BlenderFusion: 3D-Grounded Visual Editing and Generative Compositing

Jiacheng Chen[♦,1,2], **Ramin Mehran**[1], **Xuhui Jia**[1], **Saining Xie**[1,3] **and Sanghyun Woo**[1]
[1]Google DeepMind, [2]Simon Fraser University, [3]New York University, [♦]Work done during an internship at Google Deepmind

We present BlenderFusion, a generative visual compositing framework that synthesizes new scenes by recomposing objects, camera, and background. It follows a layering-editing-compositing pipeline: (i) segmenting and converting visual inputs into editable 3D entities (layering), (ii) editing them in Blender with 3D-grounded control (editing), and (iii) fusing them into a coherent scene using a generative compositor (compositing). Our generative compositor extends a pre-trained diffusion model to process both the original (source) and edited (target) scenes in parallel. It is fine-tuned on video frames with two key training strategies: (i) source masking, enabling flexible modifications like background replacement; (ii) simulated object jittering, facilitating disentangled control over objects and camera. BlenderFusion significantly outperforms prior methods in complex compositional scene editing tasks. See the project page for demos and more results: blenderfusion.github.io

1. Introduction

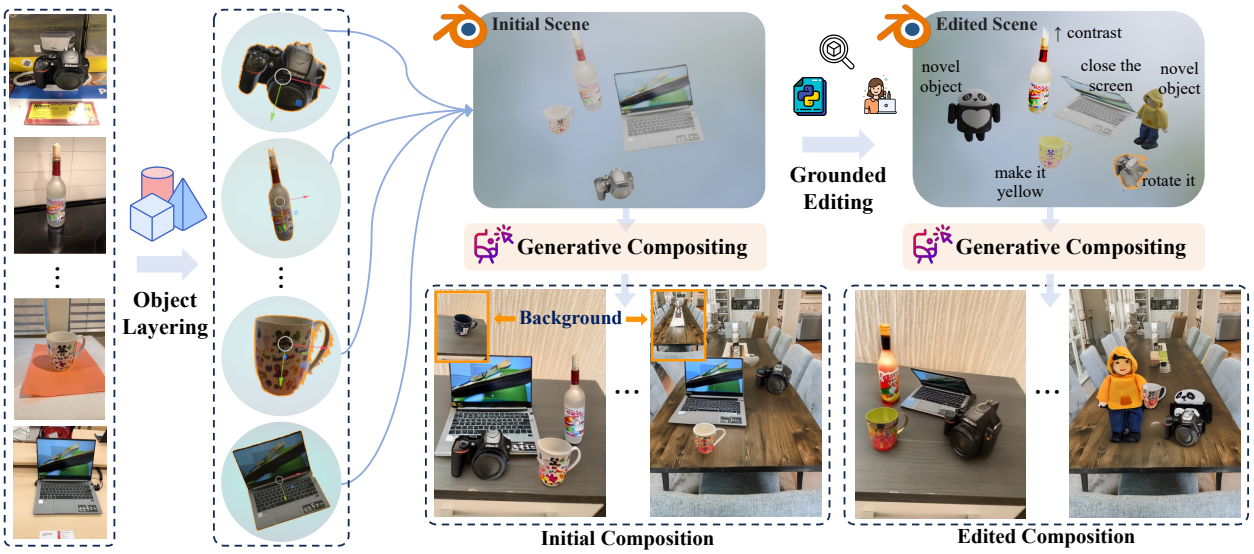

Figure 1 | BlenderFusion integrates the 3D-grounded editing capabilities of graphics software into the strong synthesis abilities of diffusion models. Despite fine-tuned only on video frames of simple object transformations with entangled camera motion, it learns precise object control, inherits Blender's rich editing functionalities (e.g., attribute modification, deformation, novel asset insertion), and generalizes to highly fine-grained multi-object editing and scene composition tasks (Figure 6).

Visual compositing is the process of constructing a scene by extracting objects from multiple images, manipulating their appearance or spatial configuration, inserting them into a new background, and adjusting the camera to produce a cohesive image or video. This enables novel visual narratives with high flexibility. While recent generative AI techniques excel at photorealistic text-to-image synthesis [3, 23, 32, 35, 37], they often fall short in complex compositing scenarios that demand

Correspondence: jca348@sfu.ca, shwoo@google.com

Method	Visual Elements			Object Control				Control Interface
	Obj	Cam	BG	Multi-Obj	Novel-Obj	Attribute Change	Non-rigid Transform	
Object 3DIT [28]	✓	✗	✗	✗	✗	✗	✗	Text
Neural Assets [55]	✓	✓	✓	✓	✗	✗	✗	Obj tokens
Image Sculpting [57]	✓	✗	✗	✗	✓	✓	✓	Blender
BlenderFusion (Ours)	✓	✓	✓	✓	✓	✓	✓	Blender

Table 1 | Capability comparisons between BlenderFusion and existing 3D-aware editing works.

precise, 3D-aware control over multiple scene elements, such as repositioning objects, modifying geometry and appearance, and adjusting viewpoint consistently.

To better characterize visual compositing tasks, we consider two key aspects: (i) the editable visual elements—objects, camera, and background—and (ii) the granularity of object-level control, including multi-object editing, novel object insertion, attribute modification, and non-rigid transformations. Some recent approaches blend objects from multiple input images while preserving identity and following coarse geometric layouts [19, 46], but their control remains implicit, lacking 3D awareness and disentanglement. A more explicit line of work incorporates 3D-aware mechanisms to enhance editing fidelity and controllability.

Object 3DIT [28] enables text-driven 3D-aware edits via synthetic training data but is limited to rigid transformations of single objects. Neural Assets [55] disentangles object and background tokens for multi-object composition and spatial manipulation, yet lacks fine-grained control. Image Sculpting [57] leverages Blender for accurate 3D edits but requires per-scene optimization and is mostly limited to editing a single object from a single input image. Table 1 summarizes these representative methods across the two aspects above, highlighting that none achieves full-scene visual compositing with fine-grained, disentangled control over all core elements.

To address these limitations, we propose BlenderFusion, a unified framework that mirrors the traditional visual compositing process through three fundamental steps (Figure 1). (1) *Layering*: off-the-shelf visual foundation models delineate foreground objects and lift them into 3D-editable entities. (2) *Editing*: native Blender operations enable fine-grained, 3D-grounded modifications of object geometry, appearance, and camera viewpoint. (3) *Compositing*: a diffusion-based visual compositor blends the Blender renderings with a background to produce a coherent image of the edited scene. By decoupling 3D control from image generation, BlenderFusion combines the strengths of graphics-based editing and generative synthesis, enabling flexible, disentangled, and 3D-aware manipulation of objects, camera, and background.

Concretely, our diffusion compositor (Figure 2) operates on two parallel input streams: a source stream containing the original scene and a target stream reflecting the edited scene. To train the compositor effectively, we leverage object-centric videos and introduce two training strategies: (1) *Source masking*: To handle large contextual changes such as object insertion or removal, we mask the modified regions in the source stream, preventing them from interfering with target composition. At test time, this allows flexible masking to expose only the valid context. (2) *Simulated object jittering*: Since training videos are often dominated by camera motion, we introduce a reconstruction training setup that jitters object positions between source and target while keeping the camera fixed, thereby enriching supervision for disentangled object control.

We evaluate our method on three video datasets—MOVi-E [14], Objectron [1], and Waymo Open Dataset [48]. Although these datasets only exhibit basic object and camera motions, our method generalizes to diverse editing scenarios, enabling precise disentanglement of visual elements, fine-grained multi-object control, and complex compositional edits (Figure 1, Figure 6).

2. Related Work

Visual Generation and Control. Modern generative modeling, including Generative Adversarial Networks (GANs) [13, 30] and diffusion models [18, 42, 45], first achieved high-fidelity content generation. The focus then shifted to user control, with natural language emerging as the primary interface [32, 35, 37]. However, text-based control is inherently limited. To address this, specialized methods were developed for more granular control over aspects like geometry [60], subject-driven generation [7, 8, 36], and aesthetics [43]. Recent efforts have moved towards unifying these disparate controls into single, powerful models such as Instruct-Imagen [19], which utilizes a versatile *text + image* $\rightarrow$ *image* framework. Despite these advancements, the reliance on text as the main control interface persists, along with its inherent limitations of ambiguity and difficulty in articulation. Our work addresses this by augmenting the generative model with a graphics engine. This approach leverages the powerful synthesis of diffusion models while gaining the precise control of a graphics engine, effectively disentangling the problems of generation and control.

Visual Compositing. Visual compositing—the process of assembling various visual elements into a single, seamless, and photorealistic image—is a crucial yet challenging task in visual generation. Early diffusion-based methods focused on simple, single-object compositions guided by layouts [9, 27, 47, 58, 59]. While recent works have advanced to handle multiple objects [50] or even manipulate semantic attributes by mixing concepts from different images [12], these approaches remain largely confined to the 2D space. A comprehensive suite for complex, multi-object, and multi-image composition with precise geometric control remains an underexplored problem, which our work addresses.

3D-aware Control. A fundamental challenge in visual generation is achieving 3D-aware control that respects the physical nature of the world. Existing methods are primarily limited to single-object manipulation and often entangle visual elements, which prevents complex scene editing. Approaches such as GeoDiffuser [38] and Diffusion Handle [29] use 3D priors to transform model activations, while Magic Fixup [2] implicitly learns world physics from video frames. More recently, 3D-Fixup [10] explicitly learn this transformation through 3D edits. A key limitation, however, is that these methods are all designed for single object control. As noted in Table 1, NeuralAsset [55] is a notable previous effort that addresses multi-object control, including the background and camera, albeit with some limitations. In contrast, our framework provides a more robust solution by moving editing into a 3D graphics engine. We lift 2D elements into a 3D space for precise manipulation, then reproject the scene to 2D for rendering with a diffusion model, enabling effective and flexible compositing.

