

Dr. Splat: Directly Referring 3D Gaussian Splatting via Direct Language Embedding Registration

Kim Jun-Seong¹GeonU Kim¹Jaesung Choe^{2*}Kim Yu-Ji¹Yu-Chiang Frank Wang²Tae-Hyun Oh^{3*}¹POSTECH ²NVIDIA ³KAIST

Abstract

We introduce Dr. Splat, a novel approach for open-vocabulary 3D scene understanding leveraging 3D Gaussian Splatting. Unlike existing language-embedded 3DGs methods, which rely on a rendering process, our method directly associates language-aligned CLIP embeddings with 3D Gaussians for holistic 3D scene understanding. The key of our method is a language feature registration technique where CLIP embeddings are assigned to the dominant Gaussians intersected by each pixel-ray. Moreover, we integrate Product Quantization (PQ) trained on general large-scale image data to compactly represent embeddings without per-scene optimization. Experiments demonstrate that our approach significantly outperforms existing approaches in 3D perception benchmarks, such as open-vocabulary 3D semantic segmentation, 3D object localization, and 3D object selection tasks. For video results, please visit : <https://drsplat.github.io/>

1. Introduction

Open-vocabulary 3D scene understanding represents a significant challenge in the field of computer vision, with applications spanning autonomous navigation, robotics, and augmented reality. This approach aims to enable the interpretation and referencing of 3D spatial information through natural language, allowing for applicability beyond a restricted set of predefined categories [2, 3, 28, 29, 33, 38, 43]. Previously, open-vocabulary 3D scene understanding has been explored using point-cloud-based methods [12, 16, 18, 27, 30, 36, 40]. Recently, the 3D Gaussian Splatting (3DGs) [17] has introduced a continuous representation integrated on explicit 3D Gaussians, which differs from traditional point-cloud approaches, enabling rapid progress in practical applications [44]. Current research has begun to explore methods for associating language-based features with 3D Gaussian splats to enhance scene understanding capabilities.

*Corresponding Authors

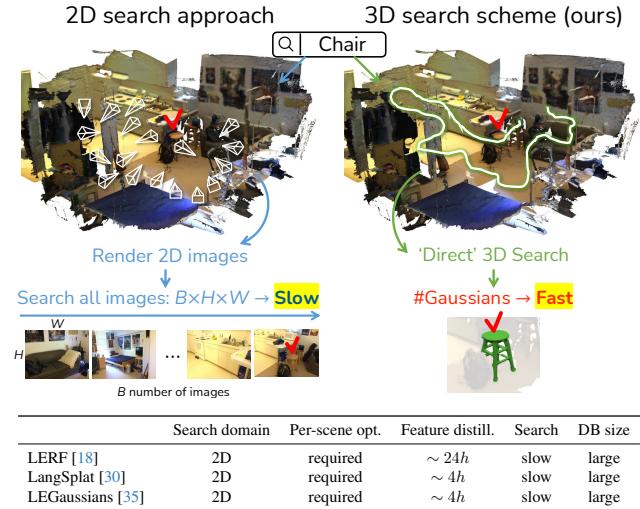


Figure 1. Comparison of 2D (left) vs. our direct 3D search (right) for open-vocabulary 3D scene understanding. The 2D approach relies on multiview rendering, incurring high computational costs. Our method directly links language features to 3D Gaussians, enabling efficient and complete spatial coverage. The table highlights Dr. Splat's superior efficiency over related methods.

Several recent approaches [30, 35, 46] introduce 3D Gaussian representation [17] into the open-vocabulary scene understanding. This unique representation uses 3D Gaussians to achieve high-quality scene rendering, offering a more structured representation that addresses some limitations of point clouds. Building on this, these methods employ 2D vision-language models to transfer language knowledge to 3D Gaussians “via rendered feature maps”.

Despite its promise, such rendering-based distillation methods [30, 35] share two limitations. First, we found that there is a discrepancy between optimized embeddings in 3D Gaussians and 2D language-aligned embeddings. This gap arises mainly from an intermediate rendering step that may distort CLIP embeddings during training. Then, the reliance on rendering impedes holistic 3D scene understanding, addi-

tional task-processing such as 3D semantic segmentation and 3D object localization, and making full spatial coverage calculations less efficient than direct 3D Gaussian methods [39] including ours as illustrated in Fig. 1.

To address this issue, this work proposes Dr. Splat. Our method bypasses the rendering stage, enabling direct interaction with 3D Gaussians for registering and referring the well-preserved language-aligned CLIP embeddings in the 3D space. This makes our Dr. Splat clearly distinguishable from prior works, facilitating a seamless integration of representative embeddings from 2D vision language models into the 3D spatial structure without compromising exhaustive rendering process that has been exploited [15, 30, 35, 44–46]. Moreover, we propose to use a Product Quantization (PQ) feature encoding method to represent embeddings compactly and efficiently without any per-scene optimization. Rather than storing full-length feature vectors or per-scene specifically compressed embeddings [15, 30, 35, 44–46], each Gaussian in our Dr. Splat stores an index from a pre-trained PQ, significantly reducing memory usage up to 6.25% compression ratio. By preserving the richness of embeddings while reducing memory usage, PQ is integral to our framework’s high scalability and its ability to perform 3D perception tasks, such as open-vocabulary 3D object localization, 3D object selection, and 3D semantic segmentation. Our contributions are summarized as follows:

- We propose Dr. Splat, direct registration and referencing of language-aligned features in 3D Gaussians, bypassing intermediate rendering and preserving feature accuracy.
- We introduce the PQ encoding method for compact feature representation, reducing memory usage while maintaining essential 3D feature properties.
- We present a novel evaluation protocol to assess accuracy of 3D localization and segmentation for 3D Gaussians, with pseudo-labeling methods and volume-aware metrics.

2. Related Work and Motivation

Language-based 3D scene understanding. Open-set 3D scene understanding has seen considerable advancements, with a focus on methods that leverage language knowledge into 3D representation such as point clouds, neural radiance fields (NeRF) [26], and Gaussian Splatting [17] for 3D comprehension. Point-based methods [5, 13, 16, 24, 27, 40, 42] in open-vocabulary settings process point cloud data trained from language embeddings [22, 31] for open-set categories.

NeRF-based approaches [7, 18, 20, 23, 32] leverage semantic embeddings from 2D foundation models, such as CLIP [31], LSeg [22] and DINO [1] for open-vocabulary understanding. While the rendering process enhances 2D perception tasks, the implicit nature of NeRF constrains the holistic understanding of 3D structures and dominantly provides ‘rendered’ feature maps.

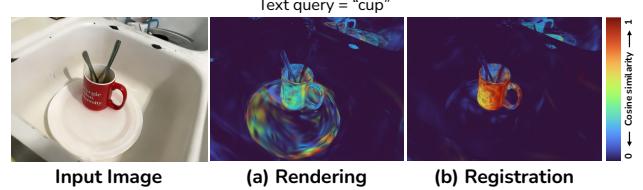


Figure 2. Visualization of discrepancy in rendered 2D features and 3D features. Color indicates a cosine similarity score between query features from a text query and either (a) 3D features distilled by 2D rendering [30] or (b) directly registered 3D features.

3D Gaussian Splatting (3DGS) [17] has emerged as a promising rendering method, as well as a novel representation for open-vocabulary 3D scene understanding. Since this research is the close related work with our work, we first elucidate the preliminary of 3DGS, followed by focusing on language embedded 3DGS as follows.

Preliminary of 3D Gaussian Splatting. 3DGS [17] encodes appearance and geometry of the target scene into the 3D Gaussian representation. Each 3D primitive representation is expressed as a 3D Gaussian distribution having mean $\mu = [x_\mu, y_\mu, z_\mu]^\top$ for 3D position and covariance matrix $\Sigma_{3D} \in \mathbb{R}^{3 \times 3}$ for 3D volume, as well as the opacity value α and the color c . In particular, the covariance matrix is decomposed into the scale matrix $S \in \mathbb{R}^{3 \times 3}$ and the rotation matrix $R \in SO(3)$, $\Sigma_{3D} = RSS^\top R^\top$. In brief, N numbers of 3D Gaussians can be parametrized as $\Theta = \{\theta_i\}_{i=1}^N = \{\mu_i, S_i, R_i, \alpha_i, c_i\}_{i=1}^N$. 3D Gaussians Θ are used to render a 2D pixel color \hat{c} computed as:

$$\hat{c}(\theta) = \sum_{i=1}^N T_i \tilde{\alpha}_i c_i, \text{ s.t. } \tilde{\alpha}_i = \alpha_i \exp\left(-\frac{1}{2} \mathbf{d}^\top \Sigma_{2D}^{-1} \mathbf{d}\right), \quad (1)$$

T_i is a transmittance, $\tilde{\alpha}_i$ is an effective opacity value computed from the Gaussian’s opacity α , the pixel distance $\mathbf{d} \in \mathbb{R}^{2 \times 1}$ from the target pixel to the projected center location of the Gaussian in pixel, and Σ_{2D} is the 2D covariance matrix in the image domain obtained from the splatting algorithm [17, 47]. The 3D Gaussian parameters Θ of a scene are optimized by minimizing the rendering loss between the input image color c and the rendered color $\hat{c}(\theta)$ in Eq. (1) as $\arg \min_{\theta} \|c - \hat{c}(\theta)\|_F^2$.

Language embedded 3D Gaussian Splatting. The basic idea of the language embedded Gaussian representation [10, 15, 21, 30, 35, 44–46] is to replace the color rendering to language embedding rendering. Language embedded 3D Gaussians are parameterized as $\Phi = \{\theta_i, \tilde{f}_i\}_{i=1}^N = \{\mu_i, S_i, R_i, \alpha_i, c_i, \tilde{f}_i\}_{i=1}^N$, where \tilde{f}_i denotes Gaussian-registered language embeddings across N numbers 3D Gaussians which will be discussed soon. Then, analogous to the color rendering Eq. (1), the language embedding rendering is expressed as:

$$\hat{f} = \sum_{i=1}^N T_i \tilde{\alpha}_i \tilde{f}_i, \quad (2)$$

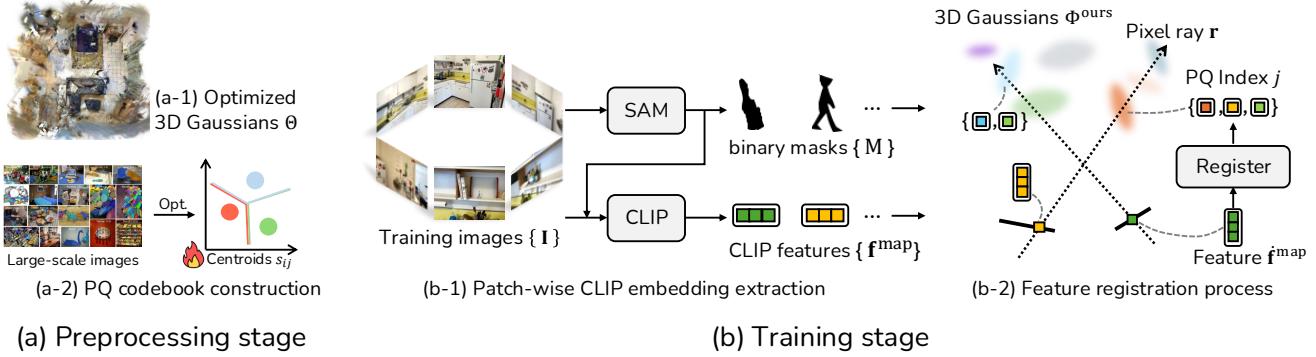


Figure 3. Overview of Dr. Splat. (a) In the preprocessing stage, we compute optimized 3D Gaussians [17] and Product Quantization (PQ) codebook construction. (b) During training, we extract CLIP embeddings from given images $\{I\}$, and then proceed feature registration process (Sec. 3.1). Finally, we obtain 3D Gaussians Φ^{ours} with PQ indices $\{j\}$ (Sec. 3.2).

where $\hat{\mathbf{f}}$ denotes a rendered language embedding. Likewise, the Gaussian-registered language embeddings $\{\tilde{\mathbf{f}}\}$ are optimized by minimizing the rendering loss between the 2D language embedding \mathbf{f} extracted from an input image and a rendered language embedding map $\hat{\mathbf{f}}$ as $\arg \min_{\{\tilde{\mathbf{f}}\}} \|\mathbf{f} - \hat{\mathbf{f}}\|_F^2$ at each corresponding pixel. This can be regarded as distilling vision language models into Gaussian-registered language embedding $\hat{\mathbf{f}}$ through volume rendering Eq. (2). The Gaussian-registered language embeddings are separately trained after pre-training and fixing the pre-trained 3DGS Θ for a scene. The language embeddings to be distilled are typically obtained from CLIP [31]. Since storing 32-bit 512-D CLIP features \mathbf{f} in every 3D Gaussians is memory-expensive, one can use a compressed feature per scene depending on the needs [15, 35, 44–46].

