This CVPR Findings paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

100Editor: 100+ Views per Batch and Minute-Scale View-Consistent 3D Editing

Cunqi Wu[1] **Peng Zhou**[†,1] **Jie Qin**[1] **Qi Tian**[2]

[1]College of Artificial Intelligence, Nanjing University of Aeronautics and Astronautics
[2]Huawei Technologies Ltd.

{cunqi_wu, zhoupeng23, jie.qin}@nuaa.edu.cn, tian.qi1@huawei.com

https://devin100086.github.io/100Editor/

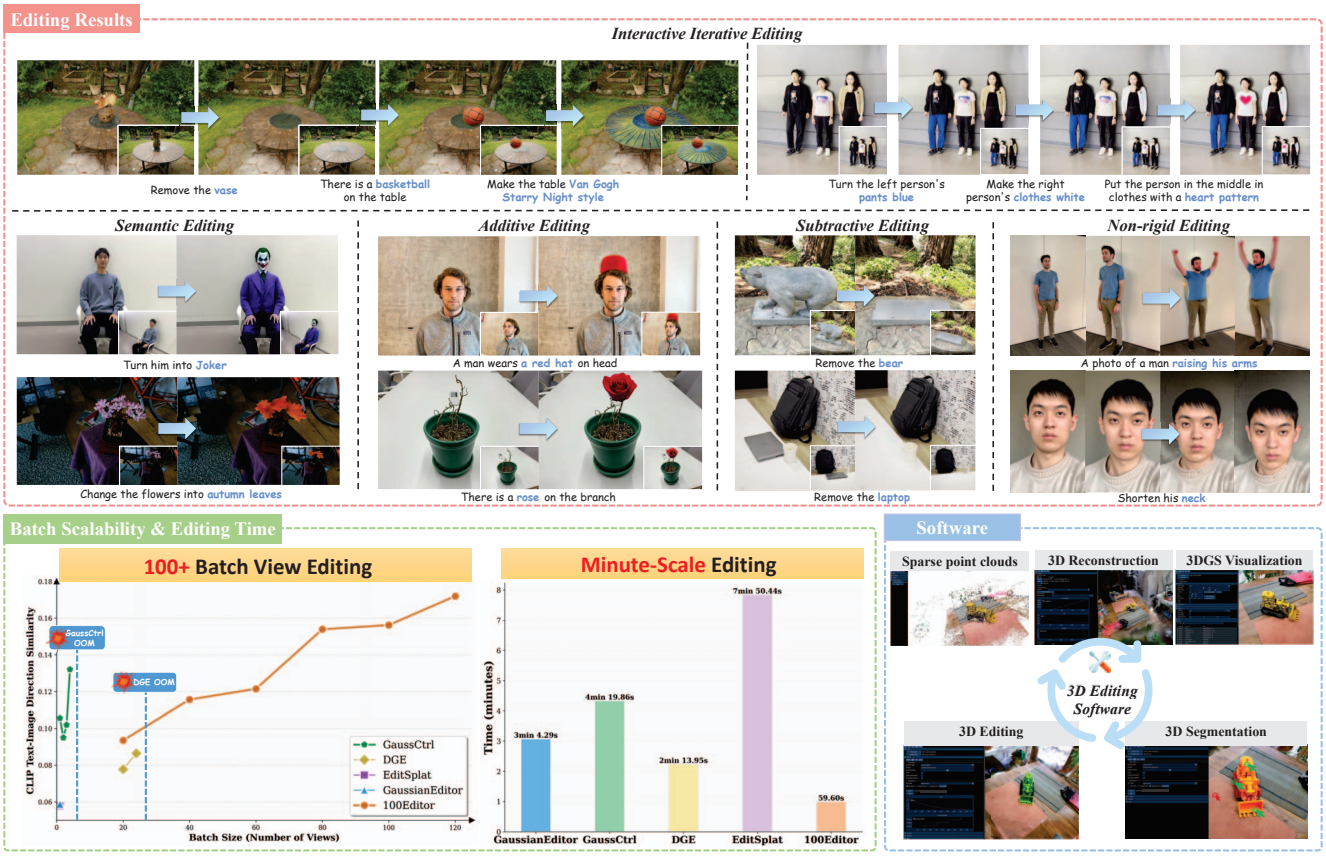


Figure 1. Our method supports diverse 3D scene edits, including semantic, additive, subtractive, and non-rigid modifications. Under a 24 GB GPU memory budget, 100Editor scales to batch sizes of over 100 views (up to 120) with minute-scale editing latency, yielding an order-of-magnitude larger batch size than existing methods. The CLIP direction scores show that edit quality consistently improves as batch size increases. To make such large-batch editing usable in practice, we provide a 3D editing software that integrates scene visualization, reconstruction, interactive 3D segmentation, and edit control in a user-friendly interface.

Abstract

Editing 3D scenes with diffusion models and 3DGS remains slow: most pipelines update one view at a time and are constrained by VRAM-limited batch sizes. We introduce 100Editor, a training-free framework that scales multi-view 3D editing to the hundred-view regime while preserving cross-view consistency. The system combines four complementary components: (1) a batch-consistent multi-view editing module that aligns overlapping content across views at the token level with chunked execution; (2) an efficiency suite that integrates a lightweight 3DGS renderer, a sparse optimizer,

† Corresponding author.

a CLIP-guided Patience Stopping (CPS) rule, and parallelized diffusion inference to reduce editing latency; (3) an interactive 3D segmentation module with point prompts and 3D back-projection for accurate, object-level local editing; and (4) a practical 3D editing software that unifies these capabilities for semantic, additive, subtractive, and non-rigid editing. Together, these designs enable large-batch and efficient 3D editing without modifying the image-editing model's weights. On a single 24 GB GPU, 100Editor edits 100+ views per batch (up to 120) and achieves minute-scale 3D scene editing latency (59.60 s). Experiments across diverse scenes and edit types show improved multi-view consistency and high perceptual quality compared to single-view and small-batch baselines, while supporting precise, interactive 3D edits.

1. Introduction

Editing 3D scenes with image-editing models, such as diffusion-based editors [5, 52, 65, 69], has progressed rapidly. Early NeRF-based pipelines [21, 31, 44] typically edit one rendered view at a time and supervise the 3D model with the edited image. Because these methods lack explicit cross-view consistency guarantees, they require many optimization iterations to converge and are further slowed by NeRF's expensive volumetric rendering with dense per-ray sampling. 3DGS [29] offers higher training and rendering throughput and has been adopted by recent 3D editors [11, 18, 35, 60, 62, 75]. However, the core limitations persist: per-view pipelines remain slow, while batch-based variants [10, 19, 36, 61, 63, 73] are constrained by GPU memory to relatively small batch sizes and often struggle to maintain cross-view consistency within the batch.