Procedural Generation. Aside from generating visuals directly from a model, notable efforts have formulated visual generation as a structured, procedural process. This involves representing the visual as a structured format, such as a Python program or script, and then relying on a graphics engine for rendering. Frameworks such as BlenderAlchemy [15, 21], FirePlace [22], and SceneCraft [20] leverage pre-trained vision-language models (VLMs) to generate desirable Blender python scripts based on the user's edit intent, which is expressed through language. These systems generate and execute code using Blender to produce the final image. In this overall process, to steer the VLM toward desirable outputs, they devise methods to incorporate external information to aid the language model or algorithm for iterative self-correction. Our framework is also loosely connected to procedural generation in the sense that we programmatically edit the projected 3D representation from 2D images using Blender. However, instead of relying on a pretrained language model to interpret user intent, our method empowers users to directly and interactively manipulate the meshes as they desire. The resulting render is then passed to our diffusion model to generate the final, high-fidelity image.

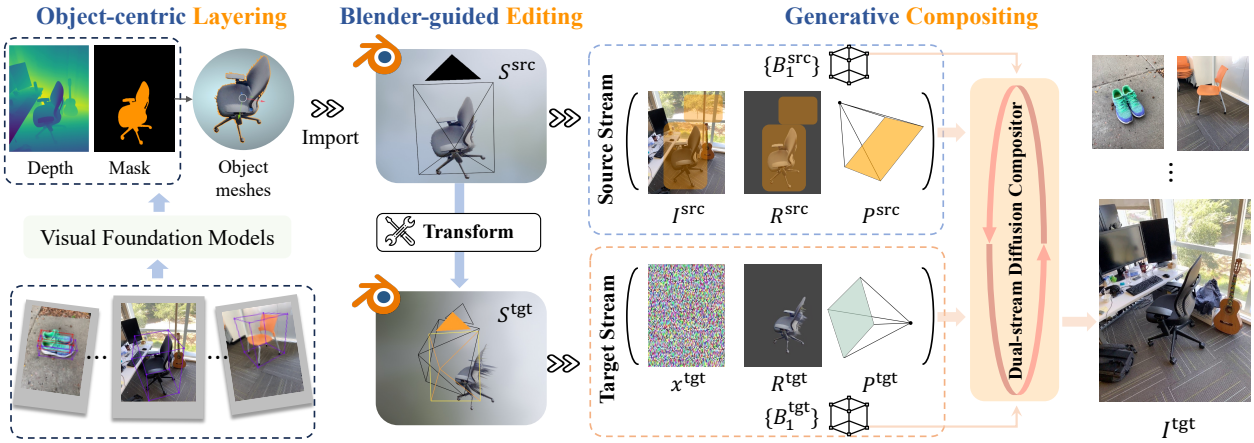

Figure 2 | BlenderFusion employs a layering-editing-compositing process: 1) segment and lift objects from the source images into editable 3D elements; 2) edit the visual elements in Blender to transform the initial scene to the target scene; and 3) use a dual-stream diffusion compositor to blend the target image. The video training data consists of object and camera pose annotations. We use these annotations to simulate the test-time transformations in Blender. The text input for each stream encodes a set of tuples consisting of object category labels and 3D bounding boxes. The source masking strategy, indicated by overlaid orange bounding boxes in I^{src} and R^{src}, is detailed in §3.2.

3. BlenderFusion

The goal of this work is to enable precise, 3D-aware visual compositing by integrating generative models with a symbolic graphics tool, such as Blender, across the three fundamental steps of compositing: layering, editing, and compositing. Concretely, we obtain editable 3D object entities from multiple images (layering), import them into Blender for versatile 3D-grounded scene modifications (editing), and use a diffusion-based visual compositor to convert coarse Blender renderings and a background into realistic final images (compositing). Figure 2 illustrates the full training pipeline.

A key component of the framework is the visual compositor, which refines the rendered target view to align with the intended edits. Since the reconstructed 3D scene S^{src} is derived from 2D images, its transformation often introduces noise, leading to artifacts in the target render R^{tgt}. The compositor corrects these artifacts with learned 3D shape priors to produce a coherent and photorealistic output. To train it effectively, we require paired scenes before and after edits. Object-centric videos naturally provide such supervision, but typically entangle object and camera motion, which limits generalization and disentangled control at test time. This motivates two simple yet effective training strategies, introduced in §3.2. We begin in §3.1 by detailing the full pipeline, including object segmentation and lifting, scene editing in Blender, and the architecture of the compositor model.

3.1. BlenderFusion Pipeline

Object-centric 3D Layering. The layering step leverages powerful off-the-shelf visual foundation models, such as SAM2 [33] and Depth Pro [5], to obtain editable reconstructions of objects of interest from input images. Our framework supports multiple input images (Figure 1, Figure 6), and each image may contain multiple objects. For simplicity and generality, we assume that each object is reconstructed from a single image, while multiple views of the same object produce higher quality reconstructions, especially with recent advances in 3D foundation models [24, 53, 54].

We begin by projecting 3D bounding boxes into the image space to obtain coarse 2D boxes. However, since these projected 2D boxes are usually loose, we refine them using Grounding DINO [26], prompting it with object category labels. In practice, if the predicted box from grounding DINO has a high overlap with the projected one (IoU > 0.5), we replace the latter. With the refined 2D box, we then prompt SAM2 to extract the object mask. Combining this mask with metric depth predictions from Depth Pro, we obtain object-wise depth. We then adjust the scale of the predicted depth to align with each object's 3D bounding box. Finally, we back-project adjusted object-wise depth maps to generate 3D point clouds and connect adjacent points to form a triangle mesh. This results in a set of 3D entities, S^{src}, which are imported into Blender for subsequent editing.

Note that the layering process described above is efficient and applied to all the data. At test time, more advanced layering operations can be included for higher-quality reconstructions. When the editing task requires a complete and high-quality 3D model (e.g., complicated intra-object part-level editing, or material change, etc.), we optionally use state-of-the-art image-to-3D models [56, 61] to produce complete object meshes. Concretely, we crop out the image patch of each object using the SAM2 object mask and run Hunyuan3D v2 [61] with the cropped image to obtain a complete textured mesh. We then align the mesh with the object's 3D box and the 2.5D surface reconstruction.

Blender-guided Editing. The output of the layering step, S^{src}, is then imported into Blender or other graphics software, where versatile transforms can be applied to objects and the camera with precise 3D grounding. BlenderFusion covers the following control tasks:

• ***Basic object control*** includes the translation, rotation, or scaling of each independent object, as well as object removal, insertion, or replacement. Since S^{src} contains per-object 3D models, all these transformations can be applied automatically and reflected in the rendering R^{tgt}.

• ***Advanced object control*** represents object attribute change (e.g., color, material), non-rigid object transforms (e.g., part-level control, deformation, etc.), and novel object insertion (e.g., not covered by the training data). These operations are inherited from Blender and can be fulfilled by user interactions in the user interface or automatic scripts (Figure 1 and Figure 7).

• ***Camera and Background control*** involves camera motion and background replacement. The new background is specified with an image to replace I^{src}, while the camera motion is simulated through the camera object in S^{src}.

After applying all these edits, both the original and edited scenes, S^{src} and S^{tgt}, are rendered into R^{src} and R^{tgt}, providing reliable 3D-grounded control signals for the compositing step. R^{src} and R^{tgt} contain the rendered RGB image and an object index mask from Blender's Object Index Pass. Although the training data only covers simple object transformations and camera motion, our generative compositor generalizes to all the above editing tasks at test time.

Generative Compositing. The generative compositor has access to two streams of input information (see Figure 2). The *source stream* consists of the image I^{src}, its rendering R^{src}, camera parameters P^{src} and object poses B^{src}. The *target stream* includes the edited rendering R^{tgt}, its camera parameters P^{tgt}, and object poses B^{tgt}. R^{tgt} may be noisy due to imperfect reconstruction and transformation. At test time, both the source and target streams may contain objects extracted from multiple input images.

To effectively process the dual-stream input, we adapt a pre-trained diffusion model with three key architectural modifications based on Stable Diffusion v2.1 [35]. First, we extend the model to a dual-stream architecture, where a single weight-shared denoising UNet processes both streams independently while enabling interaction through self-attention [11, 40, 49]. Second, we modify the first layer of the UNet to accommodate additional conditional inputs, increasing its channels

from 4 to 15 with zero-initialized weights. The original 4 channels process the VAE-encoded image or noise for the source or target stream. 5 additional channels handle Blender renderings (4 for the VAE-encoded rendering image and 1 for the instance mask). The remaining 6 channels encode camera parameters using Plücker embeddings [11, 41]. Third, each stream has an independent set of text tokens consisting of tuples of object class labels and poses. The labels are CLIP-embedded [31] while 3D boxes are converted into positional encodings [51] and processed by an MLP. The resulting embeddings are concatenated into a sequence as the text tokens for their respective streams.

