# EditVerse3D: High-Quality 3D Object Editing with Region-Aware Learning

Youtan Yin[1,2], Yanning Zhou[2], Jiacheng Wei[1], Xiaofeng Yang[1], Jun Zhang[2], Jiayang Bai[2], Jingwen Ye[2], Weidong Zhang[2], and Guosheng Lin[1]

[1] College of Computing and Data Science, Nanyang Technological University
{youtan001, xiaofeng001}@e.ntu.edu.sg, weijiacheng.gaw@gmail.com, gslin@ntu.edu.sg
[2] Tencent AIPD
ynzhou@cse.cuhk.edu.hk, {junejzhang, darcybai, jingwenye, wadewdzhang}@tencent.com

**Abstract.** Local editing of 3D objects remains a long-standing challenge. When interacting with 3D content, humans naturally tend to specify a coarse region of interest for modification rather than defining precise editing boundaries. However, previous methods rely on fully edited 2D images, precise 3D masks, or redundant pipelines, which present a gap. To bridge this gap, we propose EditVerse3D, a novel 3D editing framework that enables high-quality object editing under such coarse guidance. Our approach takes as input a 3D object to be edited, a coarse 3D bounding box indicating the target region, and a reference 2D image describing the desired modification. It produces a coherent, high-fidelity edited 3D object. To facilitate this editing, we introduce a novel region-aware adaptive loss that emphasizes hard-to-learn regions and balances the objective between target and preserved areas. Complementing our loss function, we enhance model robustness and generalization through targeted data augmentations, such as training with scaled 3D masks and filtering out unrealistic editing pairs. We construct a large-scale 3D editing dataset derived from parts information. Extensive experiments demonstrate that EditVerse3D achieves superior visual quality and quantitative performance compared to existing 3D editing approaches. Please visit our project page at https://editverse3d.github.io.

**Keywords:** 3D Editing · Region-Aware Learning · 3D Editing Dataset

## 1 Introduction

Local editing of 3D objects has been challenging due to the inherent complexity of the 3D context compared to 2D. However, the broad applicability of 3D editing has attracted significant attention from the research community [20,25,26,30,65,68,85,90,101]. Existing approaches can generally be divided into three categories.

The first approach [2,5,18,22,34,43,63,88,97] involves rendering multiple 2D views of the input 3D object, editing these views, and subsequently lifting the edits back into 3D. However, this method faces two significant challenges. First,

**Fig. 1: Editing results of our method.** Given a 3D object, a user-specified coarse 3D bounding box indicating the target editing region, and an image prompt defining the editing goal, our approach generates high-quality, coherent edits. Our method does not require fully edited 2D views, precise 3D masks, or redundant pipelines.

the back-and-forth between 3D and 2D introduces cumulative errors, resulting in a decline in editing quality. Second, editing in the 2D context ignores 3D structure, often leading to inconsistencies across multiple views. Moreover, although large vision models (LVMs) [24, 36, 47, 66, 82] have achieved remarkable results, precise 2D edits [15, 29, 32, 57, 64, 67, 103, 106] remain a considerable challenge.

The second approach [7, 9, 16, 17, 39, 48, 49, 58, 69, 108] is based on the Score Distillation Sampling (SDS) [62] scheme and its variants, which use 2D models to guide the 3D reconstruction process toward the edited 3D representation. However, these methods are computationally intensive, time-consuming, struggle with network optimization, and often fail to produce high-quality results.

Recently, the development of 3D generative models [41,44,74–77,83,86,94,99] has shown great promise, achieving impressive results in 3D generation tasks. Some methods have explored training-free 3D editing [42,61,96] based on 3D generative models, which modify the inference process to output an edited 3D object conditioned on editing instructions. These methods typically adapt training-free 2D editing techniques [35, 54, 78] developed for 2D generative models to the 3D models. However, due to the increased complexity of 3D objects, their generalization ability is limited, and they often work only in specific cases.

Leveraging advances in 3D generative models, we propose an end-to-end 3D editing framework that overcomes the limitations of existing methods that rely on complex, multi-stage pipelines. Our approach eliminates the need for additional inputs such as pre-edited 2D views, 2D masks, or precise 3D masks, while maintaining the efficiency of 3D generative models to produce high-quality results within minutes. The framework takes three key inputs: **(1)** the 3D object

to be edited, **(2)** a coarse 3D bounding box specifying the target region, and **(3)** a 2D image defining the editing goal. Using these inputs, the model directly generates the edited 3D object in a streamlined manner.

Specifically, we build our 3D editing model based on the 3D generative backbone, TRELLIS [86]. We carefully design the model architecture (Sec. 3.2) and training strategy (Sec. 3.3) to preserve the original model's generative capabilities while adapting it to support our input format. To train our model for high-quality edits under relatively loose inputs, we introduce a loss strategy that adaptively forces the model to focus more on the poorly learned region. Moreover, data augmentation techniques are applied to improve the model's generalization.

Due to the unavailability of large-scale 3D editing datasets tailored to our setting, we construct a comprehensive dataset of 3D editing pairs using 3D segmentation information (Sec. 3.4). The construction process follows a straightforward approach: we remove a part from a 3D object and treat its restoration as an "add" editing operation. Specifically, the 3D object after removal serves as the input, while the original 3D object acts as the ground truth. The removed part is used as the 3D mask, which is rendered to generate the corresponding 2D image prompt. Although this dataset primarily consists of "add" edits, our experiments demonstrate that models trained on it can generalize effectively to "replace" edits. Conceptually, "add" can be regarded as a special case of "replace," where the target region lacks any existing 3D structure.

To address the scarcity of 3D segmentation datasets, we further expand our dataset by incorporating data from existing 3D object repositories, such as Objaverse [13, 14], which contain objects naturally composed of multiple parts. As a result, we curate an extensive 3D editing dataset comprising approximately 85k meshes and 500k editing pairs.

In summary, our method has the following contributions:

- An end-to-end 3D editing framework with adaptive loss reweighting and data augmentation, enabling robust generalization and practical usability under coarse input settings.
- A large-scale 3D editing dataset that addresses the long-standing challenge of lacking supervised training data for 3D editing.
- A novel 3D editing pipeline that produces high-quality results, outperforming previous methods in both visual quality and quantitative assessment.

## 2 Related Works

### 2.1 3D Generation

With the rapid advances in diffusion [66] and flow-matching models [47] for language and 2D vision tasks, researchers have increasingly adapted these architectures for 3D generation [3, 10, 11, 27, 37, 38, 40, 41, 44, 45, 51–53, 71, 79, 81, 83, 84, 87, 94, 95, 99, 105], achieving remarkable results. The development of large-scale 3D generative models has established a strong foundation for downstream applications, accompanied by the emergence of extensive 3D object datasets [13, 14,

86, 102]. Furthermore, many studies have focused on 3D segmentation–related tasks, such as jointly generating 3D content and its corresponding semantic segmentation [6, 46, 70, 73, 93], or directly segmenting 3D assets [4, 8, 21, 23, 50, 56, 72, 89, 91, 92, 104, 107]. These tasks typically rely on 3D segmentation datasets [46,55,70,92], further bridging the gap toward 3D editing. Consequently, with the progress in 3D generative modeling and the availability of large-scale 3D data, supervised training of 3D editing models has become increasingly feasible.

### 2.2 3D Editing

3D editing has long been an actively explored research topic, yet there remains substantial room for improvement. In the absence of large-scale 3D generative models, earlier approaches [1, 2, 5, 18, 22, 34, 43, 63, 88, 97] typically render several 2D views, edit the rendered images, and then lift the results back to 3D. Naturally, this 3D–2D–3D process accumulates errors and often leads to poor 3D consistency. Other methods [7, 9, 16, 17, 39, 48, 49, 58, 69, 108] adopted score distillation sampling (SDS) [62] to optimize 3D representations under the guidance of 2D priors, but such approaches are usually hard to train and yield limited visual quality. With the emergence of 3D generative models, recent work has begun exploring training-free 3D editing [42, 61, 96]. However, these methods often apply 2D training-free editing techniques [35, 54, 78] to 3D models, still relying on edited 2D views and suffering from the same 3D–2D–3D inconsistency issues. Existing 3D editing approaches also typically require precise 3D masks or well-edited 2D views, which hinder practical usage.

