GAUSSIANS ALL THE WAY DOWN: HIERARCHICAL GAUSSIAN MIXTURE MODELS FOR OPEN-VOCAB GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

3D Gaussian Splatting (3DGS) has revolutionized high-fidelity scene synthesis. Yet, seamlessly integrating semantic knowledge in 3DGS remains a computational challenge. Existing approaches either rely on expensive, per-scene learned compression for real-time inference or avoid training entirely and work with computationally expensive scene-wide unordered feature sets. We present a framework that directly lifts 2D features into 3D Gaussians and organizes them via hierarchical Gaussian Mixture Models (GMM), clustering jointly over spatial and semantic dimensions through parallelized expectation-maximization (EM). This produces a traversable tree whose cluster centroids act as a lightweight, non-learned codebook: lifted and generated in under 10 minutes, using on average 23$\times$ less memory than storing raw features per-Gaussian, and enabling sub-millisecond semantic queries. This method is demonstrated across four semantic datasets ranging from small objects to large-scale rooms, achieving state-of-the-art semantic segmentation without any learned feature compression or per-scene training. We demonstrate the robustness of the pipeline with an extensive ablation across six different text-aligned feature backbones, multiple clustering backends, covariance constraints, and query methodologies. Combined we present an efficient hierarchical GMM data structure for joint spatial-semantic clustering; state-of-the-art open-vocabulary segmentation with an order-of-magnitude smaller on-disk footprint than raw-feature methods; and sub-millisecond query times on par with learned, per-scene optimized methods.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has emerged as a transformative representation of the world, offering high fidelity, high speed rendering, while quickly being able to reconstruct diverse and complex scenes. By representing scenes as a collection of differentiable point-based primitives, 3DGS allows rapid and high-quality reconstruction of complex environments. However, for a scene representation to be truly "*intelligent*", it must go beyond photometric and geometric fidelity and capture a rich semantic understanding of the objects and their contextual relationships within the environment.

Current efforts to embed 3DGS with semantic awareness typically rely on "lifting" high-dimensional features (such as CLIP Radford et al. (2021) or DINO Caron et al. (2021)) from 2D images into 3D space Yu et al. (2024); Cheng et al. (2025); Zuo et al. (2025). To maintain the real-time throughput expected from 3DGS, existing pipelines Li et al. (2025); Jun-Seong et al. (2025); Jang & Kim (2025) often compress these dense embeddings into small codebooks or latent spaces using learned encoders. While these methods achieve fast query times, they often come at a significant cost: the compression process tends to rely on large per-scene models optimized for a specific feature space.

In this work, we propose a shift away from learned compression in favor of an efficient, structure-driven organization of semantic data. We introduce a framework that directly lifts dense 2D features into 3D Gaussians and organizes them using a hierarchical Gaussian Mixture Model (GMM) Bishop (2006). By clustering Gaussians based on both spatial proximity and semantic similarity, we build a traversable tree that inherently captures the multi-level nature of a scene without needing to perform expensive per-scene training or costly per-Gaussian latent storage.

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.

Our approach leverages a parallelized GMM optimization process that groups semantically and spatially nearby 3D Gaussians into clusters, which are then recursively subdivided creating a GMM tree. Each cluster node in the tree stores average centroid features, acting as a lightweight, non-learned alternative to a codebook. During downstream tasks, such as open-vocabulary querying, the system can rapidly traverse this tree to isolate relevant features in sub-millisecond time. Our method enables the generation of semantically searchable 3D scenes in under 10 minutes, and thanks to the sparsity of the tree, our memory consumption is up to 34$\times$ less than pure lifting methods allowing it to run on consumer-grade hardware, while matching the sub-millisecond query of learned systems. As 3DGS and other scene representations are pushed to larger scenes and smaller compute, the need for efficient data access and generation becomes increasingly paramount. We demonstrate the robustness of this approach, we ran across various datasets ranging from small indoor scenes to the full rooms in ScanNet++ Yeshwanth et al. (2023). To show its versatility we test with CLIP Radford et al. (2021), SigLIP v2 Tschannen et al. (2025), and RADIO v4 Ranzinger et al. (2024) backbones.

We present three main contributions:

- Rapid generation of memory-efficient hierarchical semantic 3D Gaussian Splatting scenes
- An efficient formulation for high-dimensional GMMs using spatial and semantic distances
- State-of-the-art open-vocab segmentation performance without per-scene training

2 Related Work

Semantic 3D Gaussian Splatting. As 3D Gaussian Splatting Kerbl et al. (2023) has emerged as a powerful technique for representing and rendering 3D scenes, there has been increasing interest in augmenting this representation with semantic information to enable tasks such as segmentation, object recognition, and open-vocabulary querying. One of the first works to augment 3D Gaussians with features, inspired by LeRF Kerr et al. (2023), was LangSplat Qin et al. (2024), which compresses CLIP features via an encoder-decoder before associating them with each Gaussian; features render through the 3DGS rasterization step and decode back to full dimensionality in image space. In parallel, several other works Zhou et al. (2024); Liao et al. (2025); Zuo et al. (2025) extended NeRF-based semantic feature methods to 3DGS, but struggled with the curse of dimensionality while preserving fidelity and query performance. LangSplatV2 Li et al. (2025) addressed this with larger codebooks that further compress feature dimensionality, cutting rendering, querying, and post-processing time at the cost of hours of per-scene training. This trade of an expensive training pipeline for efficient inference-time operations, via learned dimensionality reduction, is a common theme.

Training free feature lifting. As an alternative to these learned approaches, LBG Chacko et al. (2025) and Occam's LGS Cheng et al. (2025) directly lifts 2D features into 3D space without any training. Specifically, LBG directly fuses foundational 2D masks onto existing fields, while Occam's LGS associates 2D features with 3D Gaussians based on camera projection and depth information. This allows mapping 2D features to 3D Gaussians in a matter of minutes and provides a more versatile system that can work with any feature space without specialized codebooks or encoders. However, it incurs significant memory and computational costs, as all Gaussians are stored with raw features. To make this more manageable, it is often combined with PCA or another lightweight dimensionality reduction technique, but the unordered set still results in slower query times than learned approaches.

Clustering and quantization. These feature mappings can be improved further through the use of clustering or hierarchical data structures to store similar Gaussian information together, allowing for more efficient querying and retrieval of relevant Gaussians for a given task. OpenGaussian Wu et al. (2024) and Dr. Splat Jun-Seong et al. (2025) use this clustering to quantize the feature space for computational efficiency, trading off semantic precision. CAGS Sun et al. (2025) leverages spatially-proximate neighborhood feature aggregation with SAM segmented region features to preserve fine-grained details in the scene. Similarly, Gaussian Grouping Ye et al. (2024) jointly reconstructs and segments scenes by assigning discrete grouping identities to Gaussians, enabling zero-shot object-level editing. More recently, ReLaGS Xie et al. (2026) extends this grouping to an inter-object 3D scene graph – structure that aids efficient data retrieval and pruning, though generating it on a full 3-million-Gaussian scene with 1024-dimensional latents per Gaussian can easily overwhelm consumer grade hardware.

Positioning. Hierarchical Gaussian mixtures fit by recursive EM are well explored Bishop (2006), and joint spatial-semantic distance metrics for clustering appear in prior 3DGS work as well Lu et al. (2024); Sun et al. (2025). Our contribution is not either component in isolation but how they are combined to close a three-way gap in the 3DGS landscape above. Learned-codebook methods Li et al. (2025); Jun-Seong et al. (2025); Wu et al. (2024) give fast, compact queries but require an expensive, scene-specific training pass; training-free direct-lifting methods Cheng et al. (2025); Chacko et al. (2025) avoid training entirely but store raw per-Gaussian features in slow-to-query unordered sets that take excess memory; existing hierarchical or clustered 3DGS methods Ye et al. (2024); Xie et al. (2026) impose structure but fix a single rigid alignment or hand-tuned spatial-semantic weighting across the whole scene, often sacrificing fidelity for reasonable generation time. Our training-free, lightweight, and computationally efficient alternative achieves the flexibility and speed of direct lifting methods and the sub-millisecond queries of learned codebooks, without memory-intensive or specially-aligned data structures – all while providing state-of-the-art segmentation across four datasets spanning orders of magnitude in physical scale and semantic content, without per-scene retraining or tuning.

