Gaussian Grouping: Segment and Edit Anything in 3D Scenes —— Supplementary Material ——

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke ♠

Computer Vision Lab, ETH Zurich

In this supplementary material, we first conduct additional experiment analysis of our Gaussian Grouping in Section 1, including multi-granularity, segmentation efficiency, quantitative editing evaluation and robustness. Then, in Section 2, we describe the detailed process of annotating our proposed LERF-Mask datasets with the visualization of annotation examples. We further provide a more detailed 3D object inpainting and style transfer pipeline description in Section 3. Finally, we illustrate the algorithm pseudocode of our Gaussian Grouping and more implementation details in Section 4, including the method limitation analysis. Please refer to the attached Gaussian-grouping-supp.mp4 for an extensive 3D results comparison.

1 Supplementary Experiments

Segmentation Efficiency In Fig. 1, we compare the segmentation results with SA3D [2] on the proposed LERF-Mask dataset for open-world 3D segmentation. Our Gaussian Grouping shows great advantages in segmentation efficiency. Using the same machine, *our Gaussian Grouping jointly segments **all objects** of the 3D scene in 9 minutes, while SA3D requires 35 minutes for **each object*** due to its inverse rendering design in 3D voxel grids. To segment each object of the 3D scene, SA3D repeatedly needs separate new training, which makes SA3D time-consuming and not user-friendly in multi-object segmentation or editing scenarios.

Multi-Granularity of Masks As in Fig. 2, our method can process SAM's anything masks at different granularity levels in the multiple views as follows:

*Step **1**).* Firstly, SAM predicts dense anything mask proposals (including both large-granularity and small-granularity ones) for each frame.

*Step **2**).* Then, these masks are scored and sorted based on their mask areas. Masks are ranked in descending (from large to small) or ascending (from small to large) order to pick varying levels of granularity for the label of each pixel.

*Step **3**).* Furthermore, we filter large-area or small-area masks through their overlapping IoU with a threshold.

*Step **4**).* For mask association, masks are temporally propagated in a bi-direction to obtain in-clip consensus. Refer to Sec 3.2 of Deva [3] for details.

♠ Corresponding Author

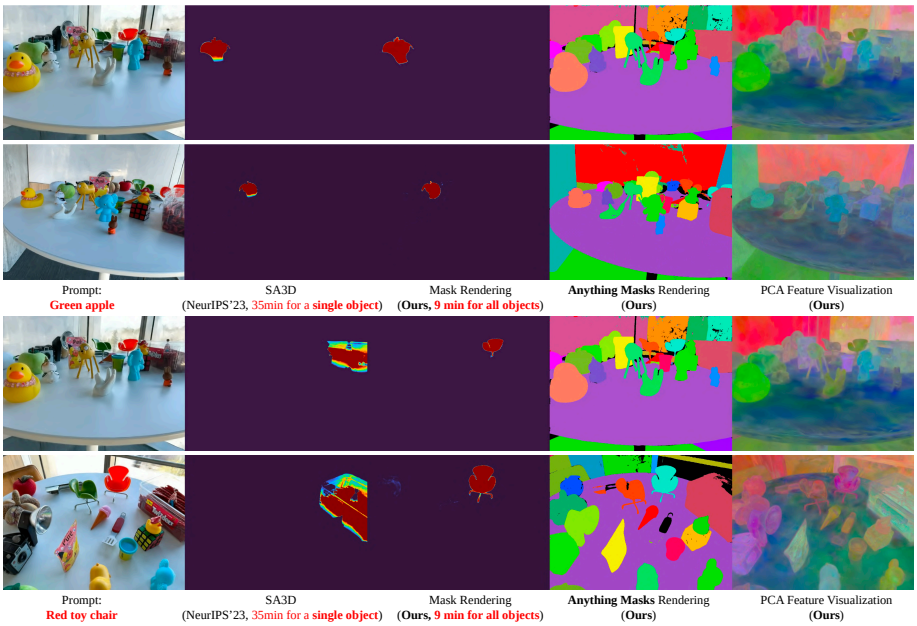

Fig. 1: Segmentation comparison between SA3D [2] and our Gaussian Grouping on the rendering view. We adopt PCA to visualize the rendered Identity Encoding features in the rightmost column. *Note that SA3D does not support concurrent multi-object segmentation* due to its design limitation in the inverse rendering, which requires training and rendering for each segmentation target for around 35 minutes (∼20min for training and ∼15min for rendering). In contrast, our Gaussian Grouping shows great efficiency by segmenting all objects in the scene only in 9 minutes.

