SPARSE CODE UPLIFTING FOR EFFICIENT 3D LANGUAGE GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

3D Language Gaussian Splatting (3DLGS) augments 3D Gaussian Splatting with language-aligned feature representations for open-vocabulary 3D scene understanding. A core challenge is efficiently associating high-dimensional vision-language embeddings with millions of 3D Gaussians while preserving efficient feature rendering for text-based querying. Existing methods either store dense features directly on Gaussians, causing high storage costs and slow rendering, or learn compact representations through expensive per-scene optimization with repeated feature rasterization. No existing method simultaneously achieves fast 3D semantic reconstruction, efficient storage, and fast rendering. We propose **SCOUP** (**S**parse **CO**de **UP**lifting), which addresses all three by decoupling language representation learning from 3D Gaussian optimization. Rather than working directly in 3D, we learn sparse codebook-based representations entirely using features associated with 2D image regions, associating each region with a sparse set of codebook coefficients. We then uplift these coefficients to 3D Gaussians with our weighted sparse aggregation using Gaussian-to-pixel associations, where each Gaussian accumulates coefficients over codebook atoms across views. Top-K filtering then extracts the most dominant multi-view coefficients per Gaussian, enabling efficient storage and fast rendering. Our method achieves more than 400× training speedup while being 3× more memory efficient during training compared to the state-of-the-art in rendering speed. Across multiple benchmarks, SCOUP matches or outperforms existing methods in open-vocabulary querying accuracy.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) has emerged as a powerful tool that enables high-quality reconstruction of 3D scenes from multi-view 2D images (Kerbl et al., 2023). 3DGS represents scene appearance and geometry with an explicit set of anisotropic Gaussians that can be rasterized in real time on modern GPUs. Beyond these visual and spatial properties, these 3D representations are increasingly extended with language-aligned visual features extracted from vision–language models (VLMs) such as CLIP (Radford et al., 2021; Ilharco et al., 2021). This creates 3D Language Gaussian Splatting (3DLGS) scenes built on top of 3DGS that can be queried with arbitrary text prompts. Recent methods (Qin et al., 2024; Shi et al., 2024; Li et al., 2025) demonstrate that 3DLGS enables open-vocabulary 3D understanding and language-guided interaction, with valuable applications in language-driven navigation, manipulation, and interaction in robotics and AR/VR settings (Huang et al., 2023; Jatavallabhula et al., 2023; Wang et al., 2024b; Tu et al., 2025).

3DLGS aims to build a 3D representation that, when rendered, produces high-dimensional features comparable with text embeddings to enable open-vocabulary scene querying. Typical pipelines use SAM (Kirillov et al., 2023) to segment each scene image into regions, then CLIP (Radford et al., 2021) to generate 512-dimensional embeddings per region (Shi et al., 2024; Qin et al., 2024). These features are then optionally compressed (e.g., via a scene-specific autoencoder) before being used for training or directly uplifted to 3D Gaussians. The Gaussians' location, covariance, and opacity are inherited from the pre-trained 3DGS (Qin et al., 2024; Marrie et al., 2025; Lee et al., 2025). During rendering, the features stored in each Gaussian are rasterized, and if compression was applied, an additional decoding step is required to recover the original feature for text-based feature comparison and querying (Qin et al., 2024; Li et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

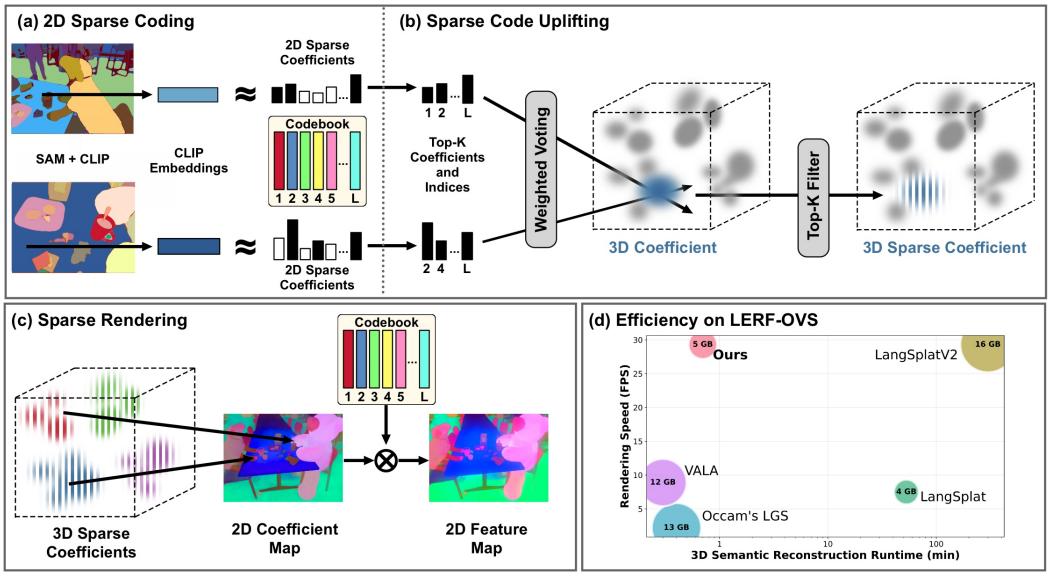

Figure 1: **SCOUP** reconstructs 3D semantic scene representations in three stages: (a) jointly optimizing a scene-aware codebook and sparse coefficient maps to represent dense 2D CLIP features; (b) a weighted semantic voting process that efficiently uplifts only non-zero coefficients to 3D Gaussians, followed by top-K filtering to enforce multi-view consistency by retaining the dominant coefficients across views. During inference, (c) the sparse codes stored in each Gaussian enable fast language feature rendering, resulting in the best efficiency (d) for 3D Language Gaussian Splatting.

There are two major challenges in 3DLGS scene reconstruction and rendering. First, 3D semantic reconstruction can incur significant latency and/or memory usage depending on the technique used. One commonly adopted approach is to directly uplift high-dimensional features from 2D images to 3D Gaussians (Marrie et al., 2025; Cheng et al., 2025; Wang et al., 2026), leveraging Gaussian-to-pixel associations readily available from the geometric and visibility priors of a pretrained 3DGS model. Despite being fast, this process can lead to high memory usage when Gaussian-pixel associations are complex, and the resulting 3DLGS model incurs high storage costs due to millions of Gaussians each storing a full high-dimensional feature vector. Alternatively, training-based methods that compress high-dimensional features using scene-specific autoencoders (Qin et al., 2024) successfully mitigate memory overhead, but incur significantly longer 3D reconstruction runtimes due to the computational burden of repeated feature rasterization during optimization. Second, rendering is typically slow due to either the need to rasterize high-dimensional features directly, or the need to decode rasterized low-dimensional features with a neural network (Cheng et al., 2025; Zhou et al., 2024; Qin et al., 2024). LangSplatV2 (Li et al., 2025) addresses this by associating each Gaussian with sparse coefficients over a shared high-dimensional codebook, replacing the expensive neural decoder with a single matrix multiplication between the coefficients and the codebook. LangSplatV2 uses the high-dimensional embeddings from 2D images as ground-truth supervision, jointly optimizing the codebook and sparse Gaussian coefficients. However, this results in slow training due to the difficulty of jointly optimizing the codebook and sparse coefficients alongside repeated rasterization during training. One alternative would be to first uplift high-dimensional features from 2D to 3D, then train the codebook and sparse coefficients against these uplifted 3D features. However, our investigations found this degrades accuracy as Gaussians associated with larger objects outnumber those associated with smaller ones. This causes larger objects to dominate the codebook updates, resulting in poor feature representation for small objects. In summary, *existing works do not achieve fast 3D semantic reconstruction, efficient storage, and fast rendering simultaneously*.

Our goal in this work is to simultaneously enable fast 3D semantic reconstruction, efficient storage, and fast rendering for 3DLGS, while achieving state-of-the-art accuracy. We propose **S**parse **Co**de **Up**lifting (SCOUP) for efficient 3DLGS. Rather than optimizing language representations directly in 3D Gaussian space, our key idea is to first jointly train a shared high-dimensional codebook and sparse coefficients in CLIP's native 2D space, where each SAM-segmented region is associated with

a sparse set of codebook coefficients to reconstruct its CLIP embedding (Fig. 1(a)). We then uplift the learned 2D region coefficients to 3D Gaussians by our weighted sparse aggregation: by assigning all pixels within a region the same set of sparse coefficients, the Gaussian-to-pixel associations from a pretrained 3DGS model can be efficiently used to aggregate coefficients from 2D to 3D without any additional optimization. As each Gaussian aggregates contributions across multiple views, the accumulated coefficients implicitly form a weighted semantic voting system: high-weight coefficients represent confident consensus, while low-weight coefficients represent multi-view noise. To denoise the uplifted coefficients, we retain only the top-K coefficients by total aggregated weight, filtering the rest to zero (Fig. 1(b)). This ensures each Gaussian stores only K coefficient weights, enabling efficient storage and fast rendering (Fig. 1(c)).

Our method achieves fast 3D semantic reconstruction by exploiting three properties. First, SAM (Kirillov et al., 2023) segments each scene into only a limited number of regions, keeping the 2D coefficient optimization significantly smaller than optimizing coefficients directly in 3D across millions of Gaussians. Second, we avoid the expensive repeated rasterization required by existing works (Li et al., 2025; Zhou et al., 2024; Qin et al., 2024), which arises from optimizing coefficients associated with 3D Gaussians against 2D image embedding supervision. Third, the learned sparse coefficients are efficiently uplifted to 3D Gaussians, requiring no additional optimization and no high-dimensional feature rasterization. Our 3D semantic reconstruction is also highly memory efficient: each Gaussian aggregates and stores only sparse coefficients instead of full high-dimensional feature vectors, avoiding the costly high-dimensional aggregation or optimization across millions of Gaussians that prior works require. Our method also matches or outperforms existing works in accuracy, as the codebook is trained to directly reconstruct 2D CLIP features, which are the representations that text embeddings are compared against at query time. Our sparse coefficient uplifting is nearly lossless given accurate Gaussian-to-pixel associations, and as we will demonstrate experimentally, treating multi-view aggregation as a voting system allows our top-K selection to act as an effective noise filter. By discarding low-weight coefficients, we intrinsically improve multi-view consistency and open-vocabulary querying performance.