Motivation. Such language-embedded radiance fields provide useful representation and language interfaces for many practical and crucial applications. While most of existing works focus on the training efficiency, the complexity in inference time has barely been discussed. Considering a scenario to text-query a 3D location of the language-embedded Gaussians, *i.e.*, 3D localization, the aforementioned methods first require rendering a 2D language embedding map at each specific camera pose. We cannot directly retrieve over the distributed embeddings $\{\tilde{\mathbf{f}}_i\}$ in 3D Gaussians, because the embeddings do not carry language information directly, but their weighted summed (rendered) features $\hat{\mathbf{f}}$ do. This issue becomes even severer with compressed features as in [30]: their decompression decoders are not designed for and incompatible with directly applying to the distributed compressed language embeddings in each 3D Gaussian, yielding degenerated CLIP decoding (refer to Fig. 2).

This introduces multiple challenges and hassles. First, it is challenging to find the best or proper camera rendering views that contain the object to find. One may attempt to pre-compute the minimal number of cameras and their camera poses that cover all the 3D Gaussians in a scene with

proper resolutions, similarly by point-based approach [12]. However, this is a well-known set covering problem [8] with constraints which is known to be an NP-hard problem.

Second, even with pre-computed rendered views, the retrieval complexity over the rendered images remains substantial [9]. Suppose a scene consisting of one million Gaussians, but just a *single* rendered language embedding map in pixel domain already has nearly a million pixels; thus, we need a dedicated system to efficiently retrieve over all the views. Third, since the retrieval is conducted in the 2D space, to find a 3D location, we need a separate mechanism to lift the localization to the 3D space, *i.e.*, increasing the system complexity. In addition, 32-bit floating 512-Dimension CLIP features for millions of Gaussian are memory intensive, which is often not manageable. To reduce this burden, the existing methods [39] apply compressions with per-scene optimized codebooks, which hinders extension or generalization to other scenes.

To overcome these, we propose a training-free algorithm for the direct allocation of language embeddings to 3D Gaussians, allowing efficient computation and interaction within the 3D space. As a concurrent work, OpenGaussian [39] tackles a similar challenge with our work, but still requires per-scene codebook construction Fig. 1.

3. Dr. Splat

This section provides details of our method. We first explain how we directly register CLIP embeddings into Gaussian-registered language embeddings, Sec. 3.1. Then, we introduce Product Quantization (PQ) into our framework to efficiently store Gaussian-registered language embeddings, Sec. 3.2. Lastly, we describe the inference stage for text query-based 3D Gaussian localization, Sec. 3.3.

3.1. Feature registration process

Our goal is to reconstruct a language embedded 3D space represented by 3D Gaussians Φ , which we can directly inter-

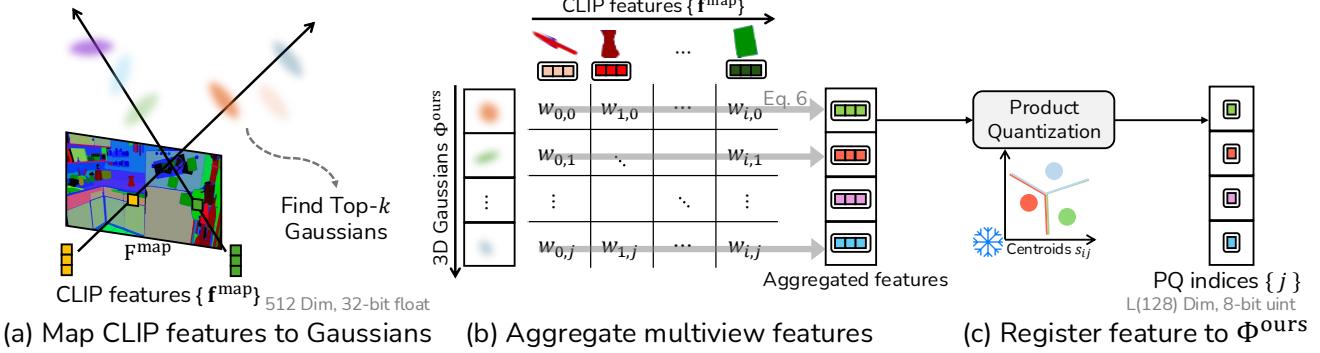


Figure 4. Feature registration process in Dr. Splat. (a) We first map per-pixel CLIP embeddings $\{f^{\text{map}}\}$ to Gaussians. Here, we only map dominant k Gaussians along pixel ray r , named Top- k Gaussians. (b) After collecting embeddings, we compute aggregated features (Eq. (6)). (c) Finally, we re-use PQ to obtain the PQ indices j of aggregated features and update Gaussian parameters Φ^{ours} .

act in 3D space without feature rendering Eq. (2). For that, following LangSplat [30], we begin by extracting per-pixel CLIP embedding maps $\mathbf{F}^{\text{map}} \in \mathbb{R}^{D \times H \times W}$ from training images of the target scenes, where D is the dimension of CLIP embeddings, H and W are the height and width of the training images. Given training images, we extract a dictionary of binary masks and language embeddings extracted from the images as: $\mathcal{F}^{\text{map}} = \{\mathbf{M}_j : \mathbf{f}_j^{\text{map}} \mid j = 1, \dots, M\}$, where $\mathbf{M}_j \in \mathbb{R}^{H \times W}$ is a binary mask extracted using SAM [19] and $\mathbf{f}_j^{\text{map}} \in \mathbb{R}^D$ is a corresponding CLIP embedding from a cropped image with \mathbf{M}_j . Each mask \mathbf{M}_j belongs to an image, and the masks are not overlapped to each other. With this dictionary, a CLIP embedding map $\mathbf{F}^{\text{map}}(\mathbf{I}, \mathbf{r})$ at a pixel \mathbf{r} in a training image \mathbf{I} is computed as:

$$\mathbf{F}^{\text{map}}(\mathbf{I}, \mathbf{r}) = \sum_{j=1}^M \mathbf{M}_j(\mathbf{I}, \mathbf{r}) \cdot \mathbf{f}_j^{\text{map}}, \quad (3)$$

where $\mathbf{M}_j(\mathbf{I}, \mathbf{r}) \in \{0, 1\}$ indicates whether the mask \mathbf{M}_j contains the pixel \mathbf{r} in the image \mathbf{I} . Using \mathbf{F}^{map} , we reconstruct language embedded 3D Gaussians via a novel feature registration process as visualized in Fig. 3.

During the feature registration process, our algorithm iterates through training images of the scene. Using projection relation, we link 3D Gaussians Φ to CLIP embeddings. Each Gaussian can link to multiple CLIP embeddings derived from different images. Then we aggregate collected embeddings to a single embedding to be assigned to each Gaussian. To ensure a consistent aggregation of the embeddings from multi-view images, we first compute a weight $w_i(\mathbf{I}, \mathbf{r})$ representing the contribution of θ_i to construct each pixel \mathbf{r} in a training image \mathbf{I} . The weights are computed with the volume rendering equation Eq. (1) as:

$$w_i(\mathbf{I}, \mathbf{r}) = T_i(\mathbf{I}, \mathbf{r}) \cdot \tilde{\alpha}_i(\mathbf{I}, \mathbf{r}), \quad (4)$$

where $T_i(\mathbf{I}, \mathbf{r})$ and $\tilde{\alpha}_i(\mathbf{I}, \mathbf{r})$ are the transmittance and the effective opacity value of θ_i for a pixel \mathbf{r} in an image \mathbf{I} ,

stated in Eq. (1). With the per-pixel weights, we calculate w_{ij} representing a weight between each Gaussian θ_i and corresponding language embedding maps $\mathbf{f}_j^{\text{map}}$, which is for aggregating CLIP embeddings from \mathbf{F}^{map} and register the embedding to each Gaussian. The weights are computed as:

$$w_{ij} = \sum_{\mathbf{I} \in \mathcal{I}} \sum_{\mathbf{r} \in \mathbf{I}} \mathbf{M}_j(\mathbf{I}, \mathbf{r}) \cdot w_i(\mathbf{I}, \mathbf{r}), \quad (5)$$

where \mathcal{I} is the set of the training images. In this iterative process, we aggregate weights only for Top- k Gaussians with the highest weights $w_i(\mathbf{I}, \mathbf{r})$, along the ray of each pixel \mathbf{r} (see Fig. 4). After aggregation, we prune the Gaussians which are not assigned any weight, i.e., $\sum_{j=1}^M w_{ij} = 0$. This summation aggregates weights between Gaussians and the CLIP embeddings by linking per-pixel weights $w_i(\mathbf{I}, \mathbf{r})$ of each Gaussian to its corresponding CLIP embeddings. With the obtained weights, we register an aggregated feature \mathbf{f}_i to each Gaussian with weighted-averaging as:

$$\mathbf{f}_i = \mathbf{f}_i / \|\mathbf{f}_i\|_2, \text{ where } \mathbf{f}_i = \sum_{j=1}^M \frac{w_{ij}}{\sum_{k=1}^M w_{ik}} \mathbf{f}_j^{\text{map}}. \quad (6)$$

This process enables 3D-aware feature registration to be consistent across various viewpoints, by aggregating features in the original high-dimensional feature space. The proposed process can be interpreted as an inverse volume rendering without gradient-based optimization, which enables our method to be faster than the prior methods requiring per-scene gradient-based optimization [27, 30, 35] for feature registration in 3D space.

3.2. Product-Quantized CLIP embeddings

Memory efficiency is a challenge in 3D scene representations, especially when associating Gaussians with high-dimensional feature vectors. LangSplat [30] addresses this by introducing an encoder-decoder network, while LeGaussian [35] and OpenGaussian [39] utilize codebook construction. However, these approaches introduce additional per-

Methods	mIoU					mAcc @ 0.25				
	waldo_kitchen	ramen	figurines	teatime	Mean	waldo_kitchen	ramen	figurines	teatime	Mean
LangSplat-m [30]	8.29	6.11	8.33	16.58	9.83	13.64	<u>14.08</u>	8.93	27.12	15.94
OpenGaussian [39]	34.60	23.87	59.33	54.44	43.06	<u>50.00</u>	35.21	<u>80.36</u>	72.88	59.61
Ours (Top-10)	37.05	24.33	<u>54.42</u>	57.35	43.29	63.64	35.21	<u>80.36</u>	77.97	64.30
Ours (Top-20)	<u>38.33</u>	<u>24.58</u>	53.94	56.19	43.26	63.64	35.21	82.14	<u>76.27</u>	64.32
Ours (Top-40)	39.07	24.70	53.36	<u>57.20</u>	43.58	63.64	35.21	80.36	<u>76.27</u>	63.87

Table 1. 3D object selection results on the LeRF-OVS dataset [18]. To measure 3D object selection performance, we calculate 2D segmentation accuracy on rendering of selected 3D Gaussians. Note that our model does not require per-scene optimization, demonstrating its robustness across diverse scenes. **Bold** and Underline stand for first and second best performance.