3 METHOD

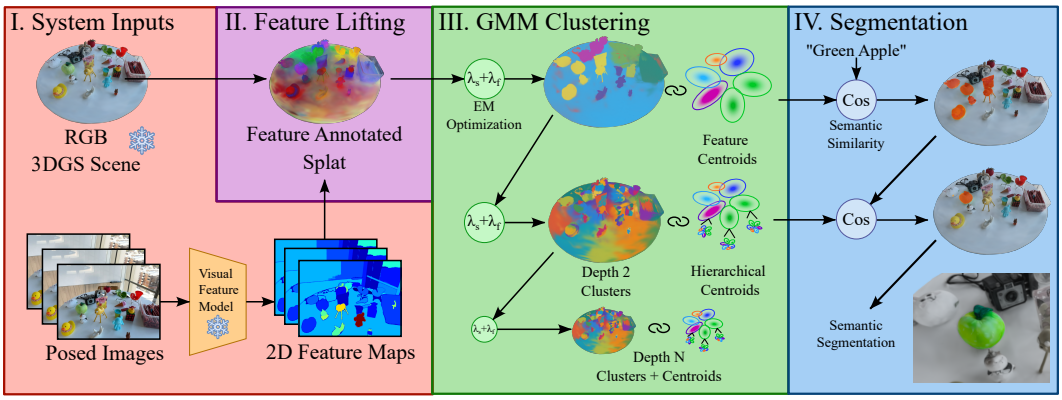


Figure 1: Overview of the proposed pipeline. From left to right: input images and lifted 2D features, construction of 3D Gaussians, hierarchical GMM clustering across spatial and semantic dimensions, and fast open-vocabulary querying via traversable cluster tree.

3.1 PRELIMINARIES

3D Gaussian Splatting 3D Gaussian Splatting Kerbl et al. (2023) is a powerful technique for representing and rendering 3D scenes using a collection of 3D Gaussians. Each Gaussian is defined by its mean μ, covariance Σ, opacity α and view dependent color C. Through the use of splatting these 3D Gaussians can be projected to 2D Gaussians. By sorting these Gaussians and applying alpha compositing, high fidelity renderings can be achieved in real time. We compute the color of each pixel by ordering Gaussians front-to-back, then using transmittance and accumulated color:

$$T_i = \prod_{j<i}(1-\alpha_j), \qquad \mathbf{c} = \sum_i T_i\,\alpha_i\,\mathbf{c}_i. \tag{1}$$

However this process is not limited to RGB color spaces, but can be applied to any high dimensional feature space, making it an attractive medium for semantic reasoning and segmentation tasks.

Gaussian Mixture Models Gaussian Mixture Models (GMMs) Bishop (2006) are a probabilistic model for representing a distribution as a mixture of multiple Gaussian components. Each component is defined by its mean μ_k, covariance Σ_k, and mixing coefficient π_k. The GMM's resulting PDF is:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k), \tag{2}$$

where $\mathcal{N}(x|\mu_k, \Sigma_k)$ is the Gaussian distribution for the k-th component. This enables dynamic clustering by optimizing Gaussians with Expectation-Maximization (EM) Dempster et al. (1977).

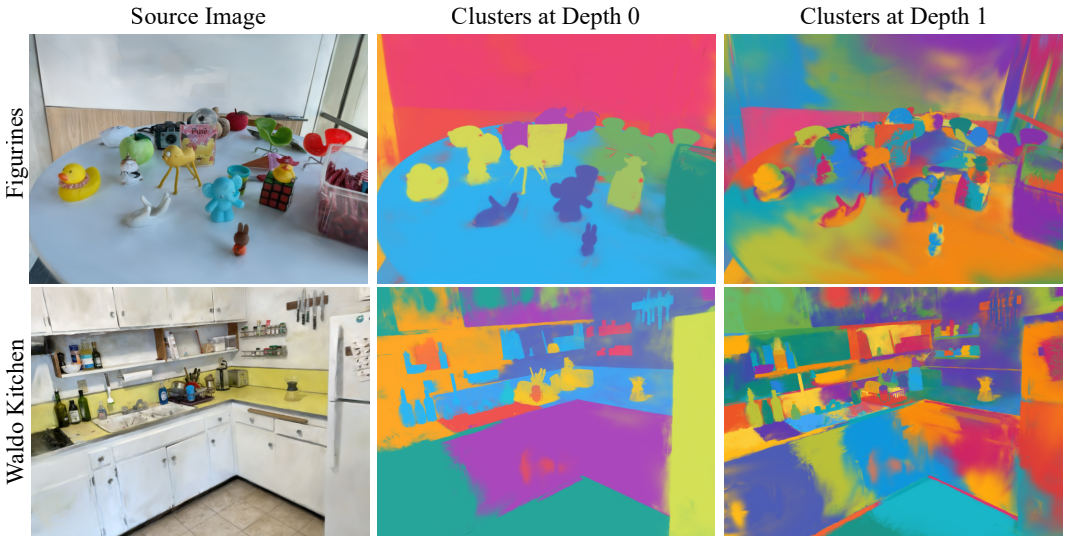


Figure 2: **Hierarchical Clusters** Each cluster colored to show how smaller clusters form around areas with more semantic variation, while larger clusters are formed in more homogeneous areas.

3.2 Feature Lifting

The proposed framework expects as input a set of images with associated camera poses and a pretrained RGB 3D Gaussian representation. 2D feature maps are then extracted from the input images using a pretrained visual feature backbone such as CLIP Radford et al. (2021) either with raw patches or in combination with SAM Kirillov et al. (2023) segmented regions. Following the formulation in Occam's LGS Cheng et al. (2025), we lift these 2D features into 3D space by associating each pixel in the input images with its corresponding 3D Gaussian based on the camera projection. We leverage the alpha transmissivity from Equation 1 to help generate the weighted likelihood of a given feature belonging to a specific Gaussian. This process is applied across all input images and all Gaussians generating a feature map for the entire scene. However, unlike the original implementation, we only gather features in the sparse set of pixels each Gaussian actually projects onto, avoiding VRAM issues when dense projecting all features across many views. This parallelized operation allows for rapid lifting of 2D features onto their 3D Gaussians.

3.3 Hierarchical GMM Clustering

In order to create a traversable tree of clusters, we apply a hierarchical GMM clustering process across the 3D Gaussians. This process optimizes the GMM parameters to cluster the Gaussians based on a weighted distance that combines both spatial proximity (euclidean distance) and semantic similarity (cosine similarity). The distance metric for clustering is defined as:

$$D(i,k) = \lambda_s \left\| \hat{\mu}_i - \hat{\mu}_k \right\|_2 + \lambda_f \left(1 - \hat{\mathbf{f}}_i \cdot \hat{\mathbf{f}}_k \right), \tag{3}$$

where μ denotes the 3D Gaussian mean, $\mathbf{f}$ the high-dimensional feature vector, and λ_s, λ_f are weighting hyperparameters. By using this formulation, the GMM can dynamically form clusters that capture both spatially close and semantically similar Gaussians, allowing for effective scene decomposition without a predefined granularity.

The scene grouping is performed top-down starting with the entire scene being divided into C children clusters. The size and span of these clusters is determined by the GMM optimization process, with the optimization centroid being stored. This centroid includes the average components across this entire cluster including a weighted center μ_c and feature $\mathbf{f_c}$. This process is then repeated for each child cluster until the maximum depth is reached or an early stopping criteria is hit (such as the number of Gaussians in a cluster is less than a minimum threshold). By storing the centroid for each

cluster, we can create a traversable tree that allows for efficient querying directing the search path down to most likely clusters containing the target. This clustering process is visualized in Figure 2.

3.4 Auto-tuned Spatial–Feature Balance

To avoid the λ_f and λ_s hyperparameters, we introduce a method for auto tuning this ratio based on local variance. We first row-normalize mean, $\hat{\mu}_i$, and feature vector, $\hat{\phi}_i$, to get a scale-independent unit vectors letting both weights be derived directly from the spread of each cluster S:

$$\lambda_f(S) = \left(\sum_{d=1}^{D} \mathrm{Var}_{i \in S}\big[\hat{\phi}_{i,d}\big] \right)^{-1/2}, \qquad \lambda_s(S) = \left(\sum_{d=1}^{3} \mathrm{Var}_{i \in S}\big[\hat{\mu}_{i,d}\big] \right)^{-1/2}. \tag{4}$$

Each weight is the reciprocal square root of its block's total per-dimension variance across S, so after weighting, spatial and feature blocks contribute *equal total variance* to the joint clustering space.

