Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Supplementary Material

This supplement is organized as follows:

- Section A contains network architecture details;
- Section B contains more details on the training and inference settings;
- Section C contains the details of the teacher features from 2D foundation models;
- Section D contains more details of Replica dataset experiment;
- Section E contains the algorithmic details of language-guided editing;
- Section F contains ablation studies of our method;
- Section G contains failure cases of complex scenes and reasoning analysis.

A. Details of Architectures

Parallel N-Dimensional Gaussian Rasterizer The parallel N-dimensional rasterizer maintains an architecture akin to the original 3DGS rasterizer. Moreover, it employs a point-based α-blending technique for rasterizing the feature map. To mitigate the issue of inconsistent spatial resolution inherent in tile-based rasterization, we ensure that both the RGB image and the feature map are rendered at matching sizes. Additionally, the parallel N-dimensional rasterizer is adaptable to various foundational models, implying that its dimensions are flexible and can vary accordingly. The detail of the Parallel N-Dimensional Gaussian Rasterization is in Algorithm 1.

Speed-up Module The primary objective of our Speed-up Module is to modify the feature map channels, enabling the relatively low-dimensional semantic features rendered from 3D Gaussians to align with the high-dimensional ground truth 2D feature map. To facilitate this, we employ a convolutional layer equipped with a 1×1 kernel, offering a direct and efficient solution. Given that we already possess the ground truth feature map from the teacher network, which serves as a target for the rendered feature map approximation, there is no necessity for a complex CNN architecture. This approach simplifies the process, ensuring effective feature alignment without the need for intricate feature extraction mechanisms. More experimental results regarding performance of the Speed-up Module are included in Sec. F.

Algorithm 1 Parallel N-Dimensional Gaussian Rasterization

$PointCloud \leftarrow$ Structure from Motion ▷ Point Cloud
$X, C \leftarrow PointCloud$ ▷ Position, Colors
$\Sigma, A, F \leftarrow InitAttributes()$ ▷ Covariances, Opacities, Semantic Features
$F_t(I) \leftarrow I$ applying Foundation Model ▷ Feature Map
$i \leftarrow 0$ ▷ Iteration Counter
repeat
  $V, I, F_t \leftarrow$ GetTrainingView() ▷ Camera Pose, Image, Feature Map
  $\hat{I}, F_s \leftarrow$ ParallelRasterizer(X, C, Σ, A, F, V) ▷ Rasterization
  $L \leftarrow$ Loss$(I, \hat{I}) + \lambda$Loss(F_t, F_s) ▷ Loss Calculation
  $X, \Sigma, C, A, F \leftarrow$ Adam(L) ▷ Backpropagation and Step
  if IsRefinementStep(i) **then**
    for Gaussians(x, q, c, α, f) **do**
      if $\alpha < \varepsilon$ or IsTooLarge(x, q) **then**
        $RemoveGaussian()$
      end if
      if $\nabla_p L > \tau_p$ **then**
        if $\|S\| > \tau_S$ **then**
          $SplitGaussian(x, q, c, \alpha, f)$ ▷ Over-reconstruction
        else
          $CloneGaussian(x, q, c, \alpha, f)$ ▷ Under-reconstruction
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$ ▷ Counter Increment
until Convergence

B. Training and Inference Details

For the training and inference pipeline, one option is to directly render a feature map with the dimension same as the ground-truth feature (512 for LSeg encoding and 256 for SAM encoding). Since rendering with such large dimension slows down the training, another option is to use our speed-up module: rendering a lower-dimensional feature map, which is later upsampled to the ground-truth feature dimension by a lightweight convolutional decoder. Similar to 3DGS [4], we use Adam optimizer for for optimization during training and use a standard exponential decay scheduling similar to [2]. For image rendering, we mainly follow the 3DGS optimization strategy by using a 4 times lower image resolution and upsampling twice after 250 and

Dimension	8	16	32	64	128	256	512
Time	6:40	7:21	8:51	12:10	19:55	48:39	1:29:42
mIoU↑	0.354	0.493	0.709	0.774	0.783	**0.791**	0.790
Accuracy↑	0.735	0.880	0.927	0.939	**0.944**	**0.944**	0.943

Table A. **Evaluation of Semantic Segmentation Performance Across Different Dimensions.** This table presents the Time, mIoU, and Accuracy corresponding to each dimension level with LSeg feature.