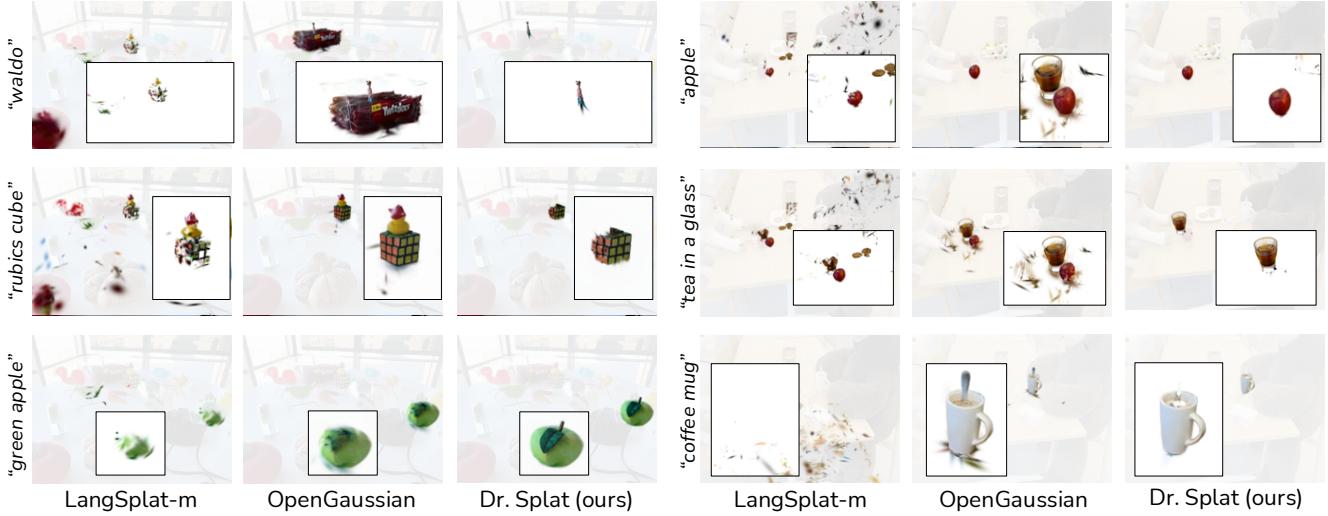


Figure 5. Qualitative results of the object selection on the LeRF-OVS dataset [18]. We visualize rendering of selected 3D Gaussians for LangSplat [30], OpenGaussian [39], and ours. For LangSplat, activations are often distributed randomly, fail to localize the target. OpenGaussian often struggles to distinguish closely situated objects. In contrast, our model shows activations precisely limited to the queried object regions, effectively localizing only the relevant areas.

scene computational costs for scene-specific parameter tuning of neural networks or codebooks (see Fig. 1). In contrast, we propose to use Product Quantization (PQ) on a large-scale image dataset, eliminating per-scene training.

Product Quantization. PQ [14] is a widely used technique for efficient embedding compression, particularly valuable in large-scale applications. The PQ process begins by dividing the original D -dimensional feature vector \mathbf{v} into L sub-vectors: $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_L]$. Each sub-vector \mathbf{v}_i is then independently quantized to a predefined number of centroids s_{ij} in a predefined codebook S_i for that sub-vector. These centroids are learned via clustering, creating a codebook for each subspace. Once the centroids are established, each sub-vector is replaced by the index of the nearest centroid in its respective codebook. The centroid indices $j_i = [j_{i1}, j_{i2}, \dots, j_{iL}]$ are optimized by minimizing $\arg \min_k \|\mathbf{v}_i - s_{ik}\|$ to quantize a given vector \mathbf{v}_i where j_{ik} is an 8-bit unsigned integer.

Then, we can measure the distance between the query and

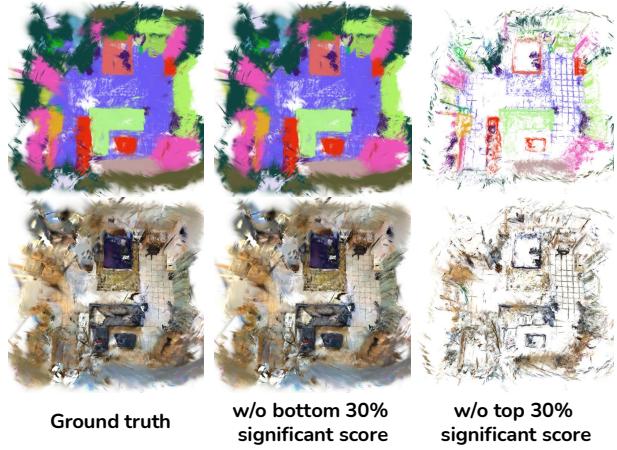


Figure 6. Limitations of point-based IoU measurement. This figure shows the effect of removing the top and bottom 30% of Gaussians according to the proposed significant score, implying that volume differences significantly impact 3D accuracy. The results highlight the need for the proposed IoU metric for 3D Gaussians.

	3D mIoU	19 classes		
		IoU > 0.15	IoU > 0.3	IoU > 0.45
LangSplat-m [30]	8.0	17.1	7.8	2.9
LEGaussians-m [35]	9.5	19.1	8.9	7.3
OpenGaussian [39]	<u>25.2</u>	<u>59.5</u>	<u>38.0</u>	18.3
Ours (Top-20)	25.0	60.7	40.3	<u>20.0</u>
Ours (Top-40)	25.4	60.7	40.3	25.6

(a) 3D object localization task.

	19 classes		15 classes		10 classes	
	mIoU	mAcc.	mIoU	mAcc.	mIoU	mAcc.
LangSplat-m [30]	2.0	9.2	4.9	14.6	8.0	23.9
LEGaussians-m [35]	1.6	7.9	4.6	16.1	7.7	24.9
OpenGaussian [39]	30.1	<u>46.5</u>	38.1	<u>56.8</u>	<u>49.7</u>	<u>71.4</u>
Ours (Top-20)	28.0	44.6	38.2	60.4	47.2	68.9
Ours (Top-40)	29.6	47.7	38.2	60.4	50.2	73.5

(b) Open-vocabulary 3D semantic segmentation task.

Table 2. Quantitative comparison in the ScanNet dataset [4]. Left: Localization prediction is defined as 3D regions with a text similarity score above threshold. Right: We assign segmentation labels by finding max activations among all classes. Note that **Bold** and Underline stand for first and second best performance, respectively.

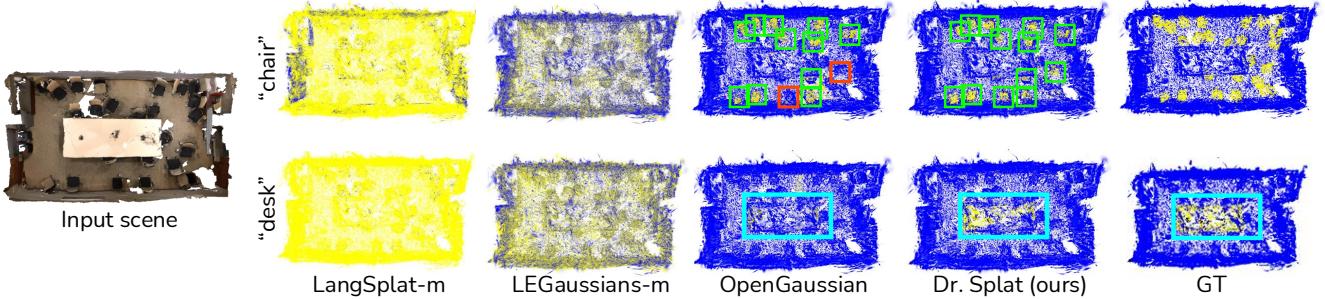


Figure 7. Qualitative results of 3D object localization. We visualize 3D localization activations (yellow) for “chair” and “desk” in the ScanNet dataset, comparing our method with others. It turns out that LangSplat-m and LEGaussians-m fail to localize objects accurately, while OpenGaussian struggles with object correspondence. Our model delivers precise and consistent localization across diverse queries.

data by adding distances between coarse centroids. Once the distances between centroids are computed as a lookup table, the computation shifts to simple indexing, which reduces the search complexity from $\mathcal{O}(D)$ to $\mathcal{O}(1)$ for a D dimension sample. This approach notably reduces computational complexity, making it suitable for large-scale search.

In our setup for language-based 3D scene understanding, we build PQ centroids based on CLIP embeddings using a large-scale image dataset, the LVIS dataset [11], that contains over 1.2M instances covering various long-tail classes and ground truth segmentation. We extract instance patches from images and collect patch-wise CLIP embeddings. After we build this CLIP embedding database, we proceed with the construction of the centroid codebook for our PQ. Once PQ is trained, any query embedding can be approximated by assigning the closest centroid for each subvector. This is a one-time procedure; once we determine the codebook, we can use it for any scene generally. In our setup, each embedding is represented as a sequence of centroid indices rather than a high-dimensional vector. Accordingly, our language embedded Gaussians are parametrized as $\Phi^{\text{ours}} = \{\phi_i^{\text{ours}}\}_{i=1}^N = \{\theta_i, j_i\}_{i=1}^N$, where the aggregated feature \bar{f}_i are converted as a quantized feature \bar{f}_i by the corresponding PQ index j_i .

3.3. Text-query based 3D localization

After training 3D Gaussians Φ^{ours} with our feature registration process and PQ, we describe the details of an inference mode that facilitates direct interaction with 3DGS upon receiving input queries, such as text. This is related to similarity score computation between a query and sources, *i.e.* Gaussian embeddings. Given a text, we first extract a query feature q using CLIP text encoder [31]. We reconstruct the quantized features $\{\bar{f}_i\}_{i=1}^N$ from the stored PQ indices $\{j_i\}_{i=1}^N$. Then, we compute a cosine similarity score between the query feature q and all quantized features.

Despite its simplicity, solely relying on the cosine similarity may result in diminished discriminability across certain similarity scores.

To address this limitation, we incorporate a re-ranking process based on relative activation with respect to the canonical feature. For this process, we adopt the relevancy scoring method proposed in LeRF [18], which enables more precise similarity analysis for a query. Specifically, each rendered language embedding, f^{map} and a text query feature q , yield a relevance score determined by, $\min_i \frac{\exp(f^{\text{map}} \cdot q)}{\exp(f^{\text{map}} \cdot q) + \exp(f^{\text{map}} \cdot f^{\text{canon}, i})}$, where (\cdot) is an element-wise dot product operator and $f^{\text{canon}, i}$ indicates CLIP embeddings of a designated canonical term selected from a set of “object,” “things,” “stuff,” and “texture”. Then, we sample 3D Gaussians based on the relevance score for downstream tasks.

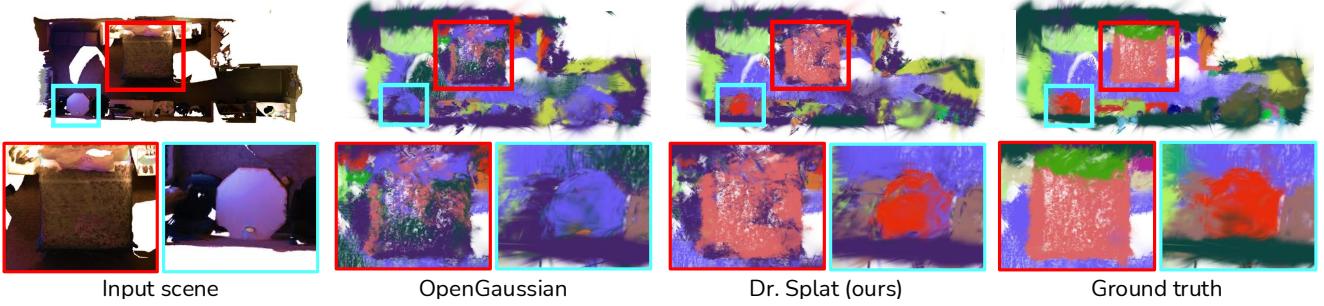


Figure 8. Visualization of open-vocabulary 3D semantic segmentation on the ScanNet dataset [4]. We visualize 3D Gaussian splat-based semantic segmentation using language features allocation of OpenGaussian [39] and Dr. Splat (ours) model on the same RGB-pretrained 3DGS. Note that, not specifically designed for segmentation, it achieves high performance as a result of language-based Gaussian updates.

4. Experiments

Dataset. We use two datasets to evaluate the 3D scene understanding performance. For the 3D object selection task (Sec. 4.1), we use the LERF [18] dataset annotated by LangSplat [30], which consists of several multi-view images of 3D scenes containing long-tail objects and includes ground truth 2D ground truth annotations for texture queries. For 3D object localization Sec. 4.2 and 3D semantic segmentation Sec. 4.3 task, we employ the ScanNet [4] dataset. ScanNet is a large-scale benchmark that provides data on indoor scenes, including calibrated RGBD images and 3D point clouds with ground-truth semantic labels. We randomly select eight scenes from ScanNet for the experiments.