Fig. 2: Consistent tracking results at multi-granularity masks. The 1st row is in coarse granularity, and the 2nd row is fine-grained.

Table 1: Quantitative comparison on three Scene Editing Tasks using the CLIP Text-Image Direction Similarity [5].

Task	Dataset	Model	CLIP Text-Image Direction Similarity
3D Object Inpainting	MipNeRF360/Kitchen	SPIN-NeRF [11]	0.126
		Ours	**0.153**
3D Object Style Transfer	Instruct-NeRF2NeRF/Bear	Instruct-NeRF2NeRF [5]	0.171
		Ours	**0.178**
3D Object Removal	Tandt/Truck	DFFs [9]	0.166
		Ours	**0.183**

Fig. 3: Tracking results for a new object in subsequent frames. The red circle highlights the new chair appearing in the 2nd frame.

Quantitative Evaluation of Editing Following Instruct-NeRF2NeRF [5], we provide the CLIP Text-Image Direction Similarity evaluation for three of our editing tasks in Table 1. Gaussian grouping supports versatile scene editing tasks, including 3D Object Inpainting, 3D Object Style Transfer and 3D Object Removal. For each of these editing tasks, Gaussian Grouping outperforms the corresponding SOTA method specifically designed for it. For inpainting, the descriptions of original and edited scenes are "Lego toy on the table" and "A flat table". For style transfer, the descriptions are "A bear statue" and "Turn the bear into a panda". For object removal, the descriptions are "A truck on urban street" and "Urban street".

Robustness of New Objects SAM+Tracking can process new objects on subsequent frames. When new high-confident segmentation doesn't match the previous objects and has high confidence, a new instance ID is assigned to it (see the new chair in the red circle in Fig. 3).

Sparse View Input For few-shot or sparse-view input, video tracking still obtains a good result for pre-processing, since it is visual feature-based and camera-motion robust. In Figure 4, we use a 3-view input for DEVA pre-processing and still get a decent tracking result. 3D reconstruction of the original Gaussian Splatting is not good with sparse-view input, but it is beyond the scope of this paper. It is retained as a component for sparse-view Gaussian Splatting reconstruction, intended for further research in the future.

2 Details on the LERF-Mask Annotation

Annotation Pipeline To measure the segmentation or fine-grained localization accuracy in the open-wold 3D scene, we construct the LERF-Mask dataset based on the existing LERF-Localization [7] evaluation dataset, where we manually

Fig. 4: Sparse 3-view tracking and reconstruction comparison.

annotate three scenes from LERF-Localization with accurate masks instead of using coarse bounding boxes. For each 3D scene, we provide 7.7 text queries with corresponding GT mask labels on average. We use Roboflow [4] platform for label annotation, and it uses SAM [8] as an auxiliary tool for interactive segmentation. Similar to the annotation used in LERF, for each of the 3 scenes, we choose 2-4 novel views for testing and annotating the rendering of novel views.

Annotation Examples All language prompts used for our LERF-Mask dataset evaluation are listed in Table 2, which contains 23 prompts in total. Also, we provide visualization on the mask annotations in Figure 5.

Scene	Text queries		
Figurines	green apple porcelain hand rubber duck with red hat	green toy chair red apple	old camera red toy chair
Ramen	chopsticks pork belly	egg wavy noodles in bowl	glass of water yellow bowl
Teatime	apple cookies on a plate sheep tea in a glass	bag of cookies paper napkin spoon handle	coffee mug plate stuffed bear

Table 2: Prompt labels used during segmentation experiments in our proposed LERF-Mask dataset (23 total).

Fig. 5: Annotation visualization of our proposed LERF-Mask dataset. We manually annotate three scenes from LERF-Localization [7] with accurate masks instead to replace the coarse bounding boxes in [5]. The text queries are detailed in Table 2.

3 Local Gaussian Editing: Steps of Object Inpainting & Style Transfer

3.1 3D Object Inpainting Pipeline

For inpainting, we remove the 3D Gaussians of the selected target by using a Gaussian Grouping model well-trained for 3D reconstruction and segmentation and add new Gaussians for finetuning. The steps are as follows:

*Step **1**).* Train the Gaussian Grouping model with our proposed 2D and 3D Identity Grouping loss.