Dimension	8	16	32	64	128	256	512
PSNR↑	36.8879	36.8871	36.9671	36.9397	**37.012**	36.9474	36.9150
SSIM↑	0.9699	0.9703	0.9706	**0.9708**	0.9706	0.9704	0.9703
LPIPS↓	0.0234	0.0230	**0.0226**	0.0229	0.0228	0.0230	0.0236

Table B. **Evaluation of Image Quality Metrics Across Different Dimensions of Lseg feature.** This table presents the PSNR, SSIM, and LPIPS values corresponding to each dimension level with LSeg feature.

500 iterations. For feature rendering, we use Adam optimizer with a learning rate of $1e-3$. For the feature decoder network in the additional Speed-up Module, we use a separate Adam optimizer with a learning rate of $1e-4$.

C. Teacher Features

LSeg Feature For LSeg, we use CLIP ViT-L/16 image encoder for ground-truth feature preparation and and ViT-L/16 text encoder for text encoding. The ground truth feature from the LSeg image encoder has feature size 360×480 with feature dimension 512. One can either choose to directly render a $h \times w$ feature with dimension 512 or use the Speed-up Module by rendering a lower-dimensional feature which is later upsampled back. In practice, we use rendered feature $dim = 128$ for Sec. 4.1 in our main paper.

To predict the semantic segmentation mask during inference, we reshape the rendered feature with shape (512, 360, 480) to (360 × 480, 512), referred as the image feature. The text feature from the CLIP text encoder has shape $(C, 512)$ where C is the number of categories. We then apply matrix multiplication between the two to align pixel-level features and a text query feature and perform semantic segmentation using LSeg spatial regularization blocks.

SAM Feature Following the image encoding details in SAM [5], we use an MAE [3] pre-trained ViT-H/16 [1] with 14×14 windowed attention and four equally-spaced global attention blocks. The SAM encoder first obtains the image resolution of 1024×1024 by resizing the image and padding the shorter side. The resolution is then $16\times$ downscaled to 64×64. Since only a portion of the 64×64 feature map contains semantic information due to the padding operation, we crop out one side of the feature map corresponding to the longer side of the original image. Specifically, suppose the original image has the resolution of $H \times W$ where $W > H$, we crop the 64×64 feature map from SAM encoder so that the new feature resolution becomes $64W/H\times64$ with the feature dimension of 256 corresponding to the output dimension of the SAM encoder. In practice, we use the Speed-up Module with the rendered feature $dim = 128$.

To obtain the results of promptable or promptless segmentation during the inference, we perform the padding operation on the rendered feature to convert from $64W/H\times64$ back to 64×64 so that the SAM decoder receive the equivalent semantic information as from the original SAM encoder.

Feature visualization As shown in Fig. D, similar to [9], we use `sklearn.decomposition.PCA` [8] in scikit-learn package for feature visualization. We set the number of PCA components to 3 corresponding to RGB channels and calculate the PCA mean by sampling every third element along $h \times w$ vectors, each with feature dimension of either 512 (for LSeg) or 256 (for SAM). The feature map is transformed using the PCA components and mean. This involves centering the features with PCA mean and then projecting them onto PCA components. We then normalize the transformed feature based on the minimum and maximum values with the outliers removed to standardize the feature values into a consistent range so that it can be effectively visualized, typically as an image. We visualize both LSeg and SAM features of scenes from the LLFF dataset [7] from different views. The feature map from LSeg encoder has size 360×480 with dimension 512 (see in the second column of Fig. D). However, as mentioned before, the feature map directly obtained from SAM encoder contains a padding re-

Dimension	8	16	32	64	128	256
PSNR↑	36.7061	36.9145	36.8793	**36.9208**	36.7815	36.9139
SSIM↑	0.9697	0.9706	0.9700	0.9701	0.9700	**0.9709**
LPIPS↓	0.0234	0.0229	0.0229	**0.0228**	**0.0228**	0.0230
FPS↑	64.7	53.9	52.7	32.8	24.2	8.3