3.2. Generative Compositor Training

The overview of the generative compositor is in Figure 2. To simulate test-time editing, we randomly sample two frames from a video: a source (original) and a target (edited). I^{src} goes through the layering step to obtain S^{src} and R^{src}. S^{src} is then transformed with the 3D object bounding boxes (object pose) and camera parameters (view changes) to obtain S^{tgt}, and re-rendered into R^{tgt}. This process introduces noise into the target render within the reconstruction, alignment, and transformation process. Given the noisy target render, the model is trained to complete missing object textures and geometry, filling in the view-changed background, using the source stream as context. While effective, this training strategy has two key limitations. First, the model struggles with edits that intensively modify the original context, such as object removal, insertion, or background change. Second, it performs poorly with disentangled object control, especially when the camera remains fixed. To this end, we introduce our two specific training strategies.

Source Masking. If the original context is altered (e.g., object is removed, replaced, or aggressively edited), the model should disregard the modified region in the source when compositing the target image. To achieve this, we introduce the source masking strategy during training, where each object in I^{src} and R^{src} is randomly masked out with a probability of 0.5 (see the overlaid orange boxes in Figure 2). At test time, we then flexibly mask both the source image and source render, or only mask the source image, depending on the concrete control task. This training strategy also has a regularization effect, which mitigates over-reliance on source information and the relative camera pose, and enforces the model to more accurately follow the intended edits in the target render (Figure 8). We also apply random masking to background regions to prevent the model from getting an inpainting bias. We refer to §A.1 for complete details.

Simulated Object Jittering. We observe that object motion supervision in training videos is limited and often strongly entangled with camera motion. For example, in object-centric videos such as Objectron, objects frequently remain static while only the camera moves. To offset this issue, we introduce an object jittering training strategy that simulates dynamic object motions under a fixed-camera setup, as shown in Figure 3. The key change from the standard video learning setup is the replacement of I^{src} and P^{src} with I^{tgt} and P^{tgt}. A random source masking strategy is applied separately to I^{tgt} and R^{src}. To infer the masked I^{tgt} from the noisy R^{tgt}, the model learns to effectively leverage the object information in R^{src}, B^{src}, and B^{tgt}, while the camera motion remains fixed. Despite its simplicity, this approach provides accurate, disentangled control of the object and camera at test time.

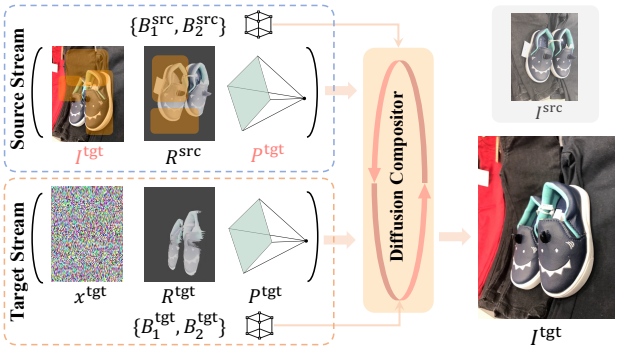

Figure 3 | The simulated object jittering training strategy for improving disentangled object control.

4. Experiments

In §4.1, we describe the implementation details and experimental setups. We then provide both quantitative and qualitative evaluations in §4.2 and §4.3, comparing BlenderFusion with leading approaches on 3D-aware control tasks of increasing complexity and finer granularity. Finally, §4.4 presents more results on human evaluation, generalization beyond training data, and ablation studies.

4.1. Experimental Settings

We implement our generative compositor using the Diffusers framework [52] and finetune the publicly available Stable Diffusion v2.1 base model. The model is trained with 8 NVIDIA A100 80GB GPUs. We use the v-prediction diffusion training objective [39]. The batch size is 320, and the model is trained for 30,000 iterations. We turn on gradient checkpointing, use gradient accumulation of two steps, and train the model with mixed bfloat16 precision. We use AdamW optimizer with a weight decay factor 1e-2 and a 500-step linear warmup. The learning rate is 5e-5 for the diffusion model and 1e-4 for the new MLP that encodes the object 3D box. For inference, we run the DDPM [18] sampler for 50 steps, and classifier-free guidance (CFG) [17] is used with scale 2.0. When encoding the camera parameters, the camera of the source stream sets the reference coordinate frame. The 3D bounding box is projected into the image plane, and each corner is represented by (x, y, depth). The ratios of vanilla video training, training with source masking, training with both source masking and simulated object jittering, and unconditional training (for CFG) are 0.35, 0.3, 0.3, and 0.05, respectively. We then introduce datasets, baselines, and metrics. At test time, the layering step only uses the 2.5D surface reconstructions without running an image-to-3D model for complete meshes, except for the complex editing tasks in Figure 7 (Bottom). Please refer to §A for complete implementation details.

Datasets. We cover three public video datasets with 3D object and camera annotations:

• *MOVi-E* is a synthetic multi-object video dataset from the Kubric [14] dataset generator. To obtain videos with more objects and camera dynamics, we run Kubric's official data generation pipeline to produce 10,000 MOVi-E videos by increasing the number of dynamic objects and the camera motion range. The image resolution is 512×512. This MOVi-E variant provides a challenging benchmark for multi-object control with 3D awareness (occlusion, novel viewpoint, etc.).

• *Objectron [1]* contains 15,000 real-world object-centric video clips across 9 categories, recorded with camera movement in real-world environments. Note that this dataset only contains static object. We keep the aspect ratio of the dataset and resize the images to 384×512.

• *Waymo Open Dataset (WOD) [48]* consists of 1,000 real-world videos captured from self-driving cars. Following prior work [55], we use the front-view camera and filter out small cars. We keep the aspect ratio of the original dataset and resize the images to 528×352.

Baselines. We use 3DIT and NA as baselines, while Image Sculpting is excluded as it is a per-scene optimization method and mainly handles single objects. For controlled experiments between baselines and ours, we re-implement 3DIT and NA within the Diffusers framework, strictly following the descriptions in their original papers. We ensure all approaches use the same base Stable Diffusion v2.1, same input information (input image, object category, source/target object 3D box), identical training setups (training loss, training iterations, learning rate, etc.), and identical sampling setups (sampler and sampler steps). As a result, the primary difference lies in their core controllability strategy. Concretely, 3DIT directly uses the text embeddings (i.e., class labels and source/target 3D boxes) to transform the source image; NA employs external DINO [6] to extract the per-object appearance and combines it with target 3D box, then controls through the text embedding interface;

BlenderFusion leverages a more explicit interface, Blender, to obtain source and target renders as 3D-grounded control signals, where many fine-grained 3D visual edits are possible. We detail the baseline approaches and their changes below.

• *Object 3DIT* [28] originally adapts Zero-1-to-3 [25] to incorporate a text instruction describing the control task and is trained on simple synthetic data, mainly targeting single-object control. For a fair comparison, we introduce several updates. The base model is replaced wit Stable Diffusion (SD) v2.1, and the source image is encoded by the SD VAE and concatenated with the input, following the original approach. Additionally, the Plücker embedding of the relative camera pose is concatenated. Instead of using a plain text instruction, we replace it with serialized object embeddings to handle multi-object control. Each object embedding consists of the CLIP embedding of its object category and the encoding of its 3D bounding box.

• *Neural Assets (NA)* [55] enables multi-object 3D editing through object tokens that encode appearance and pose features. The object appearance is obtained by applying RoIAlign [16] to DINO features of the foreground image, while the pose is obtained from the 3D object bounding box with an MLP. The background is processed separately, with the appearance extracted from the foreground-masked DINO features and the pose computed from the relative camera pose between the source and target images. Tokens for all objects and the background are serialized into a sequence, replacing the text embedding in the pre-trained Stable Diffusion v2.1 architecture.

Evaluation Metrics. We follow the evaluation metrics used by NA [55]. PSNR, SSIM, and LPIPS are computed at both image and object levels. The image-level metrics include FID, while the object-level metrics include the per-object DINO feature cosine similarity. Complete details are in the Appendix. All quantitative evaluations follow the standard video frame-based setup, which measures how well the model transforms a source frame into a target frame given camera and object poses. However, this setup is not ideal, especially for assessing fine-grained, disentangled control tasks, which are more relevant and practical in real-world editing scenarios but do not have ground truth answers. Thus, we rely on qualitative comparisons and human evaluations to better evaluate these capabilities.

Synthetic Data Pre-training. We observe that all approaches struggle to learn disentangled object rotation on WOD, primarily due to the lack of angular object motion in the dataset (i.e., most cars exhibit only simple translation across frames). To address this, we initialize WOD training with the pre-trained checkpoint from MOVi-E, which contains richer object motions and provides stronger 3D object shape priors. For all three approaches, we adopt this initialization and reduce the base learning rate to 1e−5. More analyses are in §B.2.

4.2. Standard Evaluation

Table 2 and Figure 4 summarize our comparisons with baselines in the standard video frame setup. Quantitatively, BlenderFusion consistently improves both object-level and image-level metrics across all three datasets, indicating better modeling of both foreground and background. Qualitatively, our method preserves precise object appearance and identity well, while correctly capturing geometry and shadings. In contrast, baselines often distort details (e.g., the geometry of the chair back in the first example of Objectron, the shape of the car in the first example of WOD). Additionally, the baselines struggle to handle a large number of objects with various transforms in the synthetic MOVi-E data with high dynamics, while ours shows reliable generation quality over multiple objects.