## 3 Method

We detail our approach in four stages. First, we describe the selected generative backbone, TRELLIS [86], in Sec. 3.1. Next, we explain how our network architecture supports coarse inputs in Sec. 3.2. Third, we introduce the training strategy and data augmentation techniques applied to improve the model's performance in Sec. 3.3. Finally, we describe the construction of our large-scale editing dataset in Sec. 3.4. Fig. 2 shows an overview of our method.

### 3.1 Preliminary

Our 3D editing model is based on TRELLIS, a 3D generation framework. TRELLIS generates high-quality 3D objects conditioned on either images or text. It uses an encoder-decoder architecture, where the encoder compresses 3D assets into latent representations, and the decoder reconstructs them into various 3D formats, such as 3DGS [33, 98], NeRF [59], and Mesh. To enable 3D object generation conditioned on image or text prompts, rectified flow models [47] are employed. The ground truth for these rectified flow models is the latent representation obtained from encoding the 3D object. During inference, the rectified

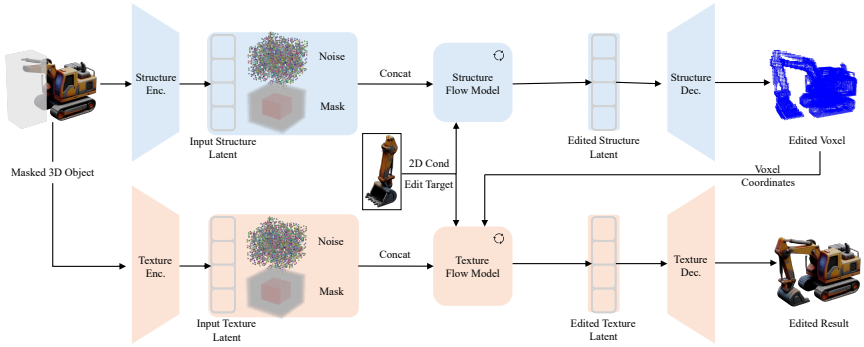


**Fig. 2: An overview of our method.** Given a masked 3D object as input, we first extract its structure and texture latents using the TRELLIS encoder. The input latents are concatenated with a binary mask and random noise along the feature channel dimension, then fed into the flow-matching model, which takes the editing target as a condition. The flow model generates edited latents, which decode into the final result.

flow model samples the latent from a noise distribution conditioned on the image or text prompt, and the decoder decodes it into the desired 3D format.

Specifically, TRELLIS handles structure and texture separately. For a given 3D object, TRELLIS first normalizes its vertices into a unit cube within the range of $[-0.5, +0.5]$, then voxelizes it into a $64^3$ voxel grid $p$ ($N \times 3$, $N$ is the number of active voxels) for the structure encoder input. The structure encoder transforms $p$ to a structure latent $L_s$ ($16 \times 16 \times 16 \times C_s$, $C_s$ is the feature channel), which is then decoded back to the voxel form $p$ by the decoder.

For texture, TRELLIS renders 150 2D views and extracts feature maps using the DINOv2 [12,31,60] model. The active voxel coordinates $p$ are projected onto the multi-view feature maps to retrieve features at the corresponding locations. The average of these 2D features $\{z, p\}$ is used as input to the texture encoder, where $z$ is the gathered DINOv2 feature ($N \times C_d$, $C_d$ is the feature channel of DINOv2 output). The texture encoder encodes $\{z, p\}$ to a texture latent $L_t$ ($N \times C_t$, $C_t$ is the feature channel). The texture decoder then decodes $\{L_t, p\}$ into the desired 3D output.

Rectified flow models are introduced to integrate image or text conditions. The flow model follows a forward-backward process similar to that of diffusion models [66]. In the forward process, noise $\epsilon$ is added to the data sample $x_0$ with $x_t = (1-t)x_0 + t\epsilon$ at each timestep $t$. The backward process is defined as:

$$\frac{dx_t}{dt} = v(x_t, t), \tag{1}$$

which moves noisy samples toward the data distribution. We learn $v$ using a neural network $\theta$ that minimizes the difference between the predicted $v_\theta(x_t, t)$ and ground-truth vector fields, computed from the forward process:

$$\mathcal{L}(\theta) = \mathbb{E}_{t, x_0, \epsilon} ||v_\theta(x_t, t) - (\epsilon - x_0)||_2^2. \tag{2}$$

Flow models for structure and texture are trained separately, with the learning targets being the corresponding structure latent $L_s$ and texture latent $L_t$.

Since TRELLIS's encoder-decoder network is highly effective at compressing and restoring 3D objects with nearly lossless perceptual quality, we directly leverage these models and train the rectified flow models for 3D editing.

### 3.2 Network Architecture

We encode 3D objects into a latent representation using TRELLIS's encoder, which produces both structure $L_s^{in}$ and texture $L_t^{in}$ latents.

**Structure.** For the structure, we concatenate the encoded latent $L_s^{in}$ ($16\times16\times16\times C_s$), the boolean mask $\mathcal{M}_s$ ($16\times16\times16\times1$), and noise $\epsilon_s$ ($16\times16\times16\times C_s$) along the feature dimension as the model input $L_s^{'}$ ($16\times16\times16\times[2C_s+1]$). The output is the edited structure latent $L_s^{out}$ ($16\times16\times16\times C_s$). We decode $L_s^{out}$ to edited geometry $p^{out}$ ($N^{out}\times3$) with TRELLIS's decoder.

**Texture.** For the texture, we first combine the input 3D object's texture latent $L_t^{in}$ ($N^{in}\times C_t$) and the coordinates of the edited voxels $p^{out}$ ($N^{out}\times3$) into $\{L_t^{in}, p^{out}\}$. Note that the input shape to be edited $N^{in}$ and the predicted output shape $N^{out}$ are misaligned. For coordinates that are not covered by the input, we pad the corresponding features with zeros. We then concatenate $\{L_t^{in}, p^{out}\}$, the boolean mask $\mathcal{M}_t$ ($N^{out}\times1$), and the noise $\epsilon_t$ ($N^{out}\times C_t$) along the feature dimension, resulting in the edited texture latent $\{L_t^{out}, p^{out}\}$ ($N^{out}\times[2C_t+1]$).

For the condition, we use images with partial elements that indicate the editing target, rather than the complete edited 2D view, for both the structure and texture models. Given an image prompt, we add the encoded latent from the input 3D object and the 3D coarse bounding box, then sample the edited structure and texture latents from noise.

### 3.3 Training Strategy

**Region-Aware Loss Reweighting.** Unlike optimizing a generative model, the editing task inherently involves an imbalance between regions that need editing and those that do not. Areas that do not require editing tend to converge more easily, while the target editing regions incur higher losses that are more difficult to reduce. The loss function for the generation task is simply an MSE loss between the predicted velocity and the ground truth

$$\mathcal{L}_{\text{gen}} = \frac{1}{n}\sum_{i=1}^{n} \left\| v\big(x_t^{(i)}, t\big) - v_\theta\big(x_t^{(i)}, t\big) \right\|_2^2. \tag{3}$$

Here, $n$ is the number of elements for the flow model's vector field, and $v_\theta$ and $v$ are the predicted vector field and ground truth, respectively. For simplicity, we abbreviate $v\big(x_t^{(i)}, t\big) - v_\theta\big(x_t^{(i)}, t\big)$ as $v - v_\theta$.

We apply a loss reweighting strategy by normalizing the loss so that both regions (masked and non-masked) receive proportional attention. We scale the loss by the number of elements in each area, ensuring that both masked and

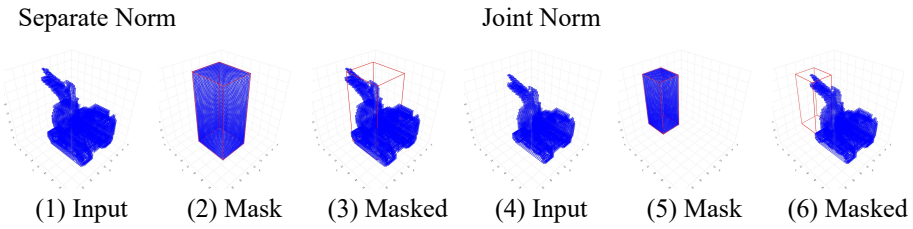


**Fig. 3: Effect of joint normalization.** Separate normalization leads to overlapping (3), whereas our joint normalization preserves the correct spatial relationship (6).

non-masked regions contribute equally to the overall loss, even if one region is significantly larger than the other. Suppose $\mathcal{M}$ is the boolean mask, and $m$ and $\bar{m}$ are the number of elements in the masked and nonmasked region, respectively. The loss terms are

$$\mathcal{L}_m = \frac{1}{m}\sum_{i=1}^{m} \| v - v_\theta \|_2^2 \, \mathcal{M}_i, \quad \mathcal{L}_{\bar{m}} = \frac{1}{\bar{m}}\sum_{i=1}^{\bar{m}} \| v - v_\theta \|_2^2 \, (1 - \mathcal{M}_i). \tag{4}$$

Through this loss decomposition, we obtain more balanced loss terms for both the unchanged region and the target editing area.

