3D-LATTE: Latent Space 3D Editing from Textual Instructions

Supplementary Material

This supplementary material presents further applications, method details, ablations and qualitative results, complementing the main paper. It is organized as follows:

- Section 1 introduces additional qualitative comparisons with the baseline methods.
- Section 2 introduces an additional application of our method for controllable synthesis with a coarse 3D geometry proxy.
- Section 3 outlines our experimental setup, including implementation details, evaluation metrics, user study and evaluation dataset, and further motivates some of our design choices.
- Section 4 provides a detailed explanation of our mask generation pipeline.
- Section 5 provides information about the runtime cost of our method and compares efficiency with related works.
- Section 6 presents additional ablations on 3D attention injection.
- Section 7 discusses how our work can potentially be adapted to different 3D backbones and representations.
- Section 8 presents limitations of our work.
- Section 9 discusses the broader impact of our work.

1. Additional Results

We provide further qualitative results of our method compared to baselines in Figure 1, Figure 2, Figure 3 and Figure 4. We observe that our method generates 3D shapes that are faithful to the editing instruction, with accurate 3D geometry and detailed appearance. It shows clear improvements over existing methods in flexibility, precision and visual fidelity.

2. Application for geometry control

Our method holds promise for additional applications, such as controllable 3D synthesis. We introduce an application in which our pipeline can generate 3D assets guided by a user-provided 3D coarse geometric proxy. Motivated by the role of self-attention in encoding 3D structure, spatial layout, and part composition, we inject self-attention from the reference geometry for 80% of the denoising process. With this guidance, the text prompt governs semantic details, while the coarse proxy guides the overall 3D shape. As illustrated in Figure 5, given a simple 3D geometry made from basic Blender primitives, our method can generate detailed variants that are faithful to the text prompt and at the same time follow the provided 3D geometry.

3. Experimental Setup

3.1. Evaluation Metrics

Let $\mathbf{v}_i^{\text{edit}}$ and $\mathbf{v}_i^{\text{input}}$ denote the normalized CLIP image embeddings for the i-th rendered frame of the edited and original 3D shapes, respectively. Similarly, let $\mathbf{t}^{\text{edit}}$ and $\mathbf{t}^{\text{input}}$ denote the normalized CLIP text embeddings of the edited and original prompts. To quantify directional alignment between the image and text edits, we compute the cosine similarity between the difference of image embeddings and the difference of text embeddings, averaged across all N rendered frames:

$$\text{CLIP}_{\text{dir-cos}} = \frac{1}{N}\sum_{i=1}^{N} \cos\left(\mathbf{v}_i^{\text{edit}} - \mathbf{v}_i^{\text{input}},\ \mathbf{t}^{\text{edit}} - \mathbf{t}^{\text{input}}\right) \tag{1}$$

Here, $\cos(\cdot,\cdot)$ denotes the cosine similarity between two vectors. This metric captures whether the visual change between the original and edited shapes aligns with the semantic direction specified by the text edit. We also adopt the CLIP-diff-noedit metric from [3] to measure whether unintended changes occur in objects across views. Specifically, let $\mathbf{t}^{\text{generic}}$ denote the normalized CLIP text embedding of the generic prompt. To construct the generic prompt we substitute the editing region with a placeholder noun (e.g., object). For example, given an edit from "a blue warrior holding a shovel" to "a blue warrior holding a flower," $\mathbf{t}^{\text{generic}}$ corresponds to "a blue warrior holding an object".

We define:

$$\text{CLIP}_{\text{diff-noedit}} = \frac{1}{N}\sum_{i=1}^{N} \left| C(\mathbf{v}_i^{\text{input}}, \mathbf{t}^{\text{generic}}) - C(\mathbf{v}_i^{\text{edit}}, \mathbf{t}^{\text{generic}}) \right|_{\text{rel}} \tag{2}$$