Our objective is to scale the per-edit batch size under a tight VRAM budget while enforcing cross-view consistency, so that a single scene edit can be executed at minute-scale latency on a single GPU and becomes practical for interactive use. At the same time, we require the pipeline to be training-free with respect to the diffusion-based image editor (no fine-tuning), while still delivering view-consistent, high-quality results. Achieving precise local edits introduces an additional challenge: reliable masks are crucial. Purely text-driven or 2D-only segmentation is often brittle on fine-grained objects or cluttered scenes, whereas native 3D segmentation approaches typically rely on costly per-scene training [7, 23, 47, 66]. We thus target precise, interactive 3D masking that is cross-view consistent, does not require per-scene training, and integrates seamlessly with our large-batch editing pipeline.

We present 100Editor, a large-batch, fast-editing framework that combines three main components. First, for large-batch, view-consistent editing, we chunk the VAE and UNet within a diffusion-based editor to avoid peak VRAM exhaustion, and employ a token-level mechanism [4, 38, 53] that merges semantically corresponding feature tokens across views within each batch. This token merging reduces the attention cost inside the diffusion model and promotes multi-view consistency, without altering the model's weights. Second, we employ a set of accelerations for both the 3D representation and the diffusion editor: an efficient 3DGS renderer [42] and a sparse Adam optimizer reduce the cost of 3DGS updates; a CLIP-guided patience stopping rule prunes non-productive iterations; and on the diffusion model, we reuse encoder features across timesteps to enable parallel decoding. Third, we provide an interactive 3D segmentation module and editing software that allow users to specify target objects through click-point prompts within a graphical user interface (GUI). Compared to text-driven or 2D/3D segmentation methods, our method yields more precise results for fine-grained objects in cluttered scenes, without per-scene training.

We highlight our contributions as follows:

- A training-free module for large-batch, view-consistent editing that encourages token-level cross-view alignment without modifying the image editor's parameters.
- A minute-scale editing pipeline that accelerates both 3DGS optimization and diffusion inference via efficient rendering, sparse optimization, CLIP-guided patience stopping, and parallel decoding.
- An interactive 3D segmentation module and accompanying 3D editing software that provide precise local control over scene edits via cross-view–consistent masks and user-friendly GUI interactions.

Experiments on public datasets demonstrate that 100Editor improves cross-view consistency and perceptual quality over per-view and small-batch baselines, and we provide extensive ablations and timing analyses. As shown in Fig. 1, on a single RTX 4090 with 24 GB of VRAM, 100Editor processes batches of over 100 views (up to 120 in our experiments) and achieves minute-scale latency (59.60 s) while delivering high-fidelity edits and precise local control. In addition, the accompanying editing software integrates our large-batch pipeline and supports semantic, additive, subtractive, and non-rigid edits, enabling fast and practical interactive workflows.

2. Related Work

Scene Reconstruction with 3DGS. With markedly improved training and rendering efficiency, 3DGS [29] has surpassed NeRF [43] as the leading paradigm for 3D scene representation. Recent work explores feed-forward 3DGS reconstruction [8, 22, 27, 54, 70], aiming to predict scene representations directly from input views. However, these methods still require considerable computational resources at reconstruction time. Other studies improve the quality and efficiency of 3DGS optimization and rendering [12,

14, 16, 20, 41, 64, 68]. For example, Speedy-Splat [20] improves 3DGS efficiency through more precise Gaussian localization and training-time pruning of primitives; DRGS [14] introduces depth regularization to mitigate overfitting in few-shot 3DGS optimization; and SparseGS [64] leverages depth priors and pruning to mitigate floater artifacts under sparse-view settings. Most closely related to our setting, 100Editor also adopts depth regularization to reduce floaters and thereby enable more accurate local editing in 3DGS.

3DGS-based 3D Scene Editing. Although NeRF-based methods [15, 17, 21, 24, 31, 44, 45, 51, 55, 58, 59] have advanced scene editing, recent works are increasingly shifting toward 3DGS [29]. Studies [26, 40, 67, 74] utilize a reference image to optimize the explicit properties of Gaussian primitives, thereby achieving style transfer for the entire 3D scene. However, they can only perform simple scene transformation operations and fail to achieve more detailed editing. Scene editing [28, 32–34, 48, 71, 76, 77] based on Score Distillation Sampling (SDS) [46] has recently attracted increasing attention. To improve consistency across edited views, recent methods [13, 57, 72] typically fine-tune 2D diffusion models. Despite their promising results, these approaches often incur substantial computational overhead and require time-consuming editing procedures.

Furthermore, several methods [9, 11, 60, 75] perform scene editing through iterative dataset updates, while others [35, 62] rely on editing a source view followed by filtering high-quality 2D results. However, these approaches still face challenges in maintaining consistency across views and remain computationally expensive. More closely related to our work, recent studies [10, 19, 36, 61, 63, 73] adopt batch view editing for full-scene 3D editing. Specifically, GaussCtrl [63] maintains multi-view consistency during editing by leveraging additional depth maps. DGE [10] improves cross-view consistency through geometry-aware multi-view editing and directly optimizing the 3DGS representation. VcEdit [61] employs a cross-attention consistency module to better align multi-view features and mitigate inconsistencies across edited views. Nevertheless, these methods remain computationally inefficient in batch editing and do not fully exploit cross-view feature interactions, which can result in less consistent edits across views. In contrast, 100Editor improves editing consistency by merging overlapping content across views, while accelerating editing with a lightweight 3DGS renderer, a sparse optimizer, and parallel diffusion inference.

3. Method

In this section, we describe the components of 100Editor and how they are composed into a full editing pipeline. An overview of our framework is shown in Fig. 2.

3.1. Large-Batch, View-Consistent Editing

Most prior methods (e.g., GaussianEditor [11], Instruct-NeRF2NeRF [21]) update a 3D representation by editing one rendered view at a time and supervising the 3D model with the edited image. Because per-view edits are not guaranteed to be multi-view consistent, many optimization iterations are required before the 3D representation converges to the edited state. A natural acceleration is to edit many views in one shot. However, existing batch-editing approaches are constrained by VRAM and typically use small batch sizes. Our goal is to scale a single editing batch to the order of hundreds of views on a 24 GB GPU (RTX 4090), while (i) preserving multi-view consistency and (ii) remaining training-free, without fine-tuning the image editing model.

Our key insight is that different views of the same scene overlap in content. This cross-view redundancy can be exploited during large-batch editing. Similar ideas have been explored in other domains, such as 3D vision (FastVGGT [53]) and video editing (VidToMe [38]). Diffusion-based editors first encode images into a VAE latent grid (tokens), and then apply stacks of self and cross-attention. If we temporarily merge tokens that correspond to overlapping regions across views, two benefits follow: (i) semantically corresponding regions share a common representation, promoting consistent edits; and (ii) the number of tokens decreases, lowering the cost of computation. This operates entirely at the token level, leaves model weights untouched, and is therefore training-free. Specifically, we (a) chunk large batches to avoid peak VRAM, (b) perform Local Views Token Merging within each chunk, (c) exchange features across chunks via a Global Token Bank for consistency propagation, and (d) run self-attention on merged tokens before unmerging them to restore the original per-view shapes for subsequent modules.