However, since this evaluation setup entangles camera and object dynamics, it is unsuitable for evaluating disentangled control capabilities. For example, a model that entirely overfits solely to camera motion (i.e., novel-view synthesis) achieve great performance on the standard Objectron

Dataset	Model	Object-level metrics				Frame-level metrics			
		PSNR ↑	SSIM ↑	LPIPS ↓	DINO ↑	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
MOVi-E [14]	Object 3DIT	14.06	0.284	0.411	0.848	17.02	0.500	0.340	15.71
	Neural Assets	13.74	0.221	0.428	0.826	16.73	0.414	0.388	23.08
	BlenderFusion	18.90	0.557	0.227	0.914	21.32	0.674	0.198	9.11
Objectron [1]	Object 3DIT	13.88	0.290	0.424	0.902	14.98	0.355	0.423	6.14
	Neural Assets	13.73	0.278	0.427	0.921	14.56	0.337	0.427	6.18
	BlenderFusion	16.06	0.389	0.291	0.959	16.54	0.413	0.323	3.25
WOD [48]	Object 3DIT	18.90	0.448	0.255	0.930	23.21	0.640	0.220	11.92
	Neural Assets	16.87	0.301	0.322	0.901	20.41	0.548	0.267	15.39
	BlenderFusion	20.93	0.596	0.185	0.956	24.11	0.676	0.189	10.02

Table 2 | Quantitative results with MOVi-E, Objectron, and Waymo Open Dataset (WOD). The model controls both the objects and the camera to transform the source frame into the target frame of a video. Please see §4.1 for details of the baselines and datasets.

evaluation setup. At test time, an important use case is to keep the camera frozen while manipulating the objects and background. Therefore, the next subsection presents further comparisons on tasks requiring more fine-grained and disentangled control capabilities.

4.3. Disentangled Control and Fine-grained Compositing

Disentangled Control. Figure 5 compares three methods across four types of disentangled control tasks: object translation, rotation, and scaling with a fixed camera, as well as jointly specified camera motion and object pose. While 3DIT performs reasonably well in the standard video setting, it fails on nearly all disentangled object manipulation tasks and tends to keep the object still. This suggests a strong entanglement between object and camera motion. NA demonstrates significantly better disentangled control than 3DIT but has two major limitations: 1) It loses appearance and geometric details for both objects and backgrounds because its DINO features are from low-resolution images, and this encoder likely discards fine details even after fine-tuning. 2) Foreground and background interfere with each other. This issue arises because RoIAlign on DINO features cannot clearly separate objects from the background. In contrast, our method precisely follows the intended 3D transformation while decently preserving both geometry and appearance details. Furthermore, when the camera remains fixed, the background remains stable, as Blender can accurately simulate the camera change.

Fine-grained Editing and Compositing. We further explore more complex 3D-aware editing and compositing tasks in Figure 6, comparing BlenderFusion with NA. As task complexity increases, the benefits of explicit 3D grounding become more pronounced. Concretely,

• In the first row, NA misaligns spatial transformations (e.g., positions and poses of shoes and bottles) and mixes appearances, whereas ours ensures both geometric and semantic consistency. NA struggles with close objects due to RoIAlign, while our layering step leverages visual foundation models for precise object delineation and lifting and performs accurate geometric transformations in Blender.

• In the second row, NA fails to duplicate objects correctly, generating only 5 out of 8 cups. Moreover, the duplicated objects exhibit undesired changes in appearance and shape, whereas we faithfully generate all objects while preserving both appearance and geometry. NA struggles with handling many objects as it falls outside its training distribution, while ours explicitly performs duplication within Blender, bypassing this limitation.

• In the third row, NA loses original object appearances during object shuffling and swapping (e.g.,

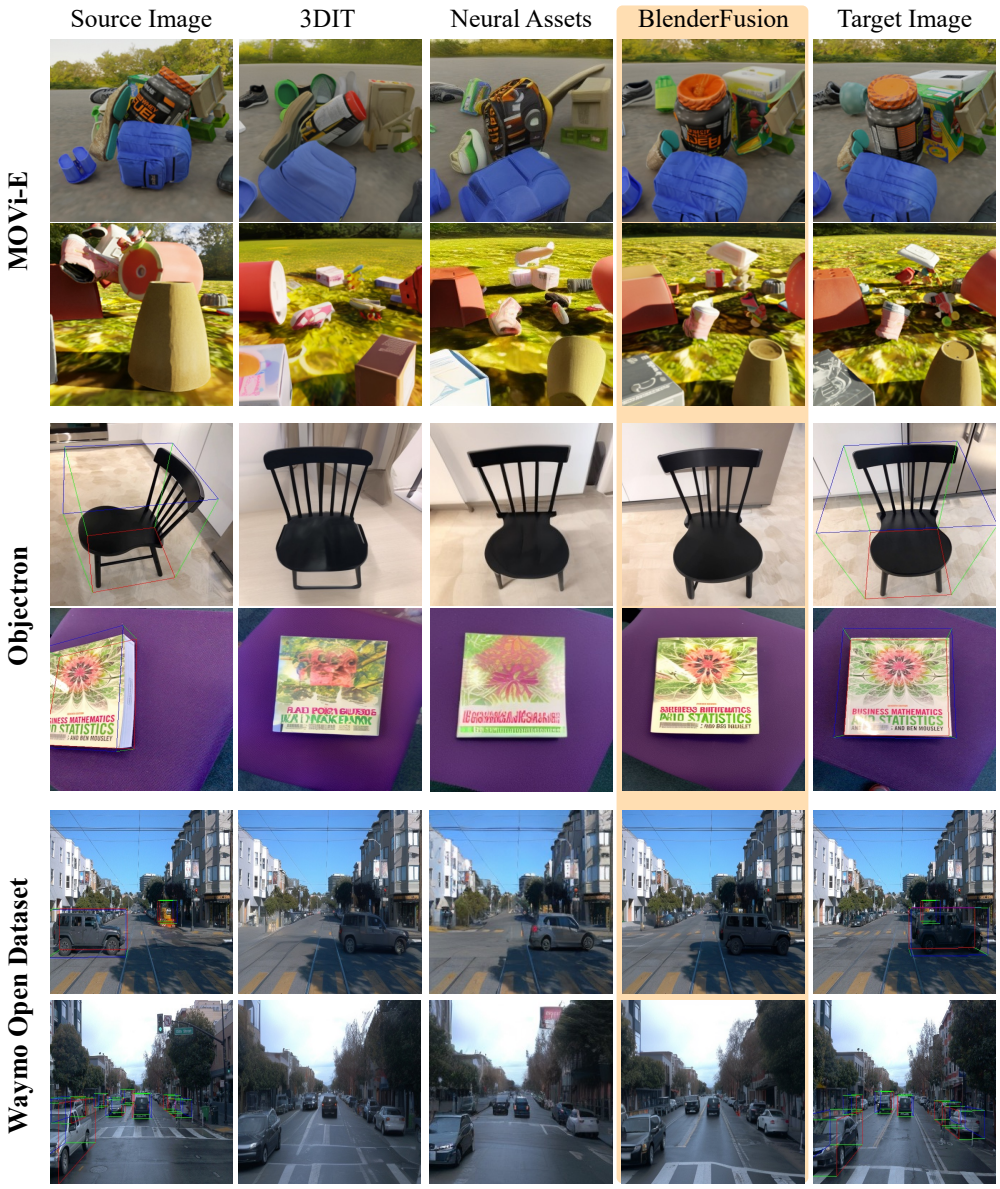

Figure 4 | Qualitative results with the standard video frame setup on three datasets. BlenderFusion outperforms the baselines in visual quality and object identity preservation. The 3D bounding boxes in MOVi-E are omitted for clear visualization.

a black chair disappears) and fails to maintain depth consistency, whereas ours preserves both appearance and natural perspective shifts.

• In the fourth row, NA produces inaccurate poses and disregards shading, whereas ours ensures both geometric accuracy and lighting consistency.

• In the WOD multi-image recomposition tasks from the last two rows, NA produces acceptable results but still loses a lot of object details due to the highly lossy DINO encoding. Ours preserves better object identity and fidelity, while showing more coherent shadings.

These results demonstrate the impact of our core idea of decoupling control from generation, leveraging Blender as a bridge to achieve precise and flexible visual compositing.