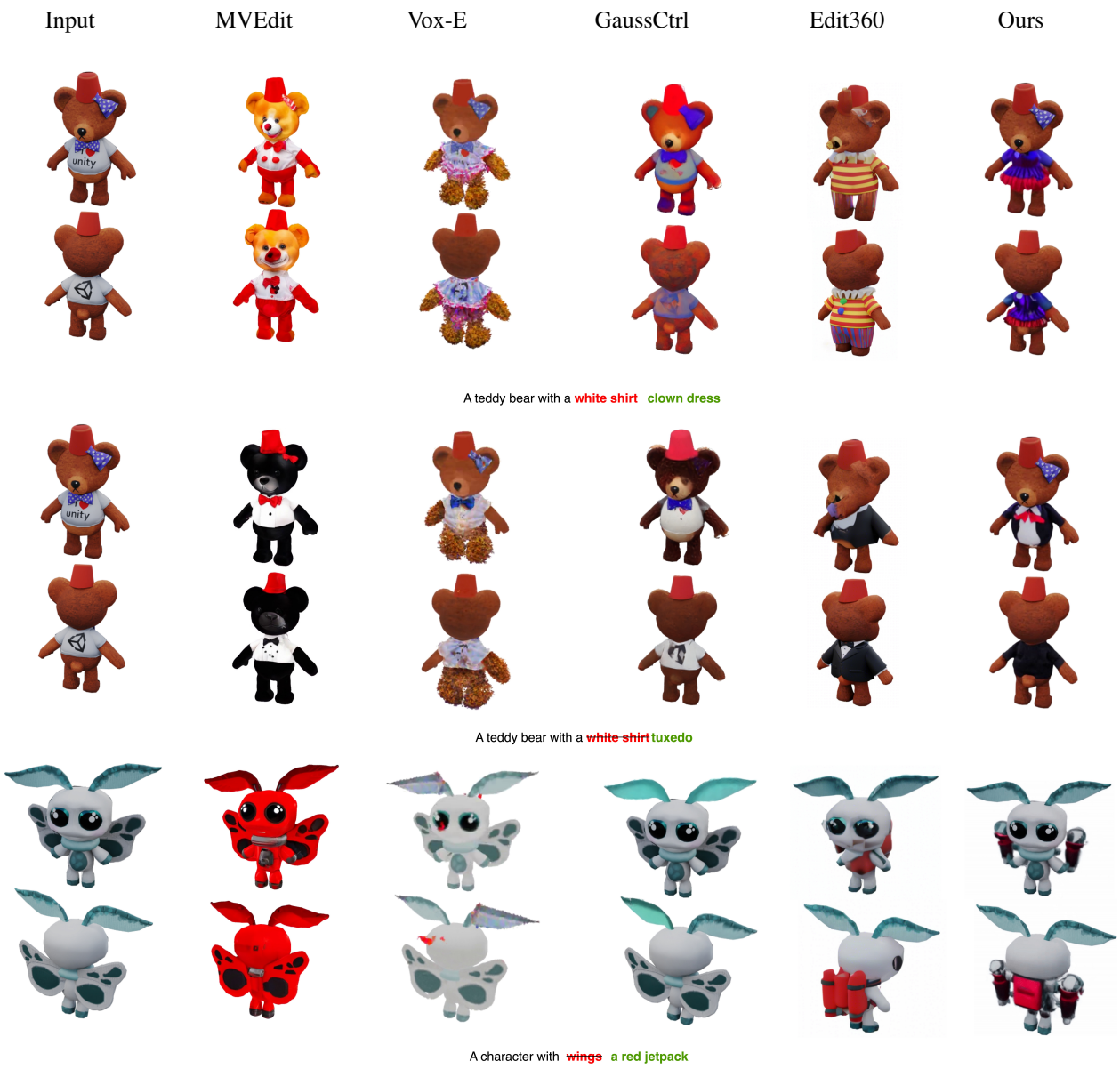

Figure 1. **Additional qualitative comparisons.**

Here, $C(\cdot, \cdot)$ denotes cosine similarity, and the relative difference is defined as:

$$|x - y|_{\text{rel}} = \frac{|x - y|}{\max(x, y)}$$

By evaluating changes in image-to-text similarity before and after the edit with respect to the generic text, we assess whether semantics are preserved across the edited renderings.