We demonstrate the effectiveness of our method across three datasets: LERF-OVS (Kerr et al., 2023), 3D-OVS (Liu et al., 2023), and Mip-NeRF360 (Barron et al., 2022). Compared to LangSplatV2, our method achieves more than $400\times$ 3D semantic reconstruction speedup while maintaining identical storage and rendering speed, and is $3\times$ more efficient in training memory. Compared to the fastest uplifting-based methods (Wang et al., 2026; Cheng et al., 2025), our reconstruction time remains under a minute, while rendering at least $3\times$ faster (Fig. 1(d)). In terms of accuracy, we outperform LangSplatV2 (Li et al., 2025) and VALA (Wang et al., 2026), the leading optimization-based and uplifting-based 3DLGS methods, by +4.5% and +2.7% in mIoU, respectively, on open-vocabulary segmentation on LERF-OVS.

In summary, our key contributions are: **(1)** To our knowledge, we propose the first framework that encodes 2D high-dimensional dense features into 2D sparse coefficient maps and uplifts them to a 3DGS scene for efficient 2D-to-3D and 3D-to-2D transfer and storage of language features. **(2)** We propose two novel techniques: (i) 2D Sparse Coding that jointly trains a shared codebook and per-region sparse codes to reconstruct CLIP features and (ii) Sparse Code Uplifting formulated as a weighted voting process over 2D codebook atoms guided by 3DGS priors, followed by top-K consensus filtering for compact storage, efficient rendering, and semantic noise reduction. **(3)** Our method achieves more than $400\times$ training speedup and uses a third of the training memory compared to LangSplatV2, while maintaining its rendering speed and storage, and matches or outperforms other baselines on 3D open-vocabulary querying across benchmarks.

2 Related Work

3D Gaussian Splatting. In contrast to the implicit representations used in Neural Radiance Fields (NeRFs) (Mildenhall et al., 2020), 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has emerged as a high-efficiency alternative for 3D scene reconstruction from 2D images. By representing 3D space through explicit anisotropic Gaussian primitives and leveraging a differentiable rasterization pipeline, 3DGS achieves real-time rendering speeds. This has motivated the research community to extend 3DGS capabilities to support various open-vocabulary tasks in 3D space, from editing (Chen et al., 2024a; Wang et al., 2024a; Kim et al., 2024; Xiao et al., 2025; Yan et al., 2024) and

generating (Chen et al., 2024b; Tang et al., 2023; Liang et al., 2024) objects using textual prompts, to embedding language representations into 3DGS to support open-vocabulary querying (Qin et al., 2024; Zhou et al., 2024; Shi et al., 2024).

3D Language Gaussian Splatting. Transferring knowledge from 2D vision-language models (VLMs) (Radford et al., 2021; Li et al., 2022; Jose et al., 2025) into 3D scene representations is a critical step for unlocking open-vocabulary capabilities. Due to the explicit nature of 3DGS, storing high-dimensional features within individual Gaussians has led to two primary paradigms for 2D-to-3D feature transfer: optimization-based (Qin et al., 2024; Zhou et al., 2024; Shi et al., 2024; Dai et al., 2025; Li et al., 2026; 2025; Qu et al., 2024; Tian et al., 2025) and optimization-free (Marrie et al., 2025; Jun-Seong et al., 2025; Cheng et al., 2025; Wang et al., 2026) methods.

Optimization-based methods learn language features through a differentiable rasterization pipeline (Kerbl et al., 2023). To speed up training and reduce the memory footprint, LangSplat (Qin et al., 2024) trains a scene-specific autoencoder to compress CLIP embeddings into a low-dimensional latent space, which is then used to distill knowledge into 3D Gaussians. While this extreme compression reduces memory requirements, decoding the rendered latent representations back to the original CLIP space results in slow inference and a loss of semantic information. LangSplatV2 (Li et al., 2025) and LEGaussians (Shi et al., 2024) adopt codebook-based approaches where Gaussians reference a global feature dictionary. Specifically, LEGaussians quantizes 2D features into a discrete codebook, then learns compact per-Gaussian features that can be decoded to codebook-index predictions. In contrast, LangSplatV2 (Li et al., 2025) represents each Gaussian as a weighted combination of multiple codebook entries. While this offers greater representational capacity than Shi et al. (2024), LangSplatV2 suffers from the computationally costly joint optimization of codebook entries and Gaussian coefficients through the rasterization pipeline. While codebook-based formulations have been explored by Qu et al. (2024), Tian et al. (2025),Alegret et al. (2026),Li et al. (2025), and Shi et al. (2024), no prior work leverages the codebook to construct 2D sparse coefficient maps.

Recent works propose training-free feature uplifting to pre-trained 3DGS models by leveraging their geometric and visibility priors (Marrie et al., 2025; Jun-Seong et al., 2025; Cheng et al., 2025; Wang et al., 2026). LUDVIG (Marrie et al., 2025) performs a weighted multi-view aggregation of 2D features with a reported speed of 2 ms per image and feature dimension. Dr. Splat (Jun-Seong et al., 2025) uses a similar formulation but assigns language representations only to the most dominant Gaussians along a viewing ray. To reduce storage requirements and improve scalability, Dr. Splat utilizes Product Quantization (PQ) to store features as compact representations. Jun-Seong et al. (2025) and Marrie et al. (2025) query objects directly in 3D space prior to rendering the results. VALA (Wang et al., 2026) speeds up feature uplifting by filtering out occluded and low-visibility Gaussians during the aggregation procedure, yet it uplifts and renders full-dimensional CLIP features. Occam's LGS (Cheng et al., 2025) adds support for arbitrary feature dimensionality uplifting with scene-specific autoencoders, as in LangSplat (Qin et al., 2024). On the other hand, CF^3 (Lee et al., 2025) moves the autoencoder directly into 3D space, learning low-dimensional latent representations for every Gaussian, which increases semantic reconstruction time. To further improve the efficiency of 3DGS, methods by Lee et al. (2025) and Marrie et al. (2025) rely on filtering or adaptive sparsification to reduce the memory footprint. In contrast, our method optimizes the CUDA implementation of feature uplifting (Marrie et al., 2025) to support sparse codes, enabling efficient uplifting of high-dimensional language representations to a 3DGS scene and supporting LangSplatV2 (Li et al., 2025) level rendering throughput in a fraction of the time.

3 PRELIMINARIES

3D Gaussian Splatting (3DGS) represents a scene as a large collection of anisotropic 3D Gaussians (Kerbl et al., 2023). Each Gaussian is defined by a 3D center coordinate $\boldsymbol{\mu}_i \in \mathbb{R}^3$, a 3D size scaling factor $\mathbf{s}_i \in \mathbb{R}^3$, a rotation quaternion $\mathbf{q}_i \in \mathbb{R}^4$, a view-dependent color $\mathbf{c}_i(v) \in \mathbb{R}^3$ described via spherical-harmonic coefficients, and an opacity value $o_i \in \mathbb{R}$. All parameters are learnable and are collectively denoted as

$$\boldsymbol{\Theta} = \{\, \boldsymbol{\mu}_i,\ \mathbf{s}_i,\ \mathbf{q}_i,\ \mathbf{c}_i(\cdot),\ o_i \,\}_{i=1}^{N}$$

for a scene with N Gaussian points.

Optimization of 3DGS parameters relies on a differentiable tile-based rasterizer to project 3D Gaussians onto a 2D image (Kerbl et al., 2023). Following Kopanas et al. (2021) and Kerbl et al. (2023), given a camera pose with viewing direction v, the estimated color of pixel p in an image $\mathbf{I}_v \in \mathbb{R}^{H\times W\times 3}$ is defined as:

$$\mathbf{I}_v(p) = \sum_{i\in\mathcal{N}_{v,p}} \mathbf{c}_i(v)\; e_i(v,p), \tag{1}$$

where $\mathcal{N}_{v,p}$ denotes the set of Gaussians whose projected splats overlap pixel p, ordered from front to back along viewing ray v, and $e_i(v,p)$ is the contribution weight of the i-th Gaussian in this ordered list. The weights are obtained via standard α-blending (Zwicker et al., 2001):

$$e_i(v,p) = \alpha_i(v,p)\;\prod_{j=1}^{i-1}\big(1-\alpha_j(v,p)\big), \tag{2}$$

where $\alpha_i(v,p) = o_i\, G_i^{2D}(v,p)$ is the alpha value of the i-th Gaussian given its opacity o_i and projected 2D footprint $G_i^{2D}(v,p)$.

Feature Uplifting (Marrie et al., 2025; Jun-Seong et al., 2025) is a training-free direct registration of language features from 2D feature maps to the 3DGS scene. Following Marrie et al. (2025), $\mathcal{S}_i$ denotes the set of all view-pixel pairs (v,p) to which the i-th Gaussian contributes during rendering. The language feature vector $\mathbf{f}_i \in \mathbb{R}^{512}$ for the i-th Gaussian is defined as:

$$\mathbf{f}_i = \frac{1}{Z_i}\sum_{(v,p)\in\mathcal{S}_i} e_i(v,p)\mathbf{F}_v(p), \tag{3}$$

where $Z_i = \sum_{(v,p)\in\mathcal{S}_i} e_i(v,p)$ is the normalization factor, and $\mathbf{F}_v(p) \in \mathbb{R}^D$ is the semantic feature vector at pixel p in view v. Storing a full D-dimensional feature vector per Gaussian scales storage linearly with D, and directly replacing the color term $\mathbf{c}_i(v)$ in Equation 1 with $\mathbf{f}_i$ similarly increases rendering time proportionally to the feature dimension.