3.5 Adaptive Depth via Spatial-Variance Fraction

The clustering process described above can be run with a fixed maximum depth $D_{\max}$ applied uniformly across the whole tree, but a single fixed depth is a poor fit for scenes that mix large homogeneous regions (walls, floors) with small, semantically dense ones (a cluttered tabletop). We address this with an adaptive stopping criterion, that lets each subtree decide independently how far to recurse based on how spatially tight it already is. Define the raw spatial variance of a node S as

$$V(S) = \sum_{d=1}^{3} \mathrm{Var}_{i \in S}\big[\mu_{i,d}\big], \tag{5}$$

and let $V_0 = V(S_{\text{root}})$ be this quantity at the root (depth 0). A node S at depth t is a leaf if any of

$$V(S) < \tau \cdot V_0, \qquad t \geq D_{\max}, \qquad |S| < n_{\min} \cdot C, \tag{6}$$

holds, where $\tau \in (0, 1)$ is a tunable hyperparameter, $n_{\min}$ is the minimum Gaussians-per-cluster floor, and C is the number of children. Otherwise S splits and each child recurses with $t{+}1$, resulting in flat, homogeneous regions (e.g. a wall) terminating early as one coarse cluster. Expressing the threshold as a fraction of the root's own variance (rather than an absolute cutoff) is what makes τ transportable across scenes with very different coordinate scales – a fixed absolute variance cutoff tuned on one scene means something entirely different on another with a different physical extent.

3.6 Segmentation and Queries

If the visual feature backbone has cross-alignment with text features such as CLIP, we can perform open-vocabulary queries by traversing the tree based on cosine similarity between the query and the cluster centroids. Starting at the root node, we compute the cosine similarity between the query feature and the centroid feature for all C child clusters. We then select the child cluster with the highest similarity and repeat this process recursively until we reach a leaf node (spatial limit) or the similarity drops off (semantic limit). This allows us to efficiently narrow down to the most relevant clusters that are likely to contain the target query, enabling fast retrieval of relevant Gaussians for rendering or segmentation tasks. Furthermore, as there is no strict granularity for the clusters, if multiple clusters return similar high similarity scores, we can return all these clusters allowing for flexible grouping of objects based on the query. This is particularly useful for queries that may correspond to parts of an object or multiple objects, as the hierarchical clustering can capture these relationships without needing explicit supervision or predefined categories. In this case multiple children clusters can be "activated" based on the semantic similarity scores. Each activated cluster can then be explored further, reducing the overall amount of comparisons needed, while allowing for tighter clusters around the target. This process is visualized in Figure 3 where a query for "Green Apple" is made and multiple clusters are returned to segment the apple at depth 1.

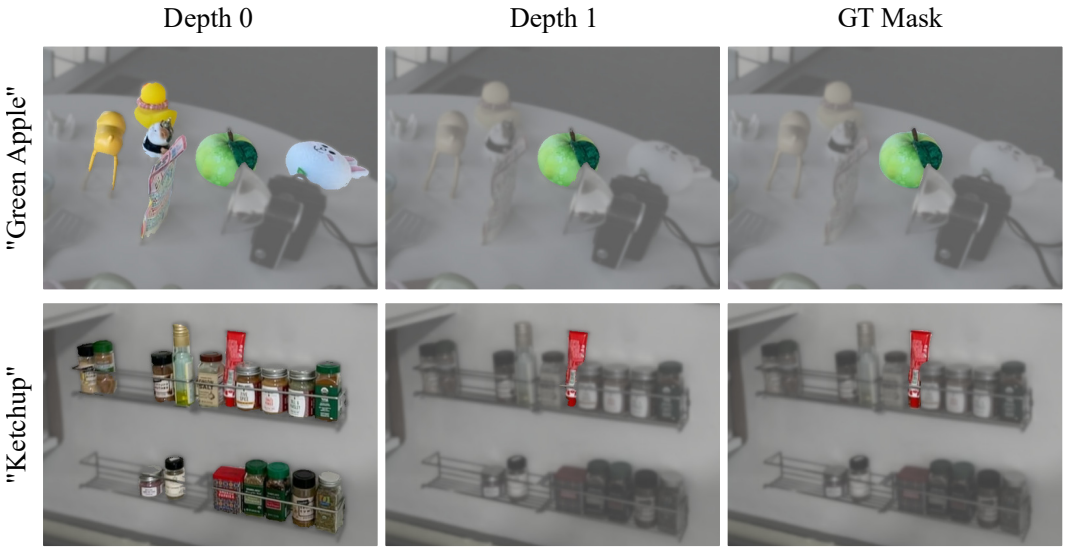


Figure 3: **Open-Vocabulary** Open-vocabulary querying process showing the ground truth mask (left), the first level of clustering (middle) and the second level of depth clusters (right). The top shows the query "Green apple" in the figurines scene while the bottom is "ketchup" in the kitchen scene.

Method	Ramen		Teatime		Figurines		Waldo Kitchen		Overall	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
OpenGaussian	28.14 (31.00)	30.24*	65.67 (60.40)	64.29*	58.52 (39.30)	80.95*	38.16 (22.70)	60.19*	47.62	58.92
Occam's LGS	51.19 (51.00)	52.55*	71.58 (70.20)	79.29*	60.59 (58.60)	78.17*	59.08 (65.30)	81.48*	60.61	72.87
LangSplatV2	48.86 (51.80)	57.75*	71.26 (72.20)	81.36*	59.66 (56.40)	75.00*	60.91 (59.10)	59.09*	60.17	68.30
ReLaGS	62.87 (51.20)	67.38*	68.60 (81.00)	75.00*	63.48 (64.70)	80.95*	71.76 (60.60)	88.89*	66.68	78.06
Ours	**71.20**	**82.86**	**82.46**	**92.86**	**73.72**	**82.14**	**74.21**	**94.44**	**75.40**	**88.08**

Table 1: **LERF-OVS.** Comparison of mIoU (%) and mAcc@25% (%) across the baselines, OpenGaussian Wu et al. (2024), Occam's LGS Cheng et al. (2025), LangSplatV2 Li et al. (2025) and ReLaGS Xie et al. (2026). Best results are **bold**, second best are underlined. Results in parentheses are from the source paper, a trailing * marks a value with no published counterpart under this exact metric (mAcc@0.25, follows LangSplat's Qin et al. (2024) localization hit and scoring IoU > 0.25 requirement for stricter evaluation). Further details can be found in Section A.1.

4 Experiments

4.1 Data

We evaluate on four datasets, chosen to span different scene scales, capture conditions, and query styles rather than a single curated benchmark. **LERF-OVS** Kerr et al. (2023) (4 scenes: Ramen, Teatime, Figurines, Waldo Kitchen) as the standard open-vocabulary segmentation benchmark with a handful of small labeled objects per scene; **3D-OVS** (5 scenes: Room, Bed, Lawn, Bench, Sofa) with mixed indoor and outdoor captures; **Mip-NeRF360** (4 scenes: Garden, Bonsai, Counter, Room) for a larger-scale, room/property-scale capture setting; and **ScanNet++** (v2) on large-scale, densely multi-class-labeled real-world indoor scan collection (a 10-scene mini-validation split, and a 50-scene extension). As the ScanNet++ ground truth labels are dominated by large background classes, we adopt SceneSplat++'s Ma et al. (2025) own evaluation convention (f-mIoU/f-mAcc, excluding wall/floor/ceiling) instead of the protocol used for the other three, described fully in Section A.1.1.

For each dataset we report per-scene mIoU and mAcc@25% as well as an aggregated Overall score. Across the three benchmarks shown in Tables 1, 2, and 3 we are state-of-the-art nearly across the board. We lose out to Occam's LGS and ReLaGS in the Room and Garden scenes respectively, however both of these scenes are dominated by lower scores in single labels. This is due to the GMM's struggle to capture sharp edges between flat objects and surfaces they sit on and is discussed further in Section 5.1 and Section A.3.

The ScanNet++ results are shown in Table 4 and compared against published baselines under SceneSplat++'s protocol on the 10-scene and 50-scene splits. Our method leads every published baseline across both splits for quality but takes a hit in accuracy when compared to SceneSplat++. This drop in accuracy is likely due to the similar clustering issues and is explored in Section A.3.

Method	Room		Bed		Lawn		Bench		Sofa		Overall	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
OpenGaussian	23.14*	46.67*	20.81*	50.00*	57.26*	66.67*	45.58*	62.86*	11.31*	16.67*	31.62	48.57
Occam's LGS	**96.16 (96.50)**	**100.00***	96.34 (96.80)	**100.00***	94.11 (97.00)	**100.00***	95.29 (95.80)	**100.00***	87.07 (88.80)	93.33*	93.79	98.67
LangSplatV2	85.28 (96.10)	96.67*	85.03 (93.00)	76.67*	96.99 (96.60)	**100.00***	95.27 (94.90)	**100.00***	77.39 (92.30)	93.33*	87.99	93.33
ReLaGS	76.75 (94.80)	83.33*	93.09 (95.40)	100.00*	92.96 (91.20)	100.00*	95.19 (94.40)	100.00*	84.81 (86.90)	100.00*	88.56	96.67
Ours	95.23	**100.00**	**96.98**	**100.00**	**97.45**	**100.00**	**96.14**	**100.00**	**93.58**	**100.00**	**95.88**	**100.00**

Table 2: **3D-OVS.** Same convention as Table 1 showing mIoU(%) and mAcc(%).