Table C. **Evaluation of Image Quality Metrics Across Different Dimensions of SAM feature.** This table presents the PSNR, SSIM, LPIPS, and FPS values corresponding to each dimension level with SAM feature.

gion (see the red areas on the bottom of the feature maps in the third column), we crop out the region on the feature map as the ground-truth feature before feature distillation (see in the last column). It is worth noting that features from LSeg models mainly capture semantic information by delineating coarse-grained boundaries, while features from SAM models show instance-level information and even fine-grained details in different parts of an object. The capability of teacher encoders determines characteristics of the feature map, thereby influencing the upper limit of the performance of the rendered features on downstream tasks.

D. Replica Dataset Experiment

Following the same selection in [9], we experiment on 4 scenes from the Replica dataset [10]: room_0, room_1, office_3, and office_4. For each scene, 80 images are captured along a randomly chosen trajectory, and every 8th image starting from the third is selected. We trained 5,000 iterations on each scene with LSeg serving as the foundational model for this experiment. We manually re-label some pixels with semantically close labels such as "rugs" and "floor". This preprocess step follows the same method in the NeRF-DFF [6]. The model is trained on the training images and was subsequently evaluated on a set of 10 test images. We test pixel-wise mean intersection-over-union and accuracy on the manually relabeled test images and we use $class = 7$ for the mIoU metric. For room_1, the last 2 test images are excluded from the results since these images do not have 7 classes in the image.

E. Editing Algorithm and Details

The editing procedure takes advantage of the 3D Gaussians so that the model is able to render a novel view image edited with a specific editing operation. As illustrated in Fig. E, starting from a set of 3D Gaussians, i.e. $\mathcal{X} = \{x_1, \ldots, x_N\}$ where each x_i is a 3D Gaussian represented by (f_i, α_i, c_i) where $f_i \in \mathbb{R}^{512}$, $\alpha_i \in \mathbb{R}$ and $c_i \in \mathbb{R}^3$ are the semantic feature, color and opacity, respectively. Guided by language, the edit algorithm takes a input text which is a list of object categories, e.g. 'apple, banana, others'. We leverage the CLIP's ViT-B/32 text encoder for text encoding to obtain the text feature $\{t_1, \ldots, t_C\}$ where $t_i \in \mathbb{R}^{512}$ and C is the number of categories. We then calculate the inner product of the text feature and semantic feature followed by a softmax function to obtain the semantic scores for each 3D Gaussian, represented by a C-dimensional vector, i.e. $scores \in \mathbb{R}^{N \times C}$. Queried by the text label l, e.g. 'apple' or a list of objects to be edited, e.g. 'apple, banana', one can either choose to apply hard selection or soft selection to perform edit operation specifically on the target region:

Soft selection: Based on the category selected by the query label $l \in \{1, 2, \ldots, C\}$ (or $l \subseteq \{1, 2, \ldots, C\}$ if l is a list of catogories), we target on the corresponding column of the score matrix, i.e. $score_l = [s_{1l}, s_{2l}, \ldots, s_{Nl}]^\top$ and apply binary thresholding on this column score vector, i.e. for any i such that $s_{il} \geq th$, we set the position i to 1 representing being selected; otherwise the position i is set to 0 representing not being selected. Then all the positions i such that $s_{li} = 1$ compose a target region to be edited. Intuitively, we mask out all the 3D Gaussians that are not selected and use those selected 3D Gaussians to update the color set $\{c_i\}_{i=1}^N$ and opacity $\{\alpha_i\}_{i=1}^N$.

Hard selection: We apply $argmax$ function to the score matrix to select the category corresponding to the highest score for each Gaussian to obtain a filtered category vector $categories = [c_1, c_2, \ldots, c_N]^\top$ where $c_i = \text{argmax}\{s_{i1}, s_{i2}, \ldots, s_{iC}\}$. Then we filter based on the query label l: for any i such that $c_i = l$ (or $c_i \in l$ if l represents a list of target objects), we set the position i to 1 representing being selected; otherwise the position i is set to 0 representing not being selected. Similarly, we select the target region to be edited by preserving only the region positions of which the highest score category is aligned with the query label.

