Supplementary Material for 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

1 Implementation Details

1.1 Texture Model Input Padding

Unlike the structure model, which samples the structure latent from noise under the guidance of 2D image conditions, the input to the texture model includes 3D voxel coordinates, along with noise and 2D conditions. Its output is a texture feature latent that corresponds one-to-one with the input coordinates. In 3D generation tasks, the input and output 3D voxels remain identical. However, editing tasks present a more complex scenario. In addition to the noise and 2D conditions, we provide the source object's texture features as guidance for the edited object's texture features. As a result, the input and output texture features occupy distinct 3D voxel coordinate spaces. Because our dataset focuses on *Add* edits, the input 3D voxel coordinates form a subset of the output 3D voxel coordinates. During training, we apply zero-padding to the texture features at coordinates present in the output but absent in the input. During inference, the source object and the edited geometry predicted by the structure model only partially overlap, meaning the source object is no longer a strict subset of the predicted geometry. To address this, we identify the coordinates shared between the source object and the predicted geometry, and apply zero-padding to all other coordinate positions. Algorithm 1 and Fig. 1 show the padding mechanism. The generative capability of our model ensures the predicted edited geometry remains highly consistent with the input geometry in unedited regions.

1.2 Dataset Preparation

Our editing dataset consists of 3D objects to be edited, 2D image prompts describing the intended edit, and 3D masks indicating the target editing region. To construct valid editing pairs, we ensure that each selected 3D asset can be

Algorithm 1 Padding Strategy for Texture Model Input

Require: Source voxel coordinates $\mathcal{V}_{src}$ and latents $\mathbf{L}_{src}$;
Ground truth coordinates $\mathcal{V}_{gt}$ and predicted coordinates $\mathcal{V}_{pred}$.
Ensure: Input coordinates $\mathcal{V}_{in}$ and texture latents $\mathbf{L}_{in}$ for the model.

function PADDING($\mathcal{V}_{src}, \mathbf{L}_{src}, \mathcal{V}_{gt}, \mathcal{V}_{pred}$, phase)
  if phase == Training **then**
    $\mathcal{V}_{in} \leftarrow \mathcal{V}_{gt}$ ▷ Use GT geometry as output
    $\mathbf{L}_{in}(v) \leftarrow \begin{cases} \mathbf{L}_{src}(v) & \text{if } v \in \mathcal{V}_{src} \\ \mathbf{0} & \text{if } v \in \mathcal{V}_{gt} \setminus \mathcal{V}_{src} \end{cases}$
  else if phase == Inference **then**
    $\mathcal{V}_{in} \leftarrow \mathcal{V}_{pred}$ ▷ Use predicted geometry as output
    $\mathbf{L}_{in}(v) \leftarrow \begin{cases} \mathbf{L}_{src}(v) & \text{if } v \in (\mathcal{V}_{src} \cap \mathcal{V}_{pred}) \\ \mathbf{0} & \text{if } v \in \mathcal{V}_{pred} \setminus \mathcal{V}_{src} \end{cases}$
  end if
  return $\{\mathcal{V}_{in}, \mathbf{L}_{in}\}$
end function

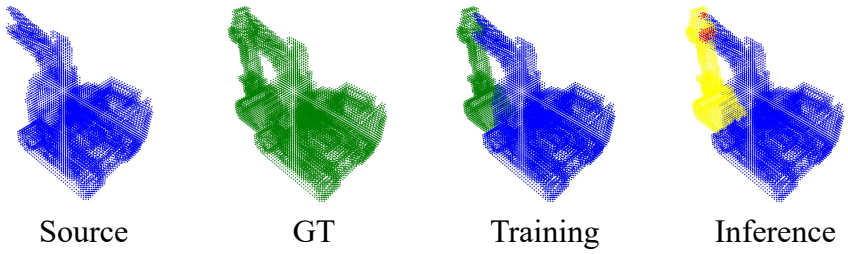

Fig. 1: Texture model input padding. *Source* (blue): The object to be edited. *GT* (green): The ground truth edited object. *Training*: Zero padding applied to the regions outside the *Source* (green). *Inference*: Zero padding applied to the regions not belonging to the *Source* (yellow+red).