Local Views Token Merging. Let $X \in \mathbb{R}^{B\times N\times C}$ be the tokenized inputs of a batch of B views, where N is the number of VAE-latent tokens per view and C is the channel dimension. To control memory, we chunk X into $\{x_i\}_{i=1}^{n}$ with $x_i \in \mathbb{R}^{b_i\times N\times C}$ and $\sum_i b_i = B$. Given a chunk $x_i \in \mathbb{R}^{b_i\times N\times C}$, split it into an anchor set x_i^{anchor} and a source set x_i^{source}. We flatten view and spatial axes so that $x_i^{\text{anchor}} \in \mathbb{R}^{(b_i^a N)\times C}$ and $x_i^{\text{source}} \in \mathbb{R}^{(b_i^s N)\times C}$ with $b_i^a + b_i^s = b_i$. Compute a cosine-similarity matrix $\mathcal{S} \in \mathbb{R}^{(b_i^s N)\times(b_i^a N)}$ between all source and anchor tokens:

$$\mathcal{S}_{k,j} = \frac{\langle (x_i^{\text{source}})_k, (x_i^{\text{anchor}})_j \rangle}{\|(x_i^{\text{source}})_k\| \, \|(x_i^{\text{anchor}})_j\|}. \tag{1}$$

For each source token $k \in \{1, \dots, b_i^s N\}$, assign a nearest neighbor: $j^*(k) = \arg\max_{j\in\{1,\dots,b_i^a N\}} \mathcal{S}_{k,j}$. Select the top-$m$ source tokens by descending $\mathcal{S}_{k,\, j^*(k)}$, where

$$m = \lfloor \rho \, b_i^s N \rfloor, \qquad \rho \in (0, 1], \qquad m \le b_i^s N. \tag{2}$$

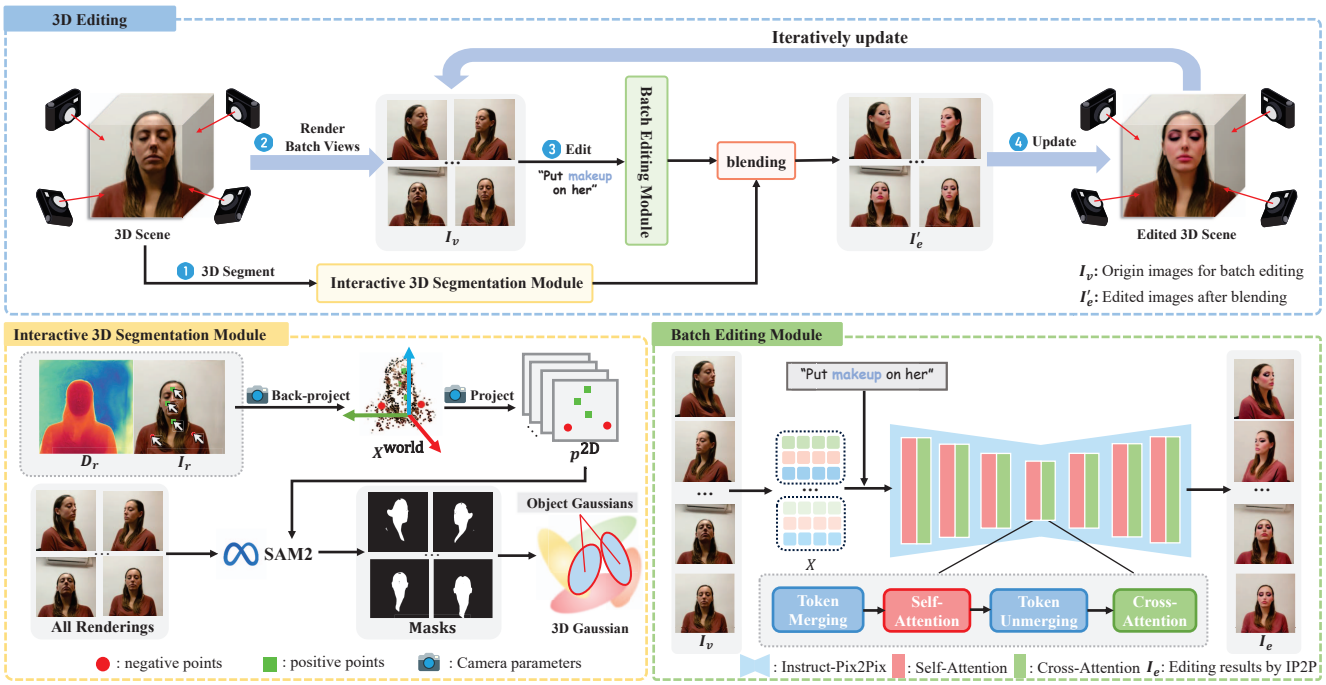


Figure 2. **Overview of 100Editor.** 100Editor primarily comprises three key modules: (1) a large-batch, view-consistent editing module (Sec. 3.1) that enhances multi-view consistency during batch editing via views token merging; (2) a fast editing module (Sec. 3.2) that accelerates both 3DGS optimization and diffusion inference, enabling minute-scale editing latency; and (3) an interactive 3D segmentation module (Sec. 3.3) that leverages point prompts and 3D back-projection to enable precise editing.

These m tokens are merged into their matched anchors, while the remaining $b_i^s N - m$ source tokens are kept. This reduces the token count within the chunk from $b_i N$ to $b_i N - m$, improving the efficiency of subsequent self-attention while aligning semantically similar regions across views. For a detailed analysis of computational complexity, see the Appendix B.1. The original tokenization is later restored by the unmerging step.

Global Views Token Merging. Local merging only ensures consistency within chunks, while insufficient cross-chunk feature alignment leads to inconsistencies in the overall batch editing. To address this, we maintain a global token bank $\mathbf{G}$ that stores representative tokens from previously processed chunks. Let x_i^{lm} denote the locally merged tokens of the i-th chunk (the output of Local Views Token Merging). When processing the i-th chunk, to mitigate ordering bias, we randomly swap the roles of x_i^{lm} and $\mathbf{G}$ as source/anchor with probability p, and then apply the same procedure as in Local Views Token Merging (cosine similarity, greedy nearest-neighbor assignment, and top-m selection) with a global ratio ρ_g to merge the source into the anchor, yielding x_i^{merge}. The unmerged source tokens are subsequently used to update $\mathbf{G}$. This cross-chunk sharing via the global bank creates long-range dependencies and enhances multi-view consistency for large-scale batch editing.

Views Token Unmerging. After the self-attention operation, we restore the compressed tokens to their original shape for subsequent processing. Token Unmerging acts as the inverse of Token Merging: the output tokens are first separated into global and local tokens, after which the local tokens are unmerged to recover the original number of view tokens for cross-attention. In this way, our method integrates seamlessly into the original pipeline without modifying any parameters of the image editing model.

3.2. Toward Minute-Scale 3D Editing

Low-latency feedback is critical for practical 3D scene editing. We target minute-scale latency on a single GPU by accelerating both 3DGS optimization and diffusion-based image editing inference.