*Step **2**).* Select the target object for inpainting. For each Identity Encoding associated with a 3D Gaussian, we acquire its linear layer classification result.

Subsequently, we remove those 3D Gaussians that are classified as the label of the selected object. Also, we remove the 3D Gaussians with position inside the convex hull of the object Gaussians.

Step 3). On the rendering views after the deletion of the object, we detect the "blurry hole" with Grounding-DINO [10] as the mask for 2D inpainting and use DEVA [3] for association. We use LAMA [12] inpainting on each view as the target for finetuning.

Step 4). After the 3D Gaussians of the target object are deleted, we clone 200K new Gaussians near the deletion region. We freeze the other Gaussians, and only finetune the newly introduced 3D Gaussians.

Step 5). During the finetuning, we employ L1 loss only in the outside regions of the object mask, and adopt LPIPS loss inside the bounding box of the object mask.

3.2 3D Object Style Transfer Pipeline

For style transfer, we finetune the 3D Gaussians belonging to the corresponding target by using a Gaussian Grouping model well-trained for 3D reconstruction and segmentation. The steps are as follows:

Step 1). Train our Gaussian Grouping model with our proposed 2D and 3D Identity Grouping loss.

Step 2). Select the target object for style transfer. For each Identity Encoding associated with a 3D Gaussian, we acquire its linear layer classification result. Subsequently, we only finetune those 3D Gaussians that are classified as the label of the selected object. Also, we finetune the 3D Gaussians with position inside the convex hull of the selected object. The Gaussians irrelevant to the editing target are frozen. During finetuning, we freeze the 3D position of Gaussains and make other Gaussian parameters (color, variance, opacity, etc.) trainable.

Step 3). During the finetuning process, we dynamically update the target images using an image-level style transfer model that has been pre-trained. Specifically, we employ InstructPix2Pix [1], introducing a noise input composed of the rendered view combined with random noise. This approach involves conditioning the diffusion model on a ground truth image to enhance accuracy and consistency.

Step 4). To preserve the spatial details of the background regions, rendering losses are exclusively performed within the mask of the style transfer target. On the 2D rendered view, we employ L1 loss inside the object mask and LPIPS loss within the bounding box that encloses the object mask.

4 More Implementation Details

4.1 More implementation details

We implement Gaussian Grouping based on Gaussian Splatting [6]. We add a 16-dimension identity encoding as a feature of each Gaussian, and implement

forward and backward cuda rasterization similar to the direct current of Spherical Harmonics. The 3D Identity Encoding has a shape of $N * 1 * 16$, where N is the number of Gaussians. We set the degree of Spherical Harmonics to zero since instance identity does not change across views. The rendered 2D Identity Encoding has a shape of $16 * H * W$. 3D Identity Encoding and 2D Identity Encoding share the same identity classification linear layer with $(16, 256)$ input and output channels.

During training, we set $\lambda_{2d} = 1.0$ and $\lambda_{3d} = 2.0$. We use the Adam optimizer for both Gaussians and the linear layer, with a learning rate of 0.0025 for identity encoding and 0.0005 for the linear layer. For 3D regularization loss, we set the nearest neighboring number to $k = 5$, and the sampling points number to $m = 1000$. To improve efficiency and avoid calculating loss at boundary points, we downsample the point cloud to 300K to calculate the loss. 3D regularization loss only affects the segmentation of identity encoding and does not affect the density of the Gaussians. We use the same adaptive density control as Gaussian Splatting. All datasets are trained for 30K iterations on one A100 GPU.

4.2 Algorithm Pseudocode

We outline the pseudocode for our Gaussian Grouping in Algorithm 1, where we highlight the introduced core components in both red and bold texts. This idea of our Gaussian Grouping is simple to implement, and straightforward but produces an effective 3D representation to efficiently support versatile downstream scene editing tasks.