Method	Garden		Bonsai		Counter		Room		Overall	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
OpenGaussian	22.04*	27.27*	19.41*	36.11*	62.96*	83.33*	41.76*	60.00*	36.54	51.68
Occam's LGS	54.72*	81.82*	74.21*	83.33*	76.05*	85.19*	66.41*	72.50*	67.85	80.71
LangSplatV2	57.02 (65.0)	66.67*	69.62 (73.1)	82.89*	74.34 (75.1)	83.78*	58.08 (64.3)	68.97*	64.77	77.08
ReLaGS	**64.50***	82.95*	75.02*	83.33*	68.67*	83.33*	72.85*	95.00*	70.26	86.15
Ours	61.05	**90.91**	**75.39**	**83.34**	**79.53**	**92.59**	**81.00**	**100.00**	**74.24**	**91.71**

Table 3: **Mip-NeRF360.** Same convention as Table 1. LangSplatV2 is the one baseline that publishes Mip-NeRF360 numbers of its own.

Method	10 scenes		50 scenes	
	f-mIoU	f-mAcc	f-mIoU	f-mAcc
Gradient-Weighted 3DGS Joseph et al. (2024)	9.25	13.91	9.00	13.28
LUDVIG Marrie et al. (2025)	9.95	20.99	15.02	33.12
Occam's LGS Cheng et al. (2025)	15.80	28.61	18.03	32.63
SceneSplat++ Ma et al. (2025)	22.63	**49.16**	28.36	**49.92**
Ours	**26.79**	38.61	**28.48**	40.63

Table 4: **ScanNet++ v2, 10-scene validation and full 50-scene splits.** f-mIoU/f-mAcc (%) are mIoU/mAcc excluding wall/floor/ceiling following SceneSplat++'s Ma et al. (2025) own Table 4.

4.2 Performance

By restricting search and comparison operations to the tree of centroids, we greatly reduce the number of comparisons needed to find relevant Gaussians for a query, allowing for rapid query times rivaling MLP based approaches without the need for training or optimization of a network. This comparison is shown in Table 5 where we show open-vocabulary query times averaged across LERF-OVS scenes. Learned approaches such as LangSplatV2 (Li et al., 2025) often achieve faster queries through the use of per-scene models or code books that are evaluated in a feed-forward manner. In comparison, direct feature embedding such as Occam's LGS Cheng et al. (2025) bypass the need for any per-scene training and instead directly store all the features for a scene directly. This results in far faster generation times, but at the cost of a larger number of comparisons across the entire scene per query. By using a hierarchical GMM clustering approach we can achieve similar query times to learned approaches while still maintaining rapid generation times of direct feature methods. Additionally, through automatic batch sizes determined via predicted VRAM usage (Section A.1.4), our pipeline is capable of running on consumer grade hardware during preprocessing and fitting, while our baselines require much more VRAM. This does come at a slight cost when compared to non-batched Occam's.

Storage efficiency. Because the hierarchical tree stores per-cluster centroids rather than a raw per-Gaussian feature copy, the achievable on-disk representation is substantially smaller than the direct feature storage approach of Occam's LGS, without any learned compression. Keeping only leaf-cluster assignments (a flat per-Gaussian leaf ID) and leaf centroids (half-precision), and dropping every internal (non-leaf) node's own centroid/variance entirely, reduces storage by $6.04\times$ on Ramen (632.72MB raw features $\rightarrow$ 104.79MB) and $34.06\times$ on Figurines (179.85MB $\rightarrow$ 5.28MB) relative to storing the raw lifted feature for every Gaussian. Averaged across all four LERF-OVS scenes, the same real reduction is $23.0\times$ (1143.31MB $\rightarrow$ 49.72MB mean, Table 5's Storage column).

	Preprocessing [min]	Feature Mapping [min]	Query [ms]	Storage [MB]	Storage [MB/1k Gaussians]
Occam's LGS Cheng et al. (2025)	**56.49**	1.28	2.43	1143.31	0.977
LangSplatV2 Li et al. (2025)	141.84	444.93	0.23	1045.89	0.732
Ours	74.04	**1.10**	**0.17**	**49.72**	**0.009**

Table 5: **Performance.** Comparison of the computational performance for different methods averaged across the LERF-OVS dataset. Preprocessing includes converting source images into segmentation masks. Feature mapping is the time to associate these 2D features with the 3D Gaussians, direct lifting for Occam's, training of the code books for LangSplatV2, and lifting + GMM construction for our method. Query is the time to find and return the segmented portion of the scene measured on NVIDIA RTX 4090 24GB. Storage is presented as space required to store the intermediate representation to disk, averaged across 4 scenes and in a per-Gaussian normalized size. For Occam's this is the raw CLIP features per Gaussian, while LangSplatV2 includes the 3 granularities of learned codebooks. Ours keeps only leaf-cluster assignments plus leaf centroids, resulting in an average 23× reduction.

4.3 Semantic Backbone Choice

We use CLIP+SAM (SAM-masked region crops encoded with CLIP) as the backbone throughout the paper, however to demonstrate the scalability for this pipeline against different backbones, we validate that choice against SigLIP v2 Tschannen et al. (2025) and NVIDIA's RADIO v4 Ranzinger et al. (2024) as well. For these experiments, Table 6 shows the average mIoU of each of the backbones (with their dense and SAM-assisted variants) across 3 datasets. We skip ScanNet++ as the SceneSplat++ data only ships with SigLIP features and does not include the GT RGB images for other backbones.

Backbone	LERF-OVS	3D-OVS	Mip-NeRF360
CLIP	**76.41**	57.35	72.88
CLIP+SAM	75.17	**95.88**	**74.24**
SigLIP	35.22	45.55	53.56
SAM+SigLIP	68.36	90.68	72.67
RADIO	51.91	92.61	68.87
SAM+RADIO	67.64	90.01	74.19

Table 6: **Semantic backbone comparison (mIoU %).** Despite having more complex latents, SigLIP and RADIO both lose out to the CLIP backbone in almost every setting. Additionally, the SAM variant's structural support helps segmentations, resulting in better scores in almost every case.

4.4 Ablation

An extensive ablation was conducted across all meaningful hyperparameters and evaluation methods in this pipeline. The complete tables and results can be found in Section A.2, while Table 7 presents an overview of what each parameter does and its impact on the results. Throughout the ablation a global recipe is derived and presented as the default option and used for the results in Section 4.

Recipe-tuning granularity. In the end tuning all of these parameters per-scene vs. with a global recipe as presented throughout this paper has a fairly small impact on overall mIoU. Table 8 presents the average mIoU across all 13 scenes in the non ScanNet++ datasets with the global recipe derived from this ablation and the SOTA results presented in the paper (Tables 1, 2, and 3), as well as how far this can be pushed with full per-scene tuning of tree shape, dimensions and adaptive depth state. Given the speed of this pipeline, the practical costs for tuning all steps after post processing is generally a few minutes, but ultimately doesn't provide much benefit compared to presented recipe.

5 Conclusion

In this work, we present a novel framework for rapidly generating and querying high-dimensional 3D Gaussian representations by leveraging hierarchical Gaussian Mixture Models for clustering across spatial and semantic dimensions. By leveraging this data structure feature-embedded 3DGS methods no longer have to choose between raw per-Gaussian features (fast to build, but expensive to store

Hyperparameter	Default	Relative Impact
Semantic backbone	CLIP+SAM	**High** – up to 38.5 pts vs. dense features[a]
Clustering backend	Batched GMM	**Low** – on mIoU ($\leq$1 pt typical) but **decisive** on speed/robustness - up to 178x faster, and dynamic batch sizes prevent OOM[b]
GMM λ selection	Auto-tuned	**High** – +12.1 pts mean, up to +47.7 pts on individual scenes[c]
Covariance type	Full	**Low** – under 2-4 pts typical per scene[d]
Feature dimensions	64	**Low** – under 2-4 pts, lower values improve performance[d]
Tree shape	Children=20, Depth=3	**Scene-dependent** – 0.25-3.6 pts on 16 of 18 scenes, but 5.9-12.1 pts on Room/Sofa[e]
Adaptive depth	Fixed	**Low** – up to +1.0 pt, costs up to -3.7 pts on Waldo Kitchen[f]
Lift aggregation	Weighted Mean	**High** – up to +19.6 pts vs. Dominant View[g]
Query recipe	Joint Query / Otsu Thresh.	**High** – up to 46.6 pts within one scene on top of an already-optimal clustering config[h]

Table 7: **Hyperparameter impact overview.** Every hyperparameter touched by the ablation campaign, its default value, and a qualitative-plus-quantitative sense of how much it typically moves mIoU, grounded in a number established in the tables in Section A.2. [a]Table 6. [b]Table 11. [c]Table 12. [d]Table 13. [e]Table 16. [f]Table 15. [g]Table 17. [h]Figure 4.