3DGS Optimization Speedups. We adopt the efficient renderer from Taming-3DGS [42] to speed up Gaussian primitive processing without degrading quality. Moreover, we use a sparse Adam optimizer that updates only parameters with non-zero gradients. These choices lower per-iteration computation and improve convergence of the 3D representation during editing. To avoid redundant iterative updates, we propose CLIP-guided Patience Stopping (CPS). We monitor several held-out viewpoints using a CLIP-based direction similarity [49] and, at fixed intervals (e.g., every $s = 20$ steps), evaluate the score. If the score decreases

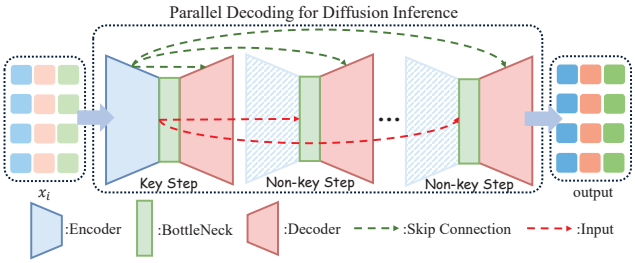


Figure 3. **Encoder-Cached Parallel Decoding.** We enable parallel decoding for multiple non-key steps to accelerate diffusion inference. Please refer to Sec. 3.2 for details.

for P consecutive checks (e.g., $P = 4$), we terminate the current 3DGS optimization round and proceed to the next batch editing round. This criterion prunes non-productive iterations while preserving quality.

Diffusion Acceleration via Parallel Decoding. For diffusion inference, the main bottleneck is the U-Net forward pass, repeated across many denoising steps per sample. Prior analysis [37] shows that the U-Net encoder changes little between adjacent timesteps, whereas decoder features vary substantially within a timestep. Leveraging this observation, we propose a parallelization scheme for InstructPix2Pix[5] that reduces inference time without compromising quality. As shown in Fig. 3, we partition the U-Net into Encoder, Bottleneck, and Decoder, and divide the sampling trajectory into key and non-key timesteps. Encoder outputs from key steps are shared with the Bottleneck and Decoder of subsequent non-key timesteps to enable parallel decoding. In addition, because encoder features change more in early steps and less in later steps, we adopt an early-biased key-timestep schedule, selecting more keys early and fewer in later steps.

Together, these components cut both 3DGS and diffusion computation. The accelerations are training-free, require no changes to the image-editing model weights, and complement the batch-consistent editing in Sec. 3.1. In practice, they enable minute-scale editing latency on a single GPU and scale to 100+ views per batch.

3.3. 3D Segmentation and Editing Software

To enable precise local edits, we develop an interactive 3D segmentation module that produces cross-view–consistent region-of-interest masks. While SAM2 [50] yields temporally consistent 2D masks in videos, it assumes sequential frames along a single camera trajectory. To support arbitrary viewpoints (not restricted to consecutive frames), we lift a few clicks in a reference view to 3D via back-projection using rendered depth, and then forward-project them to all views as 2D point prompts. Feeding these prompts to SAM2 in image mode produces per-view masks.

As illustrated in Fig. 2, let the reference image be I_r with depth D_r, a shared intrinsic matrix K used for all views, and a camera-to-world matrix T_r^{c2w}. For a clicked pixel $p_r^{\mathrm{2D}} = [u, v]^\top$ with depth $d = D_r(u, v)$, we back-project to world coordinates:

$$\begin{aligned} x_r^{\mathrm{cam}} &= d\, K^{-1}\, [p_r^{\mathrm{2D}}, 1]^\top, \\ X^{\mathrm{world}} &= T_r^{\mathrm{c2w}}\, [x_r^{\mathrm{cam}}, 1]^\top, \end{aligned} \tag{3}$$

where $X^{\mathrm{world}} \in \mathbb{R}^4$ denotes the homogeneous world-coordinate of the clicked point p_r^{2D}. To address depth artifacts induced by floaters in 3DGS, a gentle regularization is applied to the depth during the reconstruction process (additional details are provided in the Appendix A).

For any target view j with world-to-camera matrix T_j^{w2c}, the 2D point is obtained by

$$p_j^{\mathrm{2D}} \sim K\, T_j^{\mathrm{w2c}} X^{\mathrm{world}}, \tag{4}$$

where $\sim$ denotes equality in homogeneous coordinates. For brevity of notation, the homogeneous-to-inhomogeneous conversion is performed implicitly; that is, $T_j^{\mathrm{w2c}} X^{\mathrm{world}}$ is dehomogenized before multiplication by K. The propagated foreground (and optionally background) point prompts are fed to SAM2 in image mode to produce per-view masks. To precisely optimize the Gaussian primitives within specific regions, we then lift these 2D masks back to 3D and construct a mask over Gaussian primitives. A blending operation is then applied using the 2D masks to avoid affecting irrelevant regions. More details are provided in the Appendix B.

3D Editing Software. We develop a 3D editing software. It supports mouse and keyboard interactions and integrates our interactive 3D segmentation module and various editing functions (see Fig. 1): (i) semantic editing (text-guided or region-conditioned); (ii) additive editing (inserting objects into a user-selected region); (iii) subtractive editing (removing selected objects); and (iv) non-rigid editing (dragging for local geometric adjustments). Combined with 100+ views-per-batch processing and minute-scale latency, users can perform multiple precise edits on a scene without long waits, substantially improving the interactive experience. The software also provides COLMAP sparse point cloud visualization, 3DGS training and visualization. Further implementation details are provided in the Appendix D.

4. Experiments

In this section, we present implementation details, compare 100Editor with existing methods through qualitative and quantitative evaluations, and conduct ablation studies to validate the contribution of each component.

4.1. Implementation Details

Our implementation is built on a 3DGS variant, namely the efficient renderer from Taming-3DGS [42], which accelerates Gaussian primitive processing. We further extend its

