text{id}} \in \mathbb{R}^{K_{\max}\times 64}$, are zero-initialized, so Eq. 7 reduces to the identity at the start of training and the pretrained model's behavior is preserved exactly. For $K_{\max} = 3$ this adds 320 parameters in total. These parameters are stored and updated in fp32 rather than bf16: bf16 has a 7-bit mantissa, so once an embedding entry grows to $\mathcal{O}(10^{-2})$, per-step updates of magnitude $\sim 10^{-4}$ (learning rate 10^{-4}) fall below the representable increment and are rounded away, leaving the embeddings stuck near their initialization.

Depth-ordered object IDs. Object IDs are not assigned arbitrarily. Following Sec. 3.3, objects in a record are sorted by increasing depth of their source box center in the camera frame, and object k receives ID k, so $k = 1$ is always the object nearest the camera. The same ordering is used at training and test time and is stored in the data records (Appendix A.1). Because the ordering is deterministic, the ID embedding carries two pieces of information at once: it binds the source and target box of the same object, and it encodes the depth rank, which tells the model which object should occlude the other when two target boxes overlap in the image. A consequence is that ID k only receives gradient from records with at least k objects; with one third of the training records containing two or three objects (Table 4), all three ID embeddings are trained.

Constant VL context. Qwen-Image-Edit conditions the MMDiT's text stream on Qwen2.5-VL hidden states computed from the editing instruction and the source image. In our setting the edit

is fully specified by the proxy-box geometry and the source image is already present in the image stream as $\mathbf{x}_{\text{src}}$, so the VL context carries no sample-specific information that the model needs. We therefore run the VL encoder once on a single fixed prompt with no image input, drop the first 64 tokens of the chat template as in the reference implementation, and reuse the resulting 331×3584 context tensor for every sample. Because the context is independent of the sample, this introduces no approximation for our task, and it removes the VL encoder from the training loop entirely. It also avoids caching a per-sample context, which at 331×3584 bf16 values per image would require roughly 71 GB for the $30{,}000$ training records.

A.3 TRAINING

Data mixture. Table 4 summarizes the training and held-out data, grouped by the number of edited objects K per record. The training set contains $30{,}000$ records, of which one third have $K \geq 2$; $7{,}000$ records are held out for evaluation and never seen during training. We do not oversample any group: records are drawn uniformly from the concatenated training set.

Objects per record	Train	Held-out
$K = 1$	20,000	5,000
$K = 2$	5,000	1,000
$K = 3$	5,000	1,000
Total	30,000	7,000

Table 4: Data mixture by number of edited objects. No group is oversampled.

Data augmentation. No data augmentation is applied. We do not flip or crop, since both would break the correspondence between the proxy boxes and the scene geometry, and we do not permute object IDs, since the ID order is defined by depth (Appendix A.2) and permuting it would destroy the occlusion cue it carries.

Loss weighting. The object-weighted objective of Eq. 5 uses $\lambda = 3$ inside the target-object region and weight 1 elsewhere. The region Ω is obtained by downsampling the union of the per-object target masks to the 32×32 token grid of Appendix A.2; a token is assigned to Ω if any pixel of its 16×16 patch belongs to a target mask. For multi-object records the weight is applied to all edited objects with the same λ.

Trainable parameters. We apply LoRA (Hu et al., 2022) with rank $r = 32$ and $\alpha = 32$ to the four attention projections (query, key, value, and output) of the image stream in each MMDiT block, for a total of 47.19M trainable parameters. Together with the 320 embedding parameters of Eq. 7, these are the only weights updated during training; the text-stream projections, the VAE, the VL encoder, and all remaining MMDiT weights are frozen.

Optimization. We use AdamW with learning rate 10^{-4}, weight decay 10^{-2}, $(\beta_1, \beta_2) = (0.9, 0.999)$, a linear warmup of 200 optimizer steps followed by cosine decay, and gradient-norm clipping at 1.0. Records with different K have different sequence lengths (Eq. 6) and are not padded, so the per-GPU micro-batch is a single record; with 8 GPUs and 4 gradient-accumulation steps the effective batch size is 32. Training runs in bf16 autocast without a loss scaler, with gradient checkpointing enabled on all MMDiT blocks. Two implementation details matter for reproducing the reported numbers: (i) the learning-rate scheduler is stepped once per *optimizer* step, not once per micro-batch, so the warmup and decay are defined in units of effective batches; and (ii) the role and object-ID embeddings are kept outside the DDP-wrapped module (so that they can be held in fp32 while the rest of the model runs in bf16) and their gradients are all-reduced manually across ranks before each optimizer step. We train for a total of $20{,}250$ optimizer steps.

Compute. All experiments use $8\times$ NVIDIA H100 80GB GPUs. At the configuration above one optimizer step takes approximately 8.6 s, and the full run of $20{,}250$ steps required roughly 65 h of wall-clock time (18.3 h $+ 46.3$ h across two segments, the second resumed from a checkpoint of the first), of which about 16 h were spent on periodic evaluation during training.

A.4 Evaluation Protocol

Metric definitions follow Sec. 5; here we give the details of the pose-fidelity evaluation, which depends on an off-the-shelf 3D detector (Huang et al., 2026) and therefore requires several conventions to be fixed. The real-image test set is drawn from this detector's held-out evaluation split, on which it was not trained, and the detector is applied identically to every method.

