

SuperGSeg: Open-Vocabulary 3D Segmentation with Structured Super-Gaussians

Supplementary Material

This supplementary document provides additional details about our method. Section A elaborates on the design of the Super-Gaussian and demonstrates its application to downstream tasks. Section B provides further implementation details, including the MLP architectures for neural Gaussian feature decoding and the adaptation of Open-Gaussian for 2D open-vocabulary semantic segmentation comparison. Section C reports detailed efficiency analysis, including training time, inference speed, and memory consumption. Section D reports extended ablation studies on SuperG hyperparameters and module variants. Section E presents additional quantitative and qualitative results. Lastly, Section F discusses the limitations of our approach and outlines potential directions for future work.

A. Super-Gaussian Details

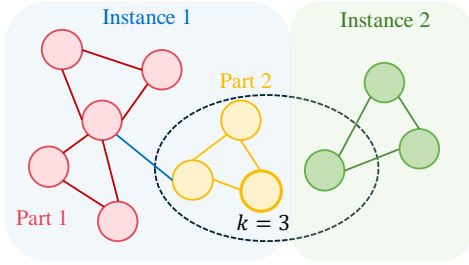


Figure 8. Example of Super-Gaussian graph. Each node represents a SuperG, connected to its k -nearest neighbors based on similarity in instance features. Through connected component analysis, the SuperG nodes are divided into two distinct instances. Within Instance 1, SuperG nodes are interconnected by similar hierarchical features, further splitting into two parts.

Module Design. As shown in Figure 3, our SuperG clustering network consists of four learnable MLPs. Inspired by SPNet [19], we design three attribute-specific learnable functions F_ϕ , F_φ , and F_ψ , each implemented as a single-hidden-layer MLP with ReLU activation. These functions independently encode the differences between an anchor and its k -nearest SuperGs in coordinates, segmentation features, and geometry, respectively, producing embeddings that reflect the relevance of each attribute for SuperG assignment. A final MLP, F_{sg} , then concatenates these embeddings and integrates spatial, semantic, and geometric cues into a probabilistic assignment.

Grouping Super-Gaussians for Instance and Hierarchical Segmentation. After training the SuperG association modules, we obtain a soft association map $\mathbf{A} \in \mathbb{R}^{N' \times k}$. During inference, each anchor point is assigned to one of its k -nearest neighbors with the highest probability, leading to a hard SuperG assignment $\bar{\mathbf{A}} \in \mathbb{R}^{N' \times 1}$. The attributes of each SuperG are then computed by averaging the attributes of its assigned anchors.

These SuperGs serve as the fundamental units for representing and interpreting the 3D scene. Specifically, as shown in Figure 8, we further construct a graph where nodes correspond to SuperGs. For instance segmentation, a node is connected to nodes within its k -nearest neighbors if their instance feature similarity exceeds a threshold τ_{Ins} . Instances are then obtained via connected component analysis on this restricted graph. Similarly, part segmentation is achieved by building a SuperG graph within each instance and identifying connected components based on hierarchical feature similarity with threshold τ_{Hier} . In practice, we set $k = 3$, $\tau_{Ins} = 0.8$, and $\tau_{Hier} = 0.9$.

B. Additional Implementation Details

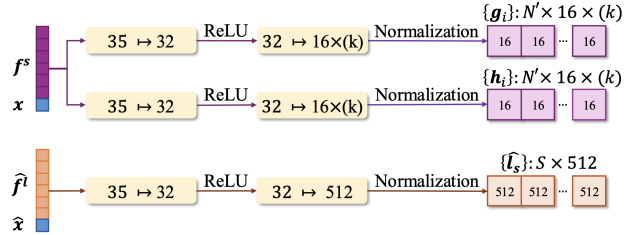


Figure 9. MLP structures for decoding different features.

Decoding Neural Gaussians from MLPs. We employ MLPs to decode latent features, as shown in Figure 9. Each MLP contains a single hidden layer of dimension 32. Their decoding targets, however, differ: instance feature decoder F_I and hierarchical feature decoder F_H decode anchor-level features, while language feature decoder F_L decodes SuperG-level features. Specifically, F_I and F_H take the anchor segmentation feature f^s and anchor position x as input, and predict the instance feature g and hierarchical feature h of the neural Gaussians spawned per anchor. In contrast, F_L predicts the CLIP-aligned feature \hat{l} for each SuperG, conditioned on its latent language feature \hat{f}^l and its center \hat{x} .