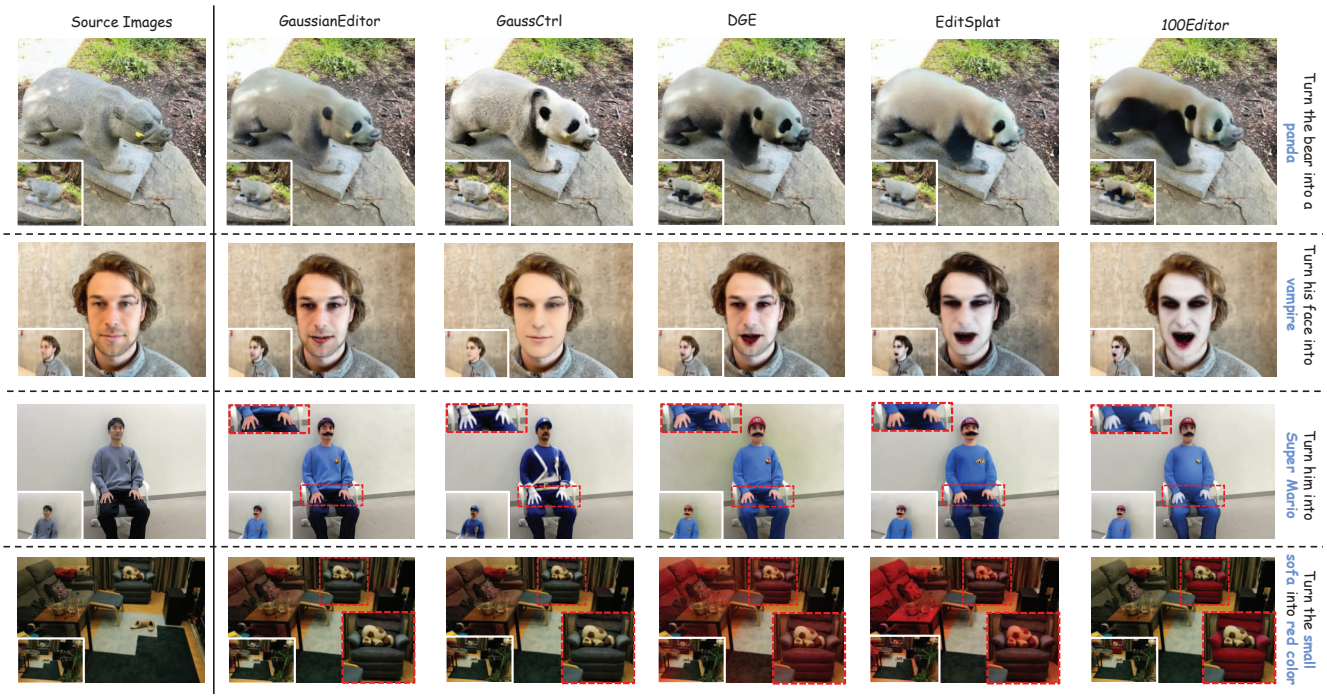


Figure 4. **Qualitative Comparison.** Compared to baseline methods, 100Editor produces more pronounced and detailed editing effects. In cluttered scenes, it enables more precise manipulation of designated objects, leading to cleaner and more faithful edits.

CUDA rasterizer to support our 3D segmentation. We evaluate 100Editor for 3D scene editing on a diverse collection of datasets, including scenes from Mip-NeRF 360 [3], IN2N [21], 3DEgo [30], NeRF-Art [59], PDS [34], scan105 from the DTU dataset [1], as well as additional real-world scenes captured by us. All experiments are conducted on a single NVIDIA RTX 4090 GPU. Additional implementation details are provided in the Appendix C.1.

4.2. Qualitative Analysis

We present qualitative comparisons of 100Editor against competing methods on semantic, additive, subtractive, and non-rigid editing tasks.

Semantic Editing. As shown in Fig. 4, we qualitatively compare 100Editor with existing 3DGS–based editing methods, including GaussianEditor [11], DGE [10], GaussCtrl [63], and EditSplat [35]. Under the prompt *"Turn his face into vampire"*, GaussianEditor, DGE, and GaussCtrl produce only subtle changes to the human face, while EditSplat introduces noticeable background artifacts. In contrast, 100Editor generates more pronounced facial edits while maintaining high visual quality and minimal artifacts. For the instruction *"Turn him into Super Mario"*, baseline methods either fail to refine details such as the hands or produce structural distortions (e.g., GaussCtrl reduces the subject's five fingers to four), whereas 100Editor enables high-fidelity 3D editing without such distortions. For cluttered scenes, we further examine the prompt *"Turn the small sofa into red color"*. Existing methods struggle to accurately localize the target object in complex environments and tend to introduce severe artifacts in the edited scenes. In comparison, 100Editor accurately identifies and edits the small sofa while preserving the surrounding content.

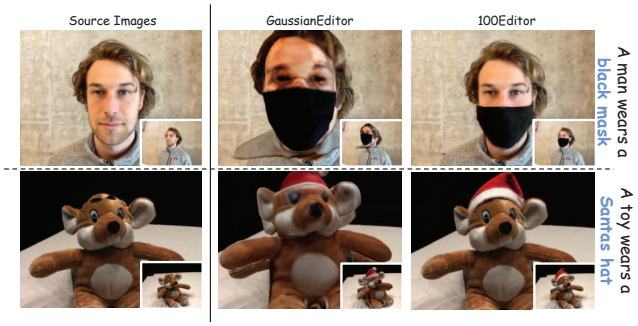


Figure 5. **Additive Editing Comparison.** Compared to GaussianEditor, 100Editor adopts a coarse-to-fine editing paradigm that combines initial object insertion with subsequent batch multi-view optimization using our large-batch editing module, resulting in higher-fidelity outputs with fewer artifacts.

Additive Editing. As shown in Fig. 5, 100Editor achieves superior additive editing compared to existing approaches. GaussianEditor [11] edits a single view and then embeds the generated mesh (from a background-removed image) back into the original 3D scene. This often fails to integrate the inserted object coherently, leading to noticeable artifacts

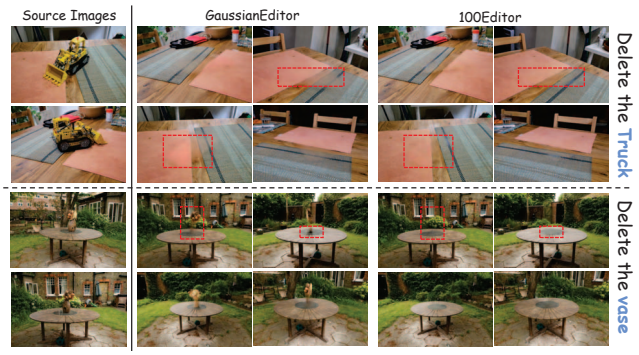


Figure 6. **Subtractive Editing Comparison.** With the interactive 3D segmentation module, 100Editor enables more precise removal of objects in scenes, while its large-batch view-consistency module effectively minimizes artifacts in the inpainted regions.

Table 1. **Quantitative comparison with other models.** Compared to other methods, 100Editor achieves superior editing quality while attaining the shortest editing time.

Method	$\text{CLIP}_{\text{sim}}\uparrow$	$\text{CLIP}_{\text{dir}}\uparrow$	MEt3R $\downarrow$	User Study $\uparrow$	Time $\downarrow$
GaussianEditor [11]	0.2222	0.1039	0.0746	12.34%	184.29s
GaussCtrl [63]	0.2389	0.1209	0.0737	9.42%	259.86s
DGE [10]	0.2455	0.1525	0.0756	20.78%	133.95s
EditSplat [35]	0.2471	0.1318	0.0771	15.58%	470.44s
100Editor	**0.2597**	**0.1726**	**0.0726**	**41.88%**	**59.60s**

and inconsistencies. In contrast, we employ a coarse-to-fine strategy for additive editing. We first apply BrushNet [39] to edit a single view, then use SAM2 [50] to segment the target object and DreamGaussian [56] to generate a coarse 3DGS representation, which is inserted into the original 3D scene. Finally, we perform batch multi-view optimization with our large-batch editing module to refine the scene, yielding more accurate and high-fidelity additive edits. Additional details are provided in the Appendix C.2 and B.2.

Subtractive Editing. As shown in Fig. 6, 100Editor delivers cleaner subtractive edits with fewer artifacts and smoother boundaries. For the prompt *"Delete the truck"*, GaussianEditor [11] introduces noticeable color shifts and blurry boundaries in the edited region. In contrast, by leveraging LeftRefill [6] which enables multi-view consistent editing guided by a single reference view, 100Editor effectively suppresses artifacts caused by inconsistent edits across views. For the prompt *"Delete the vase"*, our interactive 3D segmentation module enables complete removal of the target object, and the inpainted background produced by 100Editor is substantially more seamless and coherent than that of GaussianEditor. Additional examples and details are provided in the Appendix C.2 and B.3.