4 METHODOLOGY

4.1 2D SPARSE COEFFICIENT MAPS

Following prior work (Qin et al., 2024; Li et al., 2025), we extract high-dimensional 2D semantic feature maps using SAM (Kirillov et al., 2023) and CLIP (Radford et al., 2021). Specifically, for each input image $\mathbf{I}_v$, SAM generates a set of non-overlapping binary masks. Each masked region is encoded with CLIP to obtain a semantic feature vector, which is then assigned to all pixels within the corresponding mask, yielding a pixel-wise semantic feature map $\mathbf{F}_v \in \mathbb{R}^{H\times W\times D}$. Following existing work, we repeat this process at three SAM granularity levels: fine, middle, and coarse, to capture multi-scale semantic information. For notational clarity, the remainder of this section describes our method with respect to a single semantic level.

The semantic representation of a single 3D scene is highly redundant and occupies only a small subspace of the full CLIP feature space (Shi et al., 2024), making sparse coding a natural fit for compact scene-specific feature representation. Unlike autoencoder-based compression (Qin et al., 2024; Zhou et al., 2024), which requires a neural decoder at render time, sparse coefficients can be decoded via a simple matrix multiplication with the codebook. We therefore encode dense 2D feature maps into 2D sparse coefficient maps, where each pixel is represented as a sparse convex combination of at most K basis vectors from a learned scene-aware codebook, restricting each pixel to at most K non-zero coefficient values.

We formulate this as a 2D sparse coding problem, learning a scene-aware codebook $\mathbf{C} \in \mathbb{R}^{L\times D}$ of L basis vectors. Each pixel-wise feature $\mathbf{F}_v(p) \in \mathbb{R}^D$ is approximated as a sparse convex combination of at most K basis vectors:

$$\mathbf{F}_v(p) \approx \mathbf{W}_v(p)\;\mathbf{C}, \quad \text{s.t.} \quad \|\mathbf{W}_v(p)\|_0 \le K, \quad \mathbf{W}_v(p) \ge \mathbf{0}, \quad \mathbf{1}^\top\mathbf{W}_v(p) = 1, \tag{4}$$

where $\mathbf{W}_v(p) \in \mathbb{R}^L$ is the sparse coefficient vector at pixel p. Since CLIP features are constant within a SAM region, we parameterize coefficients per region rather than per pixel, so all pixels in a region share $\mathbf{W}_v(p)$. To enforce these constraints, we apply a normalized soft top-K softmax

over learnable region-wise logits, optimizing these logits jointly with the codebook via a cosine reconstruction loss.

Crucially, while LangSplatV2 (Li et al., 2025) couples codebook learning with 3DGS optimization, our sparse coding operates entirely in image space, prior to 3D lifting, decoupling semantic compression from the 3DGS pipeline. This places the scene-aware codebook in the same 2D domain on which CLIP (Radford et al., 2021) is trained. The number of extracted masked regions is much smaller than the number of Gaussians, yielding fewer features to compress, while region-based CLIP supervision ensures that small and large objects contribute with more balance to the codebook optimization.

4.2 Sparse Coefficient Uplifting

By exploiting the sparsity of our 2D coefficient maps, we uplift only the K non-zero coefficient values to 3D Gaussians rather than full D-dimensional feature vectors. This bounds the computational cost of the lifting process strictly by K, independent of the original feature dimension D, yielding a D/K speedup over dense feature uplifting ($K \ll D$).

We reformulate the dense feature uplifting from Equation 3 as sparse coefficient uplifting:

$$\mathbf{w}_i = \frac{1}{Z_i} \sum_{(v,p)\in\mathcal{S}_i} e_i(v,p)\ \mathbf{W}_v(p) \tag{5}$$

where $\mathbf{w}_i \in \mathbb{R}^L$ is the aggregated coefficient vector assigned to the i-th 3D Gaussian. Since each $\mathbf{W}_v(p)$ contains at most K non-zero elements, the weighted aggregation operates only over non-zero entries, reducing the time complexity of uplifting from $\mathcal{O}(|\mathcal{S}_i|D)$ in Equation 3 to $\mathcal{O}(|\mathcal{S}_i|K)$ per Gaussian.

As each viewpoint contributes a different set of active coefficient indices, the magnitude of each entry in $\mathbf{w}_i$ reflects cross-view consensus: codebook atoms supported by many views accumulate large weights, while atoms activated by only a few remain small. We therefore restore strict sparsity by retaining the K dominant coefficients of $\mathbf{w}_i$, using Top-K filtering as a consensus mechanism over the weighted voting process to suppress noisy contributions.

The final sparse coefficient vector $\hat{\mathbf{w}}_i$ is computed as:

$$\hat{\mathbf{w}}_i = \frac{\text{Top}_K(\mathbf{w}_i)}{\|\text{Top}_K(\mathbf{w}_i)\|_1} \tag{6}$$

where $\text{Top}_K(\cdot)$ retains only the K dominant elements and the normalization ensures the resulting sparse weights sum to one. Note that this final normalization subsumes the per-Gaussian scalar Z_i, which can therefore be omitted in practice. This filtering and contribution consolidation mechanism allows each 3D Gaussian to store only K coefficient values and their corresponding indices, resulting in a compact memory footprint, efficient rendering, and better multi-view consistency.

4.3 Sparse Coefficient Rendering

By storing only K non-zero coefficient values per Gaussian, rendering operates on sparse coefficients rather than full D-dimensional feature vectors, accessing only non-zero entries during weighted aggregation. Sparse coefficient rendering is defined as

$$\hat{\mathbf{W}}_v(p) = \sum_{i\in\mathcal{N}_{v,p}} \hat{\mathbf{w}}_i\ e_i(v,p), \tag{7}$$

where $\hat{\mathbf{W}}_v(p)$ is the projected sparse coefficient map on the 2D plane for view v. This sparse aggregation is directly supported by the rendering pipeline from Li et al. (2025), which efficiently accesses only non-zero coefficient entries via their corresponding sparse indices. To recover a 2D semantic feature map of CLIP embeddings from the rendered sparse coefficient map, we perform a simple matrix multiplication with the scene-aware codebook $\mathbf{C}$ learned in Section 4.1:

$$\mathbf{F}_v(p) \approx \hat{\mathbf{W}}_v(p)\ \mathbf{C}, \tag{8}$$

where the reconstructed $\mathbf{F}_v(p)$ can be directly compared with CLIP text embeddings to support open-vocabulary scene querying.

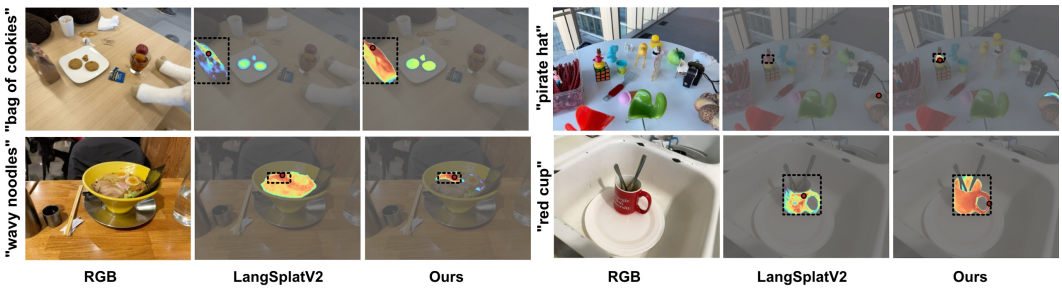

Figure 2: Qualitative results for open-vocabulary 3D object localization on the LERF-OVS dataset.

Table 1: Semantic reconstruction runtime (2D→3D), peak allocated GPU memory during reconstruction (GPU Mem.), language representation rendering speed (3D→2D), and storage requirements for all three semantic levels on LERF-OVS, Mip-NeRF360, and 3D-OVS datasets.

Method	**2D→3D** (min)			**GPU Mem.** (GB)			**3D→2D** (FPS)			**Storage** (GB)		
	LERF	Mip.	3D-OVS	LERF	Mip.	3D-OVS	LERF	Mip.	3D-OVS	LERF	Mip.	3D-OVS
LangSplatV2 (Li et al., 2025)	300	684	570	16.4	37.7	28.5	**29**	**15**	**15**	**0.1**	**0.1**	**0.1**
Occam's LGS (Cheng et al., 2025)	0.4	**0.5**	**0.1**	12.9	28.6	15.6	2	1	1	6.9	11.4	7.6
VALA (Wang et al., 2026)	**0.3**	**0.5**	**0.1**	11.6	15.1	11.6	9	4	4	0.9	3.5	1.9
Ours	0.7	1.4	0.5	**5.2**	**10.4**	**4.4**	**29**	**15**	**15**	**0.1**	**0.1**	**0.1**

5 Experiments

5.1 Datasets and Metrics

We evaluate on three datasets for 3D open-vocabulary scene understanding. **LERF-OVS** (Kerr et al., 2023; Qin et al., 2024) consists of four in-the-wild scenes with ground-truth segmentation masks and localization points for textual queries. Following Li et al. (2025), we report localization accuracy and segmentation IoU. We further evaluate segmentation on **Mip-NeRF360** (Barron et al., 2022) and **3D-OVS** (Liu et al., 2023). For Mip-NeRF360, we use ground-truth annotations provided by GAGS (Peng et al., 2024) and report IoU on four indoor and outdoor scenes. For 3D-OVS, we report segmentation on five scenes containing long-tail objects. We additionally analyze semantic reconstruction runtime and memory requirements on all three datasets.