Policy	# configs	mean mIoU (%)
Global (single config: Joint+Otsu)	1	76.87
Per-dataset	3	78.09
Per-scene (ceiling)	13	**82.49**

Table 8: **Cost of tuning granularity.** Whole-corpus (without ScanNet++) mean mIoU under three tuning policies looking at per-scene values to set a ceiling for performance, per-dataset as a cheaper alternative, and global. In the end the numbers reported for this pipeline reflect the values from the global recipe throughout.

and query), and learned per-scene codebooks (compact and fast, but requires extensive training). Instead this single training-free hierarchical GMM, efficiently fit via expectation maximization over a joint spatial-semantic distance, allows for an adaptive tree that neatly clusters the scene from large regions down to small sub-parts. This efficient tree enables sub-millisecond query times beating learned codebooks, while taking less on-disk memory, and capable of being generated faster than raw projected methods. Across four diverse datasets this method achieves state-of-the-art segmentation, and through the ablation we demonstrate the robustness of this method for various feature backbones.

5.1 Limitations

While the proposed method achieves rapid generation and efficient querying of semantic 3D Gaussian scenes, there are several limitations to consider. First, the quality of the segmentation and querying is heavily dependent on the choice of feature backbone and source 3DGS. In poorly reconstructed scenes too many overlapping semantic features due to floaters or holes can result in unstable semantics. It is hoped that further exploration of the variance within clusters can be used to identify these degenerate cases and inform active perception or perform post-processing. Second, while the hierarchical GMM allows for efficient querying, it is difficult to direct placement of the clusters to favor certain regions at different levels. This can result in subparts being well clustered at the same level as larger objects, causing slight bleeding, and can potentially be addressed via granularity constraints to the clustering operation. This is likely what causes issues in Garden and Room, as well as the mAcc drop in ScanNet++ as discussed in Section A.3. Third, while the global recipe derived from ablation achieves SOTA performance, Table 8 shows that additional performance is possible when tuning each GMM per-scene. While the system is fast enough to perform this tuning faster than the learned methods, a better approach would be to make more of the hyperparameters auto tuned like the λ values.

AI USE STATEMENT

In this work, we used generative AI tools for polishing and generating early feedback on the paper draft to highlight potential areas not well explored or ablated. Generative AI tools were also used to help set up environments for testing our method with different datasets, in the sense of making them parsable and applying mappings for different layouts, however all of these minor transformations were verified by the authors manually before deployment. We have not used generative AI tools for the development of the core framework nor mathematical claims, and the rest of the required disclosures such as proofs, dataset generation/cleanup, translation are not applicable to this work. Additionally, we used generative AI tools for getting baseline methods running a compute cluster, verifying that the reported metrics match replicated metrics within a reasonable margin on available datasets. This mainly involved environment setup and the introduction of common logging practices across various methods from literature. Additionally while generative tools were not used to make tables and figures directly, they were used to build scripts to generate plots and tables from the large database of runs and verified for correctness and parity before inclusion in this paper. We have reviewed all AI-assisted work, and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Code and samples are provided in supplemental materials, while a full, cleaned version of the code will be made open upon acceptance. We include the minimal code samples to run the method on custom data or the tested datasets (provided data and compute is available). For more convenient means of qualitatively validating the results, GMM artifacts produced from the runs, as well as a viewer script (`gmm_viser.py`) which can render the scene, clusters, and enables open vocab queries is provided. Note that this version is designed for visualization with a GPU but contains the core code for interacting with any of the GMM artifacts generated through this pipeline.

REFERENCES

Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

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 International Conference on Computer Vision (ICCV)*, 2021.

Rohan Chacko, Nicolai Häni, Eldar Khaliullin, Lin Sun, and Douglas Lee. Lifting by gaussians: A simple, fast and flexible method for 3d instance segmentation. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3497–3507. IEEE, 2025.

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`.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–38, 1977.

SungMin Jang and Wonjun Kim. Identity-aware language gaussian splatting for open-vocabulary 3d semantic segmentation. In *Proceedings of the International Conference on Computer Vision*, 2025.

Joji Joseph, Bharadwaj Amrutur, and Shalabh Bhatnagar. Gradient-weighted feature back-projection: A fast alternative to feature distillation in 3d gaussian splatting, 2024. URL `https://arxiv.org/abs/2411.15193`.

Kim Jun-Seong, Kim GeonU, 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 *CVPR*, 2025.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

Justin* Kerr, Chung Min* Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

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. *arXiv preprint arXiv:2507.07136*, 2025.

Guibiao Liao, Jiankun Li, Zhenyu Bao, Xiaoqing Ye, Qing Li, and Kanglin Liu. Clip-gs: Clip-informed gaussian splatting for view-consistent 3d indoor semantic understanding. *ACM Transactions on Multimedia Computing, Communications and Applications*, 21(8):1–24, 2025.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20654–20664, 2024.

Mengjiao Ma, Qi Ma, Yue Li, Jiahuan Cheng, Runyi Yang, Bin Ren, Nikola Popovic, Mingqiang Wei, Nicu Sebe, Luc Van Gool, Theo Gevers, Martin R. Oswald, and Danda Pani Paudel. Scenesplat++: A large dataset and comprehensive benchmark for language gaussian splatting. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025.

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.

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*, pp. 20051–20060, 2024.

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. URL https://api.semanticscholar.org/CorpusID:231591445.

Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Am-radio: Agglomerative vision foundation model reduce all domains into one. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12490–12500, June 2024.

Wei Sun, Yanzhao Zhou, Jianbin Jiao, and Yuan Li. Cags: Open-vocabulary 3d scene understanding with context-aware gaussian splatting, 2025. URL https://arxiv.org/abs/2504.11893.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Yanmin Wu, Jiarui Meng, Haijie Li, Chenming Wu, Yahao Shi, Xinhua Cheng, Chen Zhao, Haocheng Feng, Errui Ding, Jingdong Wang, and Jian Zhang. Opengaussian: Towards point-level 3d gaussian-based open vocabulary understanding. 2024.

Yaxu Xie, Abdalla Arafa, Alireza Javanmardi, Christen Millerdurai, Jia Cheng Hu, Shaoxiang Wang, Alain Pagani, and Didier Stricker. Relags: Relational language gaussian splatting. In *CVPR*, 2026.

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European conference on computer vision*, pp. 162–179. Springer, 2024.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023.

Justin Yu, Kush Hari, Kishore Srinivas, Karim El-Refai, Adam Rashid, Chung Min Kim, Justin Kerr, Richard Cheng, Muhammad Zubair Irshad, Ashwin Balakrishna, et al. Language-embedded gaussian splats (legs): Incrementally building room-scale representations with a mobile robot. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13326–13332. IEEE, 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.

Xingxing Zuo, Pouya Samangouei, Yunwen Zhou, Yan Di, and Mingyang Li. Fmgs: Foundation model embedded 3d gaussian splatting for holistic 3d scene understanding. *International Journal of Computer Vision*, 133(2):611–627, 2025.

A TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

A.1 EXPERIMENTS

A.1.1 EVALUATION PROTOCOL

mIoU / mAcc@25%. For each ground-truth query label and evaluation frame, we render the query's per-Gaussian relevancy computed as a cosine-similarity with between the Gaussian's centroid and the target query above some thresholding function's cutoff and then rendered as a binary mask. mIoU is the mean of these best-IoU scores over all (frame, label) pairs. mAcc@25% follows LangSplat's/Occam's LGS's own localization-based definition: a (frame, label) instance counts as a hit only if some predicted-positive pixel falls inside the label's ground-truth bounding box *and* we add the additional constraint that the winning candidate's IoU exceeds 0.25; mAcc@25% is the fraction of instances satisfying both conditions, averaged per label and then across labels. This additional 25% constraint is from the original LangSplat and enforces a stricter criteria for success.

Thresholding. Per-pixel relevancy is Gaussian-smoothed, min-max normalized, and then binarized against a threshold. The default, `threshold_mode=dynamic`, is Occam's LGS's own real stability-based threshold-sweep algorithm. It sweeps candidate cutoffs, tracks how the resulting mean-positive-score and mask-size change, and picks a cutoff from the widest region where both are stable and the resulting mask size is non-degenerate. A global-Otsu alternative (`threshold_mode=otsu`) is kept only for legacy/comparison runs; it was found to occasionally collapse to a near-whole-image false positive on masks whose relevancy signal does not otherwise separate the object from the background, and underperforms the dynamic threshold in most of the scenes where both were compared. Following binarization, a 7×7 majority-vote cleanup pass (`smooth_cuda`) matches LangSplatV2's and Occam's LGS's own post-threshold smoothing.

