GALA: Guided Attention with Language Alignment for Open Vocabulary Gaussian Splatting

Supplementary Material

A. Implementation Details

A.1. Hyperparameters

Gaussian Model. We adopt Scaffold-GS [23] as our appearance model. We use segmentation and geometry features of 16 dimensions each. Two separate two-layer MLPs are employed as the segmentation and geometry decoders in Eq.3, both trained with a learning rate of 0.0005. The number of neural Gaussians is set to $K = 3$ for both ScanNet-v2 [7] and LERF-OVS [14]. We further extend the appearance Gaussian model with an instance feature $\mathbf{m}_i \in \mathbb{R}^{16}$, trained with a learning rate of 0.00005, while the learning rates for other Gaussian attributes follow [23].

Attention and Codebook Module. We set the codebook parameters to $N = 64$, $d_{\text{ins}} = 16$, and $d_c = 16$, and the learning rate to 0.001. A two-layer MLP is employed as the lifting decoder $\mathcal{F}_{\text{lift}}$, producing an output with dimension $d_{\text{lang}} = 512$.

Training. By default, we train for 30k iterations in training stage 1 and 15k iterations in training stage 2. During stage 1, we enable the densification and pruning to keep both the appearance and semantic performance.

ScanNet-v2 Dataset. Following OpenGaussian [42], we selected 8 scenes from ScanNet for evaluation. For each scene, we select 1 frame every 20 frames as keyframes for training.

Preprocess Mask and Language Feature. We follow LangSplat [31] to preprocess SAM [16] masks and CLIP [33] language features from ground-truth RGB images. We use the large level of masks from SAM.

B. Evaluation

2D open-vocabulary semantics provide per-pixel classification of objects or regions in the image plane, making them highly effective for tasks such as image segmentation and navigation. However, they are view-dependent, reflecting only what the camera sees, and may yield inconsistent semantics for the same object across viewpoints due to occlusions. In contrast, 3D open-vocabulary semantics assign per-point classifications in 3D space, preserving object size, location, and spatial relationships. They are particularly useful for applications such as robotic manipulation and AR/VR, but are often sparser. To leverage the advantages of both, we report evaluation metrics in both 2D and 3D, where our method achieves strong performance across both domains. In the following, we show how the 2D and 3D evaluation are performed on different datasets.

B.1. 2D Evaluation

For 2D evaluation, we first render the 2D language feature maps and then compute mIoU and mACC for evaluation. For LERF-OVS [14], we follow the open-vocabulary query protocol of LangSplat [31].

B.2. 3D Evaluation

LERF-OVS. Following LangSplat [31], for 3D evaluation on LERF-OVS we first use the text queries to select the corresponding Gaussians based on their language features, and then rasterize the selected Gaussians into 2D for further evaluation.

ScanNet-v2. ScanNet-v2 by default provides semantic point clouds for 3D evaluation. However, during training, the positions and number of points/Gaussians are updated to improve appearance quality. In contrast, OpenGaussian [42] fixes both positions and numbers to facilitate 3D evaluation, which we argue is unfair as it causes a notable drop in appearance performance. Our evaluation is a reproduction of the evaluation protocol proposed in Dr.Splat [12] as their full code is still not accessible. We adopt a shared, volume-aware evaluation protocol that computes per-voxel intersection-over-union (IoU) and accuracy by jointly considering the ScanNet-v2 ground-truth point cloud and the optimized neural Gaussian language features within a common voxel space.

B.3. Visualization

To visualize semantic Gaussian results, the high-dimensional language features need to be converted into 3-dimensional color values. We use the autoencoder from LangSplat [31] to mapping from CLIP features to RGB.

C. Runtime Analysis

We perform single-GPU training with NVIDIA RTX 3090. Table 5 presents a runtime analysis of the Teatime scene. While our method incurs slightly longer training time due to the Scaffold-GS structure, it achieves significantly faster inference compared to other approaches. The codebook and attention module are extremely lightweight, requiring only **0.6 MB** of memory. However, our method generates a full-sized language feature map for each view (requires 2 GB) and applies the cosine similarity loss in Eq. 8 between the full-sized ground-truth and the 512-dimensional rendered feature map (requires 10 GB), which is memory-intensive.