decomposed into multiple structurally and texturally complete parts. We remove one or more parts from the asset, and the remaining part serves as the input 3D object. To obtain the image prompts, we uniformly sample 24 camera poses on a sphere centered around the removed part region and render 24 views. The removed parts naturally provide an accurate ground-truth 3D mask for the editing region. During training and inference, we apply different perturbations to this mask to enhance model generalization and simulate realistic scenarios in which users do not need to provide accurate 3D masks but prefer annotating the approximate editing regions.

For 3D segmentation data, most assets are sourced from Partverse [6]. However, the segmentation labels are provided on untextured grayscale meshes that do not align with their corresponding textured meshes. To address this mis-

Algorithm 2 Segmentation Label Mapping

Require: Untextured mesh $\mathcal{M}_{wo}$ with faces $\{f_{wo}\}$ and labels S;
Textured mesh $\mathcal{M}_w$ with faces $\{f_w\}$.
Ensure: Aligned textured mesh $\mathcal{M}_{aligned}$ and mapped labels S_w.

function MAPPING($\mathcal{M}_{wo}, S, \mathcal{M}_w$)
  $\mathcal{F}_{preserve} \leftarrow \emptyset$ ▷ Initialize the set of matched faces
  $S_w \leftarrow$ Empty mapping
  for each face $f_i \in \mathcal{M}_{wo}$ **do**
    $f^* \leftarrow \arg\min_{f_j \in \mathcal{M}_w} \text{dist}(f_i, f_j)$ ▷ Find the nearest face in $\mathcal{M}_w$
    $\mathcal{F}_{preserve} \leftarrow \mathcal{F}_{preserve} \cup \{f^*\}$
    $S_w(f^*) \leftarrow S(f_i)$ ▷ Transfer label to textured face
  end for
  $\mathcal{M}_{aligned} \leftarrow \{f \in \mathcal{M}_w \mid f \in \mathcal{F}_{preserve}\}$ ▷ Prune unmatched faces in $\mathcal{M}_w$
  return $\{\mathcal{M}_{aligned}, S_w\}$
end function

match, we locate the textured mesh's face nearest to the untextured mesh's face, and add the matched faces to a preserve set. We then remove all faces on the textured mesh that are not included in this preserve set, thereby aligning it with the segmentation annotations. Algorithm 2 explains the mapping method in detail. We also manually collect a small portion of high-quality 3D assets without segmentation labels online, for which we apply PartField [4] to produce segmentation, and retain only assets whose segmentation yields meaningful edits and complete part decompositions.

For Objaverse [1, 2] assets with native part splitting, we select objects with part counts ranging from 2 to 20 and generate editing pairs using the aforementioned pipeline. Moreover, we perform manual quality control and remove problematic assets, including those with severe geometric distortion, unreasonable topology, or missing textures. We present segmentation examples and editing pairs extracted from a segmented mesh in Fig. 2.

1.3 Training with Coarse 3D Masks

To simulate real-world usage in which users are unlikely to provide an accurate 3D mask and instead specify an approximate target editing region via a coarse 3D bounding box, we perturb the 3D mask during both training and inference. To distinguish between training and test, we use different perturbation strategies for each stage. For training, we dilate the bounding box by 1 voxel around the minimum bounding box in the $16 \times 16 \times 16$ latent space. Experimental results indicate that no additional positional augmentation is required at training time—a bounding box trained at an accurate location generalizes well to shifted boxes at test time—a small disturbance in a highly compressed latent space is significant enough for the 3D object. For inference, we enlarge the minimum bounding box of the accurate 3D mask around its center by a random scaling