Moreover, we adopt a hard-example mining strategy to emphasize regions that are more difficult to learn. Specifically, we select the hardest regions corresponding to the top $\tau\%$ of per-index losses:

$$\mathcal{L}_{\text{hard}} = \frac{1}{|\mathcal{H}|}\sum_{i\in\mathcal{H}} \| v - v_\theta \|_2^2, \quad \mathcal{H} = \text{Top-}\tau\%\Big( \| v - v_\theta \|_2^2 \Big), \tag{5}$$

where $\mathcal{H}$ denotes the set of hard examples whose per-index losses rank within the highest $\tau\%$ of all losses.

To balance each loss term's value, the overall loss is formulated as:

$$\mathcal{L}_{\text{edit}} = \mathcal{L}_m + \frac{|\mathcal{L}_m|}{|\mathcal{L}_{\bar{m}}|}\mathcal{L}_{\bar{m}} + \frac{|\mathcal{L}_m|}{|\mathcal{L}_{\text{hard}}|}\mathcal{L}_{\text{hard}}. \tag{6}$$

**Joint Normalization for Spatial Alignment.** A potential issue with the input data is normalization. If the input 3D object and the 3D mask are independently normalized to a unit cube, this can distort their relative positions. We observe that if the 3D object and the mask are not spatially aligned, the model struggles to learn the mapping between the input and the edited output. To address this, we treat the 3D object and the mask as a whole and compute a single normalization factor for both. This normalization factor is then applied uniformly across all relevant content. During inference, we use a similar normalization process to ensure consistency in spatial alignment. Fig. 3 shows the necessity of our joint normalization.

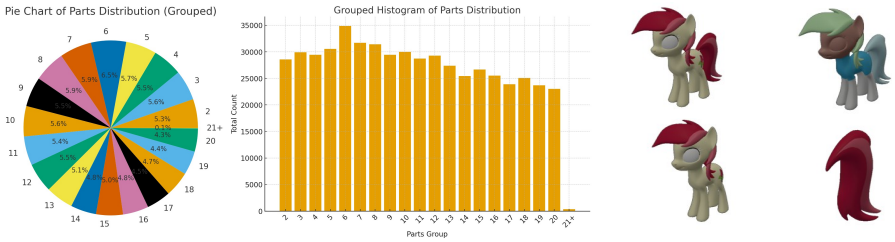


**Fig. 4: Overview of our 3D editing dataset.** Left: *Parts Distribution* across the dataset. Right: *An Example Editing Pair* illustrating the ground truth (top left), segmentation (top right), source object (bottom left), and editing target (bottom right).

**Training with Coarse 3D Masks.** In practical scenarios, users prefer to provide a coarse 3D bounding box that is larger than the actual region to be edited rather than defining precise 3D editing masks. To ensure the model can work with the arbitrary coarse 3D mask during inference, we synchronize its use during training. Experiments show that models trained with exact 3D shapes as masks converge well during training but perform poorly when tested with a coarse bounding box. Furthermore, compared to using the minimum bounding box of the accurate 3D mask, applying random disturbances to the box size and position further improves model performance.

**Filtering Unrealistic Editing Pairs.** We filter out data pairs in which the target editing region is too small based on the volume occupied by the 3D voxels. This operation aims to ensure that the training data primarily consists of cases likely to occur in real editing scenarios. Target regions with tiny volumes often correspond to editing pairs that lack practical significance. The effectiveness of this data filtering strategy is further validated in our experimental results. Please refer to the supplementary materials for more details.

### 3.4 Dataset Curation

To train our model end-to-end, we create a large-scale dataset of 3D editing pairs. The basic idea is that, for a given 3D object, we can treat its partially deleted version as the object to be edited, with the original 3D object serving as the ground truth. These two objects form an editing pair. However, directly deleting parts of the 3D object can result in missing information at cut or fracture regions, hindering model performance. To address this, we collect 3D objects with complete part information whose distinct parts still preserve a whole structure and are suitable for constructing editing pairs.

We organize our dataset using manually validated 3D segmentation data from the Partverse dataset [70] and 3D assets with part information from the Objaverse dataset [13, 14]. We clean the data by removing low-quality instances (e.g., geometric distortions or missing textures). In the end, we construct a

dataset comprising approximately 85k meshes and 500k editing pairs. We compute bounding boxes and generate image prompts from the deleted 3D parts. Fig. 4 shows an overview of our dataset.

## 4 Experiments

### 4.1 Implementation

**Train.** For each editing pair in our dataset, we use the TRELLIS encoder to extract the structure and texture latents of the input 3D object, which serve as inputs to the flow-matching model. We compute the minimum 3D bounding box of the target 3D part and apply our box augmentation strategy to formulate the 3D mask for training. Subsequently, we render 24 views of the 3D part from cameras uniformly sampled across a bounding sphere. At each training step, one of these views is randomly selected to act as the condition. The flow-matching models for structure and texture are trained independently, each for 10k steps with a batch size of 1 on 96 V100 (32GB) GPUs. The complete two-stage training process requires approximately 10k GPU hours.

**Evaluation.** We use Chamfer Distance (CD) [19] to measure geometry quality and render 32 views to assess texture quality using PSNR, SSIM [80], LPIPS [100], DINO feature similarity [60], and FID [28]. We build a test set of about 200 meshes and 1500 3D editing pairs from the PartObjaverse-Tiny [92] dataset, following our data generation pipeline, serving as the benchmark for adding. Additionally, we use the VoxHammer [42] dataset, which contains 100 meshes and 300 editing pairs, as an evaluation set for replacement. Since this dataset lacks ground-truth edits, we compute quantitative metrics only on non-edited regions. To simulate real-world scenarios in which users may not provide accurate bounding boxes, we use bounding boxes of varying sizes and positions during evaluation, rather than those used in training. Note that during training, each editing pair is associated with 24 condition views. During evaluation, to assess the robustness of our model against image prompts from varying viewing angles, we report the average performance across all 24 views.

### 4.2 Main Results

We compare our method to Instant3dit [2], TRELLIS [86] (two re-implemented local editing variants), and VoxHammer [42]. Instant3dit renders four canonical views, applies text-driven edits on those renderings, and reconstructs a 3D result from the four edited views. For TRELLIS, we re-implement two local editing strategies: **(1)** Repaint [54] — which mixes the model-predicted target region with the preserved region (the input injected with noise) at each inference timestep, and **(2)** FlowEdit [35] — which directly drives the input toward the target prompt by computing a new vector field guided by both the source and target prompts. VoxHammer adapts RF-Edit [78] to TRELLIS, achieving more accurate inversion and applying Repaint not only to the latent representation but also to intermediate-layer features.