3.2. Implementation Details

We run our experiments on an NVIDIA A100 40GB GPU using the PyTorch framework. The number of sampling steps is set to $T = 100$ for both inversion and editing, and we use a classifier-free guidance score of $c_{\text{cfg}} = 8$ during editing. We use the official DiffSplat [8] implementation and set $V_{in} = 4$. For attention injection, we apply cross-attention injection for 70% of the steps and self-attention injection for 30% of the steps. In our frequency annealing strategy, we set $s_l = 2.2$, applying low-frequency enhancement during 70% of the steps (coinciding with attention injection), and use $s_h = 1.1$ for high-frequency enhance-

Figure 2. **Additional qualitative comparisons.**

ment during the remaining steps. For geometry regularization, we set $\gamma_o = 7, \gamma_s = 7$ and $\lambda_o = 0.1, \lambda_\Sigma = 0.1$. For the 3D enhancement part, we render 24 images from our 3DGS representation at a resolution of 1024×1024, using camera views uniformly sampled along a circle looking inward at the 3D asset's center. We use Nerfstudio's

Figure 3. **Additional qualitative comparisons.**

`gsplat` [17] library and train for 7.5K iterations, updating the set of training images every 2.5K iterations. For the ControlNet-Tile updates, we set the number of diffusion steps to 20. The denoising strength, which controls the amount of noise added, and thus how much signal is retained from the original image, is randomly sampled from a fixed range of $[s_{\text{min}}, s_{\text{max}}] = [0.35, 0.45]$. We empirically validate that this range achieves a balanced trade-off between image and view consistency preservation and detail enhancement. To limit view-dependent effects, we set the order of spherical harmonics to 0.

Figure 4. **Additional qualitative comparisons.**

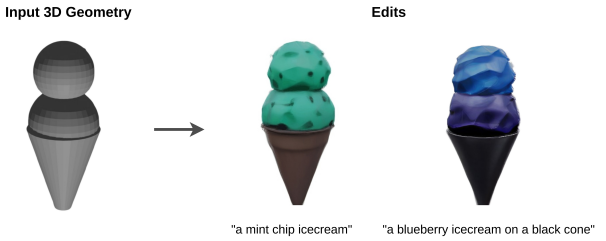

Figure 5. **Controllable text-guided synthesis from a coarse 3D proxy.** Given a coarse 3D geometry composed of simple Blender primitives, our method can generate variants while preserving the input 3D geometry.

3.2.1. Ablation on frequency modulation

In Figure 6, we show an ablation on the effects of the scale parameter value s_l used in low-frequency enhancement. Increasing the scale value enhances the surface generation quality, preserving details from the source object when needed. However, beyond a certain value the textures become oversmoothed and the model's editing ability degrades. In high-frequency enhancement we empirically observe that higher values of scale s_h lead to artifacts in the generated 3D object. Our choice to apply frequency modulation to the skip connection feature maps of the U-Net is empirically validated in our setting and is motivated by the observation of [13] that skip connection maps primarily constitute high-frequency information.

3.2.2. 3D Enhancement Robustness

Iterative update strategies are ill-suited for 3D editing, because 2D edit priors produce inconsistent signals across views. In contrast, enhancement tasks such as super-resolution focus primarily on texture refinement and detail recovery, without modifying the underlying 3D structure. As a result, they introduce far fewer multi-view inconsistencies during iterative optimization than edits that alter geometry or semantics. Thus, enhancement tasks fit naturally into iterative pipelines and converge substantially faster. We further validate empirically that our parameter choices yield fine-grained detailing without causing multi-view inconsistencies or unintended geometric changes.

3.2.3. 3D Inversion Robustness

We also investigate the 3D inversion robustness of our approach, to ensure generizability to complex and diverse 3D assets. We report the PSNR metric over rendered views across all evaluation assets. This metric captures how well the reconstruction matches the original input, serving as an inversion-error metric. In total, we test on our evaluation set as well as on an additional set of 15 objects from the Google Scanned Objects (GSO) dataset, a benchmark featuring real-world 3D scans that were not observed during the training of our 3D backbone. The average PSNR is 27.2 dB. For reference, naive inversion yields a score of 12.7 dB. This result confirms that our inversion process can generalize to out-of-distribution object-centric scenes, both real and synthetic.