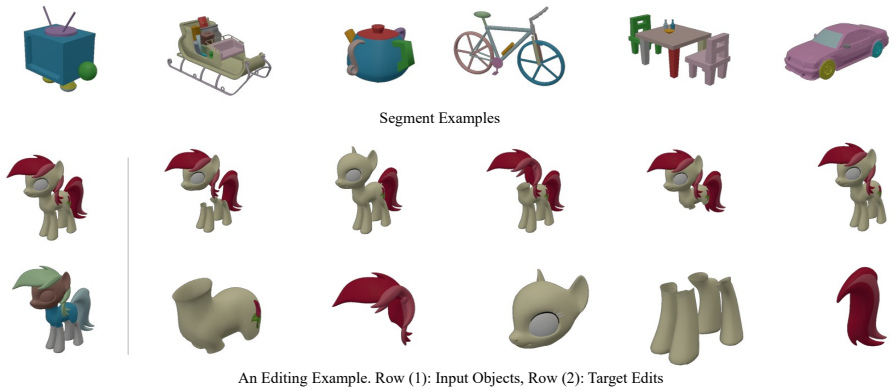

Fig. 2: Dataset snapshot. The upper section displays segmentation maps for various meshes, while the lower section illustrates editing pairs derived from a single mesh.

Fig. 3: Mask Augmentation. *BBox* denotes the minimal bounding box of the precise 3D mask. During *Training*, we employ an expanded mask to enhance generative robustness. At *inference*, we utilize a coarse 3D bounding box that is less precise than the minimal *BBox* and distinct from the *training*-phase augmentation strategy.

factor between 1.0 and 1.2, and apply a random spatial translation. Fig. 3 shows an example of different mask augmentations during training and inference.

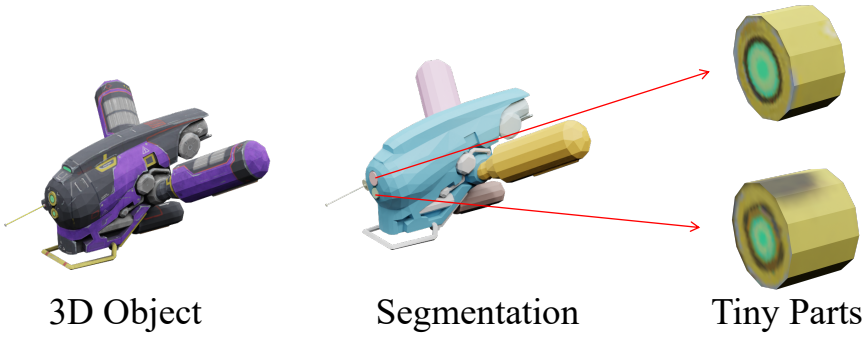

Fig. 4: Filtering Unrealistic Editing Pairs. Although the depicted object possesses accurate segmentation annotations, the two tiny cylinders are highly unlikely to be selected for real-world editing tasks. Filtering such instances from the dataset consequently enhances overall model performance.

Fig. 5: Comparison between image and text prompts. Text prompts can yield ambiguous editing targets, whereas image prompts effectively constrain generation to produce content strictly consistent with the reference image.

1.4 Filtering Unrealistic Editing Pairs

Although our editing pairs are derived from high-quality 3D segmentation data, accurate segmentations do not necessarily translate to viable 3D editing pairs. Upon analyzing the 3D data, we observed that object volume serves as a simple yet highly effective metric for evaluating 3D editing quality. As illustrated in Fig. 4, tiny parts often correspond to highly improbable editing cases in real-world applications. We evaluated various absolute voxel volumes (within a $64 \times 64 \times 64$ grid) and relative proportions of the ground truth mesh to filter out low-quality 3D parts. Through empirical tuning, we determined that an absolute volume threshold of fewer than 8 voxels yields an optimal balance between model performance and dataset scale, as overly stringent filtering would discard a substantial amount of training data, which is crucial for model efficacy.

Table 1: Ablation study for different region. Performance on the *Target Editing Region* aligns with the global evaluation. We provide a detailed explanation in the text regarding the suboptimal performance of the *Unedited Region* in Exp.#5.