During inference, the cosine similarity loss is not computed, allowing our method to achieve superior runtime efficiency.

Additionally, our training pipeline consists of only two stages, whereas LangSplat and OpenGaussian require three. Regarding memory usage, both our approach and OpenGaussian generate high-dimensional feature point clouds, while LangSplat uses a compressed feature representation. This accounts for the higher memory demand of our method.

	Memory (GB)			Train Time (min)				Inference Time (sec)
Method	S1	S2	S3	S1	S2	S3	Total	
LangSplat	5	2	6	18	7	84	109	96.00
OpenGaussian	15	9	13	43	27	5	75	0.65
Ours	12	14		107	109		216	0.31

Table 5. **Runtime Analysis.** S1–S3 correspond to stage 1, 2 and 3 of training, respectively.

D. More Ablation Study

D.1. Ablation on Number of Gaussian

As reported in OpenGaussian [42], densification is disabled during ScanNet-v2 evaluation, and appearance training is conducted at a very low resolution (160*120). This results in a significant performance drop, as evidenced in Figure 7, where OpenGaussian produces appearance results that lose many fine details compared to ours.

In contrast, we train with the default resolution as in LangSplat [31] (320*240), and Scaffold-GS introduces a parameter K to control the number of spawned Gaussians, making it unnecessary to disable densification entirely. By setting $K = 3$ and $K = 10$ for the same scene, we can flexibly adjust the number of Gaussians. As shown in Figure 7 and Table 6, reducing the number of Gaussians slightly lowers appearance quality, but the degradation is far less severe than in OpenGaussian. More importantly, segmentation performance improves noticeably. This improvement likely arises because semantics carry little or no texture information, so that semantic predictions require fewer Gaussians than appearance modeling. An excessive number of Gaussians may introduce ambiguities that negatively impact segmentation.

E. More Experiment Results

E.1. LERF-OVS

Figure 8, presents additional 2D qualitative results on the LERF-OVS dataset [14], while Figure 9 shows the corresponding 3D qualitative results.

E.2. ScanNet-v2

In Table 7, we report the per-scene 3D open-vocabulary segmentation and localization results on ScanNet-v2 [7]. In Figures 10 and Figures 11, we show additional qualitative results.

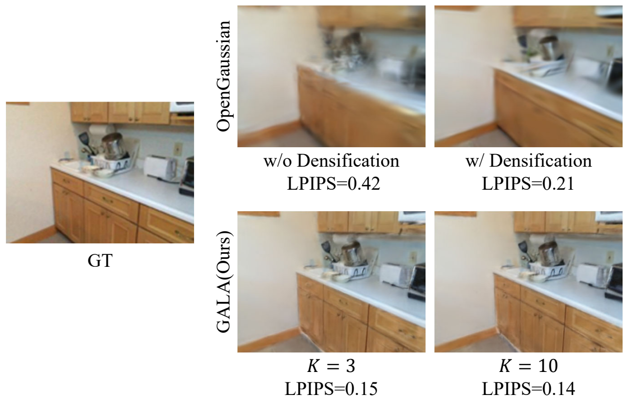

Figure 7. **Appearance Rendering.** We show the appearance rendering quality on scene0000_00 of ScanNet-v2 dataset. Compared with OpenGaussian, our method achieve much better appearance rendering.

Scenes	*Teatime*		*Scene0000_00*	
K	3	10	3	10
mIoU ↑	**53.27**	41.68	**23.82**	21.07
mAcc ↑	**84.75**	71.19	**46.83**	40.07

Table 6. **Ablation on Number of Gaussians.** We report mIoU and mAcc of our proposed method with different number of Gaussians. We report both 3D evaluation on LERF-OVS and ScanNet-v2.