3.3. User Study

To assess the perceptual quality of our edits, we conducted a user study involving 57 participants. Participants were recruited through academic and industry mailing lists and included graduate students, researchers, and industry professionals in computer vision, graphics, or related areas. Each participant was presented with images rendered from the original object alongside the corresponding text prompt used for editing. The choices included rendered views edited by our method as well as those generated by the three baseline models, all presented in a randomized order to prevent participants from inferring which model produced which result.

Participants were asked to select the best result according to the following two criteria, as presented to them:

Question 1: Instruction Faithfulness

Which edited object most closely follows the editing instruction?

This includes:

- How well the edited object reflects the edit prompt (e.g., if the prompt is "a teddy bear with a red hat", does the teddy actually have a red hat?).
- How well are the unreferenced regions preserved (e.g., is the teddy bear's body, color, or face still intact and consistent with the original?).

Question 2: Visual Quality and Shape

Which edited object has the best overall visual quality, including texture and geometry detail, and realistic/natural 3D geometry?

This includes:

- How realistic the textures look (e.g., do the materials, colors, or patterns look natural, or do they seem blurry?).
- How clean and detailed the shape is (e.g., are parts of the object sharp and well-formed?).
- Whether the object geometry is consistent from both views (e.g., does the back view match the front?).

We note that Edit360 [4] was not included in the user study due to its code being released near the submission deadline, preventing a fair evaluation with our planned participant size.

3.4. Evaluation Dataset

We constructed our evaluation dataset to cover a diverse and representative range of high-quality 3D objects, spanning categories such as buildings, animals, and furniture. The edits include both local and global modifications, targeting challenging changes in shape, appearance, or both. We will release the dataset publicly. The size of our evaluation set

enables us to assess our method on diverse and challenging assets, while remaining sufficiently compact to support a large-scale user study, the use of GPT-based metrics and comparisons with key approaches, leveraging score distillation sampling. We compare the size of our evaluation dataset to those used in prior works. Our goal was to increase both the number of object categories and the diversity of edits. Table 1 summarizes the number of unique objects and total edits used in each method.

Table 1. **Comparison of evaluation dataset size across recent 3D editing methods.** '-' indicates that the corresponding number is not reported.

Method	# Objects	# Edits
PrEditor3D [3]	18	40
Shap-Editor [1]	15	20
Vox-E [12]	8	18
GaussCtrl [15]	6	-
CMD [7]	-	50
Ours	**25**	**100**

Figure 6. **Effect of scale parameter s_l in low frequency enhancement.**

3.5. Our 3D editing setting

Our method supports both local modifications (e.g., replacing a part, altering its texture/ geometry, or adding new components) and global edits that affect overall appearance or semantics (e.g., changing the scene's style, making it look like winter, or turning a shoe into a "clown shoe"). To achieve localized edits, we illustrate our approach in the next section. We note that our text-based 3D editing pipeline is suitable for complex modifications that span multiple viewpoints and cannot be adequately expressed from a single edited rendered view.

4. Mask Generation

As described in the main paper, to obtain the masks we query GPT-4o using a rendered front view of the original object along with the original and edited prompts, following the prompt format presented below.

GPT-4o Prompt

You are given:

– An input image.
– A caption describing the image: "a blue warrior holding a shovel"
– An edit instruction: "a blue warrior holding a flower"

Task: **Editable regions**
Return a comma-separated list of all regions in the original image that must be visually changed to accomplish the edit. Each region must refer to a coherent, visible part of the original image. Do not list subcomponents that are not visually distinct. Do not include elements that only appear in the edited caption but are absent from the original image.

Task: **Changed into**
Return the name of the new object, element, or category the above region is changing into, based on the edit instruction. Do not include visual attributes like color, size, or material — just the object or element name.

This returns the name of the changing region in the original asset (e.g., the *shirt* in the example that transforms "a teddy bear with a shirt" into "a teddy bear with a tuxedo" or the *shovel* in the example that transforms "a blue warrior holding a shovel" into "a blue warrior holding a flower"). We then localize this region in the original views using GroundingDINO [9] to obtain bounding boxes. The corresponding regions are segmented and tracked across views using SAM2 [10], resulting in masks for the original object M_o. In our experiments we dilate the masks by a small factor. This process is applied across 24 views rendered around the object. To constrain the editing to the corresponding 3D region during the diffusion process, we select the masks corresponding to the $V_{\text{in}} = 4$ input views used to create the Gaussian splat grid. These masks are used to assign labels to the corresponding 3D Gaussians, thus approximating the 3D edit region in the original asset.