Relevancy scoring: `joint` vs. `independent`. We support two formulas for turning a Gaussian's (or cluster centroid's) cosine similarity to a set of text queries into a relevancy score, differing in whether queries compete against each other or are scored in isolation. `relevancy_mode=independent` is the standard LERF-style formula that Occam's LGS's own evaluation also uses: for each query independently, a two-way softmax (temperature 10) against each of four fixed generic negative phrases (*object, things, stuff, texture*), keeping the most conservative (lowest-confidence) pairing. `relevancy_mode=joint` (our default) instead scores every currently-queried label *simultaneously* in one $(Q{+}1)$-way softmax (temperature 100) against a single shared background reference (the strongest of the same four generic negatives), with per-patch mean-centering to remove global scene bias before the softmax. This joint formulation is new to this work, motivated by evaluation throughput – one softmax call scores every query at once rather than looping per label, which makes larger sets of queries faster as seen in Section 4.2. For the added speed and to give a more fair comparison, the baselines in Tables 1, 2, 3 also use this same joint scoring likely resulting in the variations to the reported numbers for their methods and why the replicated results out score some of the reported numbers.

A.1.2 DATASETS: LERF-OVS, MIP-NERF 360, AND 3D-OVS

OpenGaussian, ReLaGS and LangSplatV2 were trained from scratch from their own public code while their published numbers are transcribed in parentheses for comparisons.

Additionally, as 3D-OVS is at native 4k resolution, the images had to be downscaled to 1080p for both memory reasons with learned methods and to work around a LangSplatV2 bug that crops larger images and collapses its scores below 5% otherwise.

A.1.3 DATASETS: SCANNET++

As we use the ScanNet++ benchmark provided by SceneSplat++ Ma et al. (2025), we report against the numbers they provide for baselines. As these scenes and their segmentation in the base datasets are heavily dominated by large background classes (walls, floors, ceilings) which greatly skew the mean Intersection-over-Union metric, we use SceneSplat++'s foreground-mIoU (f-mIoU), which only looks at the foreground object segmentations.

A.1.4 MEMORY PERFORMANCE

At the scale of realistic scenes with 1-6 million Gaussians and per-Gaussian feature dimensionality of 512-1152, the practical limit on what configuration can even complete on a given GPU is VRAM capacity, not raw speed. The lifting stage from Occam's LGS is dominated by per-Gaussian accumulator buffers and the rasterizer's own tile-intersection working set, both of which scale linearly with the Gaussian count and render area but are entirely bandwidth/capacity-bound, not compute-bound. We make this explicit with a calibrated linear model, fit from an empirical sweep on real data ($N \in \{50\text{k}, \ldots, 1.5\text{M}\}$ Gaussians, $R^2 = 0.9999$): predicted peak VRAM in bytes for lifting L SAM-granularity levels of D-dimensional features onto N Gaussians at render resolution $W \times H$ is

$$M(N, D, W, H, L) \;=\; N\,L\left[4(D+1) \;+\; c_r\,\frac{WH}{W_0 H_0}\right], \tag{7}$$

where the two additive terms separate an *exact* cost (the $(D{+}1)$-wide feature and weight accumulators the lifting stage explicitly allocates per Gaussian, in fp32) from an *empirical* one ($c_r = 10,658.1$ bytes/Gaussian, the calibrated per-Gaussian cost of the rasterizer's own working memory at the calibration resolution $W_0 \times H_0 = 876 \times 584$). Equation 7's calibration is validated at single-level lifting near the calibration resolution. In practice this model is used prescriptively: given a target GPU's available VRAM, Eq. 7 is inverted to warn (and, in the batched clustering and lifting stages, to automatically chunk point-batches to a memory-safe size) before a configuration is attempted, rather than discovering the failure via an out-of-memory crash partway through a run.

A.2 ABLATION

A.2.1 NOVELTY BASELINE

First in order to isolate if any of the spatial-semantic grouping is actually contributing to the results, we test a series of basic baselines: **flat k-means**, **KD-tree**, **Hierarchical agglomerative (HAC)**. Flat k-means has no hierarchy at all (one non-recursive clustering pass); the KD-tree splits purely on 3D position at every level, never looking at semantic features; HAC is genuine bottom-up agglomerative clustering (not the top-down recursive splitting every other backend in this paper). These results are showing Table 9, showing a significant improvement by leveraging the high-dimensional hierarchical GMM. Unlike swapping the clustering algorithm *within* the existing hierarchy (Table 11, $\leq$1 point), swapping the retrieval *structure* itself costs 15-68 points depending on scene and baseline, with HAC in particular collapsing on LERF-OVS's small, densely-packed objects.

Method	Figurines		Ramen		Teatime		Waldo Kitchen		3D-OVS Bed		3D-OVS Lawn	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
Flat k-means (no hierarchy)	39.76	62.70	49.95	65.60	57.20	70.71	37.45	73.15	55.34	83.33	48.39	83.33
KD-tree (spatial-only split)	36.45	60.32	42.54	55.48	52.31	76.79	34.13	74.07	60.13	83.33	45.80	83.33
Hierarchical agglomerative (HAC)	6.11	11.11	21.08	25.41	17.73	14.29	14.41	16.67	65.25	83.33	44.97	86.67
Ours	**73.72**	**82.14**	**71.20**	**82.86**	**82.46**	**92.86**	**74.21**	**94.44**	**96.98**	**100.00**	**97.45**	**100.00**

Table 9: **Novelty baselines.** All three baselines use the same lifted CLIP+SAM features and the paper's own global-recipe hyperparameters and are evaluated through the identical harness as **Ours**.

A.2.2 CLUSTERING BACKEND

With CLIP+SAM as the backbone used throughout this campaign, we turn to the clustering backend used to build the hierarchical tree. As we are performing a clustering operation across high dimensional data we explore two main methods for generating the clusters: *K-means* and *Gaussian Mixture Models*. Table 10 shows differences between a full covariance GMM and k-means clusters across the LERF scenes. We see that while K-means is roughly 3.75$\times$ faster than GMM, GMM does result in a better overall segmentation and is therefore used throughout the paper.

Both of the clustering backend methods, K-means and GMM, can be greatly sped up via parallelization and batching (also enabling more explicit memory management as stated in Section A.1.4). As such and to ensure our batching attempts do not sway the results from standard sequential implementations we compare these against our `batched_gmm_torch` implementation as well as `batched_kmeans_torch` and `batched_kmeans_warp` variants. To account for the

Scene	k-means [mIoU %]	GMM [mIoU %]	k-means fit [s]	GMM fit [s]
Ramen	71.82 ± 0.50	**72.47 ± 0.61**	49.7	177.2
Figurines	63.46 ± 0.79	**65.88 ± 1.84**	2.4	7.8
Teatime	83.05 ± 0.78	**83.66 ± 0.53**	105.3	400.0
Waldo Kitchen	75.41 ± 1.95	**77.13 ± 0.72**	157.2	590.7
Mean	73.44	**74.54**	78.65	293.93

Table 10: **Clustering backend, GMM vs. k-means (LERF-OVS).** Both k-means and GMM are running batched variants as described in Section A.2.2. Bold marks the better backend per row.

non-determinism in these methods we ran against 5 different seeds on all four LERF-OVS scenes (Table 11).

Two findings stand out. First, `batched_gmm_torch` matches or beats every other backend's mIoU on all scenes, and it does so at a small fraction of sequential `gmm`'s fit time. Second, and more importantly: sequential `gmm` fails outright on Waldo Kitchen using full latent dimensionality on this scene's 2M Gaussians on a 24GB GPU. This is a genuine memory ceiling, not a fixable transient error: every batched backend completes the same scene comfortably (17.9-22.0GB peak) thanks to the dynamic memory batching system.