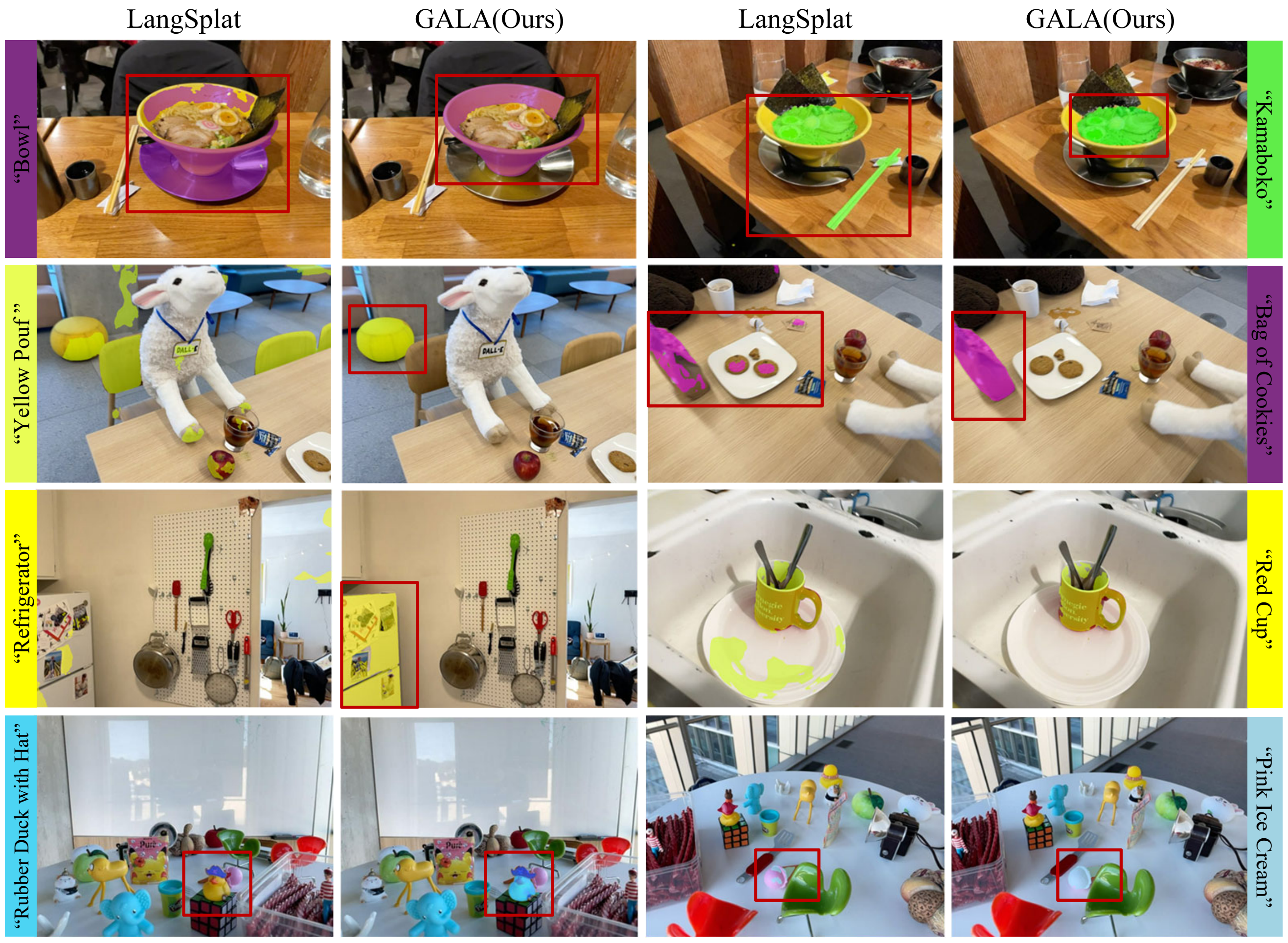

Figure 8. **More Qualitative Results of 2D Open-Vocabulary Segmentation on LERF-OVS.**