For the 3D enhancement, we also introduce a mechanism to constrain corrections to the editing region. As mentioned in Section 3.2, we render 24 views from the edited 3DGS to achieve full 360° coverage, and use these for the enhancement stage. We use the corresponding original/unedited views from the same camera poses. Using GPT-4o with the prompt described above, we identify the changing region

or concept between the original and edited images (e.g., the "shovel" in the original and the "flower" in the edited version of the blue warrior example). We localize this region in the edited views using GroundingDINO [9] to obtain bounding boxes. These bounding boxes are then used to segment and track the changing element across views using SAM2 [10], resulting in view-consistent masks M_e for the edited images. We define the union of the masks, $M = M_o \cup M_e$, where M_o were obtained in the editing step as the final masks representing the changing region. These masks M are used in the 3D enhancement module to constrain updates to the edited region, while supervising the unedited areas using the rendered views of the original object I_{src}.

5. Runtime cost

Our method achieves fast 3D edited asset generation, taking 25 seconds on average for 3D editing without the 3D enhancement step. Our results from this stage already exhibit clean 3D geometry and texture quality. The 3D enhancement step, which further enhances visual details, takes an additional 6 minutes.

In addition, we compare the runtime cost of our work with SDS-based and iterative-based methods. Specifically, we include Vox-E [12], Posterior Distillation Sampling (PDS) [5], as well as GaussianEditor[2] and InstructGS2GS [14]. Compared to related works, our method has the lowest runtime, while at the same time achieving superior editing results.

Method	**Runtime (↓)**
3D-LATTE (Ours, w/ enhancement)	6 min 25 sec
3D-LATTE (Ours, w/o enhancement)	25 sec
Vox-E	62 min
PDS	4 hours
GaussianEditor	14 min
InstructGS2GS	12 min

Table 2. Runtime comparison across methods.

6. Further ablations on 3D attention injection

Intuitively, 3D attention injection enables the model to respect the 3D structure and layout of the original scene during editing as well as the parts not referenced by the edit. Without attention injection, edits often result in unintended semantic shifts. For example, a house intended for texture modification may instead change shape, or unrelated regions may be altered. To quantitatively evaluate this effect, we report our shape preservation metric, CLIP Diff-No-Edit , both with and without attention injection. We observe that attention injection consistently improves structural preservation. To further assess the module's impact on the similarity between the original and edited objects, we additionally compute the LPIPS perceptual distance [18] (VGG) between rendered views of the original and edited shapes. We find that attention injection substantially reduces LPIPS, further supporting our intuition. The results are summarized in Table 3.

Attention Injection	**CLIP Diff-No-Edit (↓)**	**LPIPS (↑)**
w/ Injection	**0.039**	**0.14**
w/o Injection	0.061	0.11

Table 3. Effect of 3D attention injection on structural preservation and perceptual similarity.

Steps	**CLIP Directional Similarity (↑)**	**CLIP Diff-No-Edit (↓)**	**LPIPS (VGG)**
None	0.19	0.052	0.12
10%	0.18	0.049	0.12
30%	0.18	**0.039**	0.14
60%	0.15	**0.037**	**0.17**

Table 4. Effect of different steps of self-attention injection.

We also provide ablations on 3D self-attention injection. Self-attention captures 3D layout and structural information and spatial correspondences between Gaussians. We find that injecting self-attention at later denoising steps overly constrains the generation, limiting its ability to deviate from the original structure and thus compromising edit prompt alignment. At the same time, omitting it fully compromises source preservation in some cases. Thus, we experimentally identify an optimal timestep for balancing edit alignment and structure preservation. In Table 4 we show the influence of different injection steps to the CLIP similarity score, CLIP Diff-No-Edit and LPIPS (VGG).

We see that carefully injecting self-attention complements cross-attention by further reinforcing structural consistency during editing, helping preserve the global layout of the source 3D asset.