5.2 Implementation Details

The input to our model is a 3DGS scene trained for 30,000 iterations using only RGB images. Following Qin et al. (2024), we generate segmentation masks with SAM ViT-H (Kirillov et al., 2023) at three semantic levels and encode each mask with OpenCLIP ViT-B/16 (Ilharco et al., 2021) to obtain 512-dimensional CLIP features. We train the 2D sparse coding algorithm independently at each level for 4,000 epochs using Adam (Kingma & Ba, 2015) with a learning rate of 7×10^{-4}, codebook size $L = 64$, and $K = 4$ non-zero coefficients. The codebook is initialized using k-means on the CLIP-encoded masked regions. We then uplift the sparse coefficient maps to 3DGS using our sparse uplifting algorithm and render all three semantic levels simultaneously. Following Li et al. (2025), we compute similarities between the rendered language features and CLIP text embeddings to obtain relevancy masks. All experiments are conducted on an NVIDIA RTX A6000 GPU.

5.3 Runtime and Memory Analysis

To evaluate the efficacy of the SCOUP framework, we recorded 3D semantic scene reconstruction runtimes, peak GPU memory allocation, rendering speeds, and storage requirements for language feature representations across all three semantic levels. To ensure a rigorous evaluation, all baselines were re-measured locally on our GPU using synchronized CUDA clocks. As shown in Table 1, under these standardized conditions, our method achieves rendering speeds of 29 FPS on LERF-OVS and 15 FPS on Mip-NeRF360 and 3D-OVS, while rendering all three semantic levels simultaneously, matching the throughput of LangSplatV2 (Li et al., 2025). Crucially, our approach is more than

Table 2: Quantitative comparisons of open-vocabulary 3D object localization and 3D semantic segmentation on the LERF-OVS dataset.

Method	**3D Object Localization** (mAcc %)					**3D Semantic Segmentation** (mIoU %)				
	Ramen	Teatime	Kitchen	Figurines	Overall	Ramen	Teatime	Kitchen	Figurines	Overall
LEGaussian (Shi et al., 2024)	69.0	79.7	63.6	57.1	67.4	20.2	32.3	22.3	23.4	24.6
GOI (Qu et al., 2024)	56.3	67.8	68.2	44.6	59.2	33.7	55.8	54.5	23.9	42.0
GAGS (Peng et al., 2024)	69.0	88.1	90.9	78.6	81.7	46.8	60.3	55.8	53.6	54.1
LangSplat (Qin et al., 2024)	73.2	88.1	**95.5**	80.4	84.3	51.2	65.1	44.5	44.7	51.4
LangSplatV2 (Li et al., 2025)	74.7	**93.2**	86.4	82.1	84.1	51.8	72.2	59.1	56.4	59.9
Occam's LGS (Cheng et al., 2025)	74.7	**93.2**	81.8	80.4	82.5	51.0	70.2	**65.3**	58.6	61.3
VALA (Wang et al., 2026)	**75.6**	91.5	86.4	82.1	83.9	51.5	70.2	65.1	59.9	61.7
Ours	74.7	91.5	**95.5**	**85.7**	**86.9**	**57.6**	**75.3**	64.4	**60.4**	**64.4**

Table 3: Quantitative 3D semantic segmentation results on Mip-NeRF360 and 3D-OVS datasets. Occam's LGS results are shown in gray due to its different evaluation protocol on 3D-OVS.

Method	**Mip-NeRF360** (mIoU %)					**3D-OVS** (mIoU %)					
	Room	Counter	Garden	Bonsai	Overall	Bed	Bench	Room	Sofa	Lawn	Overall
GOI (Qu et al., 2024)	60.3	46.6	59.8	67.3	58.5	89.4	92.8	91.3	85.6	94.1	90.6
LangSplat (Qin et al., 2024)	53.2	68.8	51.9	55.4	57.3	92.5	94.2	94.1	90.0	96.1	93.4
LangSplatV2 (Li et al., 2025)	64.3	75.1	**65.0**	**73.1**	69.4	**93.0**	**94.9**	**96.1**	**92.3**	96.6	**94.6**
Occam's LGS (Cheng et al., 2025)	-	-	-	-	-	96.8	95.8	96.5	88.8	97.0	95.0
Ours	**66.5**	**82.3**	57.4	71.7	**69.5**	87.7	**94.9**	95.1	88.2	**97.2**	92.6

$400\times$ faster in semantic reconstruction on LERF-OVS and Mip-NeRF360 and more than $1000\times$ faster on 3D-OVS, while being $3\times$ more GPU memory efficient than LangSplatV2. While our uplifting runtime is comparable to training-free methods such as Occam's LGS (Cheng et al., 2025) and VALA (Wang et al., 2026), we report isolated semantic uplifting time for these methods, excluding hardware-dependent I/O costs associated with storing a large number of dense CLIP features in 3DGS. With 2D sparse coding, SCOUP introduces a small overhead during 3D semantic reconstruction, but this represents a deliberate trade-off: by investing slightly more time upfront, we obtain a compact language representation that renders approximately $15\times$ faster than Occam's LGS and $3\times$ faster than VALA. SCOUP stores only four coefficients and their corresponding indices per semantic level for each Gaussian in the underlying 3DGS and is the only method that remains competitive across all four efficiency metrics, making it substantially more practical for real-world deployment. We provide a detailed runtime analysis of 2D sparse coding and sparse code uplifting in Appendix E.

5.4 Main Results

LERF-OVS. Table 2 presents the 3D open-vocabulary localization accuracy and segmentation IoU scores on the LERF dataset (Kerr et al., 2023). For the localization task, our method achieves superior performance on the "Figurines" scene while remaining highly competitive with state-of-the-art results across all other scenes, resulting in the highest overall localization accuracy. Regarding 3D open-vocabulary segmentation, our approach leads to substantial overall improvements, particularly on the "Ramen" and "Teatime" scenes, where it outperforms existing methods by a significant margin. Specifically, in comparison with LangSplatV2 (Li et al., 2025), our method achieves a $+4.5\%$ gain in IoU score, while requiring significantly less time for 3D semantic scene reconstruction.

Mip-NeRF360 & 3D-OVS. Table 3 shows 3D open-vocabulary segmentation results on Mip-NeRF360 and 3D-OVS. SCOUP achieves performance comparable to LangSplatV2 across most scenes. The comparison with Occam's LGS on 3D-OVS is less direct, as Cheng et al. (2025) propose dynamic thresholding during inference, which introduces additional runtime overhead. Both datasets contain few queries per scene, and a single failure can strongly affect mIoU. Notable examples on 3D-OVS are "black leather shoe" in the Bed scene and "Pikachu" in the Sofa scene, where SCOUP either merges distinct semantic elements into a single region (*e.g.*, the black leather shoe and white sheet) or splits a coherent region into multiple fragments (*e.g.*, Pikachu clothing and body). An analysis of how SAM masks and multi-level granularity affect SCOUP is provided in Appendix I, while limitations inherited from SAM and CLIP are addressed in Appendix J. However, SCOUP is not tied to a specific segmenter or feature encoder, and either can be replaced without modifying the framework for efficient 2D-to-3D and 3D-to-2D transfer and storage of language features.

Table 4: Ablation study evaluating the impact of dense feature uplifting, sparse code uplifting, and Top-K filtering on LERF-OVS dataset.

Dense Feature Uplifting	Sparse Code Uplifting	Top-K Filtering	mIoU (%)	mAcc (%)	2D→3D (min)	3D→2D (FPS)	Storage (GB)
✓			60.5	82.0	21.3	0.5	7.8
-	✓		60.0	81.2	**0.7**	22.0	1.0
-	✓	✓	**64.4**	**86.9**	**0.7**	**29.3**	**0.1**

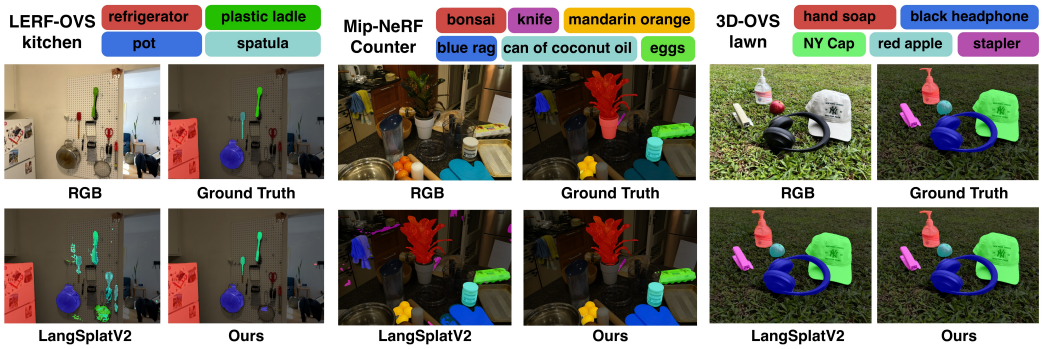

Figure 3: Qualitative results for open-vocabulary 3D object segmentation on evaluation datasets.

5.5 Ablation Study

In Table 4, we evaluate the impact of our Sparse Code Uplifting and Top-K Filtering components on the LERF-OVS dataset. Dense feature uplifting takes approximately 21 minutes on average across four scenes to uplift all three semantic levels. Our sparse uplifting drastically reduces this process to under 1 minute while maintaining performance comparable to full CLIP feature aggregation. Furthermore, applying Top-K filtering to the coefficients of each Gaussian after uplifting results in absolute performance gains of +3.9% and +4.9% in IoU score and mean accuracy, respectively, compared to using full CLIP embeddings. This demonstrates that Top-K filtering effectively acts as a denoising mechanism that improves multi-view consistency. We include further ablation studies in Appendix D and additional analysis of Top-K Filtering in Appendix F.

5.6 Qualitative Results

In Fig. 2, we present qualitative results for open-vocabulary 3D object localization on the LERF-OVS dataset. While both methods demonstrate strong performance in localizing target objects, SCOUP results in noticeably higher-confidence heatmaps and clearer object boundaries compared to LangSplatV2 (Li et al., 2025). Furthermore, Fig. 3 visualizes open-vocabulary 3D object segmentation on LERF-OVS, Mip-NeRF360, and 3D-OVS. In contrast to the 3D codebook optimization in LangSplatV2 (Li et al., 2025), in our 2D sparse coding, small and large objects contribute with more balance to the codebook learning process. This is particularly evident for challenging queries such as the "spatula" in the LERF-OVS kitchen scene and the "knife" in the Mip-NeRF360 counter scene. We include more qualitative results in Appendix K.