Hybrid selection: Since soft selection applies thresholding only based on column vector of score matrix corresponding to label l which may potentially cause incorrect selection when the dominant score exists in other columns while hard selection merely selects the highest score without any tunable threshold value. Therefore, we propose a hybrid selection method by combining both hard and soft selection to alleviate the effect of incorrectly selecting the category while making the selection tunable to adapt to different scenarios. Spcifically, we combine the selection

Gaussian masks of the two methods and apply bitwise OR operation between the two masks to obtain the final selected region.

We then update the opacity and color based on the selected edit region and a specific edit operation. We demonstrate the details of three examples: extraction, deletion and appearance modification:

(a) Extraction: for any $i \in \{1, \ldots, N\}$, if i is selected, the opacity remains to be α_i; otherwise the opacity is set to 0.

(b) Deletion: for any $i \in \{1, \ldots, N\}$, if i is selected, the opacity is set to 0; otherwise the opacity remains to be α_i. Specifically for deletion operation, we apply the hybrid selection method to select the target edit region to reduce the effect of incomplete deletion caused by the absence of target pixels.

(c) Appearance modification: for any $i \in \{1, \ldots, N\}$, if i is selected, the color c is updated to be $appearance\_func(c_i)$ where $appearance\_func(.)$ represents any appearance modification function, such as changing the green leaves to red leaves, etc.

F. Ablation Studies

We study the effect of different rendered feature dimensions using our Speed-up Module. In Tab. A, we report the performance of the semantic segmentation on Replica dataset using LSeg feature. The result shows that both rendered feature $dim = 256$ and $dim = 128$ can achieve the best performance on accuracy, and $dim = 256$ is slightly better on mIoU. However, $dim = 128$ is $\times 2.4$ faster than $dim = 256$ on training. We also report the quantitative results of novel view synthesis in Tab. B, which shows that $dim = 128$ is the best. Therefore, we choose $dim = 128$ for our Replica dataset experiment in practice. In addition, we show the performance and speed (FPS) of novel view synthesis with different dimensions of SAM feature in Tab. C.

Furthermore, Fig. A and Fig. B substantiate that our Speed-up Module not only avoids compromising performance but, in fact, resulting in time savings.

G. Failure Cases

The proposed method indeed has limitations reflected on some failure cases. In Fig. C, we showcase failure cases for scenes that are more challenging and complex. In Fig. C (a), the point-prompted segmentation mask is not perfect with a coarse boundary and small holes. This is caused by low feature quality from SAM distillation, rather than Gaussian representation. Since the boundary of the car is hard to depicted and there are multiple similar objects close to each other (multiple adjacent cars), making the scene complex. As a result, achieving a smooth and accurate mask boundary of the car becomes challenging, which could be counted as a limitation. In Fig. C (b), given the text prompt "Delete the cup", although succeeding in locating the target object, the model fails to remove the cup comprehensively. The reason behind is that in some complex scenes including various objects with multiple sizes, the 3D Gaussians corresponding to the tiny objects with sophisticated details are hard to accurately selected by the "Gaussian mask". As a result, a clean deletion is hard to perform on target object.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[2] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 1

[3] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 2

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

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 2

[6] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *NeurIPS*, 2022. 3

[7] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2, 6

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 2

[9] Karl Stelzner, Kristian Kersting, and Adam R Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv preprint arXiv:2104.01148*, 2021. 2, 3

[10] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The replica dataset: A digital replica of indoor spaces, 2019. 3

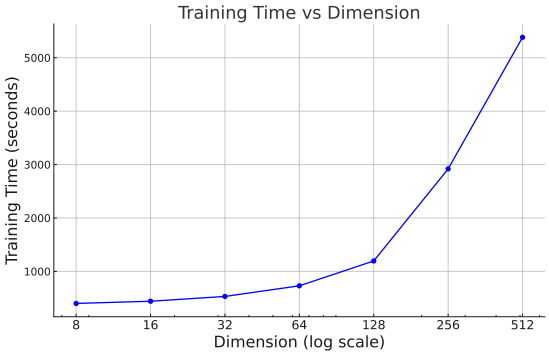


Figure A. **Training Time vs Speed-up Module Dimension** We test the training time required with different input dimension of speed-up module. In this Figure, we show that the training time can be significantly reduced with our speed-up module.