Additional Information on the Contrastive Losses. The contrastive instance feature loss \mathcal{L}_{Ins} , which is computed on the set of SAM-generated instance-level masks \mathcal{M} and the rendered 2D instance feature map \hat{G} , has been discussed in Section 3.2. Inspired by [18], we elaborate a similar yet more complicated contrastive hierarchical feature loss \mathcal{L}_{Hier} . This loss is defined on the part-level patches $\mathcal{P} = \{\mathbf{p}^p \in \mathbb{R}^{H \times W} \mid p = 1, \dots, |\mathcal{P}|\}$ and the rendered 2D hierarchical feature map $\hat{H} \in \mathbb{R}^{D_h \times H \times W}$.

Given a patch \mathbf{p}^p , we collect the rendered hierarchical features from \hat{H} at each pixel, forming a set of feature vectors \mathbf{h}^p . The mean feature of the patch is then defined as $\bar{\mathbf{h}}^p$. We define the contrastive hierarchical feature loss at the minimum unit, on a pixel t with feature $\mathbf{h}_t^p \in \mathbf{h}^p$, as:

$$\mathcal{L}^{p,t}(r) = -\log \frac{\exp(\mathbf{h}_t^p \cdot \bar{\mathbf{h}}^r / \tau_r)}{\sum_{q=1}^{|\mathcal{P}|} \exp(\mathbf{h}_t^p \cdot \bar{\mathbf{h}}^q / \tau_q)}, \quad (8)$$

where τ is the temperature of the contrastive loss, and p, r, q are indices of patches. Subsequently, the hierarchical feature loss [18] can be written as:

$$\mathcal{L}_{Hier} = \sum_{p=1}^{|\mathcal{P}|} \sum_{d=1}^{d_{\max}^p} \mathcal{L}_{p,d}, \quad (9)$$

$$\mathcal{L}_{p,d} = \frac{\lambda^{d-1}}{|\mathcal{R}_d^p|} \sum_{t=1}^{|\{\mathbf{h}^p\}|} \sum_{r \in \mathcal{R}_d^p} \max(\mathcal{L}^{p,t}(r), \mathcal{L}_{\max}^{p,t}(d-1)), \quad (10)$$

where λ^{d-1} is a hyperparameter, \mathcal{R}_d^p denotes the index set of patches at hierarchy level d of patch p , and $r \in \mathcal{R}_d^p$ refers to a patch at level d . The maximum loss at level d ensures that the contrastive loss between the pixel feature t and patches with higher correlation (lower d) is always smaller than for patches with lower correlation:

$$\mathcal{L}_{\max}^{p,t}(d) = \max_{r \in \mathcal{R}_d^p} \mathcal{L}^{p,t}(r). \quad (11)$$

Additional Technical Details. We use the SAM ViT-H model [21] to generate 2D masks from the input images and then extract language features for each instance mask using the OpenCLIP ViT-B/16 model following [7]. The training process is divided into three stages. In the first stage, we train the Scaffold-GS [36] with instance and hierarchical feature attributes for 30k iterations. In the second stage, we freeze the geometry and multi-granularity features network from stage one and train only the SuperG clustering network for another 30k iterations. Finally, in the last stage, we freeze all other parameters and optimize the language features for each SuperG for 10k iterations. The embedding dimensions for \mathbf{f}^g and \mathbf{f}^s are set to 32 [36], while instance and hierarchical features are 16-dimensional [18]. For optimization, we use the Adam [45] optimizer for the MLPs with an initial learning rate of 0.01 and an exponential an-

nealing schedule of 0.001 as in [46].

OpenGaussian Implementation. OpenGaussian [11] assigns language features to instance-level Gaussians, enabling direct language queries on 3D point clouds. However, this approach does not natively support 2D pixel-level semantic segmentation, making direct evaluation on ScanNet more challenging. To enable a fair comparison, we first identify category-relevant 3D Gaussians by iterating over all text prompts to predict language feature maps for open-vocabulary semantic segmentation. For each instance-level Gaussian cluster, we determine the corresponding text prompt ID and store these IDs in a label map, which is then used to generate the final semantic segmentation. By following this approach, occlusions at the instance-level Gaussians are not explicitly handled, leading to the occlusion artifacts observed in Figure 4.

C. Training and Inference Efficiency

In Table 5, we report training time, inference time and memory consumption for LangSplat [7] and OpenGaussian [11] on the LERF-OVS dataset.

For LangSplat [7], the training consists of two stages: in S1, the 3DGS is pretrained without any additional feature fields for 30k iterations. In S2, the pretrained 3DGS is frozen, and a language feature field is optimized for another 30k iterations. For OpenGaussian [11], S1 corresponds to pretraining the 3DGS jointly with an instance feature field for 40k iterations. In S2, Gaussian clustering is performed in a coarse-to-fine manner, requiring an additional 30k iterations. Finally, in S3, the 2D language features are directly associated with the 3D Gaussian clusters.