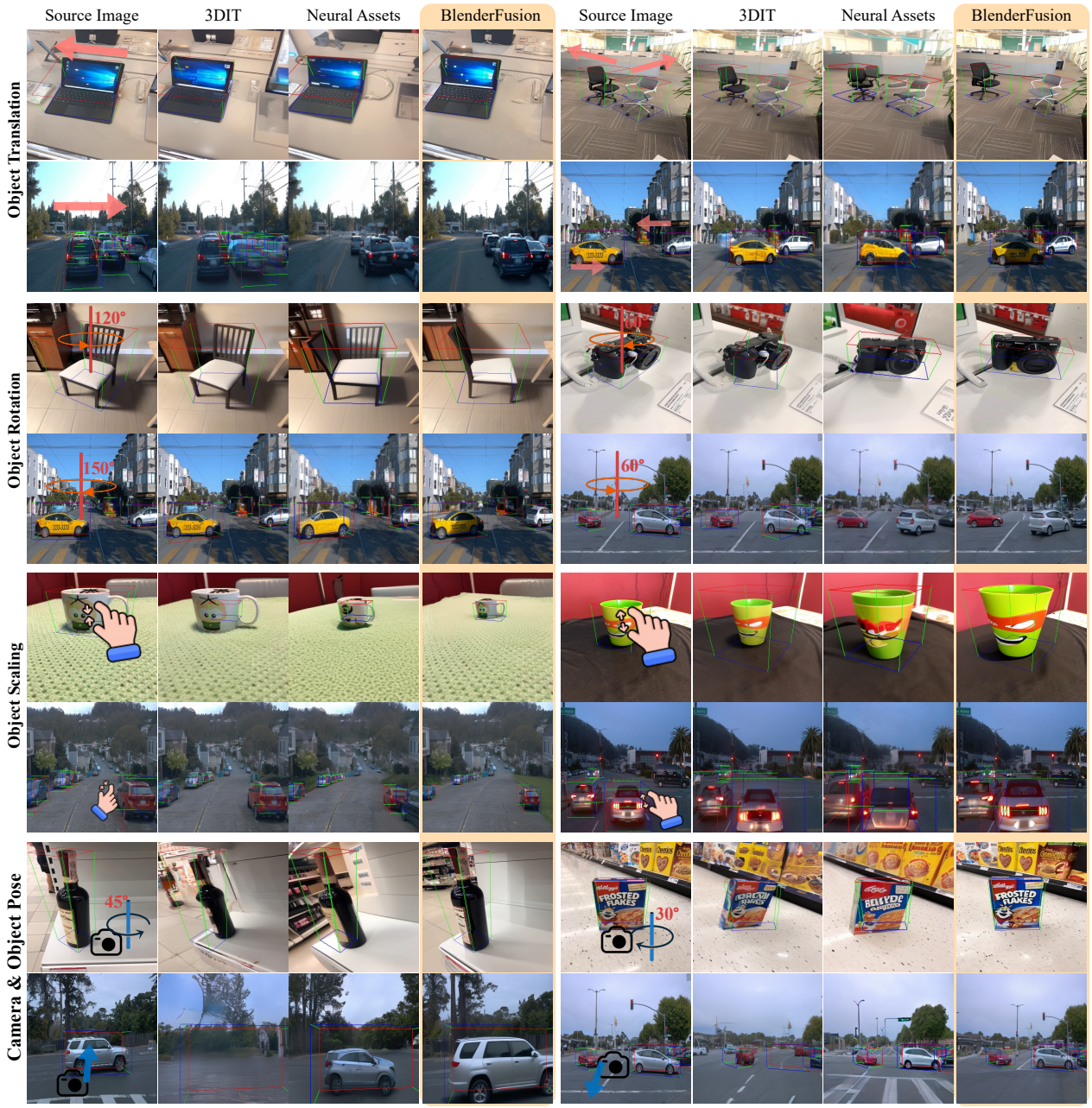

Figure 5 | Qualitative comparison results on disentangled visual control tasks, using a source image from either Objectron or WOD. BlenderFusion demonstrates more precise control, more consistent object identity, and better disentanglement of camera and object.

4.4. More Results

Human Evaluation. Based on the three evaluation settings in §4.2 and §4.3, we conduct a user study to further validate the superority of our method. We curate 54 examples for human evaluation: 18 for the standard video frame setup, 24 for disentangled object control, and 12 for complex fine-grained compositional con-

Setting	Ours (%)	Baseline (%)	Draw (%)
Overall	87.04	6.40	6.56
Video	80.79	8.80	10.42
Disentangled	88.37	6.60	5.03
Fine-grained	93.75	2.43	3.82

Table 3 | Human evaluation results for three setups.

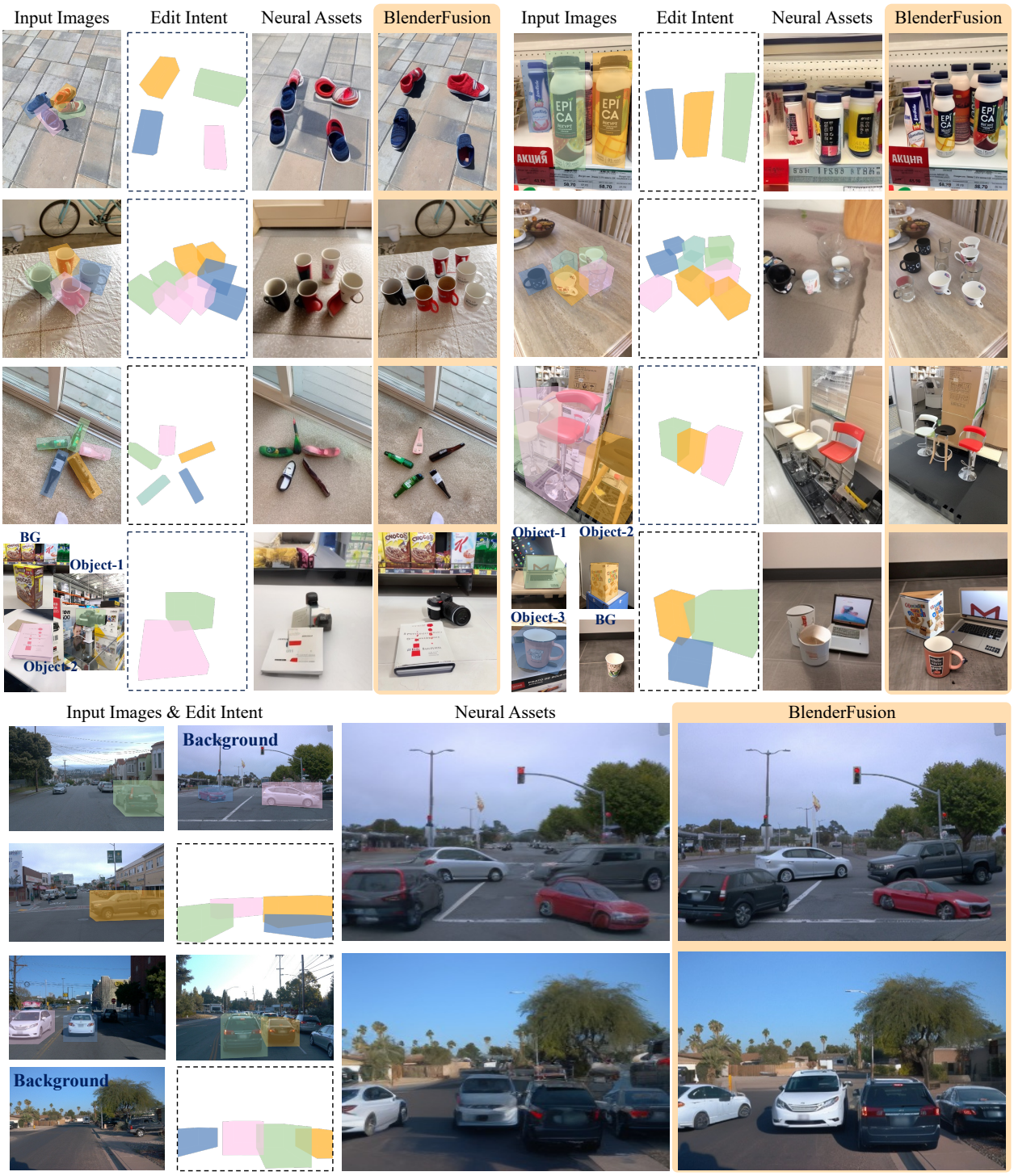

Figure 6 | Qualitative results on fine-grained multi-object editing and compositing tasks. The edit intent demonstrates the expected output geometry while the color encodes the object identity.

trol. Participants compared two shuffled generations (ours vs. Neural Assets) and selected one: A, B, or Similar. The results were collected from 1,294 selections from 24 users and are summarized in Table 3. The gap between BlenderFusion and the baseline gets larger when the editing task becomes more challenging, which further demonstrates the advantages of the 3D-grounded framework in complex and compositional visual editing.

Figure 7 | **Top:** BlenderFusion shows reasonable generalization to in-the-wild images from SUN-RGBD [44], ARKitScenes [4], and Hypersim [34] datasets. **Bottom:** our framework inherits the versatile editing capabilities of graphics software, enabling diverse object control tasks beyond the training data. Images are resized to facilitate visualization.

Generalization. We apply BlenderFusion trained on Objectron data to in-the-wild images from SUN-RGBD [44], ARKitScenes [4], and Hypersim [34] datasets. All three datasets present scenes with much higher complexity and richer details than Objectron. Figure 7 (Top) demonstrates the generalization results. Those in-the-wild images present much more complicated scene structure and object details than the Objectron training data, and Hypersim is a high-end synthetic dataset created by professional designers. BlenderFusion presents reasonable generalization capabilities, although the visual quality shows some degradation compared to in-domain results.