6 Conclusion

We present SCOUP, a framework for 3D Language Gaussian Splatting that achieves fast training, efficient storage, and fast rendering simultaneously by shifting sparse-code and codebook learning from 3D Gaussian space to 2D image space, then uplifting the learned sparse codes to the underlying 3D Gaussians through weighted sparse aggregation followed by top-K filtering, which suppresses multi-view inconsistencies. Compared to LangSplatV2, SCOUP achieves more than $400\times$ training speedup and $3\times$ lower training memory while maintaining state-of-the-art rendering speed and preserving storage efficiency and accuracy across multiple benchmarks. By reducing per-scene semantic reconstruction from hours to a minute, SCOUP lowers the practical barrier to deploying open-vocabulary 3D scene understanding in real-world applications.

AI USE STATEMENT

In this work, we used generative AI tools to polish writing and improve the readability of the manuscript, assist with writing and editing code, search for information, and provide feedback on the presentation of methodology and experiments. Generative AI tools were not used to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis. All AI-assisted work was reviewed and verified by the authors. We take responsibility for the final content of this work, including all text and code produced or edited with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For reproducibility, we provide implementation details in Section 5.2, along with additional evaluation details in Appendix C. We also provide pseudocode for the sparse code uplifting algorithm in Appendix B. Source code and configuration files will be made publicly available upon publication.

REFERENCES

Elena Alegret, Kunyi Li, Sen Wang, Siyun Liang, Michael Niemeyer, Stefano Gasperini, Nassir Navab, and Federico Tombari. GALA: Guided attention with language alignment for open vocabulary gaussian splatting. In *Thirteenth International Conference on 3D Vision*, 2026. URL https://openreview.net/forum?id=qDUUGCGL4i.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5460–5469, 2022. doi: 10.1109/CVPR52688.2022.00539.

Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21476–21485, 2024a. doi: 10.1109/CVPR52733.2024.02029.

Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21401–21412, 2024b. doi: 10.1109/CVPR52733.2024.02022.

Jiahuan Cheng, Jan-Nico Zaech, Luc Van Gool, and Danda Pani Paudel. Occam's lgs: An efficient approach for language gaussian splatting. In *36th British Machine Vision Conference 2025, BMVC 2025, Sheffield, UK, November 24-27, 2025*. BMVA, 2025. URL https://bmva-archive.org.uk/bmvc/2025/assets/papers/Paper_694/paper.pdf.

Zhenqi Dai, Ting Liu, and Yanning Zhang. Efficient decoupled feature 3d gaussian splatting via hierarchical compression. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11156–11166, 2025. doi: 10.1109/CVPR52734.2025.01042.

Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10608–10615, 2023. doi: 10.1109/ICRA48891.2023.10160969.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, Joshua B. Tenenbaum, Celso Miguel de Melo, Madhava Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. Conceptfusion: Open-set multimodal 3d mapping. *Robotics: Science and Systems (RSS)*, 2023.

Cijo Jose, Théo Moutakanni, Dahyun Kang, Federico Baldassarre, Timothée Darcet, Hu Xu, Daniel Li, Marc Szafraniec, Michaël Ramamonjisoa, Maxime Oquab, Oriane Siméoni, Huy V. Vo, Patrick Labatut, and Piotr Bojanowski. Dinov2 meets text: A unified framework for image- and pixel-level vision-language alignment. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24905–24916, 2025. doi: 10.1109/CVPR52734.2025.02319.

Kim Jun-Seong, GeonU Kim, Kim Yu-Ji, Yu-Chiang Frank Wang, Jaesung Choe, and Tae-Hyun Oh. Dr. splat: Directly referring 3d gaussian splatting via direct language embedding registration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14137–14146, 2025.

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

Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19672–19682, 2023.

Jiwook Kim, Seonho Lee, Jaeyo Shin, Jiho Choi, and Hyunjung Shim. Dreamcatalyst: Fast and high-quality 3d editing via controlling editability and identity preservation. *ArXiv*, abs/2407.11394, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

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. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992–4003, 2023.

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum*, 40, 2021.

Hyunjoon Lee, Joonkyu Min, and Jaesik Park. Cf3: Compact and fast 3d feature fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27906–27916, 2025.

Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RriDjddCLN.

Hao Li, Minghan Qin, Zhengyu Zou, Diqi He, Xinhao Ji, Bohan Li, Bingquan Dai, Dingwen Zhang, and Junwei Han. Langsurf: Language-embedded surface gaussians for 3d scene understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(10):12288–12300, 2026. doi: 10.1109/TPAMI.2026.3701287.

Wanhua Li, Yujie Zhao, Minghan Qin, Yang Liu, Yuanhao Cai, Chuang Gan, and Hanspeter Pfister. Langsplatv2: High-dimensional 3d language gaussian splatting with 450+ fps. *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2507.07136.

Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards High-Fidelity Text-to-3D Generation via Interval Score Matching . In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6526, Los Alamitos, CA, USA, June 2024. IEEE Computer Society. doi: 10.1109/CVPR52733.2024.00623. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52733.2024.00623.

Kunhao Liu, Fangneng Zhan, Jiahui Zhang, Muyu Xu, Yingchen Yu, Abdulmotaleb El Saddik, Christian Theobalt, Eric P. Xing, and Shijian Lu. Weakly supervised 3d open-vocabulary segmentation. In *Neural Information Processing Systems*, 2023.

Juliette Marrie, Romain Menegaux, Michael Arbel, Diane Larlus, and Julien Mairal. Ludvig: Learning-free uplifting of 2d visual features to gaussian splatting scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

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.

Yuning Peng, Haiping Wang, Yuan Liu, Chenglu Wen, Zhen Dong, and Bisheng Yang. Gags: Granularity-aware feature distillation for language gaussian splatting. *ArXiv*, abs/2412.13654, 2024.

Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20051–20060, 2024. doi: 10.1109/CVPR52733.2024.01895.

Yansong Qu, Shaohui Dai, Xinyang Li, Jianghang Lin, Liujuan Cao, Shengchuan Zhang, and Rongrong Ji. Goi: Find 3d gaussians of interest with an optimizable open-vocabulary semantic-space hyperplane. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, pp. 5328–5337, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706868. doi: 10.1145/3664647.3680852. URL https://doi.org/10.1145/3664647.3680852.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. Language embedded 3d gaussians for open-vocabulary scene understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5333–5343, 2024. doi: 10.1109/CVPR52733.2024.00510.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

Lei Tian, Xiaomin Li, Liqian Ma, Hao Yin, Zirui Zheng, Hefei Huang, Taiqing Li, Huchuan Lu, and Xu Jia. Ccl-lgs: Contrastive codebook learning for 3d language gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9855–9864, 2025.

Xuechang Tu, Lukas Radl, Michael Steiner, Markus Steinberger, Bernhard Kerbl, and Fernando de la Torre. Vrsplat: Fast and robust gaussian splatting for virtual reality. *Proc. ACM Comput. Graph. Interact. Tech.*, 8(1), May 2025. doi: 10.1145/3728311. URL https://doi.org/10.1145/3728311.

Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20902–20911, 2024a. doi: 10.1109/CVPR52733.2024.01975.

Sen Wang, Kunyi Li, Siyun Liang, Elena Alegret, Jing Ma, Nassir Navab, and Stefano Gasperini. Visibility-aware language aggregation for open-vocabulary segmentation in 3d gaussian splatting. In *2026 International Conference on 3D Vision (3DV)*, pp. 1791–1800, 2026. doi: 10.1109/3DV69130.2026.00170.

Yixuan Wang, Mingtong Zhang, Zhuoran Li, Tarik Kelestemur, Katherine Driggs-Campbell, Jiajun Wu, Li Fei-Fei, and Yunzhu Li. D^3fields: Dynamic 3d descriptor fields for zero-shot generalizable rearrangement. In *8th Annual Conference on Robot Learning*, 2024b.

Hanyuan Xiao, Yingshu Chen, Huajian Huang, Haolin Xiong, Jing Yang, Pratusha Prasad, and Yajie Zhao. Localized gaussian splatting editing with contextual awareness. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 5207–5217, 2025. doi: 10.1109/WACV61041.2025.00509.

Ziyang Yan, Lei Li, Yihua Shao, Siyu Chen, Wuzong Kai, Jenq-Neng Hwang, Hao Zhao, and Fabio Remondino. 3dsceneeditor: Controllable 3d scene editing with gaussian splatting. *arXiv preprint arXiv:2412.01583*, 2024.

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*, pp. 21676–21685, 2024.

M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pp. 29–538, 2001. doi: 10.1109/VISUAL.2001.964490.

Table 5: Quantitative comparison of different sparse code learning strategies for open-vocabulary 3D object localization and semantic segmentation on LERF-OVS.

Method	**3D Object Localization** (mAcc %)					**3D Semantic Segmentation** (mIoU %)				
	Ramen	Teatime	Kitchen	Figurines	Overall	Ramen	Teatime	Kitchen	Figurines	Overall
LangSplatV2 (Li et al., 2025)	**74.7**	**93.2**	86.4	82.1	84.1	51.8	72.2	59.1	56.4	59.9
3D Sparse Coding	59.2	78.0	86.4	53.6	69.3	45.4	59.4	59.5	36.6	50.2
Weighted 3D Sparse Coding	73.2	89.8	86.4	76.8	81.6	50.2	68.7	61.9	44.8	56.4
SCOUP	**74.7**	91.5	**95.5**	**85.7**	**86.9**	**57.6**	**75.3**	**64.4**	**60.4**	**64.4**

A Learning Sparse Codes in 2D versus 3D Domain

The key contribution of SCOUP is the decision to learn the sparse semantic coefficients of high-dimensional features in the 2D domain, rather than in 3D Gaussian space. This changes where semantic compression takes place and reformulates the pipeline for transferring high-dimensional features between the 2D and 3D domains, in both directions.