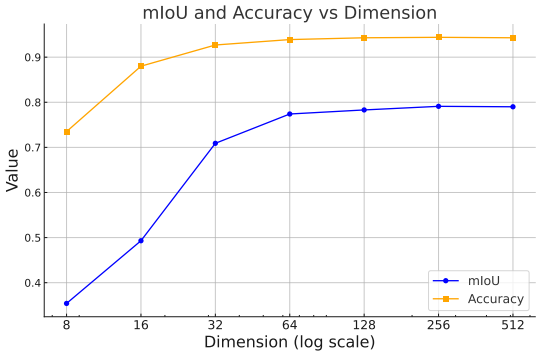


Figure B. **mIoU and Accuracy vs Dimension** In this graph, we show 2D metrics with respect to different input dimensions of speed-up module. With our speed-up module and proper input dimension, the 2D metrics are not compromised.

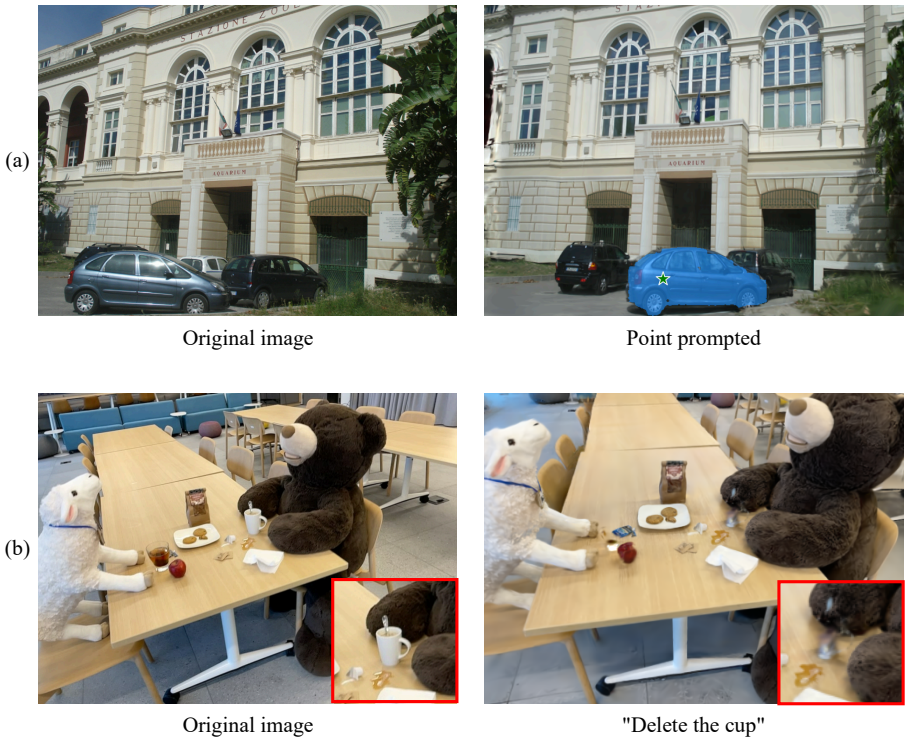
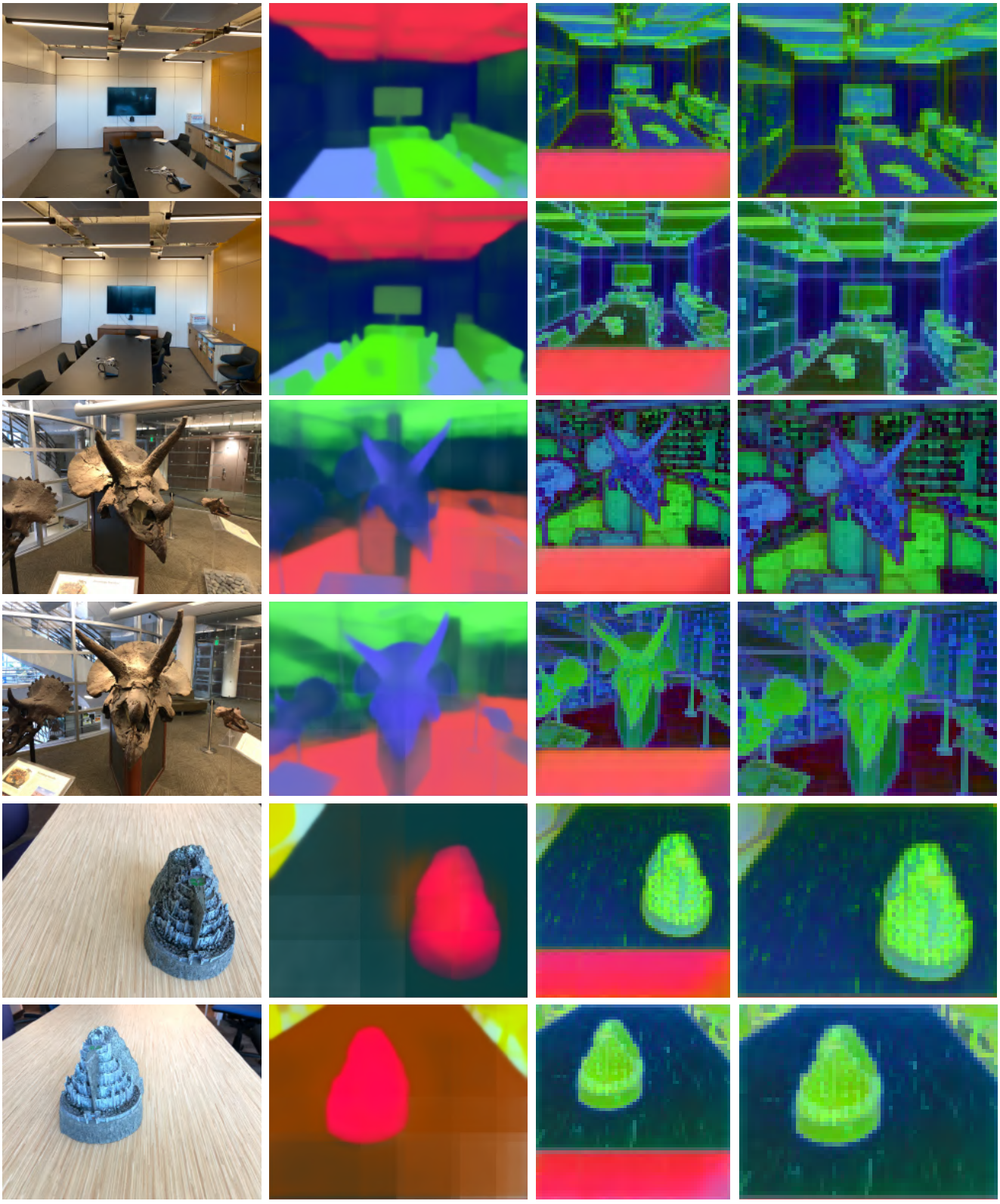


Figure C. **Failure cases in complex and challenging situations** (a) The point-prompted segmentation mask, contains flaws in the form of a coarse boundary and small holes, resulting from low-quality features. (b) The model fails to delete tiny sophisticated objects thoroughly in a complex scene in language-guided editing.



(a) Original image (b) LSeg feature (c) SAM feature (d) SAM cropped feature

Figure D. **Feature visualization on different scenes from LLFF dataset [7] from LSeg and SAM encoders**. Note that SAM features in column (d) is obtained by cropping the padding region. We resize the cropped feature for better visualization.