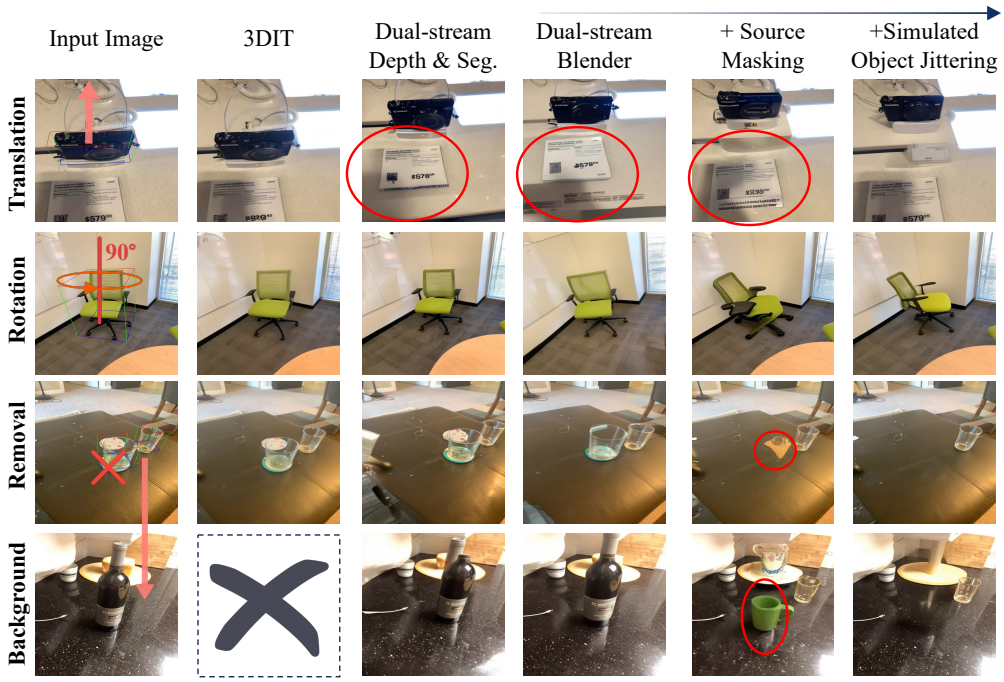

Figure 8 | Qualitative ablation results over the key designs, including using Blender renders as the 3D-grounded control signals as well as the training strategies of the diffusion compositor.

Progressive Editing. In Figure 7 (Bottom), we demonstrate progressive, step-by-step editing, where each intermediate modification in Blender is rendered using our diffusion compositor. For these examples, Hunyuan3D v2 is utilized during the layering step to generate higher-quality 3D object models, as detailed in §3.1. The first row illustrates: 1) color changes for each object, 2) rotation and text decal application, and 3) object deformation. The second row shows: 1) a color change, 2) part-level deformation, 3) texture replacement, and 4) rotation. Although we showcase simple, progressive edits here, more advanced controls supported by Blender are also possible.

Ablation Study. As discussed earlier, the standard video frame setup does not indicate disentangled control capabilities. Thus, we focus on qualitative analysis for ablation studies here and leave the quantitative results to the Appendix. Figure 8 investigates the impact of three key design choices: 1) the dual-stream architecture, 2) the source masking training strategy, and 3) the simulated object jittering training strategy. A straightforward ablation study is to use our dual-stream design while adding the depth and segmentation of the input image from Depth Pro and SAM2 into the source stream (third column). However, this model only shows improvements over the 3DIT baseline in the object translation setting. When adding the Blender renders, the dual-stream model performs reasonable object translation (first row), but the background still consistently moves along with the object. This variant also struggles with disentangled object rotation (second row) and instance-wise removal (third row), and fails in background changes (fourth row). Introducing the source masking strategy clearly strengthens object-level control, but the model still suffers from entangled camera and object states. This is mainly because the training data rarely covers object manipulation under a fixed camera. The simulated object jittering strategy addresses these remaining issues, significantly enhancing the disentangled control capabilities of both object and background. Note that the source masking and simulated object jittering are designed based on the Blender renders – without the R^{src} and R^{tgt} in the compositor's inputs, both training strategies become invalid.

5. Conclusion

This paper introduces BlenderFusion, a novel framework that integrates generative diffusion models with 3D graphics tools, enabling fine-grained and highly controllable visual compositing. BlenderFusion follows three key steps of traditional visual compositing: it segments and lifts 2D images into editable 3D entities, performs precise manipulations within Blender, and refines the resulting visuals through a specialized generative compositor. By leveraging this pipeline, we achieve superior control and realism compared to prior techniques. Experiments on real-world video datasets demonstrate that BlenderFusion significantly advances multi-object scene editing, providing a flexible and practical solution for complex visual content creation.

Acknowledgements

We thank Ziyi Wu for discussions on Neural Assets and synthetic data pre-training. We thank Thomas Kipf and Caroline Pantofaru for reviewing the paper and providing feedback.

References

[1] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. 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 (CVPR)*, 2021.

[2] H. Alzayer, Z. Xia, X. Zhang, E. Shechtman, J.-B. Huang, and M. Gharbi. Magic fixup: Streamlining photo editing by watching dynamic videos. *arXiv preprint arXiv:2403.13044*, 2024.

[3] J. Baldridge, J. Bauer, M. Bhutani, N. Brichtova, A. Bunner, K. Chan, Y. Chen, S. Dieleman, Y. Du, Z. Eaton-Rosen, et al. Imagen 3. *arXiv preprint arXiv:2408.07009*, 2024.

[4] G. Baruch, Z. Chen, A. Dehghan, T. Dimry, Y. Feigin, P. Fu, T. Gebauer, B. Joffe, D. Kurz, A. Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021.

[5] A. Bochkovskii, A. Delaunoy, H. Germain, M. Santos, Y. Zhou, S. R. Richter, and V. Koltun. Depth pro: Sharp monocular metric depth in less than a second. *arXiv preprint arXiv:2410.02073*, 2024.

[6] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2021.

[7] W. Chen, H. Hu, C. Saharia, and W. W. Cohen. Re-imagen: Retrieval-augmented text-to-image generator. *arXiv preprint arXiv:2209.14491*, 2022.

[8] W. Chen, H. Hu, Y. Li, N. Ruiz, X. Jia, M.-W. Chang, and W. W. Cohen. Subject-driven text-to-image generation via apprenticeship learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[9] X. Chen, L. Huang, Y. Liu, Y. Shen, D. Zhao, and H. Zhao. Anydoor: Zero-shot object-level image customization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2024.

[10] Y.-C. Cheng, K. K. Singh, J. S. Yoon, A. Schwing, L. Gui, M. Gadelha, P. Guerrero, and N. Zhao. 3d-fixup: Advancing photo editing with 3d priors. *arXiv preprint arXiv:2505.10566*, 2025.

[11] R. Gao, A. Holynski, P. Henzler, A. Brussee, R. Martin-Brualla, P. Srinivasan, J. T. Barron, and B. Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv preprint arXiv:2405.10314*, 2024.

[12] D. Garibi, S. Yadin, R. Paiss, O. Tov, S. Zada, A. Ephrat, T. Michaeli, I. Mosseri, and T. Dekel. Tokenverse: Versatile multi-concept personalization in token modulation space. *arXiv preprint arXiv:2501.12224*, 2025.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems (NeurIPS)*, 2014.

[14] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022.

[15] Y. Gu, I. Huang, J. Je, G. Yang, and L. Guibas. Blendergym: Benchmarking foundational model systems for graphics editing. *arXiv preprint arXiv:2504.01786*, 2025.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2017.

[17] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[18] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems (NeurIPS)*, 2020.

[19] H. Hu, K. C. Chan, Y.-C. Su, W. Chen, Y. Li, K. Sohn, Y. Zhao, X. Ben, B. Gong, W. Cohen, et al. Instruct-imagen: Image generation with multi-modal instruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2024.

[20] Z. Hu, A. Iscen, A. Jain, T. Kipf, Y. Yue, D. A. Ross, C. Schmid, and A. Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[21] I. Huang, G. Yang, and L. Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. In *European Conference on Computer Vision (ECCV)*, 2024.

[22] I. Huang, Y. Bao, K. Truong, H. Zhou, C. Schmid, L. Guibas, and A. Fathi. Fireplace: Geometric refinements of llm common sense reasoning for 3d object placement. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 2025.

[23] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2020.

[24] V. Leroy, Y. Cabon, and J. Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, 2024.

[25] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

[26] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision (ECCV)*. Springer, 2024.

[27] S. Lu, Y. Liu, and A. W.-K. Kong. Tf-icon: Diffusion-based training-free cross-domain image composition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[28] O. Michel, A. Bhattad, E. VanderBilt, R. Krishna, A. Kembhavi, and T. Gupta. Object 3dit: Language-guided 3d-aware image editing. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[29] K. Pandey, P. Guerrero, M. Gadelha, Y. Hold-Geoffroy, K. Singh, and N. 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)*, 2024.

[30] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[31] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning (ICML)*, 2021.

[32] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[33] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[34] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2021.

[35] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022.

[36] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. 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 (CVPR)*, 2023.

[37] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems (NeurIPS)*, 2022.

[38] R. Sajnani, J. Vanbaar, J. Min, K. Katyal, and S. Sridhar. Geodiffuser: Geometry-based image editing with diffusion models. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025.

[39] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[40] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023.

[41] V. Sitzmann, S. Rezchikov, B. Freeman, J. Tenenbaum, and F. Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[42] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning (ICML)*, 2015.

[43] K. Sohn, N. Ruiz, K. Lee, D. C. Chin, I. Blok, H. Chang, J. Barber, L. Jiang, G. Entis, Y. Li, et al. Styledrop: Text-to-image generation in any style. *arXiv preprint arXiv:2306.00983*, 2023.

[44] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015.

[45] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[46] Y. Song, Z. Zhang, Z. Lin, S. Cohen, B. Price, J. Zhang, S. Y. Kim, and D. Aliaga. Objectstitch: Generative object compositing. *arXiv preprint arXiv:2212.00932*, 2022.

[47] Y. Song, Z. Zhang, Z. Lin, S. Cohen, B. Price, J. Zhang, S. Y. Kim, H. Zhang, W. Xiong, and D. Aliaga. Imprint: Generative object compositing by learning identity-preserving representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[48] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2020.

[49] S. Tang, J. Chen, D. Wang, C. Tang, F. Zhang, Y. Fan, V. Chandra, Y. Furukawa, and R. Ranjan. Mvdiffusion++: A dense high-resolution multi-view diffusion model for single or sparse-view 3d object reconstruction. In *European Conference on Computer Vision (ECCV)*, 2024.