Scene	Backend	mIoU [%] (mean ± std)	Fit time [s] (mean ± std)	Peak mem [GB]
Ramen	gmm (sequential)	71.51 ± 0.17	11261.9 ± 27.4	11.10
Ramen	kmeans (sequential)	71.66 ± 0.48	6178.4 ± 982.7	8.36
Ramen	batched_kmeans_torch	71.82 ± 0.50	49.7 ± 3.4	12.96
Ramen	batched_kmeans_warp	71.72 ± 0.30	34.7 ± 3.0	7.10
Ramen	**batched_gmm_torch**	**72.47 ± 0.61**	177.2 ± 1.3	11.71
Figurines	gmm (sequential)	64.79 ± 1.48	161.6 ± 38.9	1.81
Figurines	kmeans (sequential)	64.38 ± 0.70	81.5 ± 2.4	1.70
Figurines	batched_kmeans_torch	63.46 ± 0.79	2.4 ± 0.1	4.59
Figurines	batched_kmeans_warp	62.23 ± 0.94	1.4 ± 0.2	1.71
Figurines	**batched_gmm_torch**	**65.88 ± 1.84**	7.8 ± 0.1	6.02
Teatime	**gmm (sequential)**	83.15 ± 0.77	1927.4 ± 34.9	20.15
Teatime	kmeans (sequential)	82.64 ± 0.67	681.3 ± 21.1	15.06
Teatime	batched_kmeans_torch	83.05 ± 0.78	105.3 ± 6.3	17.94
Teatime	batched_kmeans_warp	82.82 ± 0.71	66.7 ± 3.9	12.72
Teatime	batched_gmm_torch	**83.66 ± 0.53**	400.0 ± 3.2	15.83
Waldo Kitchen	gmm (sequential)	–	–	–
Waldo Kitchen	kmeans (sequential)	76.52 ± 1.45	1858.5 ± 1223.2	21.17
Waldo Kitchen	batched_kmeans_torch	75.41 ± 1.95	157.2 ± 0.8	22.03
Waldo Kitchen	batched_kmeans_warp	75.33 ± 2.06	96.8 ± 0.2	17.86
Waldo Kitchen	**batched_gmm_torch**	**77.13 ± 0.72**	590.7 ± 0.4	19.92

Table 11: **Clustering Backend, across 5 seeds on LERF-OVS with `covariance_type=full`.** Bold marks the best mIoU per scene. Every seed's `subset_seed` also controls the clustering algorithm's own random initialization, leading to the standard deviation measurements. Waldo Kitchen on `gmm (sequential)` hit OOM errors and could not complete due to the large 2M Gaussian scene and full covariance fitting/full dimensionality.

A.2.3 Automatic λ Tuning

With clustering backend fixed, we investigate whether automatic hyperparameter selection (auto-derived λ, adaptive `spatial_variance_fraction` clustering, per-scene children) produces an improvement over the project's earlier hand-tuned baselines. Table 12 shows an average mIoU rises from 70.24 to 82.38 (+17.3% relative), with the gain concentrated on scenes the hand-tuned recipe handled poorly (Ramen: 48.30 → 71.32, +47.7%; Room (3D-OVS): 52.60 → 74.96, +42.5%; Figurines: 58.80 → 78.53, +33.6%) and much smaller where the hand-tuned recipe was already near ceiling (Bench: 94.94 → 95.11, +0.2%; Lawn: 92.77 → 93.42, +0.7%). Crucially, since this auto-tuning happens per cluster per depth, this makes the system more generalizable, as a single value is not fixed per-scene, but per granular cluster. These "hand-tuned" values are via a sweep of the best

possible fixed ratios with a granularity of 0.1. Not only does this improve the segmentation results, but the automatic selection means less hyperparameters left for the user to adjust per scene/dataset.

Scene	Hand-tuned	Auto-tuned	Relative gain
Teatime	65.40	**70.37**	+7.6%
Ramen	48.30	**71.32**	+47.7%
Waldo Kitchen	58.80	**72.66**	+23.6%
Figurines	58.80	**78.53**	+33.6%
Bed	79.68	**95.94**	+20.4%
Lawn	92.77	**93.42**	+0.7%
Room	52.60	**74.96**	+42.5%
Bench	94.94	**95.11**	+0.2%
Sofa	80.84	**89.09**	+10.2%
Average	70.24	**82.38**	+17.3%

Table 12: **Effect of automatic hyperparameter selection (mIoU %).** Hand-tuned: manually chosen lambda/variance fraction with a fixed number of children across the dataset, Auto-tuned: auto-derived lambda, adaptive spatial-variance-fraction clustering, and fixed children/fraction. Both rows are our own pipeline under the same architecture.

A.2.4 Covariance Type

Given we are using a GMM, another tunable property is the covariance type to fit, as such we ablate all three options (full, diagonal, spherical). Table 13 shows the difference in compute time and performance for the different options. Despite performing a more complex fit, the time cost of a Full covariance is relatively small compared to the simplest Spherical fit and scores the best overall, resulting a Full covariance throughout the paper.

	LERF-OVS		3D-OVS		Mip-NeRF360	
Covariance	mIoU	Fit [s]	mIoU	Fit [s]	mIoU	Fit [s]
Full	**68.33**	10.55	**93.53**	61.22	78.92	47.19
Diagonal	67.32	10.73	93.39	60.91	79.65	**46.78**
Spherical	68.11	**10.64**	93.28	**60.65**	**79.78**	47.04

Table 13: **Covariance type.** A comparison of different GMM covariance types: spherical in which a uniform radius is fit, diagonal for axis-aligned scale, and full for rotation+scale fitting. Despite the differing complexities, the batched fitting has nearly the same execution time while **full** covariance results in better segmentation.

	LERF-OVS		3D-OVS		Mip-N360	
Dimensionality	mIoU	Fit [s]	mIoU	Fit [s]	mIoU	Fit [s]
64	**67.92**	**10.6**	**93.40**	**60.9**	79.45	**47.0**
128	67.10	18.7	92.65	104.9	**79.50**	81.3
Full	67.47	69.4	93.13	391.6	78.32	292.6

Table 14: **Feature dimensionality: dataset-averaged mIoU (%) and fit time (s).** Different PCA-based semantic dimensionality compression sizes are compared. This compressed latent is only used the clustering of the GMM likely resulting in the low variance in actual mIoU across dimensions, but this reduction results in far faster fitting stages.

A.2.5 Feature dimensionality.

Along with covariance type, we can greatly simplify the GMM fitting but using a PCA reduction of the semantic latent space. It should be noted this reduced space is only used for the clustering, while the full latent is still used for the centroids and queries. We test this again with the CLIP+SAM backbone with 3 GMM fitted dimensionalities: $\{64, 128, \text{full (512)}\}$ as shown in Table 14. Unlike covariance type, there is not even a slight lean toward one value, with nearly even scores at different dimensionalities, however, there is a significant speed up (and memory reduction) from using the lower 64 dimensional space when performing fitting. Therefore the remainder of these ablations and the recommended global recipe uses a clustering dimensionality of 64.

A.2.6 Adaptive Tree Depth

Tree shape had previously only been tested as a fixed (depth, children) pair on LERF-OVS. We added an adaptive alternative, `spatial_variance_fraction` (Section 3.5), which lets the tree grow

deeper only where within-cluster spatial variance warrants it, and tested it against larger fixed-depth configurations (Table 15). The result is minor loss by adding the adaptive depth, likely do loss of fine resolution in smaller clusters. Ultimately, even though this more aggressive early stopping results in slightly smaller trees, it is left turned off for the default settings in favor of the other early stopping criteria (minimum Gaussian counts) and a fixed depth to act as the maximum depth for fitting.

Dataset	# scenes	Best fixed-depth [%]	Best adaptive [%]	Δ [pts]
LERF-OVS	4	69.36	68.07	−1.29
3D-OVS	2	90.23	89.79	−0.44
Mip-NeRF360	2	82.31	82.16	−0.15
ScanNet++ (10-scene)	10	26.92	26.78	−0.14
Overall	18	49.54	49.11	−0.43

Table 15: **Adaptive depth.** Comparison of the fixed depth vs adaptive depth scores with other settings kept constant across the different datasets. The overall slightly lower score from adaptive depth means it was disabled for the global optimization (other forms of early stopping are left

A.2.7 Tree Shape

A full list of scenes' best and worst tree shape is shown in Table 16, showing a fairly small difference in the end from a poorly set child/depth and a well tuned one, likely due to the early stopping criteria.

Scene	Best (children, max_depth)	Worst (children, max_depth)	Spread [pts]
figurines	66.46 (c20,d10)	64.17 (c12,d5)	2.29
ramen	66.95 (c8,d15)	65.06 (c8,d10)	1.89
teatime	84.03 (c8,d15)	82.14 (c12,d20)	1.89
waldo_kitchen	60.00 (c6,d10)	56.37 (c8,d15)	3.64
room	85.66 (c8,d20)	79.74 (c12,d10)	5.92
sofa	94.80 (c6,d15)	82.69 (c12,d15)	12.11
counter	80.31 (c8,d10)	78.41 (c6,d10)	1.90
bonsai	84.31 (c12,d10)	81.02 (c8,d10)	3.29
09c1414f1b	25.15 (c14,d6)	24.72 (c10,d8)	0.43
0d2ee665be	28.22 (c14,d6)	27.82 (c10,d8)	0.39
38d58a7a31	37.56 (c10,d8)	36.42 (c10,d6)	1.14
3db0a1c8f3	25.90 (c10,d8)	25.11 (c10,d12)	0.80
5ee7c22ba0	34.23 (c14,d12)	31.66 (c14,d6)	2.57
5f99900f09	42.19 (c8,d12)	41.50 (c8,d6)	0.69
a8bf42d646	22.39 (c14,d8)	21.43 (c8,d8)	0.96
a980334473	20.19 (c14,d6)	19.94 (c14,d8)	0.25
c5439f4607	13.94 (c8,d8)	13.62 (c8,d6)	0.32
cc5237fd77	19.43 (c14,d6)	19.04 (c14,d12)	0.40

Table 16: **Tree shape (children×max_depth).** Best/worst mIoU across the 9-10 (children, max_depth) cells tested per scene. Most scenes show a modest 0.25-3.64-point spread, comparable to the covariance/dimensionality spreads above; Room and Sofa are clear exceptions at 5.92 and 12.11 points respectively.