Prior works (Marrie et al., 2025; Cheng et al., 2025; Wang et al., 2026) directly uplift dense 512-dimensional features to 3DGS, enabling faster 3D semantic reconstruction than optimization-based approaches, but at the cost of a large memory footprint and slow inference-time feature rendering. In contrast, learning 3D sparse codes directly in 3DGS with repeated rasterization during optimization on 2D dense feature maps, as in LangSplatV2 (Li et al., 2025), leads to hours of training time.

In our preliminary experiments, we explored a straightforward combination of these two lines of work by learning sparse representations directly in 3D after uplifting dense features. We denote this approach as *3D sparse coding*, where we first uplift dense 2D features to 3D Gaussians using the baseline method from Marrie et al. (2025) and then use them as target features for rasterization-free optimization of the corresponding 3D sparse coefficients and a 3D codebook. Although 3D sparse coding substantially reduces storage requirements and enables efficient feature rendering, it treats all Gaussians equally in codebook optimization, regardless of their actual contribution to the rendered scene. Consequently, Gaussians with low rendering influence, such as those with low opacity, are weighted equally to more visually and semantically significant Gaussians during codebook learning.

To reduce this imbalance, we also considered *weighted 3D sparse coding*, where we weight each Gaussian's contribution by its accumulated scene importance Z_i from Equation 3. Weighted 3D sparse coding improves over uniform 3D code learning, as shown in Table 5, but both 3D variants remain below SCOUP. The largest degradation is observed in the Figurines scene, which contains many small objects. Because large objects and background regions are typically represented by more Gaussians, they contribute disproportionately to codebook learning, while features corresponding to small objects can become underrepresented. Additionally, the number of Gaussians in a reconstructed scene is substantially larger than the number of masked regions extracted from the input images. Learning the sparse representation in 2D therefore operates on a considerably smaller set of feature vectors, resulting in shorter optimization runtime.

As a result, only SCOUP simultaneously achieves accurate and fast 3D semantic reconstruction, low GPU memory usage, compact storage, and efficient rendering, all of which are essential for the practical deployment of 3DLGS in real-world applications. Our shift from 3D to 2D sparse code learning is not merely an implementation choice for improved efficiency, since it also produces more balanced codebook representations across object scales and, together with our sparse code uplifting algorithm, leads to higher accuracy and IoU in open-vocabulary 3D localization and semantic segmentation, respectively.

B Sparse Code Uplifting Algorithm

The proposed sparse code uplifting procedure is summarized in Algorithm 1.

Algorithm 1 Sparse Code Uplifting

for Gaussian point $i \in \mathcal{N}$ **do**
  Initialize $\mathbf{w}_i \in \mathbb{R}^L$ as the zero vector
end for
for frame $v \in \mathcal{V}$ **do**
  for pixel $p \in \Omega_v$ **do**
    (indices $\{j_1, \ldots, j_K\}$, coefficients $\{c_{j_1}, \ldots, c_{j_K}\}$) $\leftarrow \mathrm{TopK}(\mathbf{W}_v(p))$
    for Gaussian point $i \in \mathcal{N}_{v,p}$ **do**
      for each $j \in \{j_1, \ldots, j_K\}$ **do**
        $\mathbf{w}_i[j] \leftarrow \mathbf{w}_i[j] + c_j \cdot e_i(v, p)$
      end for
    end for
  end for
end for
for Gaussian point $i \in \mathcal{N}$ **do**
  if $\mathbf{w}_i \neq \mathbf{0}$ **then**
    $\hat{\mathbf{w}}_i = \mathrm{Top}_K(\mathbf{w}_i)/\|\mathrm{Top}_K(\mathbf{w}_i)\|_1$
  end if
end for

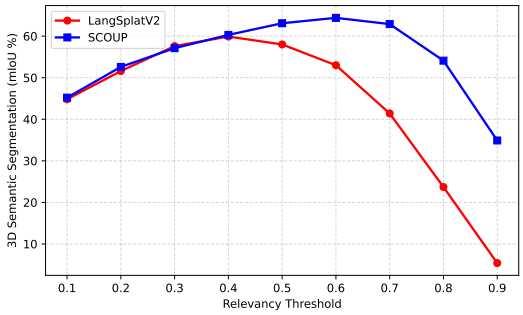

Figure 4: Sensitivity of 3D semantic segmentation results to the relevancy threshold on LERF-OVS.

C Additional Evaluation Details

We follow Qin et al. (2024) and Li et al. (2025) to compute relevancy scores between rendered language features and text queries. Specifically, the relevancy score between the rendered language feature at pixel position p, denoted by $\mathbf{F}_v(p)$, and the CLIP text embedding of the query, $\mathbf{t}_{\text{query}}$, is defined as

$$R(p) = \min_k \frac{\exp(\alpha \cdot \mathbf{F}_v(p) \cdot \mathbf{t}_{\text{query}})}{\exp(\alpha \cdot \mathbf{F}_v(p) \cdot \mathbf{t}_{\text{query}}) + \exp(\alpha \cdot \mathbf{F}_v(p) \cdot \mathbf{t}^k_{\text{canon}})}, \tag{9}$$

where α is a temperature parameter set to 10, and $\mathbf{t}^k_{\text{canon}}$ is the CLIP embedding of a predefined canonical phrase selected from "object," "things," "stuff," and "texture."

For relevancy map post-processing, we follow LangSplatV2 (Li et al., 2025), modifying only the mask thresholds and the average-pooling kernel size. We use thresholds of 0.6, 0.45, and 0.3 on LERF (Kerr et al., 2023), Mip-NeRF360 (Barron et al., 2022), and 3D-OVS (Liu et al., 2023), respectively, and set the average-pooling kernel size to 7 for all three datasets.

Sensitivity to the Relevancy Threshold. To evaluate sensitivity to the choice of relevancy threshold, we vary the threshold on LERF-OVS and compare SCOUP with LangSplatV2 (Li et al., 2025). As shown in Figure 4, SCOUP is less sensitive to the threshold choice, particularly at higher values, and maintains strong segmentation performance over a wider range of thresholds. This is consistent with the more confident relevancy heatmaps shown in Figure 2 and Figure 10. Notably, for all thresholds in the range $[0.5, 0.7]$, SCOUP outperforms the strongest baseline reported in Table 2, demonstrating robustness to the choice of relevancy threshold.

Table 6: Comparison for varying L. We report mean accuracy (%), mean IoU (%), and rendering speed (FPS) on the LERF dataset.

L	32	64	128
mean Accuracy (%)	77.2	**86.9**	<u>85.1</u>
mean IoU (%)	60.6	<u>64.4</u>	**65.5**
rendering speed (FPS)	**43**	<u>29</u>	23

Table 7: Comparison for varying K. We report mean accuracy (%), mean IoU (%), and rendering speed (FPS) on the LERF dataset.

K	2	4	8
mean Accuracy (%)	<u>82.9</u>	**86.9**	78.1
mean IoU (%)	62.5	**64.4**	<u>63.8</u>
rendering speed (FPS)	**34**	<u>29</u>	25

Table 8: Effect of codebook initialization and fine-tuning on 3D-OVS dataset.

Codebook	mAcc (%)	mIoU (%)
k-means initialization	83.3	61.1
Fine-tuned	**86.9**	**64.4**

D Additional Ablation Studies

D.1 Sparse Coding Parameters

In Table 6 and Table 7, we ablate the codebook size (L) and sparsity level (K) on the LERF-OVS dataset, respectively. We find that $L = 64$ provides the best balance between open-vocabulary performance and rendering efficiency, as larger codebooks introduce additional decoding overhead. Furthermore, setting $K = 4$ during both 2D sparse coding and Top-K filtering yields the highest mean accuracy, indicating that a larger K introduces semantic ambiguity. While these optimal parameters are consistent with those reported in LangSplatV2 (Li et al., 2025), our framework learns the codebook entirely in the 2D domain.

D.2 Effect of Codebook Initialization and Fine-Tuning

To evaluate the contribution of codebook optimization, we compare the k-means initialization against the final codebook obtained after joint optimization with the sparse coefficients. In both settings, all other components of the pipeline are kept unchanged.

As shown in Table 8, the k-means initialization achieves 83.3% mAcc and 61.1% mIoU, indicating that it already provides a suitable set of codebook atoms for sparsely representing CLIP features. Jointly fine-tuning the codebook with the sparse coefficients further improves performance to 86.9% mAcc and 64.4% mIoU. This shows that codebook optimization provides an additional benefit beyond the initial k-means representation, which is already comparable to state-of-the-art methods.

D.3 Entropy-Based Analysis of Top-K Filtering

In Table 9, we study whether entropy can be used to assess the importance of Gaussians before Top-K filtering. For each Gaussian, we compute the entropy of its coefficient distribution as

$$H(\mathbf{w}_i) = -\sum_{l=1}^{L} w_{i,l} \log w_{i,l}. \tag{10}$$

We then filter out 50% of the Gaussians based on their entropy values to evaluate their contribution to open-vocabulary 3D object localization and segmentation.

The results indicate that Gaussians with low-entropy coefficients are considerably more important. Since low entropy reflects a more confident and concentrated distribution during multi-view aggregation, these Gaussians provide more reliable semantic cues. Removing 50% of the low-entropy Gaussians leads to a substantial drop in mean IoU and mean accuracy. By contrast, removing 50% of the high-entropy Gaussians results in only a minor performance decrease. These results support the consensus interpretation underlying our Top-K filtering: Gaussians whose aggregated coefficients are concentrated on a few atoms encode more reliable multi-view semantics than those with diffuse distributions.

Table 9: Effect of entropy-based Gaussian filtering on the LERF-OVS dataset.