Competing methods. The only method available for a fair comparison with our method is the concurrent work, OpenGaussian [39]. To study the various aspects of our method, we introduce baseline methods modified from rasterization-based ones [30, 35], for direct 3D referring operation, denoted as LangSplat-m and LEGaussians-m. As discussed in Sec. 2, without modification, global search over a whole scene is quite demanding. To ensure fair evaluation, we use the same initial 3D Gaussians being trained only using RGB inputs for all comparing methods, and freeze the Gaussians during the language feature allocation process. Also, the per-pixel CLIP [31] embedding maps are unified for SAM-based [19] methods [30, 39] including ours. We follow the hyperparameter settings favorable to each respective paper.

4.1. 3D object selection

Settings. We first extract text features from an open-vocabulary text query using the CLIP model. Next, we compare text features to the 3D features embedded in each Gaussian using cosine similarity. By thresholding the similarity, we identify the 3D Gaussians that are relevant to the given text query. The selected 3D points are subsequently rendered into multi-view images using the 3DGS rasterization pipeline.

Results. We compare our model quantitatively with 3DGS-based language-embedded models as shown in Table 1. The results demonstrate that our method performs better object selection in most scenes, showing an improvement of over 0.5 in mIoU and more than 4.5 in mAcc compared to counterpart models. Notably, the rasterization-based method, LangSplat-m, often underperforms in most scenes.

Qualitative results are shown in Fig. 5. For LangSplat-m, the activations often show random 3D Gaussians or fail to localize entirely (e.g., see “coffee mug”), highlighting the limitations of rasterization-based methods and their unsuitability for 3D understanding, aligning the observation from Fig. 2. OpenGaussian frequently exhibits false activations with incorrect text-object pairs (e.g., “apple” and “tea in a glass”) and struggles to distinguish between nearby objects (e.g., “waldo,” “rubik’s cube”). This artifacts can be attributed to use of spatial clustering and limited encoder capacity.

In contrast, our model leverages general image features thanks to the general PQ, maintaining feature distinctiveness regardless of scene complexity. Our feature registration considers the 3D geometry of the 3D Gaussians, which results in superior performance in 3D scene understanding tasks.

4.2. 3D object localization

Settings. Similar to the 3D object selection task, we calculate the cosine similarity between text query and 3D features embedded in each Gaussian. By thresholding the similarity, we identify the 3D Gaussians relevant to the given text query. To measure volume-aware localization evaluation, we propose a protocol to measure the IoU of 3D Gaussians that expands the traditional metric of point cloud-based approaches by incorporating volumetric information of 3D Gaussians.

Novel evaluation protocol for 3D localization in 3DGS. Unlike conventional evaluation protocol for the 3D localization task in point clouds, it is tricky to evaluate 3D localization performance in 3D Gaussians [17]. This is primarily

due to the un-deterministic structure of Gaussian distribution. To address this issue, we compute 3DGS pseudo-labels for evaluating the 3DGS localization in a volume-aware way. The details can be found in the supplementary material.

Given the ground truth, we measure IoU considering the spatial significance of each Gaussian and define a significant score d_i for each Gaussian θ_i with its scale $s_i = [s_{ix}, s_{iy}, s_{iz}]$ and opacity α_i as $d_i = s_{ix}s_{iy}s_{iz}\alpha_i$, where $s_{ix}s_{iy}s_{iz}$ denotes a relative ellipsoid volume of a Gaussian θ_i . With the obtained significant scores $\mathbf{d} = [d_1, d_2, \dots, d_N]$, we compute weighted IoU of 3D Gaussians to approximate volumes. The proposed metric is designed to assign a larger weight to the Gaussians with higher significant scores, when measuring IoU. Figure 6 shows that the impact of each Gaussian on the scene extremely varies depending on their significant scores, which demonstrates the necessity of the proposed IoU metric on 3D Gaussians that regards unequal contributions of each Gaussian.

Results. We report the 3D localization performance on the Scannet dataset in Table 2a. The 2D rasterization-based methods [30, 35] struggle to achieve precise activations for 3D localization. They inherently face challenges when applying for 3D tasks because they need to render 2D images for the scene interaction. Even with the 3D space search method, OpenGaussian [39], our model consistently demonstrates superior performance and achieves higher accuracy in localization. Figure 7 also shows that LangSplat-m and LEGaussians-m fail to properly localize the objects, and OpenGaussian misses queried objects in the scene.

4.3. 3D semantic segmentation

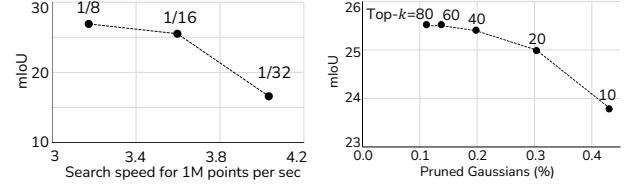
Settings. For a given set of open-vocabulary text labels, we perform segmentation by assigning each Gaussian a label having the highest activation among the known label set.

Results. The numerical comparison is presented in Table 2b. Although not explicitly designed for semantic segmentation, our model achieves notable performance in this task as a result of accurately updating each Gaussian with language features. Consistent with previous observations, rasterization-based 3DGS models exhibit lower segmentation performance. While OpenGaussian performs position-based clustering, our model demonstrates comparable performance, surpassing the baseline as the Top- k value increases. Our model also achieves better segmentation results, with a visual comparison of the segmented scene shown in Fig. 8.

4.4. Ablation study

We conduct an ablation study using the ScanNet dataset on different hyper-parameters of Dr. Splat to measure the contribution of each component.

Product Quantization. PQ introduces a trade-off between memory usage, computational efficiency, and accuracy. To better understand the balance between computational cost



(a) Ablation on PQ parameters. (b) Ablation on Top- k .

Figure 9. Ablation study on (a) PQ and (b) Top- k Gaussians.

and localization quality, we conduct an ablation study by varying the number of sub-vectors. We evaluate performance at sub-vector sizes of 64, 128, and 256. Notably, these settings correspond to bit-size reductions of 1/32, 1/16, and 1/8 of the original CLIP feature, respectively. We measure the query distance computation time for one million data points, averaging results over 100 iterations for efficiency measure. Our findings reveal a favorable trade-off between quantization performance and accuracy (see Fig. 9-(b)) in the Pareto front with our PQ configurations. This achieves a balance that maximizes memory and computational efficiency while minimizing any loss in accuracy.

Top- k Gaussians. We examine the influence of the number of Gaussians assigned per ray. This parameter affects both memory requirements and computation, serving as a critical factor in overall performance. The ratio of pruned Gaussians and the mIoU results from different k are presented in Fig. 9a. We observe that increasing the aggregating number of Gaussians per ray improves localization performance; however, it results in higher memory consumption and the number of occupied Gaussians, indicating a clear trade-off.

5. Discussion and Conclusion

We present Dr. Splat, which is a novel approach for open-vocabulary 3D scene understanding by directly registering language embeddings to 3D Gaussians, eliminating the need for an intermediate rendering process. Compared to the previous 2D rendering-based methods [30, 35], which have limited search domain and capacity, our method directly searches 3D space while preserving the fidelity of language embeddings. This operation is further accelerated by the integration of Product Quantization (PQ)

Experimental results validate Dr. Splat's superior performance across various 3D scene understanding tasks, including open-vocabulary 3D object selection, 3D object localization, and 3D semantic segmentation. These findings highlight Dr. Splat's ability to transform 3D scene understanding by achieving a balance between highly representative quality and computational efficiency. This breakthrough paves the way for advanced applications in robotics, autonomous navigation, and augmented reality.

References

- [1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2
- [2] Jaesung Choe, Chunghyun Park, Francois Rameau, Jaesik Park, and In So Kweon. Pointmixer: Mlp-mixer for point cloud understanding. In *European Conference on Computer Vision*, pages 620–640. Springer, 2022. 1
- [3] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019. 1
- [4] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 6, 7, 13, 15
- [5] Runyu Ding, Jihan Yang, Chuhui Xue, Wenqing Zhang, Song Bai, and Xiaojuan Qi. Pla: Language-driven open-vocabulary 3d scene understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7010–7019, 2023. 2
- [6] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvassy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024. 11
- [7] Francis Engelmann, Fabian Manhardt, Michael Niemeyer, Keisuke Tateno, Marc Pollefeys, and Federico Tombari. Open-Nerf: Open Set 3D Neural Scene Segmentation with Pixel-Wise Features and Rendered Novel Views. In *ICLR*, 2024. 2
- [8] Michael R. Garey and David S. Johnson. Computers and intractability. a guide to the theory of np-completeness. *W. H. Freeman and company*, 174, 1979. 3
- [9] Antoine Guédon, Tom Monnier, Pascal Monasse, and Vincent Lepetit. Macarons: Mapping and coverage anticipation with rgb online self-supervision. In *CVPR*, 2023. 3
- [10] Jun Guo, Xiaoqian Ma, Yue Fan, Huaping Liu, and Qing Li. Semantic gaussians: Open-vocabulary scene understanding with 3d gaussian splatting. *arXiv preprint arXiv:2403.15624*, 2024. 2
- [11] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvls: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019. 6
- [12] Zhening Huang, Xiaoyang Wu, Xi Chen, Hengshuang Zhao, Lei Zhu, and Joan Lasenby. Openins3d: Snap and lookup for 3d open-vocabulary instance segmentation. In *European Conference on Computer Vision*, pages 169–185. Springer, 2025. 1, 3
- [13] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Osama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023. 2, 16
- [14] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010. 5, 15
- [15] Yuzhou Ji, He Zhu, Junshu Tang, Wuyi Liu, Zhizhong Zhang, Yuan Xie, and Xin Tan. Fastlgs: Speeding up language embedded gaussians with feature grid mapping. *arXiv preprint arXiv:2406.01916*, 2024. 2, 3
- [16] Li Jiang, Shaoshuai Shi, and Bernt Schiele. Open-vocabulary 3d semantic segmentation with foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21284–21294, 2024. 1, 2
- [17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023. 1, 2, 3, 7, 11, 13, 14
- [18] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023. 1, 2, 5, 6, 7, 12
- [19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. In *ICCV*, 2023. 4, 7, 11
- [20] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems*, 35:23311–23330, 2022. 2
- [21] Hyunjee Lee, Youngsik Yun, Jeongmin Bae, Seoha Kim, and Youngjung Uh. Rethinking open-vocabulary segmentation of radiance fields in 3d space, 2024. 2
- [22] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022. 2
- [23] Kunhao Liu, Fangneng Zhan, Jiahui Zhang, Muyu Xu, Yingchen Yu, Abdulkotaleb El Saddik, Christian Theobalt, Eric Xing, and Shijian Lu. Weakly supervised 3d open-vocabulary segmentation. *Advances in Neural Information Processing Systems*, 36:53433–53456, 2023. 2
- [24] Minghua Liu, Yinhao Zhu, Hong Cai, Shizhong Han, Zhan Ling, Fatih Porikli, and Hao Su. Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21736–21746, 2023. 2
- [25] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Ming Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. In *CVPR*, 2024. 16
- [26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2
- [27] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023. 1, 2, 4, 16