**Table 1: Quantitative comparison of different edits.** *Repaint* and *FlowEdit* denote two 2D editing approaches applied to TRELLIS. For *Replace*, since ground truth is unavailable for edited regions, the results shown correspond only to the preserved regions. For *Add*, we report metrics across the entire edited results.

| Method | Replace | | | | | | Add | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CD↓ | PSNR↑ | SSIM↑ | LPIPS↓ | DINO↑ | FID↓ | CD↓ | PSNR↑ | SSIM↑ | LPIPS↓ | DINO↑ | FID↓ |
| Instant3dit | 0.297 | 13.06 | 0.868 | 0.255 | 0.732 | 73.86 | 0.110 | 8.075 | 0.664 | 0.501 | 0.681 | 81.78 |
| Repaint | 0.007 | 35.96 | 0.992 | 0.010 | 0.980 | 29.32 | 0.008 | 27.44 | 0.960 | 0.038 | 0.978 | 3.408 |
| FlowEdit | 0.071 | 17.93 | 0.917 | 0.109 | 0.902 | 40.76 | 0.017 | 19.46 | 0.886 | 0.102 | 0.945 | 6.129 |
| VoxHammer | 0.022 | 24.51 | 0.961 | 0.043 | 0.955 | 36.28 | 0.023 | 21.16 | 0.923 | 0.095 | 0.919 | 8.281 |
| Ours | **0.005** | **36.32** | **0.995** | **0.008** | **0.981** | **28.34** | **0.005** | **28.67** | **0.962** | **0.029** | **0.984** | **2.960** |

**Table 2: Quantitative comparison for different regions**. Since some texture metrics (e.g., LPIPS, DINO feature similarity) do not support irregularly shaped images, the *Target Editing Region* is computed with the minimal 2D bounding box. The *Unedited Region* is evaluated by treating the edited region as consistent with the ground truth. The above estimation yields a higher metric for actual performance but does not compromise consistent comparison. Because the *Target Editing Region*'s sizes vary, FID is not reported because it does not support inputs of different sizes.

| Method | Target Editing Region | | | | | Unedited Region | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | CD↓ | PSNR↑ | SSIM↑ | LPIPS↓ | DINO↑ | CD↓ | PSNR↑ | SSIM↑ | LPIPS↓ | DINO↑ |
| Instant3dit | 0.052 | 3.698 | 0.119 | 0.194 | 0.406 | 0.105 | 8.529 | 0.746 | 0.425 | 0.660 |
| Repaint | 0.019 | 19.00 | 0.726 | 0.032 | 0.862 | 0.006 | 31.16 | 0.945 | 0.012 | 0.983 |
| FlowEdit | 0.020 | 14.31 | 0.512 | 0.055 | 0.783 | 0.022 | 23.01 | 0.931 | 0.057 | 0.952 |
| VoxHammer | 0.039 | 14.26 | 0.499 | 0.087 | 0.696 | 0.019 | 25.86 | 0.964 | 0.044 | 0.948 |
| Ours | **0.009** | **21.01** | **0.811** | **0.030** | **0.899** | **0.002** | **35.45** | **0.987** | **0.008** | **0.990** |

**Quantitative.** Our method outperforms existing approaches on both replacement and addition tasks (Tab. 1), in terms of both edit-region fidelity and preservation of unedited regions (Tab. 2). **(1)** Instant3dit: Its 2D editing capability is limited, and its 2D-to-3D process relies solely on reconstruction from four sparse canonical views rather than dense views, and lacks mechanisms to resolve 3D inconsistencies. These factors contribute to a significant performance degradation, resulting in the worst quantitative results among all evaluated baselines. **(2)** Repaint & FlowEdit: Repaint significantly outperforms FlowEdit. This superiority stems from Repaint's mask-based fusion, which leverages the unedited region to constrain the generated content within the target editing area. In contrast, FlowEdit predicts vector fields corresponding to the source and target prompts to compute the editing direction. This guidance mechanism is relatively weak. **(3)** VoxHammer: By utilizing intermediate network features, it should achieve better performance than the direct application of Repaint. However, based on the authors' implementation, we observed slightly inferior results. This discrepancy may arise from differences in our specific evaluation configurations.

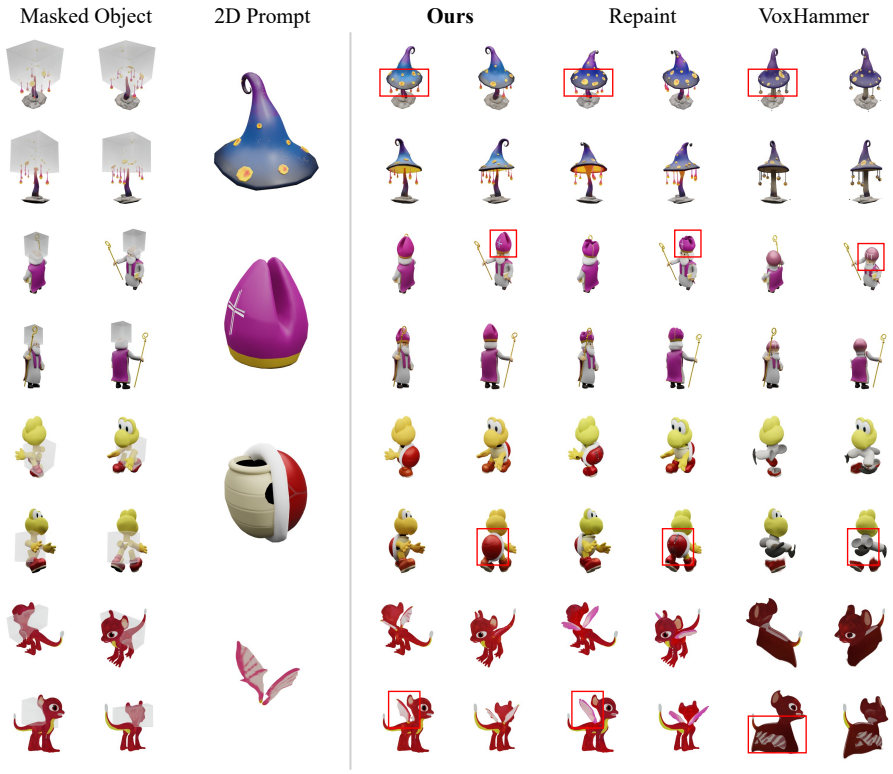


**Fig. 5: Qualitative comparison.** Our method yields prompt-aligned edits while preserving unedited regions. *Repaint* aligns structures in rows (1, 4) but misaligns textures in (2, 3). *VoxHammer* shows partial consistency in (1, 2) but fails in (3, 4).

**Qualitative.** We visualize the editing results produced by our method and existing approaches in Fig. 5. As shown, our method delivers clearly superior visual quality. Additional results are presented in Fig. 6, demonstrating the strong generalization ability and robustness of our approach. Across diverse test cases, our method consistently produces realistic and high-quality 3D editing results.

### 4.3 Ablation Study

To validate our method, we perform a detailed ablation study on the PartObjaverse Tiny test set. Variant without *Joint Normalization* fails to converge and is thus omitted. Beyond the components discussed in Sec. 3.3, we also examine the impact of data *Segmentation Quality* and training conditions' *View Variance*. Tab. 3 shows that our training strategy consistently improves performance, with Exp.#5 achieving the best results. Notably, segmentation quality has a limited impact at the same data scale (Exp.#5 vs Exp.#6), indicating that 3D assets

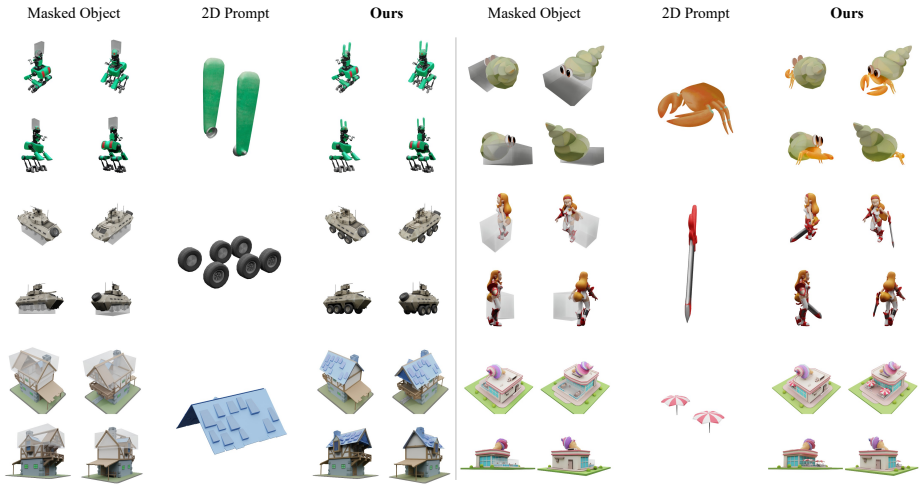


**Fig. 6: Qualitative results.** See additional examples in the supplementary material.

**Table 3: Ablation study of our method.** The *Mask* column indicates the scale of the 3D mask used during training: *Exact* uses the precise shape, *BBox* uses the minimal enclosing bounding box, and *BBox*$^+$ applies additional perturbations to simulate real-world scenarios. ✓/✗ denotes whether a data filter is applied. *MSE* and *Ours* refer to the original generation loss and our optimized loss, respectively. *Seg* indicates the quality of data segmentation, with *High* representing human-annotated 3D segmentation datasets and *Low* representing 3D assets with built-in part information. *Cond* refers to the strategy for selecting 2D conditions during training described by *View Variance*. The CD metric is scaled by $10^2$ for better visualization.