Filtering strategy	mIoU (%)	mAcc (%)
No filtering	**64.4**	**86.9**
Remove 50% of the highest-entropy Gaussians by $H(\mathbf{w}_i)$	<u>61.0</u>	<u>86.5</u>
Remove 50% of the lowest-entropy Gaussians by $H(\mathbf{w}_i)$	33.1	78.2

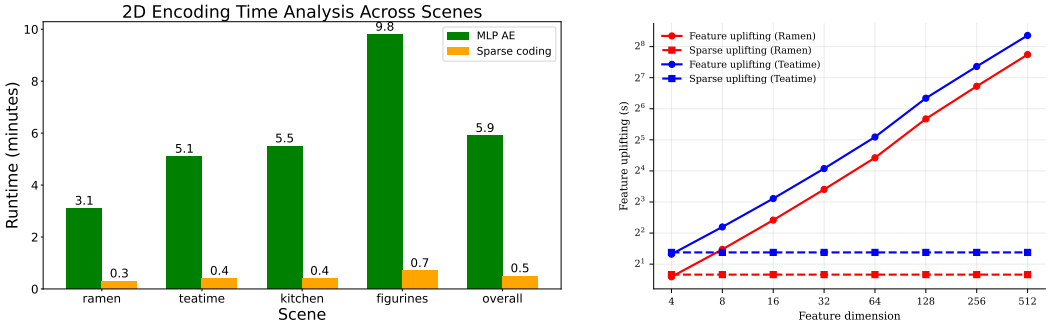

Figure 5: Left: 2D encoding runtime of the MLP autoencoder from LangSplat (Qin et al., 2024) compared to our 2D sparse coding. Right: sparse coefficient uplifting runtime of SCOUP compared to full-dimensional feature uplifting on the LERF-OVS dataset (NVIDIA RTX A6000 GPU).

E Detailed Runtime Analysis

In Fig. 5 (left), we compare our 2D Sparse Coding algorithm against the MLP autoencoder used in LangSplat (Qin et al., 2024). Our approach is over $10\times$ faster at encoding CLIP language embeddings for each SAM-segmented region while avoiding the information loss typically associated with MLP-based compression.

Furthermore, we analyze the scalability of our sparse code uplifting on the LERF-OVS dataset. As shown in Fig. 5 (right), uplifting sparse coefficient maps requires only 2.5 seconds and 1.5 seconds per semantic level on the Teatime and Ramen scenes, respectively, regardless of the feature dimensionality. This indicates that approximately 80% of the total computational runtime in 3D semantic reconstruction stems from the 2D sparse coding of the feature maps, a step we can efficiently compute prior to the uplifting process.

F Multi-View Consistency from Top-K Filtering

To evaluate whether the Top-K filter improves multi-view consistency rather than simply denoising through sparsification, we designed an additional controlled corruption experiment.

Before uplifting, we randomly select a fraction of the input views and permute the sparse coefficients among the SAM regions within each selected view. For example, the coefficients associated with an apple region may be exchanged with those of a table region. This deliberately breaks cross-view semantic consistency, so that corrupted views cast incorrect votes during sparse code uplifting. We progressively increase the fraction of corrupted views and compare 3D open-vocabulary segmentation (mIoU %) performance with and without Top-K filtering on LERF-OVS.

As shown in Fig. 6, as the fraction of corrupted views increases, performance degrades substantially more slowly with Top-K filtering than without it. At 20% corruption, Top-K filtering preserves nearly the same 3D segmentation performance as in the uncorrupted setting, while at 40% corruption it still outperforms the uncorrupted variant without Top-K filtering. At 100% corruption, no consistent cross-view signal remains, and both variants collapse. This indicates that our method preserves coefficients that are consistently supported across the majority of views, rather than acting solely as a generic denoising mechanism.

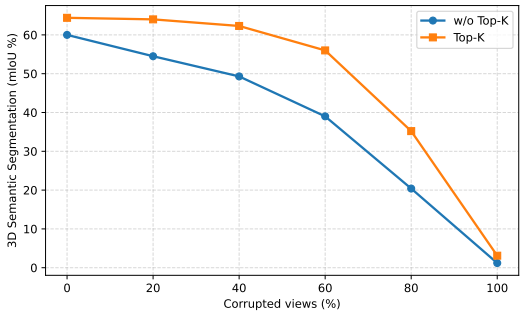

Figure 6: Effect of Top-K filtering on multi-view consistency under controlled view corruption.

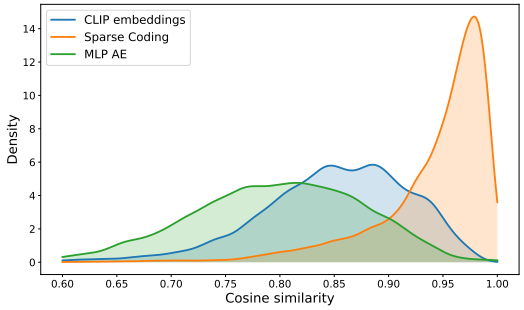

Figure 7: Comparison of multi-view consistency among ground-truth CLIP features, decoded MLP autoencoder features from LangSplat (Qin et al., 2024), and decoded sparse coding features from SCOUP on the *Teatime* scene from the LERF (Kerr et al., 2023) dataset.

In summary, our corrupted-view analysis, together with the entropy analysis from Appendix D.3, shows that Top-K filtering serves as an effective multi-view consistency mechanism by suppressing inconsistent cross-view contributions, resulting in substantial improvements in 3D open-vocabulary localization and segmentation performance, as shown in Table 4.

G Multi-View Consistency from 2D Sparse Coding

In Fig. 7, we evaluate the multi-view consistency of three types of features on the coarse semantic level of the *Teatime* scene from LERF-OVS (Kerr et al., 2023): ground-truth CLIP features, decoded features from the 2D autoencoder of LangSplat (Qin et al., 2024), and decoded features from the 2D sparse coding method used in SCOUP. To measure multi-view consistency, we use sparse 3D points from COLMAP (Schönberger & Frahm, 2016; Schönberger et al., 2016) that we used to initialize 3DGS training. For each sparse 3D point, we identify its corresponding 2D observations in multiple views and compute the cosine similarity between the associated 2D features. The results indicate that 2D sparse coding improves feature consistency across views by increasing the cosine similarity between matched 2D features. By contrast, the MLP autoencoder may lead to a loss of semantic information due to the aggressive compression of CLIP embeddings, resulting in smaller cosine similarity between corresponding 2D features. These results justify our adoption of 2D sparse coding, as it preserves semantic structure across views more faithfully than the autoencoder baseline, while also improving training efficiency, as shown in Appendix E.

H Robustness to Inaccuracies in Gaussian-to-Pixel Associations

To evaluate SCOUP's practical robustness to imperfect Gaussian-to-pixel associations, we uplift the same 2D semantic representation onto 3DGS checkpoints of progressively increasing quality, trained for 3k, 7k, 15k, and 30k iterations. Earlier checkpoints exhibit less accurate geometry and noisier Gaussian-to-pixel associations, allowing us to assess how semantic uplifting behaves as the

Table 10: Impact of the number of 3DGS training iterations on SCOUP.

Iterations	3DGS training time	PSNR ↑	mAcc (%) ↑	mIoU (%) ↑
3k	1:02	22.94	69.5	35.7
7k	3:12	26.41	70.9	59.0
15k	9:19	27.31	85.4	61.8
30k	20:04	30.20	86.9	64.4

Table 11: Performance on LERF-OVS for different SAM mask granularities.

SAM granularity	mAcc (%)	mIoU (%)
Fine	69.4	42.5
Middle	82.0	60.8
Coarse	81.3	61.6
Multi-level	86.9	64.4

underlying reconstruction improves. We report reconstruction time in minutes, 3DGS scene quality (PSNR), and downstream semantic performance on LERF-OVS (mAcc and mIoU).

As shown in Table 10, at 15k iterations (50% of standard training), SCOUP already reaches 85.4% mAcc and 61.8% mIoU, and matches or exceeds the best fully trained baseline methods from Table 2. A substantial loss is observed only at the earliest checkpoint, where the underlying 3DGS reconstruction remains highly incomplete.

I Effect of SAM Mask Granularity

In this work, the use of multi-level SAM masks follows established practice in 3DLGS (Qin et al., 2024; Li et al., 2025; Cheng et al., 2025; Wang et al., 2026) and enables a controlled comparison with these methods. We extract SAM regions at fine, middle, and coarse semantic granularities and uplift all three levels to 3DGS. In Table 11, we isolate the performance at each level as an additional analysis of two complementary mask failure modes: merging distinct semantic elements into a single region and splitting a coherent region into multiple fragments.

The coarse level, which is more likely to merge distinct semantic elements, remains competitive in terms of the overall localization and segmentation results. Its main limitation arises for fine-grained or part-level queries. For example, if a bear and its nose belong to the same coarse region, querying "bear nose" may retrieve the entire bear rather than only the requested part.

The larger degradation occurs at the fine level, where SAM (Kirillov et al., 2023) frequently fragments coherent objects into small regions. Such fragments contain less semantic context, producing less reliable CLIP features and sparse coefficients. Over-segmentation therefore appears to be the more substantial source of degradation, although both failure modes remain possible.

Multi-level granularity improves overall performance, but limitations remain for certain queries, as illustrated by the 3D-OVS examples in Fig. 8. Additionally, multi-level granularity introduces additional rendering and storage overhead, as a separate language representation is stored for each level.

SCOUP itself is not tied to SAM (Kirillov et al., 2023): the sparse-coding and uplifting procedure can be applied to masked regions produced by another segmentation model without modifying the proposed framework.

J Limitations and Broader Impact

Despite its efficiency, SCOUP inherits the computational overhead of the large visual foundation models used for pre-processing. Specifically, the heavy feature extraction required by SAM and CLIP remains a bottleneck in pre-processing. Furthermore, since our sparse feature uplifting relies



Figure 8: Limitation of merging distinct semantic elements (left) and splitting a coherent region into multiple fragments (right) on the 3D-OVS dataset.