Exps	Mask	Filter	Loss	Seg	Cond	Target Editing Region					Unedited Region				
						CD×10^2↓	PSNR↑	SSIM↑	LPIPS↓×10^2	DINO↑	CD×10^2↓	PSNR↑	SSIM↑	LPIPS↓×10^2	DINO↑
#1	Exact	✗	MSE	High	random	4.351	12.41	0.409	9.918	0.639	3.283	30.15	0.982	2.027	0.965
#2	BBox	✗	MSE	High	random	1.671	16.43	0.645	4.619	0.800	0.406	35.27	0.986	0.907	0.986
#3	BBox$^+$	✗	MSE	High	random	1.587	16.41	0.647	4.318	0.808	0.578	35.01	0.985	0.900	0.984
#4	BBox$^+$	✓	MSE	High	random	1.513	16.46	0.651	4.330	0.807	0.461	35.00	0.984	0.867	0.984
#5	BBox$^+$	✓	Ours	High	random	**1.487**	**16.67**	**0.657**	**4.175**	**0.813**	0.408	35.16	0.984	**0.841**	0.984
#6	BBox$^+$	✓	Ours	Low	random	1.488	16.56	0.653	4.236	0.811	0.449	35.11	0.984	0.876	0.984
#7	BBox	✗	MSE	High	traverse	2.090	16.20	0.636	5.210	0.773	**0.371**	**35.35**	**0.987**	0.858	**0.987**

1.5 The Reason for Using Image Prompt

Many existing methods [3, 5] in the research community use text prompts for editing rather than our proposed input formulation. Although the text prompt is concise, it significantly compromises control over editing. Compared to image-based conditioning, text descriptions struggle to convey precise editing targets. To illustrate this, we provide a simple example in Fig. 5. Even with highly detailed text prompts (A central spherical orb featuring a detailed, traditional East Asian stylized cloud and wave motif in shades of light blue, sky blue, and white. The intricate, calligraphic swirling patterns cover the smooth, polished surface seamlessly, like ceramic. The background is pure white.), the model yields multiple variations of a ball with textures that diverge significantly from a desired target. In contrast, utilizing an image prompt ensures that the generated editing region remains strictly consistent with the input reference.

2 Experiments

2.1 Checkpoint Selection

For checkpoint selection, we evaluate the model on the PartObjaverse-Tiny [7] test set using 24 2D condition views, compute the average loss with respect to the ground truth, and select the checkpoint with the lowest loss for metric computation and visual display.

2.2 Region Comparison for Ablation Study

For the ablation study, we provide metrics that separately evaluate edited and preserved regions in Tab. 1. We observe that for the *Target Editing Region*, the conclusion is consistent with the average metrics in the main paper, where applying our training strategy (Exp.#5) achieves the best performance. In contrast, the behavior in the *Unedited Region* is more nuanced. The mask augmentation strategy from *BBox* (Exp.#2) to *BBox*$^+$ (Exp.#3) introduces noticeable variations (This behavior is expected, as expanding the mask size inadvertently encroaches upon *Unedited Region*), and incorporating the *traverse* view selection

further improves consistency in the preserved region. Nevertheless, since Exp.#5 yields the best overall result, and the editing task primarily prioritizes the quality of the edited region, we adopt the Exp.#5 strategy combination in our final model. At the same time, Exp.#5's performance on the preserved region remains competitive with other effective strategies.

2.3 More Quality Results

Fig. 6 demonstrates the ability to apply consistent edits across different meshes within the same semantic space. Fig. 7 illustrates our method's ability to apply sequential edits to a single mesh. We include additional visual results (Fig. 8-11) to further demonstrate the effectiveness of our method.