7. Generalization across 3D representations

Our core idea is not inherently tied to 3D gaussian splat grids and can be adapted to other 3D backbones and representations. We focus our discussion here on the recent and widely-used Structured LATent (SLAT) representation introduced in TRELLIS[16]. For 3D inversion, inversion for rectified-flow models [11] can be applied at both the structure and the splat latent stage. This inversion technique is also used in the concurrent work [6]. The 3D attention mechanism in [16] can capture the correspondences between text tokens and 3D voxel latents as well as among the 3D voxel latents themselves. Thus, in the structure stage, when coarse geometry is generated, 3D attention injection

can be applied in a manner analogous to our approach.

8. Limitations

Since we rely on DiffSplat [8] as a 3D diffusion backbone which is not fine-tuned on scenes our method can only operate on object-centric 3D assets, rather than full scenes with backgrounds. In addition, we observe that large structural changes that alter pose (e.g., a teddy bear lifting its arms or sitting) are challenging and may sometimes fail. This could be attributed to the text-conditioning signal as current text-conditioned diffusion models are mostly trained on datasets that lack the scale and pose diversity of 2D data.

Moreover, our approach inherits certain limitations from DiffSplat, including low supervision resolution in 3D Gaussian generation, which can lead to minor artifacts in certain edits and limit edit quality on humans and human faces.

9. Broader Impact

Our work contributes to the growing field of 3D generative modeling and editing, by providing artists, designers and researchers with the tools to perform flexible and controllable 3D editing. We acknowledge that, like other generative tools, our method could be misused to create misleading or inappropriate content and might reproduce biases present in large diffusion models.

References

[1] Minghao Chen, Junyu Xie, Iro Laina, and Andrea Vedaldi. Shap-editor: Instruction-guided latent 3d editing in seconds. In *CVPR*, 2024. 7

[2] 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 *CVPR*, 2024. 8

[3] Ziya Erkoç, Can Gümeli, Chaoyang Wang, Matthias Nießner, Angela Dai, Peter Wonka, Hsin-Ying Lee, and Peiye Zhuang. Preditor3d: Fast and precise 3d shape editing. In *CVPR*, 2024. 1, 7

[4] Junchao Huang, Xinting Hu, Shaoshuai Shi, Zhuotao Tian, and Li Jiang. Edit360: 2d image edits to 3d assets from any angle. In *ICCV*, 2025. 6

[5] Juil Koo, Chanho Park, and Minhyuk Sung. Posterior distillation sampling. In *CVPR*, 2024. 8

[6] Lin Li, Zehuan Huang, Haoran Feng, Gengxiong Zhuang, Rui Chen, Chunchao Guo, and Lu Sheng. Voxhammer: Training-free precise and coherent 3d editing in native 3d space. *arXiv preprint arXiv:2508.19247*, 2025. 8

[7] Peng Li, Suizhi Ma, Jialiang Chen, Yuan Liu, Chongyi Zhang, Wei Xue, Wenhan Luo, Alla Sheffer, Wenping Wang, and Yike Guo. Cmd: Controllable multiview diffusion for 3d editing and progressive generation. *arXiv preprint arXiv:2505.07003*, 2025. 7

[8] Chenguo Lin, Panwang Pan, Bangbang Yang, Zeming Li, and Yadong Mu. Diffsplat: Repurposing image diffusion models for scalable 3d gaussian splat generation. In *ICLR*, 2025. 2, 9

[9] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *ECCV*, 2024. 7, 8

[10] 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 Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv:2408.00714*, 2024. 7, 8

[11] Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Semantic image inversion and editing using rectified stochastic differential equations. In *ICLR*, 2025. 8

[12] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *ICCV*, 2023. 7, 8

[13] Chenyang Si, Ziqi Huang, Yuming Jiang, and Ziwei Liu. Freeu: Free lunch in diffusion u-net. In *CVPR*, 2024. 6

[14] Cyrus Vachha and Ayaan Haque. Instruct-gs2gs: Editing 3d gaussian splats with instructions, 2024. 8

[15] Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Prisacariu. GaussCtrl: Multi-View Consistent Text-Driven 3D Gaussian Splatting Editing. In *ECCV*, 2024. 7

[16] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *CVPR*, 2025. 8

[17] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. 2025. 4

[18] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 8