Non-rigid Editing. We achieve non-rigid editing by building on PDS [34] and SC-GS [25], with examples shown in Fig. 1. For PDS, an extension of SDS, we integrate our batch-consistent multi-view editing module into the original framework, yielding higher-fidelity deformations with fewer artifacts. For SC-GS, which manipulates 3DGS via sparse control points but often causes unintended distortions in unrelated regions, we incorporate our interactive 3D segmentation module to restrict dragging to user-specified areas, enabling more precise local deformations. Further details are provided in the Appendix C.2 and B.4.

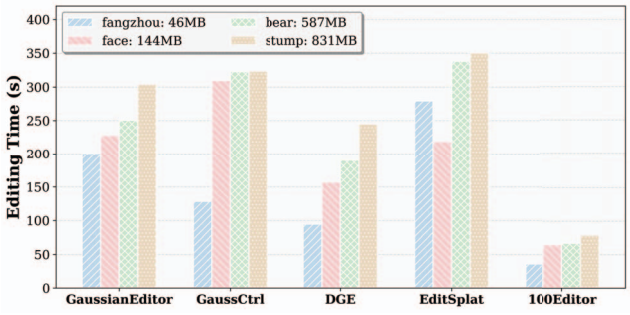


Figure 7. **Editing Time across Scenes of Different Sizes.** Editing time (in seconds) of each method on four scenes whose complexity increases with the size of the reconstructed 3DGS model. The legend in the top-left corner shows the `.ply` file size (MB) of the 3DGS reconstruction before editing for each scene. 100Editor consistently achieves the lowest editing time across diverse scenarios, keeping the editing time around one minute for both small and large scenes.

4.3. Quantitative Analysis

As reported in Table 1, we evaluate editing quality using the CLIP [49] similarity score CLIP_{sim} and directional score CLIP_{dir} for text-image semantic alignment, MEt3R [2] for multi-view consistency, and editing time for efficiency. Specifically, MEt3R score is computed on adjacent frame pairs in the rendered image sequence, and we report the average score over all adjacent pairs as the measure of multi-view consistency. In addition, we conduct a user study in which, for each test scene, participants are shown the results of all methods and asked to select the best one. 100Editor consistently outperforms competing methods on both CLIP metrics, indicating better text–image consistency. Moreover, 100Editor achieves the lowest MEt3R score, suggesting improved cross-view 3D consistency after editing. Together, these gains support the effectiveness of the large-batch multi-view editing strategy. In the user study, 100Editor is chosen as the best result in 41.88% of all votes, reflecting better perceptual quality. In terms of efficiency, 100Editor achieves the fastest editing time, reducing the time overhead by up to $7.89\times$ compared to EditSplat. More metric evaluation details are provided in the Appendix C.3.

Editing Time across Scene Sizes. Fig. 7 reports the editing time of different methods on 3D scenes of varying sizes, where more complex scenes require more Gaussian primitives and thus lead to larger 3DGS `.ply` files

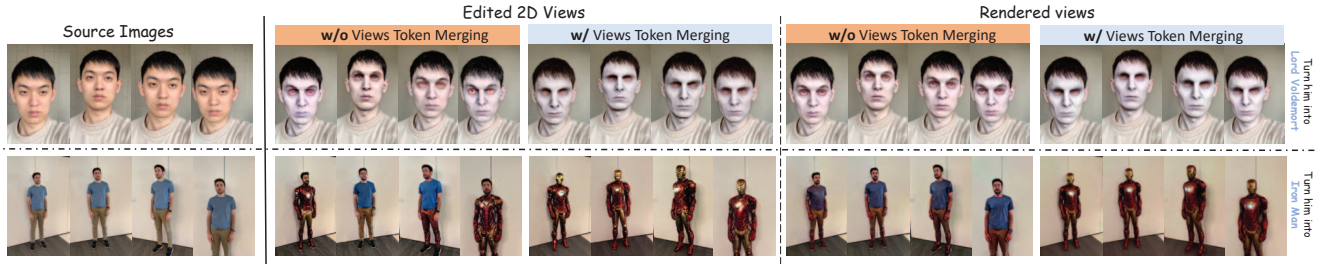


Figure 8. **Ablation on Views Token Merging.** With views token merging enabled, batch-edited 2D views exhibit strong cross-view consistency, leading to clearer and more faithful renderings of the final edited scenes. In contrast, disabling this component causes noticeable inconsistencies among the batch-edited images and ultimately results in lower-quality 3D editing outputs.

Table 2. **Comparison of Batch Scalability.** 100Editor supports multi-view editing with larger batch sizes while achieving more efficient memory usage.

Batch	VRAM[GB]				
	GaussianEditor[11]	GaussCtrl[63]	DGE[10]	EditSplat[35]	100Editor
1	8.23	17.01	**7.00**	15.88	7.51
10	-	OOM	14.07	-	**11.85**
20	-	OOM	16.62	-	**12.09**
40	-	OOM	OOM	-	**12.57**
60	-	OOM	OOM	-	**13.05**
100	-	OOM	OOM	-	**13.98**

(in MB). Here, the `.ply` size refers to the reconstructed 3DGS model before editing begins. Across all scene scales, 100Editor consistently achieves the shortest editing time. For larger scenes, GaussCtrl [63], DGE [10], and EditSplat [35] frequently run into out-of-memory issues, whereas our chunk-based strategy effectively manages GPU memory, enabling editing on larger scenes and with larger view batches.

Batch Scalability Evaluation. As reported in Table 2, we evaluate the VRAM scalability of 100Editor for multi-view editing. With a batch size of 1, its VRAM consumption (7.51 GB) is comparable to that of DGE [10] (7.00 GB). As the batch size increases, the advantage of 100Editor becomes more evident: when the batch size reaches 40, all baseline methods encounter out-of-memory (OOM) failures, whereas 100Editor remains stable and can scale to a batch size of 100 with a VRAM footprint of only 13.98 GB. In principle, 100Editor can support even larger batch sizes. In the experiments, the maximum batch size is limited by the number of available views (slightly over 100), and we provide additional tests on more scenes in the Appendix C.3.

4.4. Ablation Studies

To assess the effectiveness of our design, we conduct a series of ablation studies. Fig. 8 highlights the role of the views token merging strategy in maintaining multi-view editing consistency. Without views token merging, the edits become noticeably inconsistent across viewpoints, leading to incomplete changes and visible artifacts in the final reconstructions. In contrast, enabling views token merging yields more consistent edits across views, with higher color fidelity and more refined local details.

Table 3. **Impact of Individual Modules on Editing Time.** We report ablations over batch editing, lightweight renderer, parallel decoding, and the CLIP-guided patience stopping (CPS) strategy. Each component contributes to reducing the overall editing time, and their combination yields the fastest configuration.