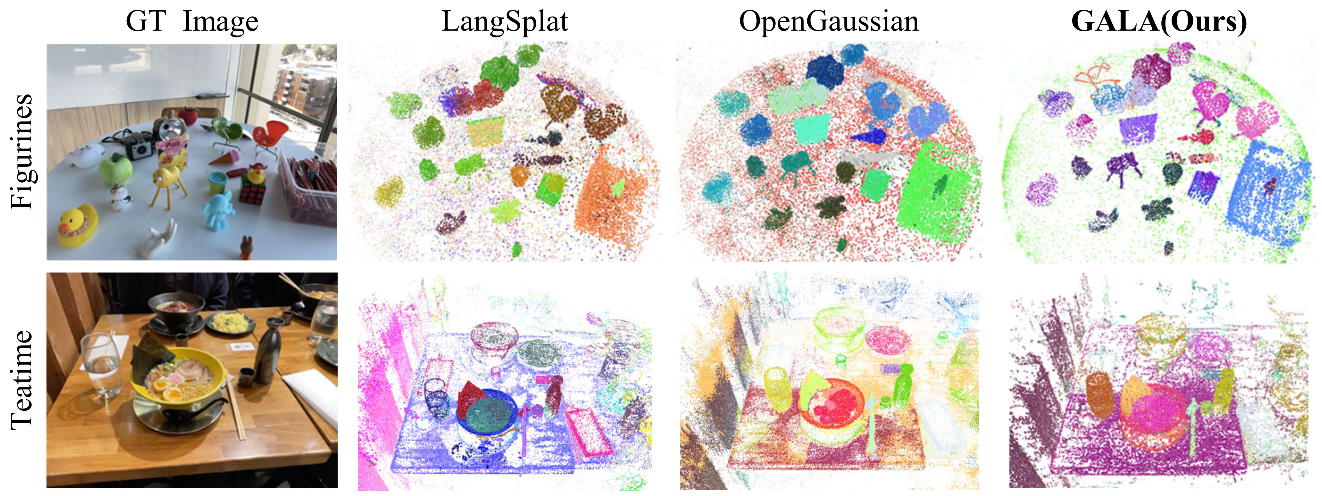

Figure 9. **More Qualitative Results of 3D Open-Vocabulary Segmentation on LERF-OVS.** We visualize the language feature point cloud by compressing the features into the RGB point cloud. Note that the colors for visualization are consistent only within each method and not method-to-method.

Method	**Mean**		*Scene0000_00*		*Scene0062_00*		*Scene0070_00*		*Scene0097_00*		*Scene0140_00*		*Scene0347_00*		*Scene0590_00*		*Scene0645_00*	
	mIoU ↑	mAcc ↑	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
Number of Classes: 19																		
LangSplat [31]	2.94	11.63	2.72	7.88	2.89	11.92	1.47	8.31	6.67	14.38	2.88	16.59	2.78	16.36	1.42	8.92	2.74	8.76
OpenGaussian [42]	15.47	26.04	16.12	27.67	18.02	27.42	**19.01**	**31.64**	9.01	13.61	15.39	28.49	23.22	35.93	11.84	18.56	11.22	25.06
Ours	**21.54**	**37.47**	**19.04**	**40.67**	**20.14**	**41.01**	18.54	28.88	**18.01**	**38.74**	**25.55**	**38.13**	**43.11**	**50.13**	**11.55**	**29.61**	**16.40**	**32.64**
Number of Classes: 15																		
LangSplat [31]	3.80	13.98	3.18	9.35	3.73	16.84	2.18	12.64	8.40	18.47	3.40	13.98	3.94	20.04	3.43	11.81	2.13	8.71
OpenGaussian [42]	17.42	28.82	19.18	33.77	**18.65**	28.71	21.16	26.89	10.76	17.31	18.64	29.86	20.50	36.85	15.78	28.20	14.73	29.02
Ours	**25.20**	**42.06**	**23.82**	**46.83**	18.23	**44.79**	**29.43**	**34.22**	**12.75**	**37.79**	**30.50**	**43.31**	**50.85**	**59.61**	**16.62**	**34.99**	**19.41**	**34.95**
Number of Classes: 10																		
LangSplat [31]	6.60	22.24	5.04	14.35	5.78	25.27	4.34	16.90	18.15	36.69	4.20	16.82	7.13	38.72	4.97	16.59	3.19	12.61
OpenGaussian [42]	23.46	37.73	22.70	39.32	28.34	43.10	29.09	37.55	21.72	29.88	21.91	35.11	20.34	46.22	**25.56**	36.17	18.09	34.55
Ours	**35.85**	**57.02**	**27.61**	**57.53**	**30.02**	**66.91**	**51.76**	**67.65**	**24.65**	**52.81**	**45.26**	**60.62**	**56.03**	**56.20**	25.17	**43.74**	**26.36**	**50.74**

Table 7. **3D Evaluation on ScanNet-v2.** We report per scene mIoU and mAcc on the ScanNet-v2 dataset [7], following the evaluation protocol of Dr.Splat [12].