A.2.8 Lift Aggregation

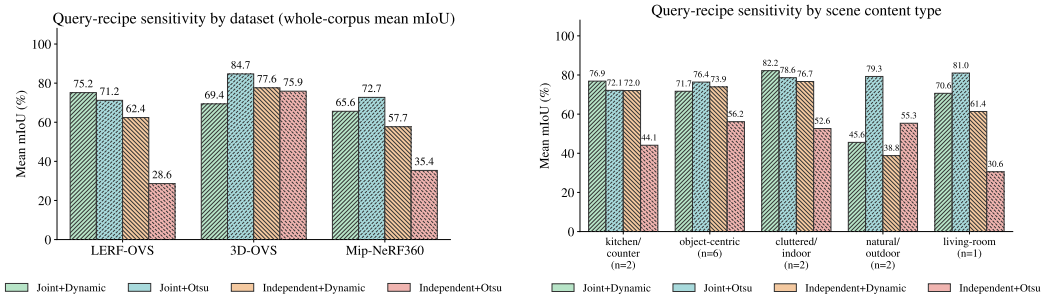
A small sweep (Table 17) tested the per-Gaussian feature-lifting aggregation rule. **Weighted Mean** in which we weight each observing view's contribution by its significance in line with Occam's LGS. As well as a simpler **Dominant View**, in which we keep only the single highest-weight view, as presented in Dr. Splat Jun-Seong et al. (2025). Weighted mean wins on all 3 scenes tested, by a wide enough margin the improved computation of dominant view was not worth the cost and was dropped from the global recipe.

Scene	Weighted Mean [%]	Dominant View [%]	Gap [pts]
figurines	66.05	46.50	+19.55
room	85.72	84.08	+1.64
counter	81.19	67.04	+14.15

Table 17: **Phase 3: weighted_mean vs. dominant_view, one representative scene per dataset.** As this stage impacts the slowest stage of our pipeline, preprocessing, we perform a smaller ablation looking at one scene per dataset. Under **Weighted mean** we combine the features from all views but weigh each based on the significance of this view using the formulation from Occam's LGS. **Dominant view** instead sees if we can bypass this expensive operation to speed up preprocessing by only using the most significant view to lift the semantic features from their 2D maps to 3D Gaussians, similar to Dr. Splat. Ultimately, the dominant view far underperformed the weighted mean and was dropped from the default configuration.

A.2.9 Relevancy Scoring

Beyond clustering, the relevancy-query recipe itself (joint vs. independent SAM querying, Otsu vs. dynamic thresholding) is also tunable per scene. Figure 4 shows the breakdown per dataset and per scene content for the different evaluation modes. The joint evaluation mode outperforms their independent counterparts resulting joint being used as the default configuration. Additionally, for fairness this was used with the baselines resulting in the slight variations from the reported numbers in the tables. Dynamic vs Otsu thresholding has less of an overall impact in most cases, but Otsu far outperforms its Dynamic counterpart on nature scenes leading to its inclusion in the default recipe.



(a) By dataset.

(b) By scene content type.

Figure 4: Query-recipe sensitivity, grouped by dataset (left) and by scene content type (right). Joint scoring (used by every baseline and Ours by default) dominates Independent scoring in nearly every group, and the best threshold mode is context-dependent, however the strong outdoor performance of Otsu lead to its use as default.

A.3 Additional Limitations

A.3.1 Thin Object Merging

Certain objects with either thin small regions or that present larger unobserved boundaries between flat objects can fail to produce sufficiently constrained clusters. This is particularly noticeable in the "dinosaur" case in the Room scene where the long tail of the dinosaur is too thin and long relative to the rest of the clusters at this scale resulting in portions of it being misallocated to the table top. This could be addressed by a denser 3DGS scene or by taking over from the autotuning defaults to force deeper levels with smaller min Gaussian counts.

We observe something similar in the "flowerpot on the table" label in the Garden scene, in which a background flower pot having small regions at the borders merging with background clusters. Again for well observed objects with enough Gaussians this issue is not as prominent.

A.3.2 ScanNet++ mAcc

While we beat all baselines' f-mIoU in ScanNet++, we do fall behind SceneSplat++ in f-mAcc. Similar to the issue above but on a larger scale, we struggle with a lot of finer classes that are flat against larger surfaces such as posters or ceiling lights. The poor clustering here ends up reducing the our localization hits on certain objects to below the cutoff threshold for success, resulting in a handful of objects scoring near 0. This averaged out with the rest of the objects scoring >80% results in the comparatively lower score. Compared to SceneSplat++ however, we score far better on the labels we are able to localize, rather than scoring in the midrange across most labels. Additionally, the clustering issues are exacerbated with the SigLIP feature set (likely resulting in the fairly low scores in Table 6) as the relative variance in text is far less. The cross cosine-similarity for 33 of the 100 classes are between 0.499 and 0.542. This limited variance means even with our row-normalized weighting the semantic information does not provide a strong enough signal ensure clustering and in some cases may even trigger early stopping.

A.4 Additional Qualitative Samples

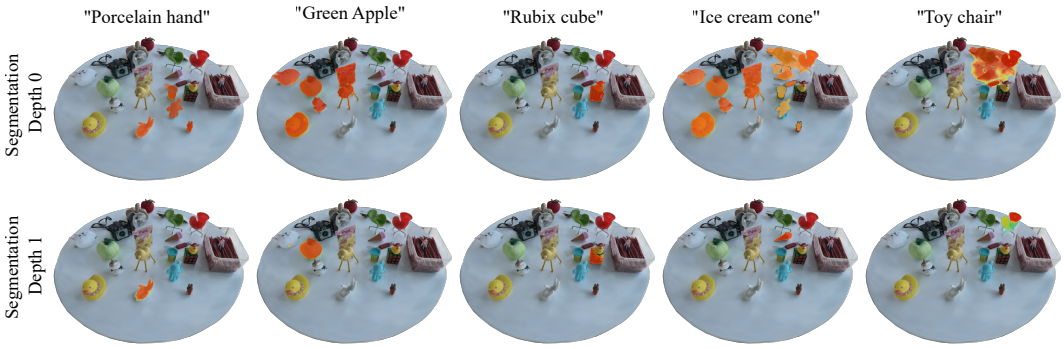


Figure 5: **Figurines Segmentations.** Renders of segmentation queries at different depths.



Figure 6: **LERF-OVS – Teatime.** Real renders logged during evaluation: RGB, prediction/GT overlay, relevancy heatmap for "three cookies", and one hierarchical-clustering level with different clusters colored differently

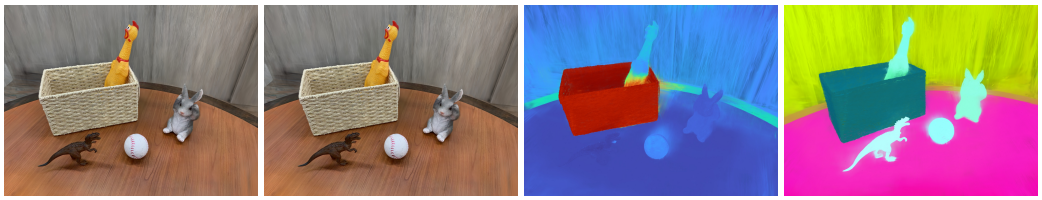


Figure 7: **3D-OVS – Room.** Real renders logged during evaluation: RGB, prediction/GT overlay, relevancy heatmap for "weaving basket", and one hierarchical-clustering level with different clusters colored differently



Figure 8: **Mip-NeRF360 – Counter.** Real renders logged during evaluation: RGB, prediction/GT overlay, relevancy heatmap for "baking gloves", and one hierarchical-clustering level with different clusters colored differently



Figure 9: **ScanNet++ – cc5237fd77.** Real renders logged during evaluation: RGB, prediction/GT overlay, relevancy heatmap for "floor",and one hierarchical-clustering level with different clusters colored differently