4.3 Limit Analysis

Our Gaussian Grouping segments “anything masks” with the assistance of SAM. But the “anything mask labels" by original SAM [8] have no direct semantic language information. We adopt the Grounding-DINO [10] for open vocabulary segmentation to pick the 2D object, and match our anything masks rendering. When some language prompts are very complicated, the Grounding-DINO can not acquire the correct mask from the input text prompt and will give a wrong mask prediction. In this case, even if we provide the correct mask in anything mask rendering, we do not obtain explicit category information. Also, the zero-shot 2D association accuracy of DEVA [3] will also limit the open-world 3D segmentation performance of Gaussian Grouping. This can be solved by further improvement of the vision language detection model and better association schemes in the future.

Algorithm 1 *Gaussian Grouping*

$p \leftarrow$ SfM Points ▷ 3D Positions
$m = (m_1, m_2, \ldots, m_K) \leftarrow$ SAM ▷ **SAM's Masks** at Various K Views
$(\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_K) \leftarrow$ Zero-shot Tracking(m) ▷ Multi-view **Associated Masks**
$s, \alpha, c, \mathbf{e} \leftarrow$ InitAttributes() ▷ Covariances, Opacities, Colors, **Identity Encodings**
$i \leftarrow 0$ ▷ Iteration Count
while not converged **do**
  $V, \hat{I}, \hat{M} \leftarrow$ SampleTrainingView() ▷ Camera View V, Image and **Mask**
  $I, \mathbf{E}_{\text{id}} \leftarrow$ Rasterize(p, s, a, c, $\mathbf{e}$,V) ▷ Rendered Image and **Identity Encoding**
  $\mathcal{L}_{\text{image}} \leftarrow \mathcal{L}(I, \hat{I})$ ▷ Original Image Rendering Loss
  $L_{\text{id}} \leftarrow \lambda_{2d}\mathcal{L}_{\text{2d}}(\mathbf{E}_{\text{id}}, \hat{M}) + \lambda_{3d}\mathcal{L}_{\text{3d}}(\mathbf{e})$ ▷ **Identity Grouping Loss**, Eq. **??**
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{image}} + \mathcal{L}_{\text{id}}$ ▷ Total Loss
  p, s, a, c, $\mathbf{e} \leftarrow$ Adam($\nabla\mathcal{L}$) ▷ Backprop & Step
  if IsRefinementIteration(i) **then**
    for all J Gaussians $(p_j, s_j, \alpha_j, c_j, \mathbf{e}_j)$ in $(p, s, a, c, \mathbf{e})$ **do**
      if $\alpha < \epsilon$ or IsTooLarge(p_j, s_j) **then** ▷ Pruning
        RemoveGaussian()
      end if
      if $\nabla_p L > \tau_p$ **then** ▷ Densification
        if $\|S\| > \tau_S$ **then** ▷ Over-reconstruction
          SplitGaussian$(p_j, s_j, \alpha_j, c_j, e_j)$
        else ▷ Under-reconstruction
          CloneGaussian$(p_j, s_j, \alpha_j, c_j, e_j)$
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while

References

1. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. In: CVPR (2023)
2. Cen, J., Zhou, Z., Fang, J., Yang, C., Shen, W., Xie, L., Zhang, X., Tian, Q.: Segment anything in 3d with nerfs. In: NeurIPS (2023)
3. Cheng, H.K., Oh, S.W., Price, B., Schwing, A., Lee, J.Y.: Tracking anything with decoupled video segmentation. In: ICCV (2023)
4. Dwyer, B., N.J.S.J.e.a.: Roboflow (version 1.0) [software]. https://roboflow.com. (2022)
5. Haque, A., Tancik, M., Efros, A., Holynski, A., Kanazawa, A.: Instruct-nerf2nerf: Editing 3d scenes with instructions. In: ICCV (2023)
6. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM TOG **42**(4), 1–14 (2023)
7. Kerr, J., Kim, C.M., Goldberg, K., Kanazawa, A., Tancik, M.: Lerf: Language embedded radiance fields. In: ICCV (2023)
8. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. In: ICCV (2023)
9. Kobayashi, S., Matsumoto, E., Sitzmann, V.: Decomposing nerf for editing via feature field distillation. In: NeurIPS (2022)
10. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. arXiv preprint arXiv:2303.05499 (2023)
11. Mirzaei, A., Aumentado-Armstrong, T., Derpanis, K.G., Kelly, J., Brubaker, M.A., Gilitschenski, I., Levinshtein, A.: Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In: CVPR (2023)
12. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. In: WACV (2022)