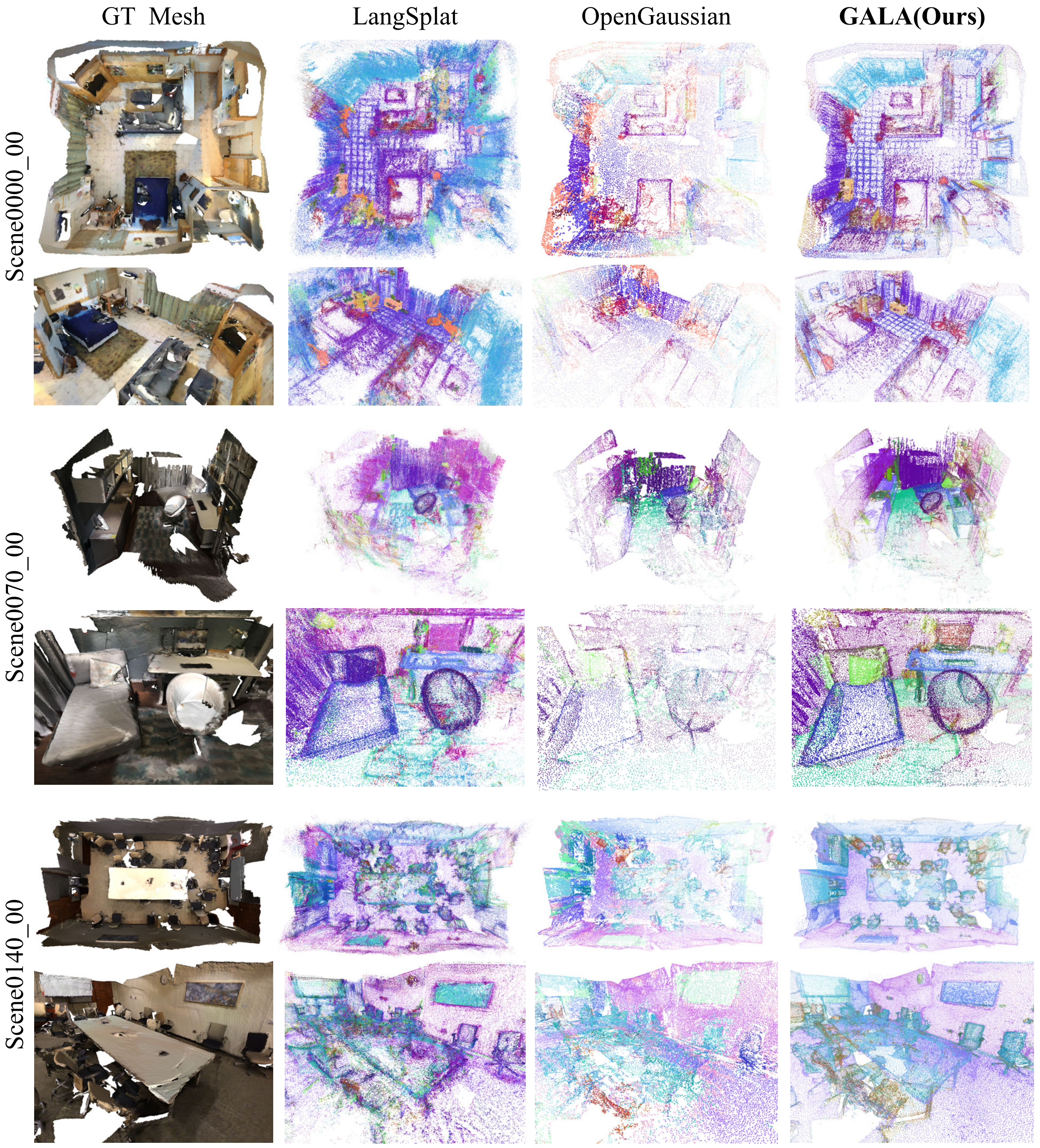

Figure 10. **More Qualitative Results of 3D Open-Vocabulary Segmentation on ScanNet-v2.** We visualize the language feature point cloud by compressing the features into the RGB point cloud. Note that the colors for visualization are consistent only within each method and not method-to-method. OpenGaussian doesn't enable densification, leads to sparser point cloud.