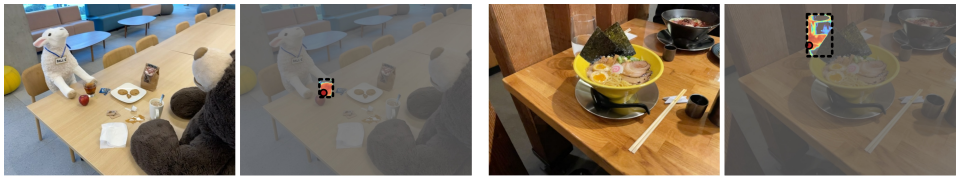

Figure 9: Limitation of direct feature uplifting in 3DLGS on the LERF-OVS dataset.

on 3DGS priors optimized for visual appearance, objects with high transparency pose a challenge. Gaussians representing transparent surfaces often have lower opacity values, leading to less confident semantic heatmaps on objects like glass, as shown in Fig. 9.

Along with the computational overhead of VLMs, our method inherits their semantic biases, which propagate to the downstream task. At the same time, SCOUP's efficiency carries a positive environmental impact: it makes per-scene open-vocabulary 3D reconstruction substantially cheaper, reducing semantic reconstruction time from hours to a minute and lowering memory and GPU utilization during training.

K Additional Qualitative Results

We present additional qualitative examples of open-vocabulary 3D object localization on the LERF-OVS (Kerr et al., 2023) dataset in Fig. 10.

We present additional qualitative examples of open-vocabulary 3D object segmentation on the LERF-OVS (Kerr et al., 2023), Mip-NeRF360 (Barron et al., 2022), and 3D-OVS (Liu et al., 2023) datasets in Fig. 11, Fig. 12, and Fig. 13, respectively.

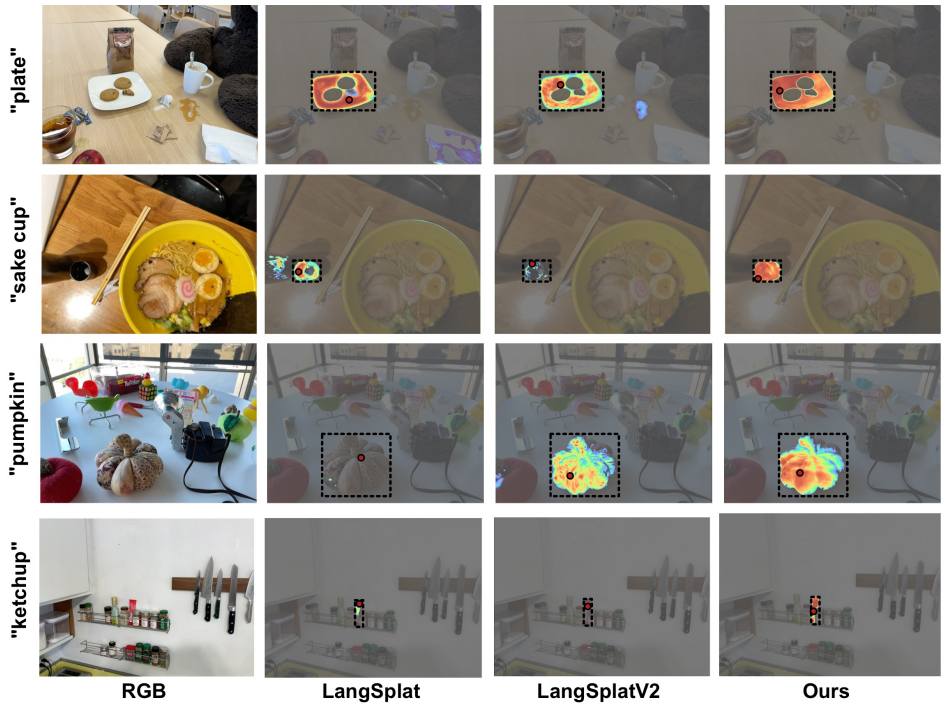

Figure 10: Additional qualitative results for open-vocabulary 3D object localization on the LERF-OVS dataset.

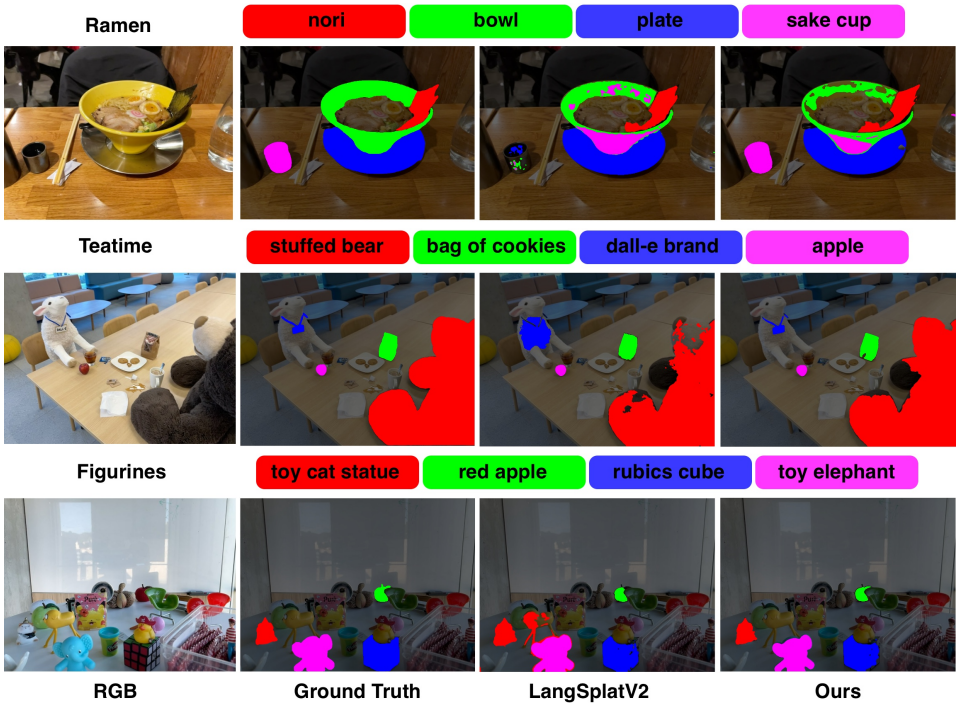

Figure 11: Additional qualitative results for open-vocabulary 3D object segmentation on the LERF-OVS dataset.

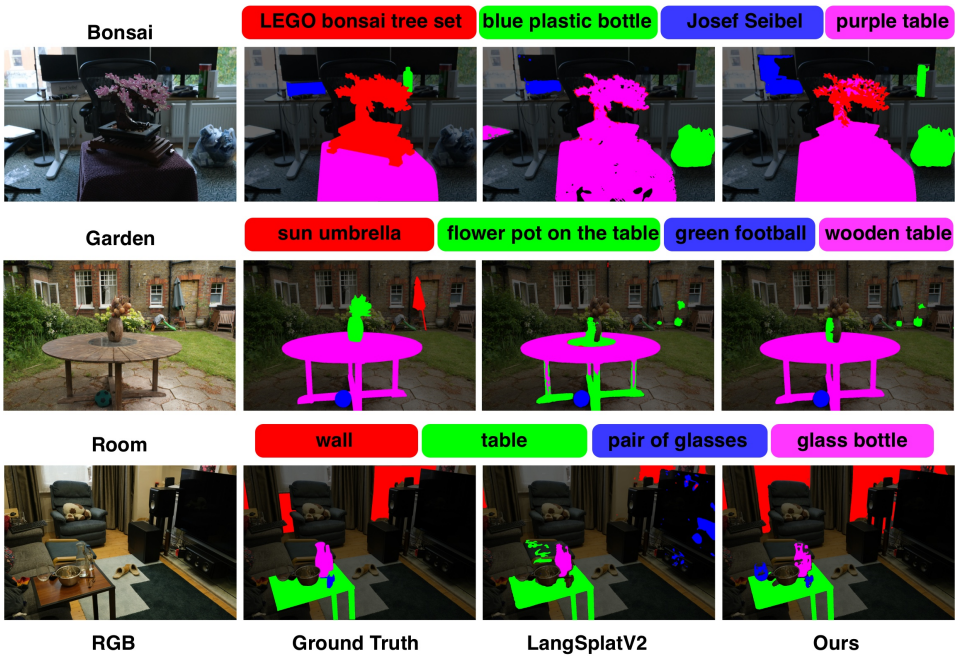

Figure 12: Additional qualitative results for open-vocabulary 3D object segmentation on the Mip-NeRF360 dataset.

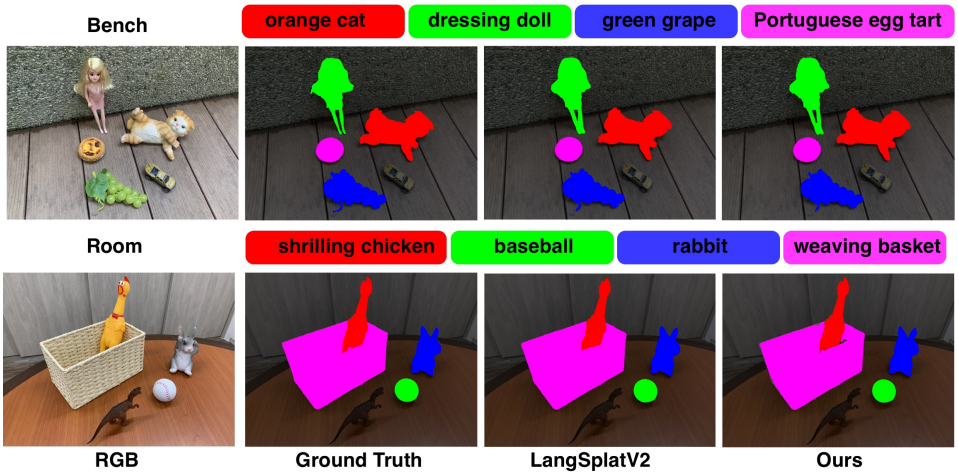

Figure 13: Additional qualitative results for open-vocabulary 3D object segmentation on the 3D-OVS dataset.