- [28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1
- [29] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in neural information processing systems*, 35:23192–23204, 2022. 1
- [30] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024. 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 16, 17, 18
- [31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2, 3, 6, 7, 12
- [32] Adam Rashid, Satvik Sharma, Chung Min Kim, Justin Kerr, Lawrence Yunliang Chen, Angjoo Kanazawa, and Ken Goldberg. Language embedded radiance fields for zero-shot task-oriented grasping. In *7th Annual Conference on Robot Learning*, 2023. 2, 16
- [33] Damien Robert, Hugo Raguet, and Loic Landrieu. Efficient 3d semantic segmentation with superpoint transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17195–17204, 2023. 1
- [34] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *ECCV*, 2022. 16
- [35] Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. Language embedded 3d gaussians for open-vocabulary scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5333–5343, 2024. 1, 2, 3, 4, 6, 7, 8, 11, 12, 16, 17, 18
- [36] Ayça Takmaz, Elisabetta Fedele, Robert W Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. Open-mask3d: Open-vocabulary 3d instance segmentation. *arXiv preprint arXiv:2306.13631*, 2023. 1
- [37] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 16
- [38] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024. 1
- [39] Yanmin Wu, Jiarui Meng, Haijie Li, Chenming Wu, Yahao Shi, Xinhua Cheng, Chen Zhao, Haocheng Feng, Errui Ding, Jingdong Wang, et al. Opengaussian: Towards point-level 3d gaussian-based open vocabulary understanding. *arXiv preprint arXiv:2406.02058*, 2024. 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 15, 16, 17, 18
- [40] Jihan Yang, Runyu Ding, Weipeng Deng, Zhe Wang, and Xiaojuan Qi. Regionplc: Regional point-language contrastive learning for open-world 3d scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19823–19832, 2024. 1, 2
- [41] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024. 16
- [42] Junbo Zhang, Runpei Dong, and Kaisheng Ma. Clip-fo3d: Learning free open-world 3d scene representations from 2d dense clip. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2048–2059, 2023. 2
- [43] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 1
- [44] Yuhang Zheng, Xiangyu Chen, Yupeng Zheng, Songen Gu, Runyi Yang, Bu Jin, Pengfei Li, Chengliang Zhong, Zeng-mao Wang, Lina Liu, Chao Yang, Dawei Wang, Zhen Chen, Xiaoxiao Long, and Meiqing Wang. Gaussiangrasper: 3d language gaussian splatting for open-vocabulary robotic grasping. *IEEE Robotics and Automation Letters*, 2024. 1, 2, 3
- [45] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024.
- [46] Xingxing Zuo, Pouya Samangouei, Yunwen Zhou, Yan Di, and Mingyang Li. Fmgs: Foundation model embedded 3d gaussian splatting for holistic 3d scene understanding. *IJCV*, 2024. 1, 2, 3
- [47] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus H. Gross. EWA volume splatting. In *Visualization Conference*, 2001. 2

Dr. Splat: Directly Referring 3D Gaussian Splatting via Direct Language Embedding Registration

Supplementary Material

A Implementation Details

B Experiment Setup

C Evaluation Protocols

D Search-time Experiments

E Additional Results

E.1 Additional results on presented 3D tasks

E.2 Experiments on the ScanNet200 dataset

E.3 Experiments on the city-scale dataset

F Broader Applications and Limitations

Supplementary Material

In this supplementary material, we provide additional details omitted from the manuscript. Sec. A covers implementation and evaluated 3D tasks. Sec. B outlines the experimental setup, and Sec. C explains our Gaussian-friendly evaluation protocol. Sec. D presents search-time experiments, while Sec. E includes qualitative results, annotation analyses, and city-scale dataset evaluations. Sec. F addresses limitations and future directions. We also provide a supplementary video that highlights city-scale experiments.

A. Implementation Details

Overall, our method consists of (1) a pre-processing stage that constructs the codebooks in Product Quantization, and pre-training of 3DGS, (2) a training stage that aggregates multiview CLIP embeddings into unified Gaussian-registered embeddings, and (3) Inference stage that directly referring to language-embedded 3D Gaussians for the downstream task.

Pre-Training stage. In the pre-processing stage, we need to extract per-patch CLIP embeddings to build PQ codebooks. It consists of a patch extraction step, and a CLIP embedding extraction step. To obtain patches, we utilize the LVIS dataset, a large-scale dataset having ground truth image segmentations. From the segmentation mask given in the LVIS dataset, we identify object regions and crop them into individual image patches. Each cropped patch is then processed and encoded using the OpenCLIP ViT-B/16 model. Based on these predictions, we continue to build PQ codebooks. We utilize FAISS [6] open-source library for our Product Quantization implementation. We use 128 sub-vectors per embedding, with each subvector assigned to one of 256 centroids, yielding an 8-bit index per subvector. This process is illustrated in Fig. S1.

3D Gaussian parameters Θ [17] are also optimized during

the pre-processing stage. We typically care about this initialization of the 3D Gaussians, which can potentially impact the performance of the 3D scene understanding tasks. So, we follow the original 3D Gaussian Splatting method and utilize the optimized 3D Gaussians as our initial parameters. In other words, the pre-training is conducted using the default hyperparameters from 3DGS [17] framework, running 30,000 iterations. Also, we consistently apply this paradigm across different methods for fair comparison. Especially, for LeGaussian [35] that employs mutual training, we disabled 3D Gaussian updates during feature assignment in our experiments.

Training stage. Based on the PQ and the initial 3D Gaussian parameters Θ , we begin the training stage. All competing models and our proposed model are trained and evaluated on a single NVIDIA RTX A6000 GPU to ensure fair performance comparison. The training stage consists of three main steps: extracting pixel-wise CLIP embeddings from training images, the feature aggregation stage, and lastly feature registration stage.

Given multi-view images, we extract dense CLIP features assigned to each pixel. To obtain per-pixel CLIP features, we adopt the feature extraction scheme by LangSplat [30], which utilizes SAM [19]. To collect per-patch embeddings, we followed the OpenGaussian framework and used a single-level mask, while LangSplat utilized multi-level masks.

Once these CLIP features are extracted for all images, we proceed to the feature registration step. In the feature registration step, we iteratively measure the contribution (weights) of pre-trained Gaussians for each ray assigned to the pixels in the training images, and update the 3D Gaussian embeddings. These weights are determined according to the volume rendering equation, which defines their influence during the color rasterization process (see Sec. 3.2 of the manuscript). After the registration process, we normalize the embeddings by dividing embeddings with L2 norms.

Lastly, we register aggregated features to 3D Gaussians. For memory efficiency, we quantize the aggregated Gaussians using the pre-trained PQ codebooks to encode features to indices, a set of 128-channel 8-bit integer indices (Fig. S1). While registration, Gaussians that were never selected in the top-k process are pruned to reduce noise and memory consumption. At the end of the training, we retain a set of assigned Gaussians with 128 8-bit integer indices.

Inference stage. Finally, in the inference stage, the PQ-assigned Gaussians from the previous steps are used. By recalling the PQ index list assigned to each 3D Gaussian,

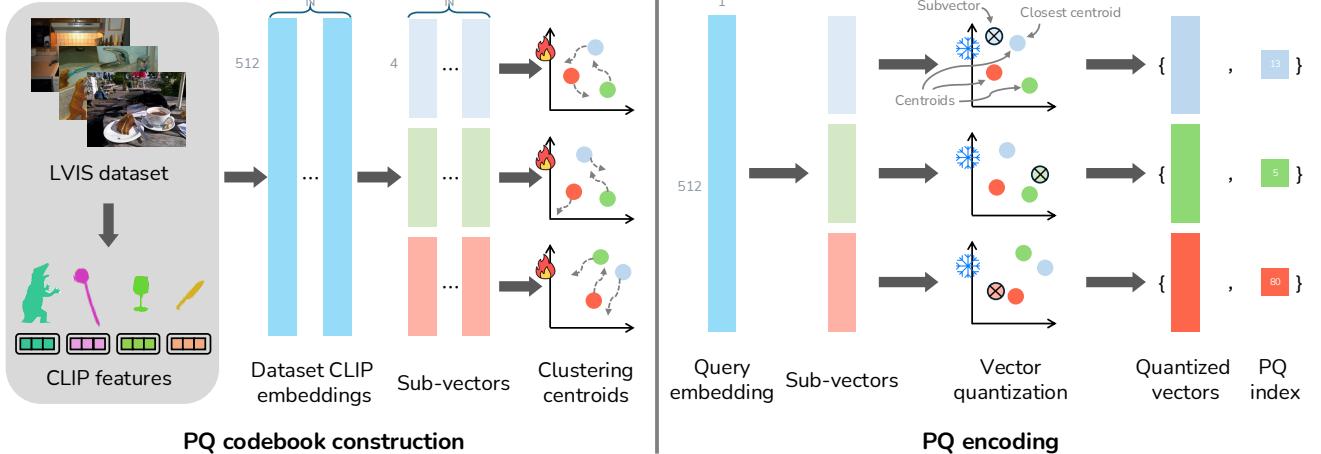


Figure S1. We illustrate the process of construction and encoding process of Product Quantization we used in Dr. Splat. (left) We first construct by update subvector centroids using CLIP features extracted from large-scale object images. (right) After constructing PQ codebook, centroids for each sub-vectors are kept frozen. For query feature, we divide into sub-vectors and are encoded into centroid indices by finding nearest neighbor.

cosine similarity is computed between the embeddings of a given text query, extracted using the same CLIP encoder, and each 3D Gaussian. Detailed steps are provided in Sec. 3.3 of the manuscript. As the subvector norms do not sum to 1, normalization by the sum of the subvector L2 norms is applied. We can apply this by using the `search` function in the Faiss library. The resulting similarity scores are then used to perform various 3D tasks, as evaluated in the study. The following sections explain how the computed activation values are applied in each task.

B. Experiment Setup

We conduct experiments on three different tasks: 3D object selection task, open-vocabulary 3D object localization task, and open-vocabulary 3D semantic segmentation task. These tasks are closely related to the 3D search as described in Fig. 1 of the manuscript as well as the 3D scene understanding tasks [39].

3D object selection. To evaluate the model’s 3D awareness capability, we evaluate a 3D object selection task. We first extract text features from an open-vocabulary text query using the CLIP text encoder [31]. Next, we compare these text features to the 3D Gaussian embeddings by computing the cosine similarity score. By thresholding the similarity, we identify the 3D Gaussians that are relevant to the given text query. The threshold value for each method is determined through a grid search to identify the optimal performance.

We use the LeRF-OVS dataset [18] with annotations by LangSplat [30]. As the LeRF-OVS dataset lacks 3D ground truth, we follow the 2D segmentation-based evaluation method proposed by OpenGaussian [39]. This approach evaluates 3D understanding by measuring multi-view 2D seg-

mentation accuracy between the rendered occupancy mask from the selected 3D Gaussians and the GT object masks. Ground truth segmentation masks are manually annotated corresponding to text queries as described in [30]. We evaluate the IoU and localization accuracy for the metric.

Open-vocabulary 3D object localization. Given an open-vocabulary text query, we use the CLIP text encoder [31] to extract a text feature of the given text query. Then, we compute the cosine similarity score between the query text feature and the Gaussian-registered embeddings. Finally, we select highly relevant 3D Gaussians by thresholding the obtained cosine similarities. We set the threshold of each method individually by searching the thresholds that show the best mIoU on the scenes used for evaluation.

Open-vocabulary 3D semantic segmentation. We further evaluate our method using the open-vocabulary 3D semantic segmentation task. For a given set of open-vocabulary text queries representing categories, we use the CLIP text encoder to extract a language embedding for each query. We then compute the cosine similarity scores between the 3D Gaussian embeddings and the language features from the given text queries. Using the obtained cosine similarity scores, we assign each 3D Gaussian to the category with the highest cosine similarity score.