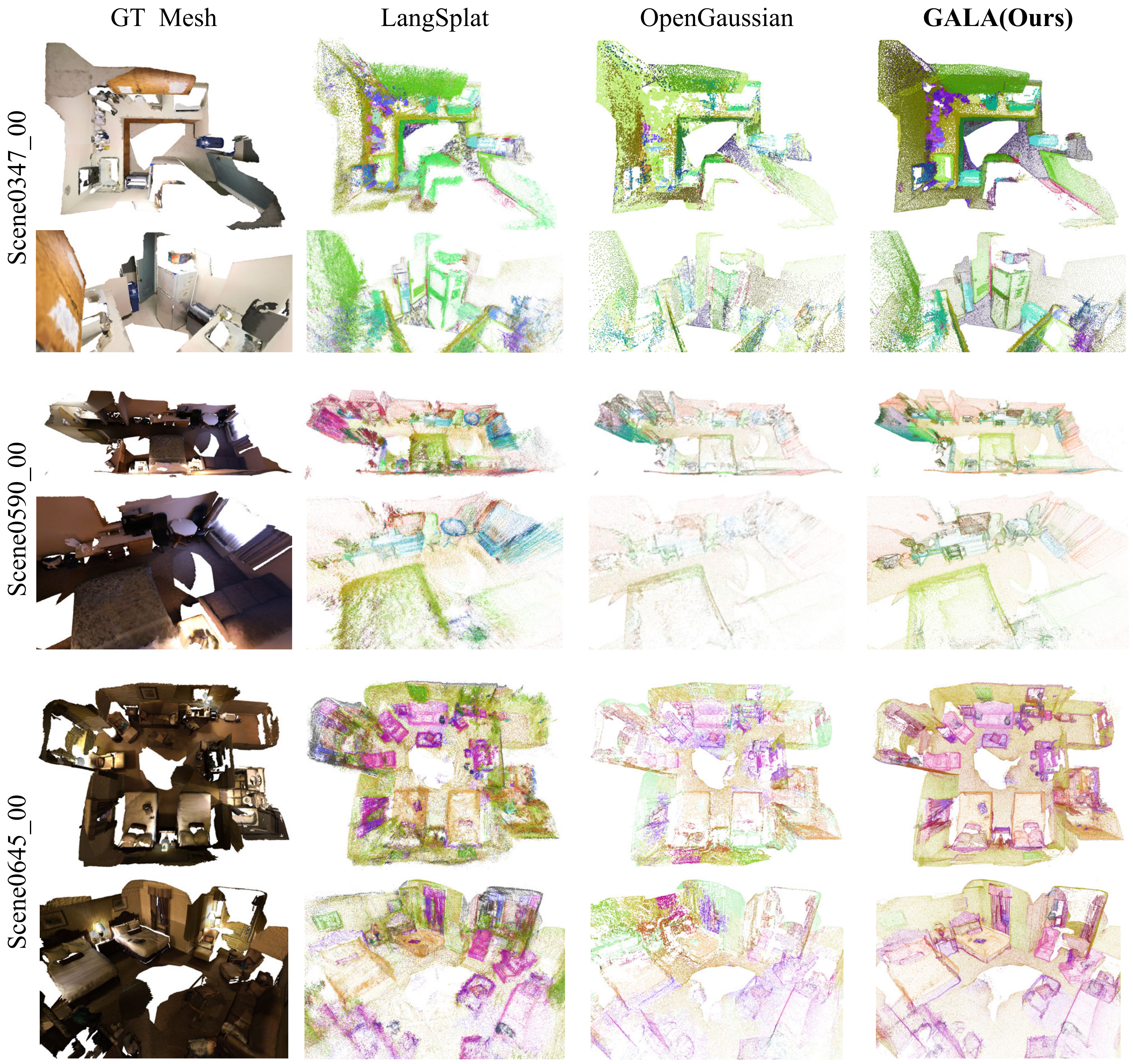

Figure 11. **More Qualitative Results of 3D Open-Vocabulary Segmentation on ScanNet-v2.** We visualize the language feature point cloud by compressing the features into the RGB point cloud. Note that the colors for visualization are consistent only within each method and not method-to-method. OpenGaussian doesn't enable densification, leads to sparser point cloud.