| Exps | Mask | Filter | Loss | Seg | Cond | CD$\times 10^2\downarrow$ | PSNR↑ | SSIM↑ | LPIPS↓ | DINO↑ | FID↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| #1 | Exact | ✗ | MSE | High | random | 2.502 | 21.03 | 0.918 | 0.113 | 0.902 | 9.329 |
| #2 | BBox | ✗ | MSE | High | random | 0.701 | 24.65 | 0.946 | 0.057 | 0.960 | 3.709 |
| #3 | BBox$^+$ | ✗ | MSE | High | random | 0.692 | 24.75 | 0.946 | 0.056 | 0.962 | 3.573 |
| #4 | BBox$^+$ | ✓ | MSE | High | random | 0.658 | 24.82 | 0.947 | 0.055 | 0.962 | 3.542 |
| #5 | BBox$^+$ | ✓ | Ours | High | random | **0.635** | **25.01** | **0.947** | **0.054** | **0.963** | 3.496 |
| #6 | BBox$^+$ | ✓ | Ours | Low | random | 0.636 | 24.90 | 0.947 | 0.054 | 0.963 | **3.474** |
| #7 | BBox | ✗ | MSE | High | traverse | 0.850 | 24.41 | 0.946 | 0.059 | 0.953 | 4.129 |

with partial information are usable, enabling us to construct a large-scale dataset despite the scarcity of 3D segmentation datasets.

**Region-Aware Loss Reweighting.** As shown in Tab. 3, our improved loss function (Exp.#5) outperforms the vanilla generation MSE loss (Exp.#4).

**Training with Coarse 3D Masks.** We observe that as the training mask progresses from the exact mask shape (Exp.#1) to the minimal enclosing bounding box (Exp.#2), and further to the dilated bounding box (Exp.#3), model performance steadily improves. Notably, although training with the exact mask shape

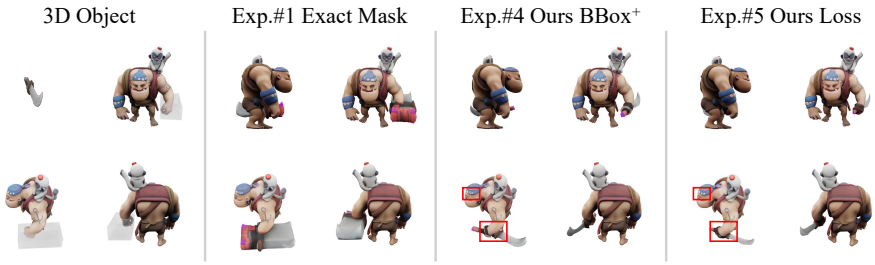


**Fig. 7: Qualitative ablation results.** *3D Object* displays the masked object alongside the target prompt (a sword). *Exact Mask* (Exp.#1) is trained with precise 3D masks. *Ours BBox*$^{+}$ (Exp.#4) utilizes our mask augmentation strategy. *Ours Loss* (Exp.#5) further incorporates our region-aware loss.

achieves good results on the training set, it generalizes poorly to the test set and produces meaningless outputs.

**Filtering Unrealistic Editing Pairs.** Filtering out tiny abnormal instances (Exp.#4) improves performance compared to the unfiltered dataset (Exp.#3).

**Segmentation Quality.** Regarding data sources, datasets with segmentation annotations yield higher-quality segmentation (Exp.#5) than 3D assets with part information (Exp.#6). Experiments show that models trained on these two sources achieve comparable performance.

**View Variance.** In real-world scenarios, the orientation of the 3D object and the 2D prompt are often misaligned. To improve generalization to the 2D prompt's view angle, we render 24 views from the partial 3D object and randomly select one per training iteration. Comparing this random selection strategy (Exp.#2) with traversing all 24 renderings (Exp.#7) shows that the random approach performs better, a counterintuitive result.

**Quality.** Fig. 7 shows that exact 3D masks yield blurred geometry in the target area, whereas our mask augmentation strategy significantly improves visual quality. Furthermore, our region-aware loss refines fine-grained details. It preserves unedited attributes (e.g., hat dots, arm tattoos). It enhances consistency between the image prompt and the edited geometry, particularly at transitional boundaries (e.g., the sword's hilt-to-blade junction).

**Robustness to Input Variance.** Fig. 8 demonstrates the robustness of our method against input perturbations. Our approach consistently produces plausible 3D editing results, despite variations in the positions and scales of the coarse 3D bounding boxes and in the viewing angles of the image prompt.

### 4.4 Inference Efficiency

Tab. 4 presents a comparison of inference efficiency between our method and the baselines. **(1)** Instant3dit: It is slow due to inefficient 3D reconstruction and impractical because it demands exact 3D masks. **(2)** Repaint & FlowEdit:

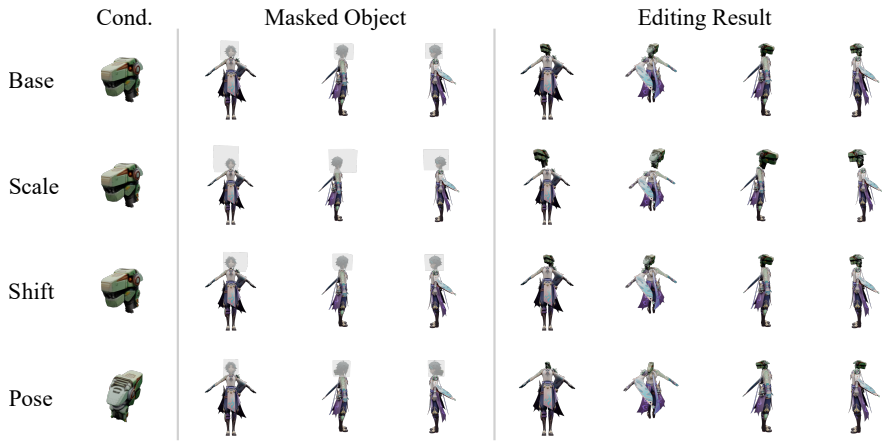


**Fig. 8: Robustness to input variations.** *Base* shows editing results using an ideal bounding box and a 2D prompt. *Scale* enlarges the baseline box by 1.5×. *Shift* translates the scaled box to the right. Finally, *Pose* demonstrates results when the image prompt (Cond.) features a different viewing angle than the *Base*.

**Table 4: Efficiency Comparison.** *Exact* and *Bbox*$^+$ indicate whether the method requires an accurate 3D mask or only a coarse 3D bounding box, respectively. A ✓ denotes that the resource is required, while a ✗ indicates that it is not.

| Method | TRELLIS | Instant3dit | Repaint | FlowEdit | VoxHammer | Ours |
|---|---|---|---|---|---|---|
| 3D Mask | – | Exact | BBox$^+$ | ✗ | Exact | BBox$^+$ |
| 2D Mask | – | ✓ | ✗ | ✗ | ✓ | ✗ |
| Edited 2D | – | ✓ | ✓ | ✓ | ✓ | ✗ |
| Runtime(s) | 20 | 30 | 20 | 20 | 120 | 20 |

The modifications applied at each inference timestep do not introduce computational overhead, thereby maintaining the efficiency of vanilla TRELLIS. However, utilizing resampling steps repetition introduces a trade-off between generation quality and inference speed. Regarding inputs, both variants require a pre-edited 2D view as guidance. Specifically, while Repaint does not strictly mandate an exact mask, providing one yields higher-quality results by ensuring the faithful preservation of larger unedited regions. Conversely, FlowEdit operates entirely without a mask, but this lack of spatial constraint renders its inference trajectory highly uncontrollable. **(3)** VoxHammer: It exhibits a significantly longer inference time due to the computational burden of integrating inner-layer features. Furthermore, it shares Repaint's stringent input requirements, putting it at a distinct disadvantage compared to our approach. In contrast, **(4)** our proposed method operates as a streamlined, end-to-end framework, preserving inference efficiency comparable to the vanilla TRELLIS while reducing the input burden.

## 5 Conclusion

In this work, we propose a novel end-to-end pipeline for local 3D object editing, which takes as input a 3D object, a coarse bounding box, and a reference 2D image. Our approach achieves high-fidelity results by introducing region-aware adaptive loss and data augmentation techniques. Extensive experiments demonstrate that our approach achieves state-of-the-art performance, outperforming existing methods in both editing quality and efficiency. Furthermore, we construct a large-scale dataset of 3D editing pairs, facilitating supervised training and future research in 3D editing.