C. Evaluation Protocols

Limitations of existing evaluation protocols. Compared to the previous works, such as LERF [18], LEGaussian [35], and LangSplat [30], our method challenges to leverage the 3D Gaussian representation into the 3D scene understanding tasks. Similar to ours, OpenGaussian [39] is a concurrent work that aims at the open-vocabulary 3D semantic seg-

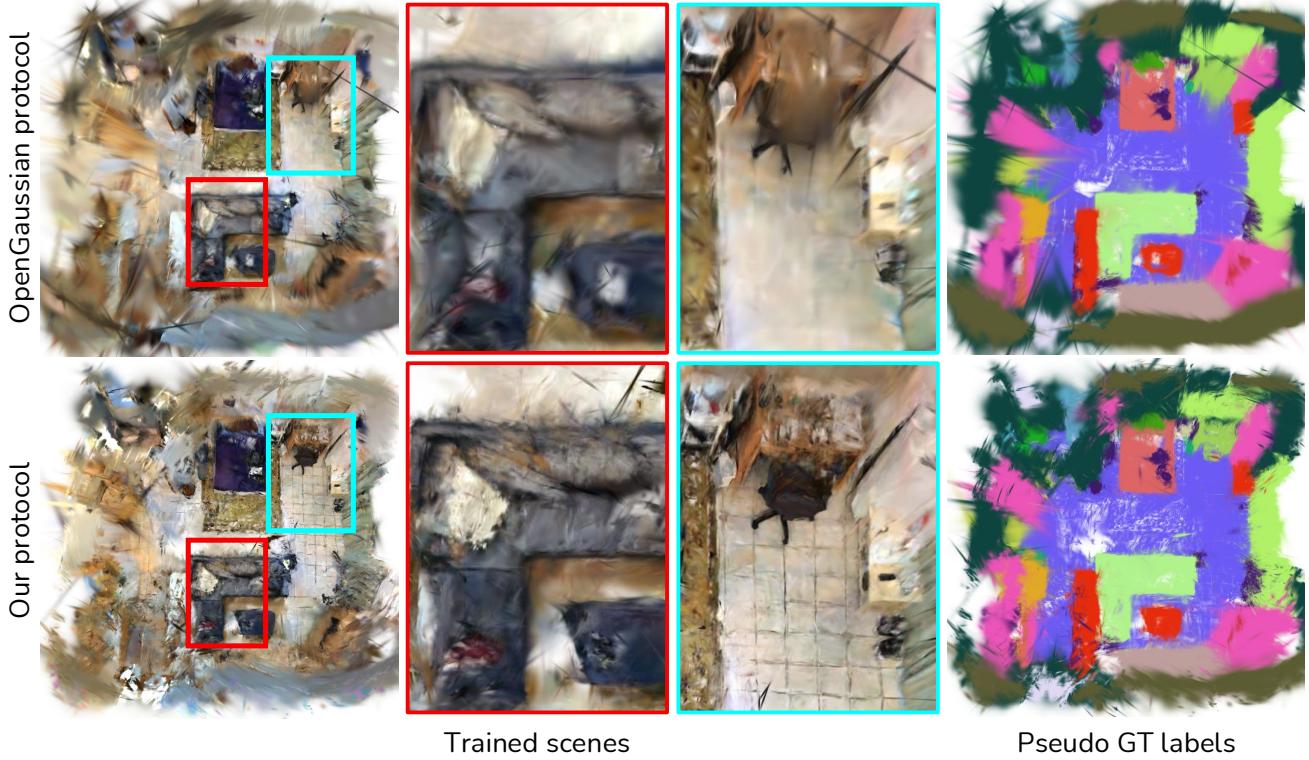


Figure S2. We compare the quality of the scenes and pseudo ground truth labels obtained from different evaluation protocols. (top) Trained scenes following OpenGaussian evaluation protocol, which fix the positions and the number of the initial points during training. (bottom) Trained scenes following our evaluation protocol, which dose not require any constraint during training.

mentation task as well. However, unlike OpenGaussian, we introduce a new evaluation criterion specialized for the 3D Gaussians, instead of using point cloud-specific evaluations.

OpenGaussian [39] computes evaluation metrics directly from 3D Gaussians, using ScanNet [4] ground truth point clouds with semantic labels. It aligns Gaussian centers μ with dataset points $[x, y, z]$ and keeps both μ and the number of Gaussians N fixed during parameter optimization. This differs from vanilla 3D Gaussian Splatting [17]. As shown in Fig. S2, their approach introduces significant quality issues, influenced by the evaluation metric. However, the reason behind this optimization trick is related to evaluation.

The evaluation by OpenGaussian involves predicting labels for each Gaussian and measuring their alignment with the ground truth point cloud using Intersection over Union (IoU). To compute IoU, the overlap (intersection) and total extent (union) of the points are calculated between the 3D Gaussians' center locations $\{\mu\}$ and the ground truth point clouds at fixed positions. As we discussed, since OpenGaussian does not update the locations of the 3D Gaussians, which is identical to the locations of the 3D ground truth points, they simply count the overlap and union without considering the volumetric properties of the 3D Gaussians.

We claim that such an evaluation protocol has two dominant issues. First, by pre-defining the number of Gaussians as well as the center locations of the 3D Gaussians, the optimized 3D Gaussians produce degraded rendering quality as shown in Fig. S2, which is not a practical solution. Second, the aforementioned IoU is calculated only with the number of 3D Gaussians, which does not consider the significance of each Gaussian having different shapes and densities.

Our Gaussian-friendly evaluation protocol. To address these limitations, we propose a novel evaluation protocol to compute IoU from 3D Gaussians. Our evaluation protocol follows the original 3D Gaussian Splittings' optimization scheme [17] by updating the location of the 3D Gaussians as well as the number of 3D Gaussians. After we obtain the optimized Gaussians Θ , these parameters are used to train language-embedded Gaussians. Then, the following question is how we assign the ground truth semantic labels for each Gaussian from the existing per-point semantic annotations provided by the ScanNet dataset [4].

Starting from the given Q numbers of point cloud $\mathcal{P} = \{\mathbf{p}_k\}_{k=1}^Q$ and a set of semantic labels $\mathcal{S} = \{\mathbf{s}\}$, we compose a paired set of points and their labels as $\{\mathbf{p}_k, \mathbf{s}^{\mathbf{p}_k}\}_{k=1}^Q$, which is provided by the official datasets. We mea-

sure the Mahalanobis distances between the language-embedded 3D Gaussian parameters $\Phi = \{\theta_i, \tilde{\mathbf{f}}_i\}_{i=1}^N = \{\boldsymbol{\mu}_i, S_i, R_i, \alpha_i, \mathbf{c}_i, \tilde{\mathbf{f}}_i\}_{i=1}^N$ (Sec. 2 of the manuscript) and ground truth point clouds. Note that the Mahalanobis distances is already used in the 3DG [17] when computing effective alpha values, as stated in the Eq. 1 of the manuscript. We maintain to use this equation to calculate the Mahalanobis distance $\mathbf{d}^{\text{mahal}}(\cdot)$ between volumetric 3D Gaussian θ and 3D point \mathbf{p} as:

$$\mathbf{d}^{\text{mahal}}(\mathbf{p}, \theta) = (\mathbf{p} - \mu)^\top \Sigma^{-1} (\mathbf{p} - \mu). \quad (7)$$

Using the Mahalanobis distance, we determine the semantic label of each Gaussian as below:

$$\mathbf{s}^{\theta_i} = \arg \max_{\mathbf{s} \in \mathcal{S}} \left(\sum_{\mathbf{p}_k \in \mathcal{P}} \mathbb{1}\{\mathbf{s}^{\mathbf{p}_k} = \mathbf{s}\} \cdot \mathbf{d}^{\text{mahal}}(\mathbf{p}_k, \theta_i) \right), \quad (8)$$

where \mathbf{s}^{θ_i} is the semantic label of the i -th 3D Gaussian θ_i , $\mathbb{1}\{\mathbf{s}^{\mathbf{p}_k} = \mathbf{s}\}$ is an indicator function returning 1 only when k -th point label is identical to a semantic label $\mathbf{s} \in \mathcal{S}$. In shorts, this equation determines the semantic label of each 3D Gaussian from the specific semantic label \mathbf{s} that has the highest sum of the Mahalanobis distances from the ground truth point to each 3D Gaussian.

The proposed assignment process enables generally applicable evaluation of 3D Gaussians without any constraints. Fig. S2 shows the quality degradation of the trained scene following the OpenGaussian evaluation protocol, which fixes the position and the number of initial points during training. On the other hand, our generalizable evaluation protocol does not impose any constraints during the training of Gaussians, and it also enables high-quality scene reconstruction, effectively capturing detailed areas.

With the obtained N number of pseudo GT 3D Gaussians, we measure IoU by considering the volumetric significance of each Gaussian. We define the significant score d_i for each Gaussian θ_i with its scale $\mathbf{s}_i = [s_{ix}, s_{iy}, s_{iz}]^\top$ and opacity α_i as $d_i = s_{ix}s_{iy}s_{iz}\alpha_i$ where $s_{ix}s_{iy}s_{iz}$ denotes a relative ellipsoid volume of a Gaussian θ_i . With the obtained significant scores $\mathbf{d} = [d_1, d_2, \dots, d_N]^\top$, we calculate IoU of i -th 3D Gaussians for the label as:

$$\begin{aligned} \text{Intersection}_i &= \mathbf{d} \cdot (\mathbf{l}_i^{\text{pred}} \odot \mathbf{l}_i^{\text{gt}}), \\ \text{Union}_i &= \mathbf{d} \cdot (\mathbf{l}_i^{\text{pred}} + \mathbf{l}_i^{\text{gt}} - (\mathbf{l}_i^{\text{pred}} \odot \mathbf{l}_i^{\text{gt}})), \\ \text{IoU}_i &= \text{Intersection}_i / \text{Union}_i, \end{aligned} \quad (9)$$

where $\mathbf{l}_i^{\text{pred}} \in \mathbb{R}^N$ and $\mathbf{l}_i^{\text{gt}} \in \mathbb{R}^N$ are binary vectors indicating whether the predicted/GT label of each Gaussian is the n -th label, \mathbf{s}^θ in Eq. (8). The proposed metric is designed to assign a larger weight to the Gaussians with higher significant scores when measuring IoU, and the significant score endows our metric with volume-awareness.

Volume awareness of the proposed metric. To validate that the proposed metric can effectively approximate the

volumetric IoU of the 3D scene, we compare our metric with another volume-aware IoU measurement based on voxel representation. Before measuring IoU with voxels, we train 3D Gaussians and generate labeled pseudo-GT 3D Gaussians with Eq. (8). Then we first sample voxels in the scene, and allocate a GT label to each voxel with the labeled 3D Gaussians. We obtain the most likely label of each voxel by defining the label score. The label score l_{jn}^{voxel} is computed with the opacity α_i and the density $\mathcal{N}(\mathbf{v}_j | \boldsymbol{\mu}_i, \Sigma_i)$ of each Gaussian at the position of a voxel \mathbf{v}_j as:

$$l_{jn}^{\text{voxel}} = \sum_{\theta_i \in \Theta} \alpha_i \cdot \mathbb{1}\{\mathbf{s}^{\theta_i} = \mathbf{s}\} \cdot \mathcal{N}(\mathbf{v}_j | \boldsymbol{\mu}_i, \Sigma_i), \quad (10)$$

where $\mathbb{1}\{\mathbf{s}^{\theta_i} = \mathbf{s}\}$ is an indicator function determining whether a Gaussian θ_i is assigned to the n -th label and $\det(\Sigma_i)$ is the determinant of Σ_i . With the obtained score, we first filter out empty voxels by thresholding with: $p_j = \sum_{n=1}^L l_{jn}^{\text{voxel}}$, where L is the total number of the labels, which can be interpreted as a density of each voxel \mathbf{v}_j . Then we assign a label with the highest score, as the GT label of each voxel. We can also generate predicted labels of voxels using the predicted labels of Gaussians in the same manner, and can evaluate IoU by comparing the GT and predicted labels of the voxels one-to-one.

Volume awareness is inherent in this voxel-based IoU evaluation as the voxels explicitly represent the volume of the scene. We show the volume-awareness of our evaluation metric by showing a correlation between our metric and voxel-based metric in Fig. S3. As can be seen, our metric obtains a high correlation with the voxel-based IoU evaluation metric by considering the significant score when calculating IoU. This result shows the necessity of the significant score, which endows our metric with volume awareness.

Although the voxel-based IoU evaluation effectively measures volume-aware IoU of the scene, the computational cost to assign labels is too expensive. Each time new labels of Gaussians are predicted, the process of assigning them to the voxels is required for evaluation. Different from voxel-based IoU evaluation, our IoU evaluation protocol has a low computational cost, since there is no repeated assignment process after we once generate the labeled pseudo-GT Gaussians. In other words, our proposed IoU evaluation protocol is a fast and volume-aware evaluation for measuring the IoU of scenes represented by 3D Gaussians.

D. Search-time Experiments

In addition to its memory efficiency, Product Quantization significantly enhances search speed. Product quantization can approximate distances between vectors using quantized sub-vectors. By precomputing and storing distances between subvector centroids in a Look-Up Table (LUT), distance calculations between query and database vectors during the

	OpenGaussian evaluation	Our evaluation	OpenGaussian	Ours
	OpenGaussian	Ours	OpenGaussian	Ours
IoU > 0.15	52.7	54.3	57.8	52.6
IoU > 0.30	36.4	39.4	38.0	40.3
IoU > 0.45	14.7	15.5	18.3	25.6
3D mIoU	23.1	25.0	25.2	25.4

Table S3. We compare different metrics for measuring IoU, proposed by OpenGaussian [39] and our work.