Modules enabled				Time [s]	Δ [s]
Batch Editing	Lightweight Renderer	Parallel Decoding	CPS		
	∅ (baseline)			184.29	–
✓				116.24	**68.05**
✓	✓			95.24	21.00
✓	✓	✓		86.83	8.41
✓	✓	✓	✓	**59.60**	27.23

Table 3 reports the impact of each component on editing time. Introducing the batch editing strategy first reduces the total time by 68.05 seconds. Adding the lightweight renderer and parallel decoding yields additional savings of 21.00 seconds and 8.41 seconds, respectively. Finally, integrating the CPS module contributes a further reduction of 27.23 seconds, bringing the overall editing time down to 59.60 seconds. These results show that all modules are important for accelerating the pipeline, with the batch editing strategy providing the largest gains.

5. Conclusion

We propose 100Editor, a training-free framework for large-batch 3D scene editing. The method combines a batch-consistent editing module with an efficiency-oriented design that includes a lightweight 3DGS renderer, a sparse optimizer, CLIP-guided patience stopping, and parallelized diffusion inference, enabling hundred-view batches under a single 24 GB GPU with minute-scale latency. An interactive 3D segmentation module and unified 3D editing software further provide precise object-level control, supporting semantic, additive, subtractive, and non-rigid edits.

6. Acknowledgments

This research was primarily supported by the National Natural Science Foundation of China (No. 62402222, No. U25A20533, No. 62276129), the Natural Science Foundation of Jiangsu Province (No. BK20241400, No. BK20250082), the China National Postdoctoral Program for Innovative Talents (No. BX20240484), the Fundamental Research Funds for the Central Universities (No. NE2025010), the New Generation Artificial Intelligence-National Science and Technology Major Project (No. 2025ZD0122903), and the Jiangsu Outstanding Postdoctoral Program.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, 120(2):153–168, 2016. 6

[2] Mohammad Asim, Christopher Wewer, Thomas Wimmer, Bernt Schiele, and Jan Eric Lenssen. Met3r: Measuring multi-view consistency in generated images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6034–6044, 2025. 7

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 6

[4] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[5] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 2, 5

[6] Chenjie Cao, Yunuo Cai, Qiaole Dong, Yikai Wang, and Yanwei Fu. Leftrefill: Filling right canvas based on left reference through generalized text-to-image diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7705–7715, 2024. 7, 5

[7] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3d gaussians. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1971–1979, 2025. 2

[8] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19457–19467, 2024. 2

[9] Jun-Kun Chen and Yu-Xiong Wang. Proedit: Simple progression is all you need for high-quality 3d scene editing. In *Advances in Neural Information Processing Systems*, pages 4934–4955, 2024. 3

[10] Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. In *European Conference on Computer Vision*, pages 74–92, 2024. 2, 3, 6, 7, 8, 4, 9

[11] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21476–21485, 2024. 2, 3, 6, 7, 8, 9

[12] Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. Dashgaussian: Optimizing 3d gaussian splatting in 200 seconds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11146–11155, 2025. 2

[13] Yufeng Chi, Huimin Ma, Kafeng Wang, and Jianmin Li. Disco3d: Distilling multi-view consistency for 3d scene editing. *arXiv preprint arXiv:2508.01684*, 2025. 3

[14] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 811–820, 2024. 3

[15] Jiahua Dong and Yu-Xiong Wang. Vica-nerf: View-consistency-aware 3d editing of neural radiance fields. In *Advances in Neural Information Processing Systems*, pages 61466–61477, 2023. 3, 6

[16] Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Linning Xu, Zhilin Pei, Hengjie Li, et al. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26652–26662, 2025. 3

[17] Haruo Fujiwara, Yusuke Mukuta, and Tatsuya Harada. Style-nerf2nerf: 3d style transfer from style-aligned multi-view images. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–10, 2024. 3

[18] Eyal Gomel and Lior Wolf. Diffusion-based attention warping for consistent 3d scene editing. *arXiv preprint arXiv:2412.07984*, 2024. 2

[19] Lanqing Guo, Yufei Wang, Hezhen Hu, Yan Zheng, Yeying Jin, Siyu Huang, and Zhangyang Wang. Mastering regional 3dgs: Locating, initializing, and editing with diverse 2d priors. *arXiv preprint arXiv:2507.05426*, 2025. 2, 3

[20] Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21537–21546, 2025. 3

[21] 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*, pages 19740–19750, 2023. 2, 3, 6

[22] Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. Pf3plat: Pose-free feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2410.22128*, 2024. 2

[23] Xu Hu, Yuxi Wang, Lue Fan, Chuanchen Luo, Junsong Fan, Zhen Lei, Qing Li, Junran Peng, and Zhaoxiang Zhang. Sagd: Boundary-enhanced segment anything in 3d gaussian via gaussian decomposition. *arXiv preprint arXiv:2401.17857*, 2024. 2

[24] Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18342–18352, 2022. 3

[25] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4220–4230, 2024. 7, 5

[26] Sahil Jain, Avik Kuthiala, Prabhdeep Singh Sethi, and Prakanshul Saxena. Stylesplat: 3d object style transfer with gaussian splatting. *arXiv preprint arXiv:2407.09473*, 2024. 3

[27] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics*, 44(6):1–16, 2025. 2

[28] Hiromichi Kamata, Yuiko Sakuma, Akio Hayakawa, Masato Ishii, and Takuya Narihira. Instruct 3d-to-3d: Text instruction guided 3d-to-3d conversion. *arXiv preprint arXiv:2303.15780*, 2023. 3

[29] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2, 3

[30] Umar Khalid, Hasan Iqbal, Azib Farooq, Jing Hua, and Chen Chen. 3dego: 3d editing on the go! In *European Conference on Computer Vision*, pages 73–89, 2024. 6

[31] Umar Khalid, Hasan Iqbal, Nazmul Karim, Muhammad Tayyab, Jing Hua, and Chen Chen. Latenteditor: Text driven local editing of 3d scenes. In *European Conference on Computer Vision*, pages 364–380, 2024. 2, 3

[32] Hayeon Kim, Ji Ha Jang, and Se Young Chun. Robust 3d-masked part-level editing in 3d gaussian splatting with regularized score distillation sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5501–5510, 2025. 3

[33] Jiwook Kim, Seonho Lee, Jaeyo Shin, Jiho Choi, and Hyunjung Shim. Dreamcatalyst: Fast and high-quality 3d editing via controlling editability and identity preservation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[34] Juil Koo, Chanho Park, and Minhyuk Sung. Posterior distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13352–13361, 2024. 3, 6, 7, 5

[35] Dong In Lee, Hyeongcheol Park, Jiyoung Seo, Eunbyung Park, Hyunje Park, Ha Dam Baek, Sangheon Shin, Sangmin Kim, and Sangpil Kim. Editsplat: Multi-view fusion and attention-guided optimization for view-consistent 3d scene editing with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11135–11145, 2025. 2, 3, 6, 7, 8, 9

[36] Ruihuang Li, Liyi Chen, Zhengqiang Zhang, Varun Jampani, Vishal M Patel, and Lei Zhang. Syncnoise: Geometrically consistent noise prediction for instruction-based 3d editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4905–4913, 2025. 2, 3