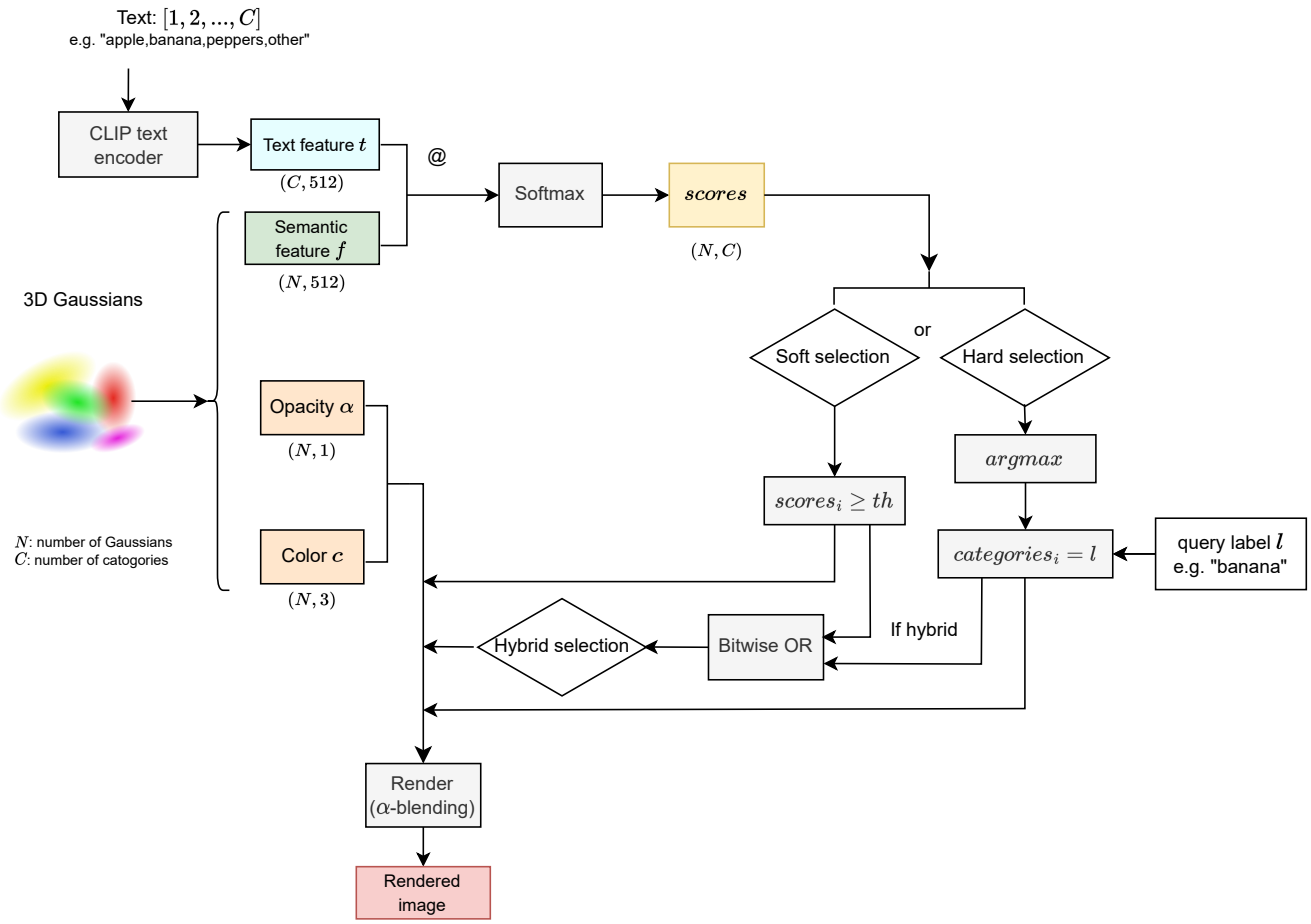


Figure E. **Language-guided editing procedure using 3D Gaussians.** We calculate the inner product between the semantic feature and the text feature from CLIP encoder followed by a softmax to obtain a score matrix and query the feature field to apply editing on target regions (obtained from soft selection / hard selection / hybrid selection) by updating opacity and color from 3D Gaussians before rendering.