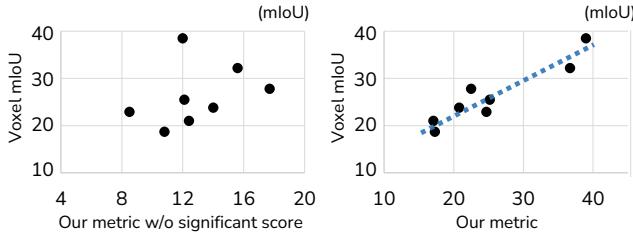


Figure S3. Scatter plot of mIoUs with different mIoU evaluation protocols, measured from eight scenes of the ScanNet [4] dataset. (left) Low correlation between voxel-based metric and our metric without significant score, *i.e.*, same score d_i for all Gaussians. (right) High correlation between voxel-based metric and our metric.

search phase are reduced to simple indexing operations. The precomputation shifts the complexity of vector distance calculations from $O(ND)$ for a D dimensional vector to $O(N)$ per subvector.

Use of LUT can be described as follows. For trained PQ centroids c_{lj} , for $l = 1, 2, \dots, L$, and $j = 1, \dots, 2^k$, where L is number of sub-vectors, and k refers the number of bits used for indexing each centroids. LUT is stored as follows:

$$\text{LUT}_l[i, j] = \|c_{li} - c_{lj}\|_2^2, \text{ where } i, j \in \{1, 2, \dots, 2^k\}. \quad (11)$$

Then for vectors, $\mathbf{v}_1 = [\mathbf{v}_{11}, \dots, \mathbf{v}_{1L}]$, $\mathbf{v}_2 = [\mathbf{v}_{21}, \dots, \mathbf{v}_{2L}]$ mapped to indices $j_1 = [j_{11}, j_{12}, \dots, j_{1L}]$, and $j_2 = [j_{21}, j_{22}, \dots, j_{2L}]$, distance is computed as summation of each retrieved LUT values following each PQ indices:

$$d(\mathbf{v}_1, \mathbf{v}_2) = \sum_{l=1}^L \text{LUT}_l(j_{1l}, j_{2l}). \quad (12)$$

We can also compute the cosine similarity of the vectors, by computing inner products rather than distances, following normalization by the sum of each norm of each sub-vector. Despite the quantization errors, previous literature [14] shows that these errors remain within certain quantization bounds, preserving the correlation between the approximated and actual distances.

The scalability and speed of the proposed approach make it particularly suitable for handling complex 3D data. We compared search speed between computing cosine similarity

of CLIP features and distance computation in product quantization (see Fig. S4). Under identical hardware conditions, the proposed LUT-based approach demonstrated substantial speed improvements compared to cosine similarity computation between CLIP features: with a subvector size of 128, 64, 32 search performance improved by approximately $2\times$, $6.6\times$, $14.1\times$ respectively. These improvements underscore the computational advantages of the proposed method.

Considering that rendering-based methods require significantly greater computation compared to 3D data processing, Dr. Splat's approach demonstrates its superior efficiency in search efficiency, and establishes itself as a practical and scalable solution for 3D data search and processing at scale.

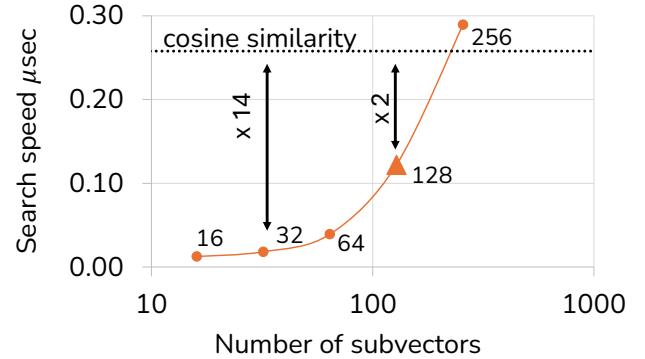


Figure S4. We compare inference speed between product quantization LUT based method and ordinary cosine similarity calculation. We calculate average inference time spent over one million feature points. We report mean values over 100 repeated experiments

E. Additional Results

In this section, we present additional results that are not shown in the manuscript due to space constraints.

E.1. Additional results on presented 3D tasks

We first show more experimental results for the 3D object selection task in Fig. S5, and the 3D localization task in Fig. S6 which are not included in the manuscript due to the space limit. Consistent with our earlier observations, the LangSplat model struggles to learn accurate 3D features. While it occasionally follows feature patterns, it frequently produces significant noise, making it unsuitable for real-world applications such as localization, object grabbing, or 3D image editing. Additionally, we observe persistent spatial bias in the OpenGaussian method, as previously noted, see red cup, plate, or wavy noodles, and bed cases in Fig. S6, it fails to select relevant regions in others. In contrast, our proposed method, which allows direct search and inference in 3D space, consistently identifies favorable localization performance. This demonstrates the robustness and practicality of our approach compared to competing methods.

	3D mIoU	200 classes		
		IoU > 0.15	IoU > 0.3	IoU > 0.45
LangSplat-m [30]	3.9	7.6	3.5	0.8
LEGaussians-m [35]	4.0	7.4	3.8	1.4
OpenGaussian [39]	14.7	34.2	18.9	11.0
Ours (Top-20)	14.6	36.3	18.6	9.4
Ours (Top-40)	14.9	36.0	19.3	14.0

Table S4. We compare evaluate our method with previous methods on the ScanNet-200 dataset.

In the Sec.C, we demonstrated that our metric provides superior volumetric alignment compared to existing approaches. To further validate the superiority of our model, we also evaluated its performance using the metric proposed by OpenGaussian. We confirm that our method outperforms even using other evaluation protocols as shown in Table S3

E.2. Experiments on the ScanNet200 dataset

The proposed model and its counterparts are designed to operate effectively in open-vocabulary settings. To evaluate performance under more comprehensive open-vocabulary cases, we conducted additional experiments using the ScanNet-200 annotation [34], which extends the ScanNet limited-label of 20 to 200 semantic categories, including tail categories such as armchair and windowsill. These rare classes provide a closer approximation to real-world scenarios and enable a robust assessment of the models’ generalization capabilities. For consistency, experiments are conducted using the same scenes as previous benchmarks, following ground truth annotations as described in Sec. C.

The results, summarized in Table S4, demonstrate that the proposed model consistently outperforms its counterparts, which highlights superior generalization across diverse object spaces. The results validate the proposed model’s ability to excel across both constrained and diverse object spaces, emphasizing its potential for practical application in complex real-world scenarios.

E.3. Experiments on the city-scale dataset

The proposed method is further evaluated in a large-scale scenario using the Waymo San Francisco Mission Bay dataset [37], which features expansive spatial contexts. For each scene, the dataset comprises approximately 12,000 images captured by 12 cameras, providing a challenging and diverse testing environment for 3D localization tasks. We select 3 blocks of the scene for large-scale scene tests.

We conducted comparisons against the LangSplat-m model for the 3D text-query localization task as shown in Fig. S7. Our evaluation focused on qualitatively assessing how well each model performs in localizing queries within the 3D space. Our method consistently succeeds in localizing diverse text queries, demonstrating robust and accurate performance across various contexts. In contrast, LangSplat-

m struggles to make precise predictions, particularly with its 3D Gaussian representations failing to align with the expected ground-truth values. These findings are consistent with our earlier observations regarding the limitations of LangSplat-m’s approach.

As shown in Fig. S8, we can see that the results reflect not only objects, but also attributes like color to some extent. Additional visualizations of the results can be found in Fig. S9 and the supplementary video, which provides a more comprehensive view of the qualitative differences between the methods. We strongly encourage readers to refer to these supplementary materials for further insights.

The differences between the methods become even more pronounced when considering search speed in large-scale scenarios. For example, the Waymo dataset contains over 2.9M Gaussians, with individual images requiring nearly 1M computations per image for over 100 images. The computational efficiency of the proposed method allows it to handle such large-scale data more effectively, highlighting its scalability and practical applicability in real-world scenarios.

F. Broader Applications and Limitations

Broader application. The proposed method offers the potential for broader applications across diverse scenarios. Similar to works that explore the application in point cloud [13, 27] and MLP-based methods [32], our approach, using 3DGS, can be extended to support various input modalities, such as click or image queries, by leveraging a self-referencing mechanism. Additionally, integrating our method with Large Language Models (LLMs) could facilitate dialogue-based interactions, allowing users to dynamically issue commands or explore the environment. This integration suggests promising avenues for developing 3D interactive systems that go beyond simple search tasks.

Furthermore, applying the method to canonical forms could support dynamic 3D scenes [25, 41]. This adaptation would extend the applicability of our approach beyond static environments, demonstrating its versatility in handling complex, real-world scenarios.

Limitation. While our method has demonstrated robust performance across diverse combinations of nouns and adjectives (e.g., “tea in a glass,” wavy noodles,” and red light” in Fig. S5 and Fig. S9) as well as unfamiliar nouns (e.g., nori,” waldo,” and safety cone”), without additional training, generalization remains an area for improvement. Exploring additional training techniques for Product Quantization (PQ) could further enhance the method’s capabilities. Further exploration of Product Quantization (PQ) training, such as using more diverse datasets or finer-grained query representations, could enhance adaptability across varied contexts.

Despite its advantages, some limitations of the proposed method have also been identified, particularly related to

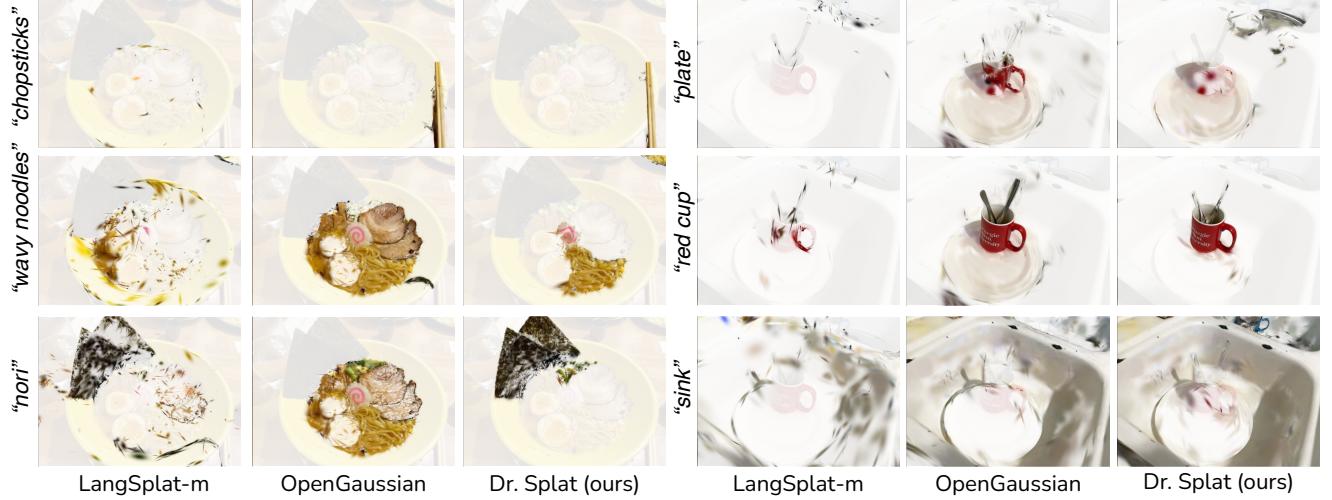


Figure S5. We compare 3D object selection task in LeRF dataset with Langsplat [30], and OpenGaussian [39]. We visualize selected Gaussians with high similarity to query text. Langsplat shows noisy, 3D uncorrelated activations, and Opengaussion often show false positive activations, while our method show accurate localization showing superiority on generalizability.

CLIP features. Occasionally, related but distinct objects are simultaneously activated for a given query. For instance, the query “red apple” might activate non-red apples or unrelated red objects. This stems from CLIP’s semantic associations and could be mitigated with post-processing techniques like re-ranking to improve query specificity.