[50] G. C. Tarrés, Z. Lin, Z. Zhang, H. Zhang, A. Gilbert, J. Collomosse, and S. Y. Kim. Multitwine: Multi-object compositing with text and layout control. *arXiv preprint arXiv:2502.05165*, 2025.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems (NeurIPS)*, 2017.

[52] P. von Platen, S. Patil, A. Lozhkov, P. Cuenca, N. Lambert, K. Rasul, M. Davaadorj, D. Nair, S. Paul, W. Berman, Y. Xu, S. Liu, and T. Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[53] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. Vggt: Visual geometry grounded transformer. *arXiv preprint arXiv:2503.11651*, 2025.

[54] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[55] Z. Wu, Y. Rubanova, R. Kabra, D. Hudson, I. Gilitschenski, Y. Aytar, S. van Steenkiste, K. Allen, and T. Kipf. Neural assets: 3d-aware multi-object scene synthesis with image diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[56] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.

[57] J. Yenphraphai, X. Pan, S. Liu, D. Panozzo, and S. Xie. Image sculpting: Precise object editing with 3d geometry control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[58] Z. Yuan, M. Cao, X. Wang, Z. Qi, C. Yuan, and Y. Shan. Customnet: Zero-shot object customization with variable-viewpoints in text-to-image diffusion models. *arXiv preprint arXiv:2310.19784*, 2023.

[59] B. Zhang, Y. Duan, J. Lan, Y. Hong, H. Zhu, W. Wang, and L. Niu. Controlcom: Controllable image composition using diffusion model. *arXiv preprint arXiv:2308.10040*, 2023.

[60] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2023.

[61] Z. Zhao, Z. Lai, Q. Lin, Y. Zhao, H. Liu, S. Yang, Y. Feng, M. Yang, S. Zhang, X. Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

Appendices

The appendix provides remaining implementation details and additional experimental results as mentioned in the main paper:

- §A presents the remaining implementation details of BlenderFusion, and describes full details about the baselines and datasets.
- §B provides more qualitative results in a similar style of Figure 4 and Figure 5 of the main paper, the quantitative ablation results corresponding to Figure 8, and failure case analyses.

A. Remaining Implementation Details

A.1. Remaining Details of BlenderFusion

Source Masking Details. During training, the source masking is always consistently applied to the source image and the source Blender render. Concretely, we randomly mask each foreground object with an (independent) probability of 0.5, then apply random masking to the background using boxes with a similar aspect ratio as the corresponding foreground objects. The mask is applied by masking out the 2D bounding box derived from the 3D object bounding box, with a dilation operation. The background masking here effectively alleviates the object inpainting bias, which might hurt the performance in object removal. When an object is masked out, the corresponding source bounding box information is also dropped for the source stream. The simulated object jittering training also applies the same source masking strategy, while the masks applied to the "source" image and source render are different – because the "source" image in this reconstruction setting is actually the target frame, while the source render still comes from the true source frame.

Frame Sampling Strategy for Training Data. To train BlenderFusion, we need to prepare the Blender renderings of the source and target frames. For both datasets, we sample the source frame uniformly with a fixed stride. Then, for each source stream, we randomly sample a set of target frames. For each pair of source and target frames, we obtain the object 3D models with the 3D lifting process, import them into Blender, and simulate the object and camera transforms. In this way, we obtain the source and target renders for training our dual-stream diffusion model. Note that since the two baselines do not rely on the reconstruction and re-render steps, there is no limitation on the sampling of the source and target frames, and they actually train with more diverse data than BlenderFusion. The strong quantitative and qualitative performance of our method suggests that training with stronger 3D grounding can also improve the data efficiency.

Test-time Details. For the standard video evaluation setting, all conditions are directly passed to the model without any masking or dropping. For disentangled object manipulation with fixed camera (including translation, rotation, and scaling), the source masking is applied to two regions in the *source image*: 1) the region of the original object and 2) the expected region of the target object. The *source render* and the *source object 3D bounding box* are only masked/dropped if the object is intended to be removed or replaced, otherwise, they are preserved and will be used as the main reference for object appearance and geometry. The source masking at test time encourages better disentanglement between foreground objects and background contexts.

When applying the advanced object-level control inherited from Blender (e.g., attribute change, deformation), as described in §3.1, those edits are always reflected in both the source render R^{src} and target render R^{tgt} in the dual-stream inputs. Concretely, when we close the screen of the laptop like

in Figure 1, the edit is always first reflected in the initial scene S^{src} and then rendered to update R^{src}, then, further edits like object rotations or translations transform S^{src} into S^{tgt} and we render S^{tgt} to get R^{tgt}. In this way, the difference between the target and source stream is always the basic object transformations, which is consistent with the training setting and makes the framework generalizes well to much more complex multi-object editing and scene composition tasks at test time.

A.2. Remaining Details of Datasets and Baselines

MOVi-E Details. Instead of using the original 10K MOVi-E videos released by the Kubric dataset generator [14], we update the generation config to generate a new set of 10K videos with more extensive object and camera motions. Specifically, we update the number of static objects from 10-20 to 5-10, and the number of dynamic objects from 1-3 to 5-10. For the camera movement, we reset the maximum camera movement from 4 units in Kubric's 3D space to 8 units. This new variant is more suitable for evaluating the model 3D controllability over multiple objects and the camera. Additionally, it makes the synthetic MOVi-E data a more effective pre-training resource for real-world datasets with limited object and camera motions (e.g., WOD has highly imbalanced object poses).

Objectron Details. Simialr to NA's setting, we drop the bike category and use the remaining data for training and evaluation. For each video, we randomly sample 60 frames without repeating to get a clip. Then, we randomly sample source and target frames for each clip. The test set contains 2812 videos, and we sample 67,092 pairs for evaluation. The original resolution of the dataset is 480×640 (height is larger than the width). We keep the aspect ratio and use 384×512.

Waymo Open Dataset (WOD) Details. Similar to the settings of NA, we only take the front-view camera of WOD, and use the same filtering strategies to filter out the videos without large objects. Concretely, the pair of source and target frames is considered as valid only when there is at least one large object that occupies more than 1% image area in both frames. We sample 6976 pairs of source and target frames from the test set for evaluation.

Baseline Implementation Details. As described in §4.1, we re-implement the two major baselines, 3DIT and Neural Assets (NA), and carefully control the experimental settings to focus on the essential difference between the model designs for object/scene controllability. The re-implemented 3DIT uses the same base model and image resolution as ours, and shares exactly the same training recipes and inference setups. It can be considered as ours without the two-stream model architecture, the grounded Blender renders, and the two accompanying training strategies. In our experiments, we observe that training NA with high generation resolutions (i.e., 384×512 for Objectron, 528×352 for WOD) does not bring improvement on quantitative metrics, but produces apparent object appearance shifts compared to the original 256×256 setting. This might be related to the resolution inconsistency between its DINO encoder (i.e., 224×224) and diffusion model, and is not trivially resolved by jointly fine-tuning all modules. Increasing DINO's input resolution would significantly raise the training GPU memory cost and make the training unaffordable, which also has the risk of invalidating the pre-trained priors. Therefore, for better qualitative results under DINO's pre-defined input resolution, we choose to keep NA's original 256×256 generation resolution in all experiments.

B. Additional Experimental Results

B.1. Quantitative Ablation Study

Table 4 provides the quantitative ablation evaluation results with the standard video setting. The dual-stream design alone can improve the baseline significantly by more effectively leveraging the source

and target information. Incorporating the Blender renders in both streams further unlocks the model's capacity by providing reliable 3D groundings. However, as presented in Figure 8, these variants still cannot achieve disentangled object control, struggling to manipulate objects with a fixed camera or to do significant modifications to the source image (e.g., object removal, background change).

The source masking can slightly improve the quantitative results, as it can be considered as a data augmentation strategy to the basic video training setting. While being the core design to acquire disentangled object control, the simulated object jittering training strategy slightly lowers the quantitative results on the standard video setting. This is expected because it is essentially an image reconstruction training, and has a different source stream setting from the standard video setup (i.e., source camera is the same as the target camera).

Method	Object-level Metrics			
	PSNR↑	SSIM ↑	LPIPS ↓	FID ↓
3DIT (one-stream)	13.88	0.290	0.424	6.14
Dual-stream (DS)	15.90	0.378	0.310	3.52
DS + Depth & Seg.	16.04	0.382	0.313	3.74
DS + Blender	16.05	0.389	0.292	2.93
+ Source Masking	16.18	0.393	0.290	2.64
+ Sim. Obj Jittering	16.06	0.389	0.291	3.25

Table 4 | The quantitative ablation studies on the key elements of our diffusion model compositor, corresponding to Figure 8 of the main paper.

B.2. Failure Cases

In our experiments, one main failure mode is that the model sometimes has difficulty achieving accurate manipulation for disentangled object rotation (Figure 9). On WOD, this is largely alleviated by using the MOVi-E pre-trained model as the initialization to compensate for the lack of pose difference between the source and target frames in the training data (as most cars keep going straight). The top row of Figure 9 shows typical failure cases of NA and ours when not using MOVI-E pre-training.