[37] Senmao Li, Taihang Hu, Joost van de Weijer, Fahad S Khan, Tao Liu, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of the encoder for diffusion model inference. In *Advances in Neural Information Processing Systems*, pages 85203–85240, 2024. 5

[38] Xirui Li, Chao Ma, Xiaokang Yang, and Ming-Hsuan Yang. Vidtome: Video token merging for zero-shot video editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7486–7495, 2024. 2, 3

[39] Yaowei Li, Yuxuan Bian, Xuan Ju, Zhaoyang Zhang, Junhao Zhuang, Ying Shan, Yuexian Zou, and Qiang Xu. Brushedit: All-in-one image inpainting and editing. *arXiv preprint arXiv:2412.10316*, 2024. 7, 5

[40] Kunhao Liu, Fangneng Zhan, Muyu Xu, Christian Theobalt, Ling Shao, and Shijian Lu. Stylegaussian: Instant 3d style transfer with gaussian splatting. In *SIGGRAPH Asia 2024 Technical Communications*, pages 1–4. 2024. 3

[41] Xi Liu, Chaoyi Zhou, and Siyu Huang. 3dgs-enhancer: Enhancing unbounded 3d gaussian splatting with view-consistent 2d diffusion priors. In *Advances in Neural Information Processing Systems*, pages 133305–133327, 2024. 3

[42] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 2, 4, 5

[43] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421, 2020. 2

[44] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Marcus A Brubaker, Jonathan Kelly, Alex Levinshtein, Konstantinos G Derpanis, and Igor Gilitschenski. Watch your steps: Local image and scene editing by text instructions. In *European Conference on Computer Vision*, pages 111–129, 2024. 2, 3

[45] Jangho Park, Gihyun Kwon, and Jong Chul Ye. ED-NeRF: Efficient text-guided editing of 3d scene with latent space NeRF. In *The Twelfth International Conference on Learning Representations*, 2024. 3

[46] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[47] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024. 2

[48] Yansong Qu, Dian Chen, Xinyang Li, Xiaofan Li, Shengchuan Zhang, Liujuan Cao, and Rongrong Ji. Drag your gaussian: Effective drag-based editing with score distillation for 3d gaussian splatting. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–12, 2025. 3

[49] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2021. 4, 7

[50] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025. 5, 7

[51] Sara Rojas, Julien Philip, Kai Zhang, Sai Bi, Fujun Luan, Bernard Ghanem, and Kalyan Sunkavalli. Datenerf: Depth-aware text-based editing of nerfs. In *European Conference on Computer Vision*, pages 267–284, 2024. 3

[52] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 2

[53] You Shen, Zhipeng Zhang, Yansong Qu, Xiawu Zheng, Jiayi Ji, Shengchuan Zhang, and Liujuan Cao. FastVGGT: Fast visual geometry transformer. In *The Fourteenth International Conference on Learning Representations*, 2026. 2, 3

[54] Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs. *arXiv preprint arXiv:2408.13912*, 2024. 2

[55] Liangchen Song, Liangliang Cao, Jiatao Gu, Yifan Jiang, Junsong Yuan, and Hao Tang. Efficient-nerf2nerf: streamlining text-driven 3d editing with multiview correspondence-enhanced diffusion models. *arXiv preprint arXiv:2312.08563*, 2023. 3

[56] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *The Twelfth International Conference on Learning Representations*, 2024. 7, 5

[57] Zeng Tao, Zheng Ding, Zeyuan Chen, Xiang Zhang, Leizhi Li, and Zhuowen Tu. C3editor: Achieving controllable consistency in 2d model for 3d editing. *arXiv preprint arXiv:2510.04539*, 2025. 3

[58] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 3

[59] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):4983–4996, 2023. 3, 6

[60] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20902–20911, 2024. 2, 3

[61] Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-consistent 3d editing with gaussian splatting. In *European Conference on Computer Vision*, pages 404–420, 2024. 2, 3

[62] Minghao Wen, Shengjie Wu, Kangkan Wang, and Dong Liang. Intergsedit: Interactive 3d gaussian splatting editing with 3d geometry-consistent attention prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26136–26145, 2025. 2, 3

[63] Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Adrian Prisacariu. Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing. In *European Conference on Computer Vision*, pages 55–71, 2024. 2, 3, 6, 7, 8, 9, 24

[64] Haolin Xiong, Sairisheek Muttukuru, Hanyuan Xiao, Rishi Upadhyay, Pradyumna Chari, Yajie Zhao, and Achuta Kadambi. Sparsegs: sparse view synthesis using 3d gaussian splatting. In *2025 International Conference on 3D Vision*, pages 1032–1041, 2025. 3

[65] Sihan Xu, Yidong Huang, Jiayi Pan, Ziqiao Ma, and Joyce Chai. Inversion-free image editing with natural language. *arXiv preprint arXiv:2312.04965*, 2023. 2

[66] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European Conference on Computer Vision*, pages 162–179, 2024. 2

[67] Xin-Yi Yu, Jun-Xin Yu, Li-Bo Zhou, Yan Wei, and Lin-Lin Ou. Instantstylegaussian: Efficient art style transfer with 3d gaussian splatting. *arXiv preprint arXiv:2408.04249*, 2024. 3

[68] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 3

[69] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 2

[70] Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21936–21947, 2025. 2

[71] Yuhong Zhang, Han Wang, Yiwen Wang, Rong Xie, and Li Song. Freeinsert: Personalized object insertion with geometric and style control. In *Proceedings of the 33rd ACM In-*

ternational Conference on Multimedia, pages 10361–10369, 2025. 3

[72] Canyu Zhao, Xiaoman Li, Tianjian Feng, Zhiyue Zhao, Hao Chen, and Chunhua Shen. TINKER: Diffusion's gift to 3d–multi-view consistent editing from sparse inputs without per-scene optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. 3

[73] Yian Zhao, Wanshi Xu, Yang Wu, Weiheng Huang, Zhongqian Sun, and Wei Yang. Progdf: Progressive gaussian differential field for controllable and flexible 3d editing. *arXiv preprint arXiv:2412.08152*, 2024. 2, 3

[74] Yian Zhao, Rushi Ye, Ruochong Zheng, Zesen Cheng, Chaoran Feng, Jiashu Yang, Pengchong Qiao, Chang Liu, and Jie Chen. Tune-your-style: Intensity-tunable 3d style transfer with gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19032–19042, 2025. 3

[75] Peng Zhou, Dunbo Cai, Yujian Du, Runqing Zhang, Bingbing Ni, Jie Qin, and Ling Qian. Edit3d: Elevating 3d scene editing with attention-driven multi-turn interactivity. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3401–3410, 2024. 2, 3

[76] Haiming Zhu, Yangyang Xu, Chenshu Xu, Tingrui Shen, Wenxi Liu, Yong Du, Jun Yu, and Shengfeng He. Stable score distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16597–16606, 2025. 3

[77] Jingyu Zhuang, Di Kang, Yan-Pei Cao, Guanbin Li, Liang Lin, and Ying Shan. Tip-editor: An accurate 3d editor following both text-prompts and image-prompts. *ACM Transactions on Graphics*, 43(4):1–12, 2024. 3