Lastly, similar to previous methods [30, 35, 39], ours also requires to set an appropriate threshold. In this study, we utilize a fixed similarity threshold employed a fixed similarity threshold across all scenes, ensuring stable and reproducible results. However, optimizing thresholds for specific scenarios or implementing dynamic adjustments could further refine localization accuracy in diverse environments.

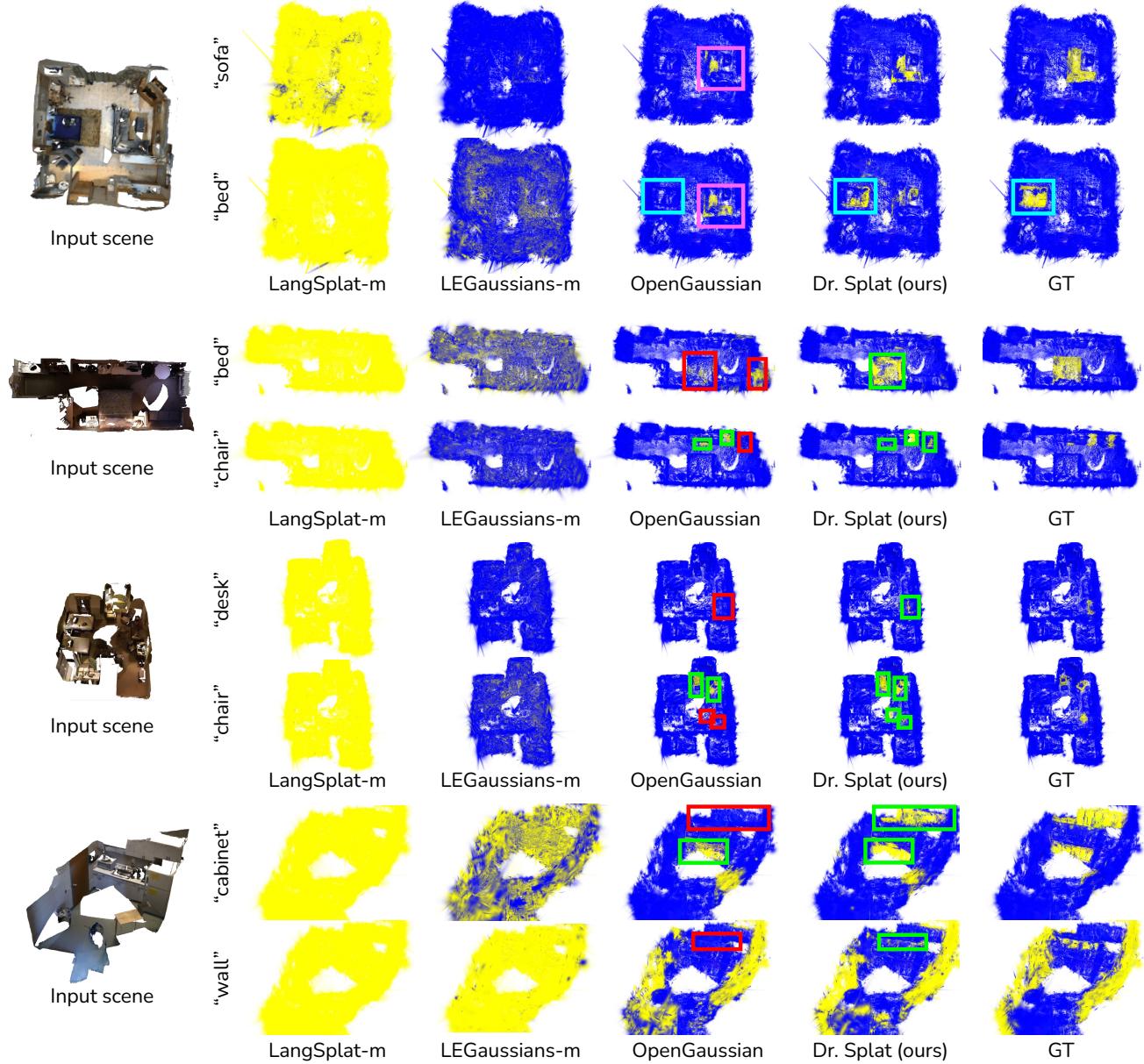


Figure S6. We compare 3D object localization results between competing methods [30, 35, 39] with Dr. Splat. 3DGs with similarity above the threshold (0.562) are shown in yellow, while those below the threshold are displayed in blue. Green box indicates successful localization, while red boxes indicates missing or false positive in 3D localization.

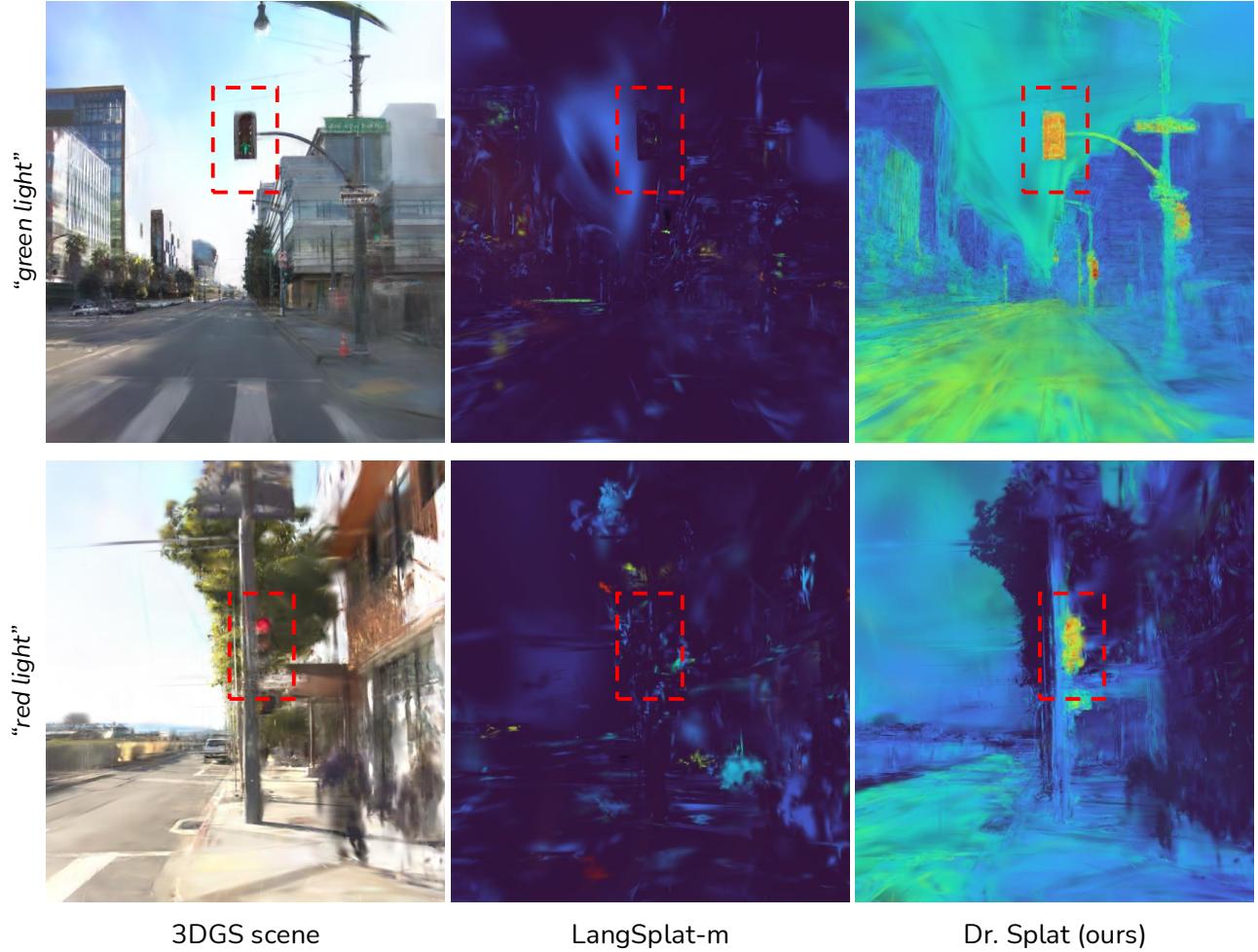


Figure S7. We compare 3D localization between rendering based LangSplat-m with registration based Dr. Splat. While LangSplat-m shows randomly distributed activations, fail to localize the target, ours model successfully detect the target in both cases.

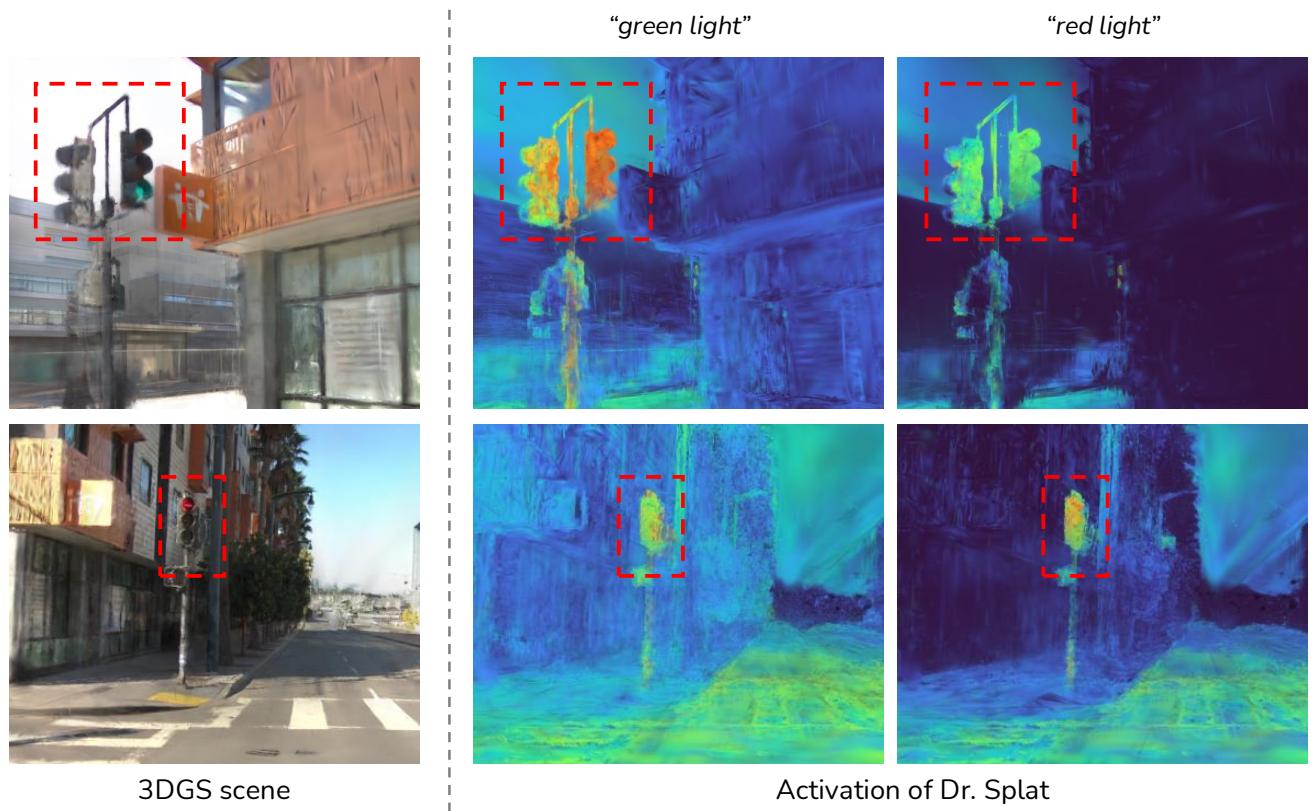


Figure S8. Visualization of 3d localization in different attributes (e.g., color) given as query. The result highlights the ability of Dr. Splat (ours) to effectively distinguish attributes such as “green light” and “red light” in scenes based on text queries, demonstrating the robustness in open-vocabulary understanding.



Figure S9. Qualitative results of Dr. Splat on 3D localization task in city-scale data showcasing Dr. Splat’s generalization performance across diverse text queries includes various target objects and concepts.