When comparing our method to LangSplat and OpenGaussian across different training stages S1, S2, and S3, we find that our approach, while requiring longer training times in S1 and S2, achieves comparable efficiency in S3, indicating a trade-off due to the joint learning of instance and hierarchical features. GPU memory usage varies, with our method consuming more resources in S1 and S2 but significantly less in S3, showcasing the compact memory footprint enabled by our proposed SuperGs. Conversely, LangSplat exhibits consistent memory efficiency in S1 and S2, while OpenGaussian’s memory demands vary across different training stages. Most notably, our method consistently outperforms both baselines in inference speed across all scenarios, a critical advantage for real-time applications. This blend of rapid inference and flexible resource utilization highlights our method’s robustness for practical deployment in diverse computer vision tasks. In addition, our method supports multi-granularity scene understanding. Specifically, it enables semantic-level queries to retrieve groups of objects sharing the same language description, instance-level segmentation of a specific object,

Ablation	mean		<i>figurines</i>		<i>teatime</i>		<i>ramen</i>		<i>waldo_kitchen</i>	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
$S = 250$	25.75	39.42	22.70	35.71	35.93	50.85	15.44	21.13	28.92	50.00
$S = 500$	34.05	50.03	27.44	48.21	57.42	77.97	20.42	23.94	30.93	50.00
$S = 1000$	35.94	52.02	43.68	60.71	55.31	77.97	18.07	23.94	26.71	45.45
$S = 2000$	27.61	40.60	27.64	46.43	47.58	62.71	12.46	16.90	22.76	36.36
$k = 3$	35.94	52.02	43.68	60.71	55.31	77.97	18.07	23.94	26.71	45.45
$k = 5$	33.70	46.76	21.59	35.71	68.75	84.75	16.96	21.13	27.48	45.45
$k = 10$	33.81	46.88	43.56	62.50	41.56	55.93	21.82	28.17	28.31	40.91

Table 4. Additional ablation studies on the LERF-OVS dataset [7] about the parameter choices for the SuperG Clustering Network. We use $s = 1000$ SuperGs and $k = 3$ for k -nearest neighbor in our implementation by default.

	LangSplat		OpenGaussian			Ours		
	S1	S2	S1	S2	S3	S1	S2	S3
Train	20m	45m	50m	20m	10m	90m	85m	30m
Memory	8G	6G	14G	17G	22G	18G	14G	4G
Inference	3.28s		5.55s			0.56s		
Memory	18GB		9GB			4GB		

Table 5. Comparison of the time and GPU memory requirements during training (top rows) and inference (bottom rows).

and further decomposition of this instance into fine-grained parts. In contrast, LangSplat [7] only supports semantic-level queries. OpenGaussian [11] extends to instance-level understanding but, similar to LangSplat, does not support finer-grained part segmentation. This demonstrates that our method provides a more comprehensive representation for 3D scene understanding.

D. Additional Ablation Studies

In Section 4, we perform ablation studies to evaluate the necessity of SuperGs and analyze the performance of the instance feature field and hierarchical feature field. In this section, we further investigate the necessity of our proposed SuperG clustering network and measure the impact of its individual components.

Super-Gaussian Clustering Approaches. Given a pre-trained scene, our objective is to group anchors into meaningful SuperGs using their coordinates, segmentation features, and geometric properties. We explore two alternative approaches. First, we evaluate a simple K -means clustering algorithm by concatenating the aforementioned attributes and clustering them into $k = 1000$ SuperGs. Second, we experiment with a traditional supervoxel generation approach. Specifically, we map the anchors to a point cloud, using the concatenated features as normals. We then apply the Voxel Cloud Connectivity Segmentation (VCCS) algorithm [32] to compute the SuperGs. Finally, we compare

these two non-learning-based approaches with our learning-based method for Anchor-to-SuperG association.

We observed that K -means fails to prevent the overlap of resulting SuperGs across instances. Meanwhile, VCCS [32], originally designed for dense point clouds, struggles with the sparse structure of Gaussians. Its region-growing mechanism incorrectly clusters a large number of anchors together, which hinders the learning of the language feature field. The results in Table 6 show that our method is better suited for grouping Gaussians, achieving better performance.

Method	mIoU \uparrow	mAcc. \uparrow
K -means	53.77	67.80
VCCS[32]	0.45	0.00
Ours	55.31	77.97

Table 6. Ablation study of SuperG clustering approaches on the *teatime* scene of LERF-OVS.

w/ F_ϕ	w/ F_φ	w/ F_ψ	mIoU \uparrow	mAcc. \uparrow
✓			32.41	40.68
			48.29	67.80
	✓		58.07	75.66
		✓	37.12	62.71
✓	✓	✓	55.31	77.97

Table 7. Ablation study on the SuperG clustering network and its components on the *teatime* scene of LERF-OVS.