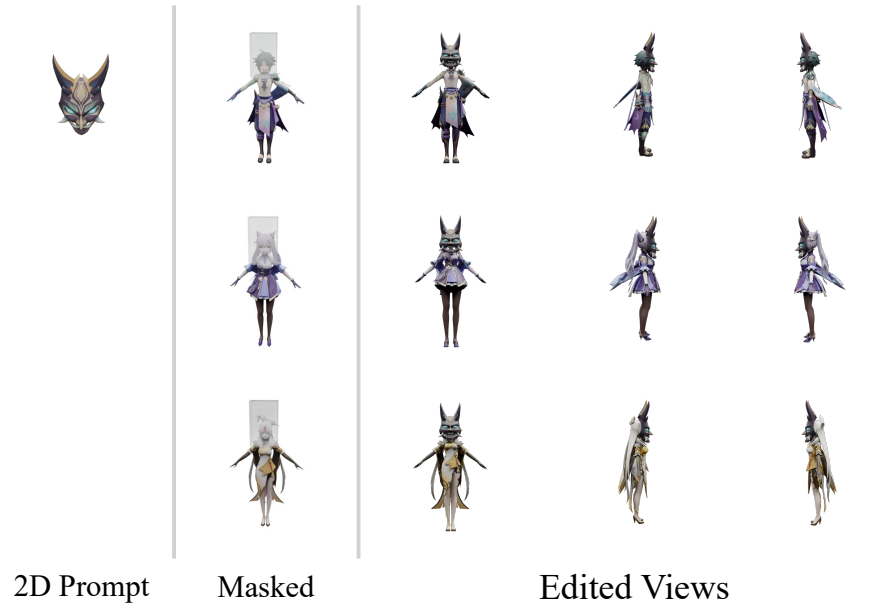

Fig. 6: Co-Editing. We apply an identical mask to the head regions of various objects.

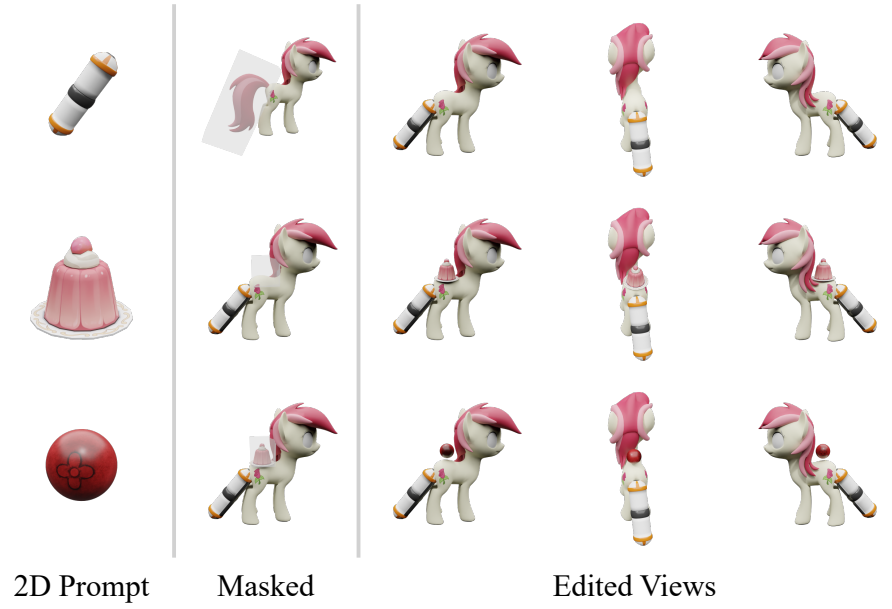

Fig. 7: Multi-Turn Editing. We initially replace the horse's tail, subsequently place a cake on its back, and finally replace the cake with a red sphere.

Fig. 8: The left 2×2 grid shows the image prompt (top left), three masked views, and the right 2×2 grid shows four corresponding edited views.

Fig. 9: The left 2×2 grid shows the image prompt (top left), three masked views, and the right 2×2 grid shows four corresponding edited views.

Fig. 10: The left 2×2 grid shows the image prompt (top left), three masked views, and the right 2×2 grid shows four corresponding edited views.

Fig. 11: The left 2×2 grid shows the image prompt (top left), three masked views, and the right 2×2 grid shows four corresponding edited views.

References

1. 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)
2. 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)
3. 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
4. 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)
5. 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)
6. 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)
7. 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)