## Acknowledgements

This research is supported by the MoE AcRF Tier 2 grant (MOE-T2EP20223-0001) and the MoE AcRF Tier 1 grant (RG14/22). This research is also supported by a Tencent research grant (04IDS001541N022).

## References

1. Bar-On, R., Cohen-Bar, D., Cohen-Or, D.: Editp23: 3d editing via propagation of image prompts to multi-view (2025), https://arxiv.org/abs/2506.20652
2. Barda, A., Gadelha, M., Kim, V.G., Aigerman, N., Bermano, A.H., Groueix, T.: Instant3dit: Multiview inpainting for fast editing of 3d objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16273–16282 (2025)
3. Bokhovkin, A., Meng, Q., Tulsiani, S., Dai, A.: Scenefactor: Factored latent 3d diffusion for controllable 3d scene generation (Dec 2024). https://doi.org/10.48550/arXiv.2412.01801
4. Cen, J., Fang, J., Yang, C., Xie, L., Zhang, X., Shen, W., Tian, Q.: Segment any 3d gaussians (Feb 2025). https://doi.org/10.48550/arXiv.2312.00860
5. Chen, H., Shi, R., Liu, Y., Shen, B., Gu, J., Wetzstein, G., Su, H., Guibas, L.: Generic 3d diffusion adapter using controlled multi-view editing (2024), https://arxiv.org/abs/2403.12032
6. Chen, M., Shapovalov, R., Laina, I., Monnier, T., Wang, J., Novotny, D., Vedaldi, A.: Partgen: Part-level 3d generation and reconstruction with multi-view diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5881–5892 (2025)
7. Chen, M., Xie, J., Laina, I., Vedaldi, A.: Shap-editor: Instruction-guided latent 3d editing in seconds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 26456–26466 (2024)
8. Chen, T., Yu, C., Li, J., Zhang, J., Zhu, L., Ji, D., Zhang, Y., Zang, Y., Li, Z., Sun, L.: Reasoning3d – grounding and reasoning in 3d: Fine-grained zero-shot open-vocabulary 3d reasoning part segmentation via large vision-language models (May 2024). https://doi.org/10.48550/arXiv.2405.19326
9. Chen, Y., Hu, T., Tang, Y., Chen, S., Chen, A., Yi, R.: Plasticine3d: 3d non-rigid editing with text guidance by multi-view embedding optimization (Jul 2024). https://doi.org/10.48550/arXiv.2312.10111

10. Chen, Y., Li, Z., Wang, Y., Zhang, H., Li, Q., Zhang, C., Lin, G.: Ultra3d: Efficient and high-fidelity 3d generation with part attention (Jul 2025). https://doi.org/10.48550/arXiv.2507.17745
11. ChoiChangwoon, LeeJaeah, ParkJaesik, Min, K.: 3doodle: Compact abstraction of objects with 3d strokes. ACM Transactions on Graphics (TOG) (Jul 2024). https://doi.org/10.1145/3658156
12. Darcet, T., Oquab, M., Mairal, J., Bojanowski, P.: Vision transformers need registers (2023)
13. Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., VanderBilt, E., Kembhavi, A., Vondrick, C., Gkioxari, G., Ehsani, K., Schmidt, L., Farhadi, A.: Objaverse-xl: A universe of 10m+ 3d objects. arXiv preprint arXiv:2307.05663 (2023)
14. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13142–13153 (2023)
15. Deng, Y., He, X., Mei, C., Wang, P., Tang, F.: Fireflow: Fast inversion of rectified flow for image semantic editing. In: Forty-Second International Conference on Machine Learning (Jun 2025)
16. Dinh, N.A., Lang, I., Kim, H., Stein, O., Hanocka, R.: Geometry in style: 3d stylization via surface normal deformation (2025), https://arxiv.org/abs/2503.23241
17. Dong, S., Ding, L., Huang, Z., Wang, Z., Xue, T., Xu, D.: Interactive3d: Create what you want by interactive 3d generation (2024), https://arxiv.org/abs/2404.16510
18. Erkoç, Z., Gümeli, C., Wang, C., Nießner, M., Dai, A., Wonka, P., Lee, H.Y., Zhuang, P.: Preditor3d: Fast and precise 3d shape editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 640–649 (2025)
19. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 605–613 (2017)
20. Fang, S., Wang, Y., Tsai, Y.H., Yang, Y., Ding, W., Zhou, S., Yang, M.H.: Chat-edit-3d: Interactive 3d scene editing via text prompts. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G. (eds.) Computer Vision – ECCV 2024. pp. 199–216. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-72946-1_12
21. Fischer, M., Georgiev, I., Groueix, T., Kim, V.G., Ritschel, T., Deschaintre, V.: Sama: Material-aware 3d selection and segmentation (Nov 2024). https://doi.org/10.48550/arXiv.2411.19322
22. Gao, W., Wang, D., Fan, Y., Bozic, A., Stuyck, T., Li, Z., Dong, Z., Ranjan, R., Sarafianos, N.: 3d mesh editing using masked lrms (2025), https://arxiv.org/abs/2412.08641
23. Gao, Z., Yi, R., Huang, Y., Chen, W., Zhu, C., Xu, K.: Self-supervised learning of hybrid part-aware 3d representations of 2d gaussians and superquadrics (Jul 2025). https://doi.org/10.48550/arXiv.2408.10789
24. Google: Google gemini. https://gemini.google.com/app (2025)
25. Han, X., Tian, R., Tong, Y., Yu, F., Liu, D., Zhang, Y.: Arap-gs: Drag-driven as-rigid-as-possible 3d gaussian splatting editing with diffusion prior (Apr 2025). https://doi.org/10.48550/arXiv.2504.12788

26. He, K., Wu, C.H., Gilitschenski, I.: Ctrl-d: Controllable dynamic 3d scene editing with personalized 2d diffusion. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 26630–26640 (2025)
27. He, X., Zou, Z.X., Chen, C.H., Guo, Y.C., Liang, D., Yuan, C., Ouyang, W., Cao, Y.P., Li, Y.: Sparseflex: High-resolution and arbitrary-topology 3d shape modeling (Mar 2025). https://doi.org/10.48550/arXiv.2503.21732
28. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
29. Hong, Y., Cai, X., Zeng, P., Zhang, S., Song, J., Gao, L., Shen, H.T.: Towards generalized and training-free text-guided semantic manipulation (Jul 2025). https://doi.org/10.48550/arXiv.2504.17269
30. Jincheng, J., Cai, Y., Liu, L.: Craftmesh: High-fidelity generative mesh manipulation via poisson seamless fusion (Sep 2025). https://doi.org/10.48550/arXiv.2509.13688
31. Jose, C., Moutakanni, T., Kang, D., Baldassarre, F., Darcet, T., Xu, H., Li, D., Szafraniec, M., Ramamonjisoa, M., Oquab, M., Siméoni, O., Vo, H.V., Labatut, P., Bojanowski, P.: Dinov2 meets text: A unified framework for image- and pixel-level vision-language alignment (2024)
32. Ju, X., Liu, X., Wang, X., Bian, Y., Shan, Y., Xu, Q.: Brushnet: A plug-and-play image inpainting model with decomposed dual-branch diffusion. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G. (eds.) Computer Vision – ECCV 2024. pp. 150–168. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-72661-3_9
33. Kerbl, B., Kopanas, G., Leimkuehler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4), 1–14 (Aug 2023). https://doi.org/10.1145/3592433
34. Kim, J., Lee, S., Shin, J., Choi, J., Shim, H.: Dreamcatalyst: Fast and high-quality 3d editing via controlling editability and identity preservation. In: The Thirteenth International Conference on Learning Representations (Oct 2024)
35. Kulikov, V., Kleiner, M., Huberman-Spiegelglas, I., Michaeli, T.: Flowedit: Inversion-free text-based editing using pre-trained flow models (Jul 2025). https://doi.org/10.48550/arXiv.2412.08629
36. Labs, B.F., Batifol, S., Blattmann, A., Boesel, F., Consul, S., Diagne, C., Dockhorn, T., English, J., English, Z., Esser, P., Kulal, S., Lacey, K., Levi, Y., Li, C., Lorenz, D., Müller, J., Podell, D., Rombach, R., Saini, H., Sauer, A., Smith, L.: Flux.1 kontext: Flow matching for in-context image generation and editing in latent space (2025), https://arxiv.org/abs/2506.15742
37. Lai, Z., Zhao, Y., Zhao, Z., Liu, H., Wang, F., Shi, H., Yang, X., Lin, Q., Huang, J., Liu, Y., Jiang, J., Guo, C., Yue, X.: Unleashing vecset diffusion model for fast shape generation (Mar 2025). https://doi.org/10.48550/arXiv.2503.16302
38. Lan, Y., Zhou, S., Lyu, Z., Hong, F., Yang, S., Dai, B., Pan, X., Loy, C.C.: Gaussiananything: Interactive point cloud flow matching for 3d generation. International Conference on Representation Learning **2025**, 97419–97443 (May 2025)
39. Le, D.H., Pham, T., Kembhavi, A., Mandt, S., Ma, W.C., Lu, J.: Preserving identity with variational score for general-purpose 3d editing (Jun 2024). https://doi.org/10.48550/arXiv.2406.08953
40. Li, H., Tian, Y., Wang, Y., Liao, Y., Wang, L., Wang, Y., Zhou, P.Y.: Text-to-3d generation by 2d editing (Mar 2025). https://doi.org/10.48550/arXiv.2412.05929