Detector box convention. The detector returns a box as a rotation matrix and an extent triple. Rather than assume that its output follows the annotation convention of Appendix A.1 (length, height, width), we verify the extent-to-column assignment empirically. We calibrate this once on the ground-truth target images of the synthetic test split: over the six possible assignments of extents to columns, we select the one that simultaneously minimizes yaw and scale error against the annotated target boxes. Exactly one assignment is minimal for both, and it is used unchanged for all methods and both datasets.

Axis alignment. To compare a detected box with $\mathcal{B}_{\mathrm{tgt}}$, the detected axis closest to the ground up vector is taken as the height axis; of the remaining two axes, the longer is taken as the principal horizontal axis. Scale error is computed over the aligned extents, and yaw error is the angle between the principal horizontal axes of the detected and ground-truth boxes, measured in the ground plane.

Symmetry. Yaw error is folded by a per-category symmetry period (default 180°, reflecting the front/back ambiguity of most box-shaped objects). Categories that are rotationally symmetric about the vertical axis (e.g., barrels, cones) are assigned an infinite period and contribute a yaw error of zero. The same table is applied to every method.

Ground anchoring (real images only). On real images the evaluation detector's depth estimate is its least reliable output, and WildDet3D scenes span tens of metres, so depth errors would otherwise dominate the translation metric. Before comparison we therefore rescale each detected box along its viewing ray until its bottom face touches the per-image ground plane estimated in Appendix A.1. This resolves only the depth ambiguity along the ray; the box's image projection is unchanged, so the operation cannot make a wrongly placed object appear correct. No category-dependent size prior is used. On the synthetic split the detector output is compared to the ground-truth box directly, without anchoring.

Matching and misses. The detection assigned to $\mathcal{B}_{\mathrm{tgt}}$ is the one with the highest projected IoU against it; if no detection of the object's category overlaps the target box, the sample counts as a miss and is excluded from the pose averages. The multi-object protocol is described below.

Multi-object correspondence. On the multi and overlap splits we evaluate each edited object through its *slot*, the 2D projection of its target box $\mathcal{B}_{\mathrm{tgt}}$. Correspondence is decided in two steps. *(i) Occupancy.* We run the evaluation detector in text-prompted mode, prompting with the categories of all K objects in the sample, and assign to each slot the detection with the highest 2D IoU against it; a slot is *occupied* if this IoU is at least 0.3 and *empty* otherwise. Occupancy is decided by position alone and does not use the detected category. *(ii) Identity.* For every source object we crop I_{src} by the bounding box of its source mask and extract a DINOv2-L CLS feature; for every occupied slot we crop the output image by the projected target box and do the same. Occupied slots are matched to the K source objects by Hungarian assignment with cost $1 - \cos$ between the two features.

Each slot then falls into exactly one of three exclusive classes: **Corr** (occupied and matched to its own source object), **Swap** (occupied but matched to a different source object, i.e. the right position holds the wrong object), or **Vanish** (empty). These sum to 100%. **Merge** is recorded separately and is not exclusive with the other three: when two or more occupied slots are claimed by the same detection, every slot involved is counted as merged, and is still classified as Corr or Swap by identity. Translation and yaw errors in Table 2 are computed over Corr objects only, since comparing a swapped object against a box that does not belong to it is not meaningful.

Three properties of this protocol matter for reading the numbers. Vanish includes the detector's own misses; on ground-truth target images it is about 9%, which is the GT ref. row of Table 2. Identity is decided relative to the K source objects of the sample, so a slot occupied by an unrelated object

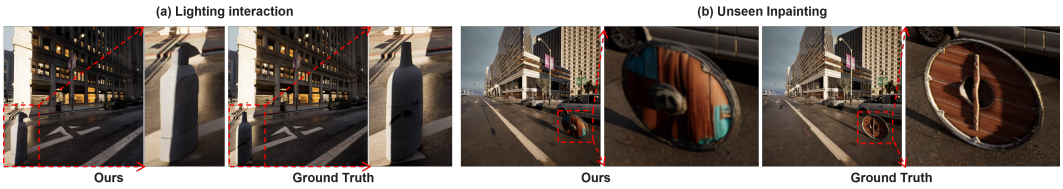

Figure 5: **Failure cases.** Left: ours; right: ground truth; red boxes mark the zoomed regions. **(a) Lighting interaction.** Under strong directional sunlight, the relocated object is rendered with plausible local shading that differs from the ground truth. **(b) Unseen inpainting.** When the edit exposes a surface that is not visible in the source view, the model produces a plausible but incorrect appearance.

is assigned to the most similar source object and counted as Corr or Swap rather than Vanish; for the same reason a sample with a single occupied slot can still produce a Swap. Finally, if a source mask is empty and no feature can be extracted, identity matching is skipped for that sample and all occupied slots are counted as Corr.

A.5 Failure cases and limitations

Figure 5 shows the two dominant failure modes. The first is *lighting interaction* (Fig. 5a). The moved object is re-synthesized with plausible appearance, but its interaction with scene illumination is only approximately recovered: the proxy cuboid conveys pose and extent and nothing about light direction, so cast shadows and the object's own shading are inferred from context and can differ from the ground truth in direction, length, and contrast. The second is *unseen inpainting* (Fig. 5b). When an edit exposes surfaces that were not visible in the source view, the model has to invent their texture; the completion is semantically consistent, but the specific pattern is unconstrained and often differs from the true object. Beyond these two modes, the training set of 30k pairs is small relative to the visual diversity of real photographs, and edits on inputs far from the synthetic distribution, such as non-photorealistic styles (paintings, sketches, cartoons) or unconventional objects with no close counterpart in Objaverse, degrade more noticeably; scaling the data pipeline is the most direct remedy.