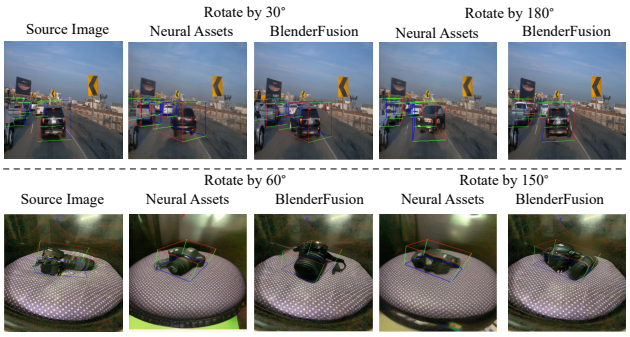

Figure 9 | Failure cases on object rotation.

On Objectron, the failure of object rotation usually comes with wrong or obviously altered object geometry, and is caused by two factors: 1) the model lacks accurate 3D understanding for complex object geometry (e.g., high-end cameras), and 2) when the object reconstruction is 2.5D, the renders can be unreliable when the object is rotated significantly. For the latter case, using 3D-Gen meshes resolves the problem in most cases, albeit at the cost of a longer processing time for running an image-to-3D model and an object pose alignment step. Pre-training on MOVi-E does not help with this failure case, probably because the objects in MOVi-E only present simple geometry. Two potential directions can be explored in future works: 1) pre-training on datasets with diverse object geometries and motions, or 2) extending the pipeline to multi-view or video input so that a more complete 3D reconstruction can be obtained with state-of-the-art multi-view reconstruction algorithms.

B.3. Additional Qualitative Results

Figure 10 and Figure 11 provide additional qualitative comparison, extending Figure 4 and Figure 5 of the main paper. These examples further demonstrate the superior object control capability of BlenderFusion.

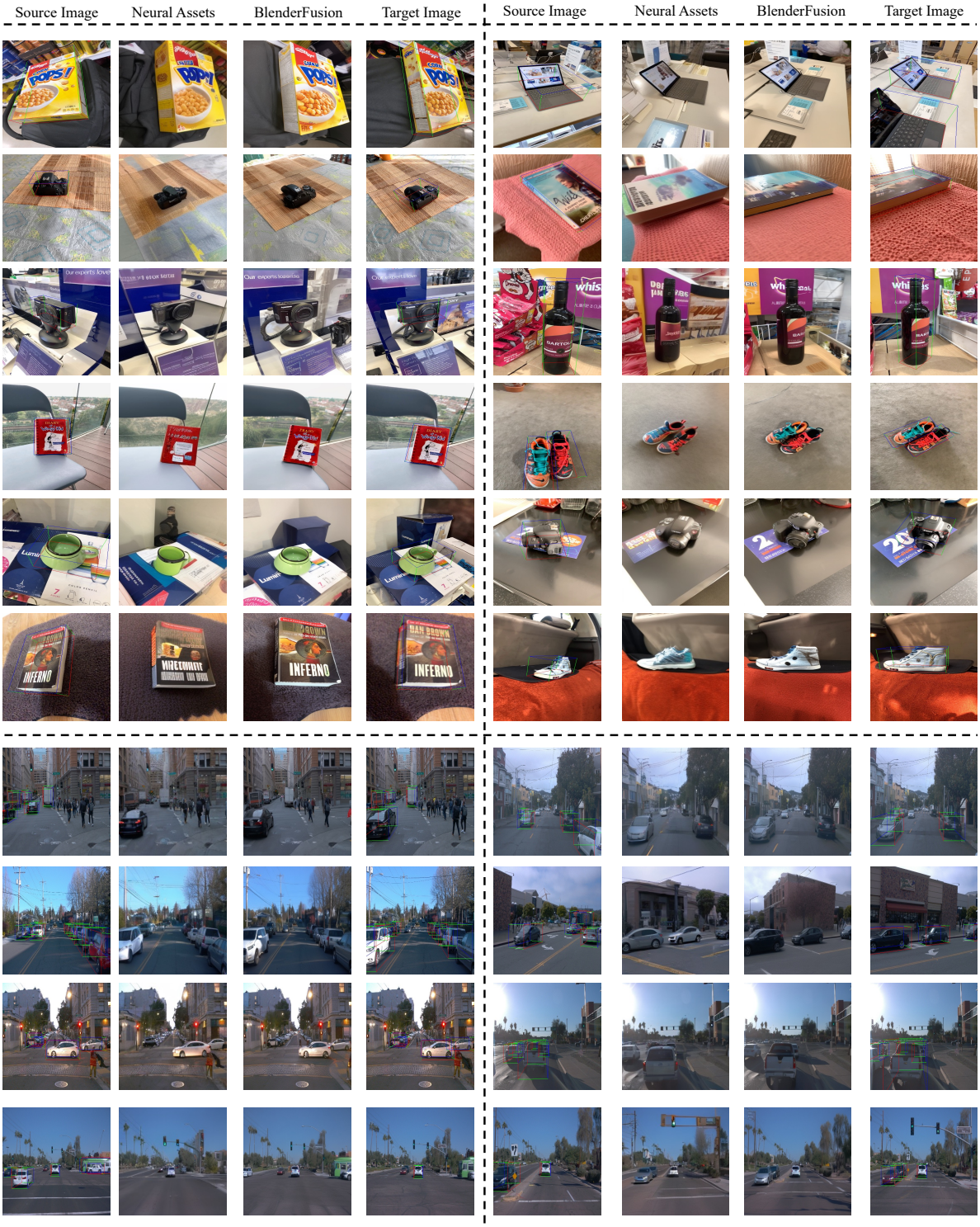

Figure 10 | Additional qualitative results of the standard evaluation setting (source and target frames from a video), extending Figure 4 of the main paper. 3DIT is omitted in this figure.

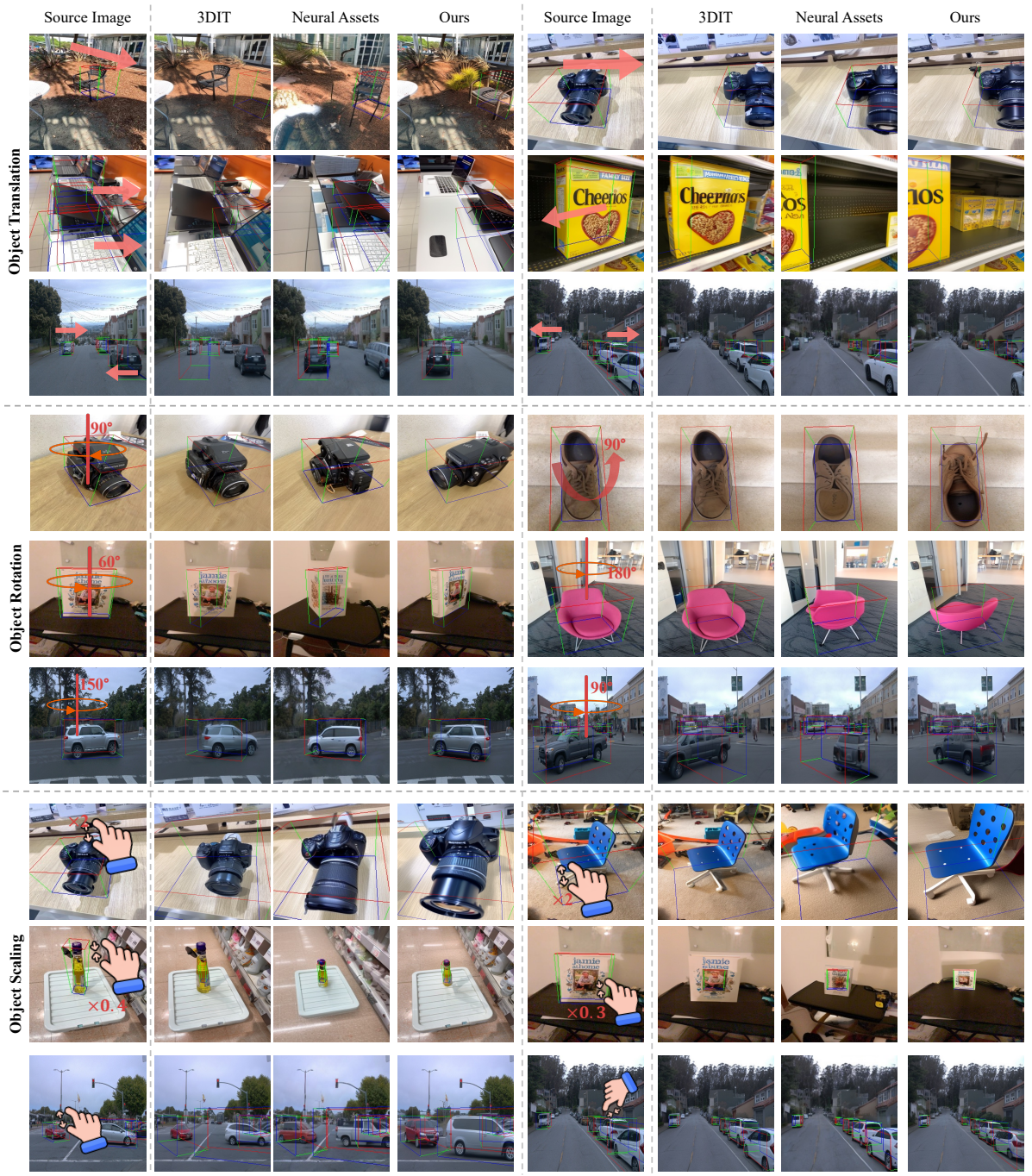

Figure 11 | Additional qualitative results of disentangled object control with fixed camera, extending Figure 5 of the main paper.