41. Li, H., Erkoc, Z., Li, L., Sirigatti, D., Rozov, V., Dai, A., Nießner, M.: Meshpad: Interactive sketch-conditioned artist-reminiscent mesh generation and editing (Aug 2025). https://doi.org/10.48550/arXiv.2503.01425
42. Li, L., Huang, Z., Feng, H., Zhuang, G., Chen, R., Guo, C., Sheng, L.: Voxhammer: Training-free precise and coherent 3d editing in native 3d space (Aug 2025). https://doi.org/10.48550/arXiv.2508.19247
43. Li, P., Ma, S., Chen, J., Liu, Y., Zhang, C., Xue, W., Luo, W., Sheffer, A., Wang, W., Guo, Y.: Cmd: Controllable multiview diffusion for 3d editing and progressive generation. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers. pp. 1–10. ACM, Vancouver BC Canada (Aug 2025). https://doi.org/10.1145/3721238.3730722
44. Li, W., Zhang, X., Sun, Z., Qi, D., Li, H., Cheng, W., Cai, W., Wu, S., Liu, J., Wang, Z., Chen, X., Tian, F., Pan, J., Li, Z., Yu, G., Zhang, X., Jiang, D., Tan, P.: Step1x-3d: Towards high-fidelity and controllable generation of textured 3d assets (May 2025). https://doi.org/10.48550/arXiv.2505.07747
45. Lin, J., Yang, X., Chen, M., Xu, Y., Yan, D., Wu, L., Xu, X., Xu, L., Zhang, S., Chen, Y.C.: Kiss3dgen: Repurposing image diffusion models for 3d asset generation. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 5870–5880 (2025)
46. Lin, Y., Lin, C., Pan, P., Yan, H., Feng, Y., Mu, Y., Fragkiadaki, K.: Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers (Jun 2025). https://doi.org/10.48550/arXiv.2506.05573
47. Lipman, Y., Chen, R.T.Q., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling (2023), https://arxiv.org/abs/2210.02747
48. Liu, F., Wang, H., Chen, W., Sun, H., Duan, Y.: Make-your-3d: Fast and consistent subject-driven 3d content generation (2024), https://arxiv.org/abs/2403.09625
49. Liu, F.L., Fu, H., Lai, Y.K., Gao, L.: Sketchdream: Sketch-based text-to-3d generation and editing. ACM Transactions on Graphics **43**(4), 1–13 (Jul 2024). https://doi.org/10.1145/3658120
50. Liu, M., Uy, M.A., Xiang, D., Su, H., Fidler, S., Sharp, N., Gao, J.: Partfield: Learning 3d feature fields for part segmentation and beyond (2025)
51. Long, X., Guo, Y.C., Lin, C., Liu, Y., Dou, Z., Liu, L., Ma, Y., Zhang, S.H., Habermann, M., Theobalt, C., Wang, W.: Wonder3d: Single image to 3d using cross-domain diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9970–9980 (2024)
52. Lu, R., Chen, Y., Ni, J., Jia, B., Liu, Y., Wan, D., Zeng, G., Huang, S.: Movis: Enhancing multi-object novel view synthesis for indoor scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 26767–26778 (2025)
53. Lu, Y., Tian, Y., Jiang, Z., Zhao, Y., Yang, Y., Ouyang, H., Hu, H., Yu, H., Shen, Y., Liao, Y.: Orientation matters: Making 3d generative models orientation-aligned (Jun 2025). https://doi.org/10.48550/arXiv.2506.08640
54. Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., Van Gool, L.: Repaint: Inpainting using denoising diffusion probabilistic models. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11451–11461 (Jun 2022). https://doi.org/10.1109/CVPR52688.2022.01117
55. Ma, C., Li, Y., Yan, X., Xu, J., Yang, Y., Wang, C., Zhao, Z., Guo, Y., Chen, Z., Guo, C.: P3-sam: Native 3d part segmentation. arXiv preprint arXiv:2509.06784 (2025)

56. Ma, Z., Yue, Y., Gkioxari, G.: Find any part in 3d (Mar 2025). https://doi.org/10.48550/arXiv.2411.13550
57. Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.Y., Ermon, S.: Sdedit: Guided image synthesis and editing with stochastic differential equations. In: International Conference on Learning Representations (Oct 2021)
58. Mikaeili, A., Perel, O., Safaee, M., Cohen-Or, D., Mahdavi-Amiri, A.: Sked: Sketch-guided text-based 3d editing. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14607–14619 (2023)
59. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (Jan 2022). https://doi.org/10.1145/3503250
60. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.Y., Xu, H., Sharma, V., Li, S.W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: Dinov2: Learning robust visual features without supervision (2023)
61. Parelli, M., Oechsle, M., Niemeyer, M., Tombari, F., Geiger, A.: 3d-latte: Latent space 3d editing from textual instructions (Sep 2025). https://doi.org/10.48550/arXiv.2509.00269
62. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. In: The Eleventh International Conference on Learning Representations (Sep 2022)
63. Qi, Z., Yang, Y., Zhang, M., Xing, L., Wu, X., Wu, T., Lin, D., Liu, X., Wang, J., Zhao, H.: Tailor3d: Customized 3d assets editing and generation with dual-side images (2024), https://arxiv.org/abs/2407.06191
64. Ren, Y., Jiang, Z., Zhang, T., Forchhammer, S., Süsstrunk, S.: Fds: Frequency-aware denoising score for text-guided latent diffusion image editing. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 2651–2660 (2025)
65. Rojas, S., Philip, J., Zhang, K., Bi, S., Luan, F., Ghanem, B., Sunkavalli, K.: Datenerf: Depth-aware text-based editing of nerfs. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G. (eds.) Computer Vision – ECCV 2024. pp. 267–284. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-73247-8_16
66. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10684–10695 (Jun 2022)
67. Rout, L., Chen, Y., Ruiz, N., Caramanis, C., Shakkottai, S., Chu, W.S.: Semantic image inversion and editing using rectified stochastic differential equations. In: The Thirteenth International Conference on Learning Representations (Oct 2024)
68. Sella, E., Atia, N., Mokady, R., Averbuch-Elor, H.: Blended point cloud diffusion for localized text-guided shape editing. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19119–19129 (2025)
69. Sella, E., Fiebelman, G., Hedman, P., Averbuch-Elor, H.: Vox-e: Text-guided voxel editing of 3d objects. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 430–440 (2023)
70. Shaocong, D., Lihe, D., Xiao, C., Yaokun, L., Yuxin, W., Yucheng, W., Qi, W., Jaehyeok, K., Chenjian, G., Zhanpeng, H., Zibin, W., Tianfan, X., Xu, D.: From one to more: Contextual part latents for 3d generation. In: ICCV (2025)