Super-Gaussian Clustering Network. As introduced in Section A, we employ three MLPs F_ϕ , F_φ , and F_ψ to capture the coordinate, segmentation, and geometric relationships between anchors and their k -nearest neighbors. To evaluate the contributions of these MLPs, we conduct an ablation study. Notably, in experiments where none of these MLPs are used, we directly concatenate the differences of

Method	mIoU mean	mAcc	mIoU wall	mAcc	mIoU floor	mAcc	mIoU cabinet	mAcc	mIoU table	mAcc	mIoU desk	mAcc	mIoU curtain	mAcc
LERF [23]	38.5	60.4	35.2	82.8	60.1	68.8	52.0	82.7	10.2	80.1	14.4	16.1	70.2	77.8
LEGaussians [8]	8.7	33.2	17.9	53.1	14.6	20.6	2.7	18.6	0.0	0.0	0.5	13.5	1.9	10.4
OpenGaussian [11]	24.1	68.7	13.4	96.6	31.2	74.4	0.3	22.9	0.1	1.0	30.6	35.6	17.7	79.2
LangSplat [7]	27.6	48.3	45.3	72.6	43.3	45.6	24.8	56.7	21.9	87.4	0.1	6.4	46.8	66.5
SuperGSeg [ours]	54.7	74.7	58.8	92.9	53.6	86.5	69.8	83.8	35.7	54.8	15.0	16.7	61.8	64.5
	toilet		counter		refrigerator		chair		sink		window		door	
LERF [23]	25.2	25.2	24.4	42.8	69.9	90.2	10.9	10.9	25.8	37.1	11.5	11.5	64.5	67.5
LEGaussians [8]	13.7	16.3	10.7	27.0	9.0	74.3	0.4	28.7	0.3	0.4	0.0	44.4	1.4	4.7
OpenGaussian [11]	73.0	98.4	3.0	9.3	88.0	98.3	36.5	83.4	3.0	3.7	75.0	88.8	75.4	97.0
LangSplat [7]	0.1	5.4	10.7	34.7	0.7	33.3	18.0	48.5	0.0	0.0	0.0	0.1	55.6	66.3
SuperGSeg [ours]	26.9	26.9	14.0	59.1	79.4	80.2	80.4	83.8	11.7	12.0	54.7	77.0	58.2	58.3

Table 8. Comparison of mIoU and mAcc for various methods on each class of the ScanNet v2 dataset [20].

the attributes as input to F_{sg} for predicting the association matrix. The results presented in Table 7 demonstrate that each MLP contributes to improving the SuperG assignments. The MLP F_ϕ , which accounts for the segmentation feature differences between the anchor and the SuperG, has the most significant impact. In particular, using only F_ϕ yields a relatively high mIoU, emphasizing its effectiveness in aligning semantic features. However, our full setup that integrates the F_ϕ and F_ψ for coordinate and geometric feature information further enhances mAcc. This suggests that incorporating additional spatial and geometric context refines the SuperG assignments, leading to a more precise understanding of the scene.

Parameters in Super-Gaussian Clustering Network.

We conduct ablation studies on the parameters involved in generating SuperGs using the SuperG clustering network. One crucial parameter is the total number of SuperGs predefined, denoted as S . Too few SuperGs fail to distinguish all instances, causing a single SuperG to span multiple instances, which undermines semantic accuracy. Conversely, too many SuperGs may introduce additional noise. Another parameter is the number of neighboring SuperGs k considered for each anchor when computing the association matrix between anchors and SuperGs.

As shown in Table 4, these two parameters are highly scene-specific, with the optimal number of SuperGs S and neighbors k varying across different scenes. Notably, for fair comparisons, we use the same parameter values, $S = 1000$ and $k = 3$, for all scenes. This parameter choice achieves optimal performance on average.

E. Additional Results

Qualitative Results in Occlusion Cases. As illustrated in Figure 10, our method queries objects directly in 3D space, effectively mitigating occlusion issues (e.g., the bear

leg under the table can be retrieved). Moreover, the queried objects exhibit multi-view consistency, enabling comprehensive scene understanding in 3D. For additional qualitative results, please refer to the accompanying videos.

Additional Quantitative Results on ScanNet. We report the results on more categories in the ScanNet dataset in Table 8.



Figure 10. The language-queried 3D masks rendering to arbitrary viewpoints remain multi-view consistent. Benefit from the 3D understanding, we enable render regions that are originally occluded and invisible in 2D.

F. Limitations

Despite the advancements achieved by our method, certain limitations remain. First, our approach inherits biases from the original visual foundation models, which may constrain performance and limit generalization to diverse or unseen scenarios. Second, our method is tailored for scene-specific language representation, requiring significant modeling time for each scene. This limits its applicability in tasks that demand rapid adaptation or broad generalization, such as in-the-wild scene understanding. Future work could focus on mitigating inherited biases and optimizing training pipelines to enhance scalability and generalization.