71. Song, G., Zhao, Z., Weng, H., Zeng, J., Jia, R., Gao, S.: Mesh silksong: Autoregressive mesh generation as weaving silk (Jul 2025). https://doi.org/10.48550/arXiv.2507.02477
72. Tang, G., Zhao, W., Ford, L., Benhaim, D., Zhang, P.: Segment any mesh (Mar 2025). https://doi.org/10.48550/arXiv.2408.13679
73. Tang, J., Lu, R., Li, Z., Hao, Z., Li, X., Wei, F., Song, S., Zeng, G., Liu, M.Y., Lin, T.Y.: Efficient part-level 3d object generation via dual volume packing (Jun 2025). https://doi.org/10.48550/arXiv.2506.09980
74. Team, T.H.: Hunyuan3d 1.0: A unified framework for text-to-3d and image-to-3d generation (2024)
75. Team, T.H.: Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation (2025)
76. Team, T.H.: Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details (2025), https://arxiv.org/abs/2506.16504
77. Team, T.H.: Hunyuan3d-omni: A unified framework for controllable generation of 3d assets (2025), https://arxiv.org/abs/2509.21245
78. Wang, J., Pu, J., Qi, Z., Guo, J., Ma, Y., Huang, N., Chen, Y., Li, X., Shan, Y.: Taming rectified flow for inversion and editing. In: Proceedings of the 42nd International Conference on Machine Learning. pp. 64044–64058. PMLR (Oct 2025)
79. Wang, R., Xu, S., Dai, C., Xiang, J., Deng, Y., Tong, X., Yang, J.: Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 5261–5271 (2025)
80. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (Apr 2004). https://doi.org/10.1109/TIP.2003.819861
81. Wei, S.T., Wang, R.H., Zhou, C.Z., Chen, B., Wang, P.S.: Octgpt: Octree-based multiscale autoregressive models for 3d shape generation. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers. pp. 1–11. ACM, Vancouver BC Canada (Aug 2025). https://doi.org/10.1145/3721238.3730601
82. Wu, C., Li, J., Zhou, J., Lin, J., Gao, K., Yan, K., ming Yin, S., Bai, S., Xu, X., Chen, Y., Chen, Y., Tang, Z., Zhang, Z., Wang, Z., Yang, A., Yu, B., Cheng, C., Liu, D., Li, D., Zhang, H., Meng, H., Wei, H., Ni, J., Chen, K., Cao, K., Peng, L., Qu, L., Wu, M., Wang, P., Yu, S., Wen, T., Feng, W., Xu, X., Wang, Y., Zhang, Y., Zhu, Y., Wu, Y., Cai, Y., Liu, Z.: Qwen-image technical report (2025), https://arxiv.org/abs/2508.02324
83. Wu, S., Lin, Y., Zhang, F., Zeng, Y., Yang, Y., Bao, Y., Qian, J., Zhu, S., Cao, X., Torr, P., Yao, Y.: Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention (May 2025). https://doi.org/10.48550/arXiv.2505.17412
84. Wu, T., Zheng, C., Guan, F., Vedaldi, A., Cham, T.J.: Amodal3r: Amodal 3d reconstruction from occluded 2d images (Mar 2025). https://doi.org/10.48550/arXiv.2503.13439
85. Xia, R., Tang, Y., Zhou, P.: Towards scalable and consistent 3d editing (Oct 2025). https://doi.org/10.48550/arXiv.2510.02994
86. Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., Yang, J.: Structured 3d latents for scalable and versatile 3d generation. arXiv preprint arXiv:2412.01506 (2024)

87. Xu, Y., Shi, Z., Yifan, W., Chen, H., Yang, C., Peng, S., Shen, Y., Wetzstein, G.: Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G. (eds.) Computer Vision – ECCV 2024. pp. 1–20. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-72633-0_1
88. Xu, Y., Zhu, L., Yang, Y.: Gg-editor: Locally editing 3d avatars with multimodal large language model guidance. In: ACM Multimedia 2024 (Jul 2024)
89. Yan, X., Xu, J., Li, Y., Ma, C., Yang, Y., Wang, C., Zhao, Z., Lai, Z., Zhao, Y., Chen, Z., et al.: X-part: high fidelity and structure coherent shape decomposition. arXiv preprint arXiv:2509.08643 (2025)
90. Yan, Z., Li, L., Shao, Y., Chen, S., Wu, Z., Hwang, J.N., Zhao, H., Remondino, F.: 3dsceneeditor: Controllable 3d scene editing with gaussian splatting (Dec 2024). https://doi.org/10.48550/arXiv.2412.01583
91. Yang, Y., Guo, Y.C., Huang, Y., Zou, Z.X., Yu, Z., Li, Y., Cao, Y.P., Liu, X.: Holopart: Generative 3d part amodal segmentation (Apr 2025). https://doi.org/10.48550/arXiv.2504.07943
92. Yang, Y., Huang, Y., Guo, Y.C., Lu, L., Wu, X., Edmund Y., L., Cao, Y.P., Liu, X.: Sampart3d: Segment any part in 3d objects. arXiv preprint arXiv:2411.07184 (2024)
93. Yang, Y., Zhou, Y., Guo, Y.C., Zou, Z.X., Huang, Y., Liu, Y.T., Xu, H., Liang, D., Cao, Y.P., Liu, X.: Omnipart: Part-aware 3d generation with semantic decoupling and structural cohesion (Jul 2025). https://doi.org/10.48550/arXiv.2507.06165
94. Yao, K., Zhang, L., Yan, X., Zeng, Y., Zhang, Q., Xu, L., Yang, W., Gu, J., Yu, J.: Cast: Component-aligned 3d scene reconstruction from an rgb image. ACM Transactions on Graphics **44**(4), 1–19 (Aug 2025). https://doi.org/10.1145/3730841
95. Ye, J., He, Y., Zhou, Y., Zhu, Y., Xiao, K., Liu, Y.J., Yang, W., Han, X.: Primitiveanything: Human-crafted 3d primitive assembly generation with auto-regressive transformer. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers. pp. 1–12. ACM, Vancouver BC Canada (Aug 2025). https://doi.org/10.1145/3721238.3730732
96. Ye, J., Xie, S., Zhao, R., Wang, Z., Yan, H., Zu, W., Ma, L., Zhu, J.: Nano3d: A training-free approach for efficient 3d editing without masks (Oct 2025). https://doi.org/10.48550/arXiv.2510.15019
97. Yin, Y., Fu, Z., Yang, F., Lin, G.: Or-nerf: Object removing from 3d scenes guided by multiview segmentation with neural radiance fields (2023), https://arxiv.org/abs/2305.10503
98. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19447–19456 (2024)
99. Zhang, B., Tang, J., Nießner, M., Wonka, P.: 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. ACM Trans. Graph. **42**(4), 92:1–92:16 (Jul 2023). https://doi.org/10.1145/3592442
100. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 586–595 (Jun 2018). https://doi.org/10.1109/CVPR.2018.00068

101. Zhang, X., Lee, J., Joslin, C., Lee, W.: Advancing 3d gaussian splatting editing with complementary and consensus information (Mar 2025). https://doi.org/10.48550/arXiv.2503.11601
102. Zhang, Y., Zhang, L., Ma, R., Cao, N.: Texverse: A universe of 3d objects with high-resolution textures (Sep 2025). https://doi.org/10.48550/arXiv.2508.10868
103. Zhang, Z., Xie, J., Lu, Y., Yang, Z., Yang, Y.: In-context edit: Enabling instructional image editing with in-context generation in large scale diffusion transformer (Sep 2025). https://doi.org/10.48550/arXiv.2504.20690
104. Zhao, W., Cao, Y.P., Xu, J., Dong, Y., Shan, Y.: Assembler: Scalable 3d part assembly via anchor point diffusion (Jun 2025). https://doi.org/10.48550/arXiv.2506.17074
105. Zheng, K., Zha, R., Xu, Z., Gu, J., Yang, J., Wang, X.E.: Constructing a 3d scene from a single image (Oct 2025). https://doi.org/10.48550/arXiv.2505.15765
106. Zhou, Z., Deng, Y., He, X., Dong, W., Tang, F.: Multi-turn consistent image editing (May 2025). https://doi.org/10.48550/arXiv.2505.04320
107. Zhu, Z., Wan, L., Xu, R., Zhang, Y., Chen, H., Dou, Z., Lin, C., Liu, Y., Wei, M.: Partsam: A scalable promptable part segmentation model trained on native 3d data. arXiv preprint arXiv:2509.21965 (2025)
108. Zhuang, J., Kang, D., Cao, Y.P., Li, G., Lin, L., Shan, Y.: Tip-editor: An accurate 3d editor following both text-prompts and image-prompts. ACM Transactions on Graphics **43**(4), 1–12 (Jul 2024). https://doi.org/10.1145/3658205