

EmbodiedSplat: Online Feed-Forward Semantic 3DGS for Open-Vocabulary 3D Scene Understanding

Supplementary Material

Table of Contents

A Additional Explanations.	1
A.1. Implementation Details.	1
A.2. EmbodiedSplat	2
A.3. EmbodiedSplat-fast.	4
B Discussions.	4
B.1. 2D VLM	4
B.2. More Baselines.	5
B.3. Limitations	6
C Additional Experiments.	6
C.1. 3D Semantic Segmentation	6
C.2. 2D-rendered Semantic Segmentation	7
C.3. Novel View Synthesis	7
C.4. Ablations on Memory Compression Rate	8
C.5. Qualitative Results	8

A. Additional Explanations.

In this section, we supplement the additional details about our study including the experimental settings (Sec. A.1) and further details for the overall framework of our EmbodiedSplat (Sec. A.2) and EmbodiedSplat-fast (Sec. A.3).

A.1. Implementation Details.

Model & Training Details. EmbodiedSplat is built on top of pretrained FreeSplat++ [41], which is the feed-forward 3DGS that supports whole-scene 3D reconstruction. 3D sparse U-Net [7] with temporal-aware memory adapter [46] are attached to obtain 3D geometric-aware features $\hat{\mathbf{g}}$ where Minkowski Res16UNet18A [7] is adopted as 3D U-Net. The training of EmbodiedSplat consists of two stages which are explained in Sec. 5. Memory-based adapter is trained only in the second stage, where it is zero-initialized to enable a smooth fine-tuning, following [46]. We adopt Adam [1] optimizer with an initial learning rate of $1e-4$ followed by cosine decay for both stages.

Keyframe Selection. For the experiments, we select keyframes for each testing scene that cover the entire scene from the full set of multi-view images, following [11, 16, 36]. Specifically, we calculate the pose distance between two cameras as follows:

$$\text{dist}(\mathbf{T}_{\text{rel}}) = \sqrt{\|\mathbf{t}_{\text{rel}}\|^2 + \frac{2}{3}\text{tr}(\mathbb{I} - \mathbf{R}_{\text{rel}})}, \quad (9)$$

Dataset	Scene ID	Num of multi-view images	Num of keyframes
ScanNet [9]	scene0000_01	5920	363
	scene0046_00	2480	280
	scene0079_00	1196	119
	scene0158_00	1920	83
	scene0316_00	770	45
	scene0389_00	1415	205
	scene0406_00	1414	120
	scene0521_00	1566	84
	scene0553_00	1500	61
	scene0616_00	3027	226
ScanNet++ [47]	09c1414f1b	2391	428
	9071e139d9	1221	310
	a24f64f7fb	652	138
	c49a8c6cff	1188	253
Replica [38]	office0	900	230
	office1	900	230
	office2	900	230
	office3	900	230
	office4	900	230
	room0	900	230
	room1	900	230
	room2	900	230

Table 7. Configurations of the three benchmarks [9, 38, 47].

where $\mathbf{T}_{\text{rel}} = [\mathbf{R}_{\text{rel}}|\mathbf{t}_{\text{rel}}]$ denotes the relative pose between two cameras. If the pose distance between current frame and last keyframe exceeds 0.1, we append the current frame to the list of keyframes. Collected keyframes are treated as streaming images for the experiment of our EmbodiedSplat. Tab. 7 presents the list of testing scenes for each benchmark with the number of selected keyframes. Note that the same set of keyframes is used to train all per-scene optimized baselines [6, 21, 22, 26, 32, 37, 43] to ensure fair comparisons.

Point clouds Annotation. To evaluate the performance on 3D semantic segmentation, we assign text labels to the ground-truth point clouds using the optimized semantic 3DGS, following [17, 18]. Given M semantic Gaussians and C text labels, we first compute the per-Gaussian semantic logits $\mathbf{P} \in \mathbb{R}^{M \times C}$. To obtain the semantic logit for a point $\mathbf{p} \in \mathbb{R}^3$, we aggregate the semantic logits of multiple Gaussians weighted by their Mahalanobis distance [10] to \mathbf{p} . Specifically, the scaled semantic logit contributed by i -th Gaussian to point \mathbf{p} is formulated as:

$$\hat{\mathbf{P}}_i = \exp(-\frac{1}{2}(\mathbf{p} - \mu_i)^\top \Sigma_i^{-1}(\mathbf{p} - \mu_i))\mathbf{P}_i, \quad (10)$$

where \mathbf{P}_i , μ_i and Σ_i are the semantic logits, 3D mean vector and 3D covariance of i -th Gaussian, respectively. The final semantic logits for a point \mathbf{p} is then computed by summing the weighted logits from every neighboring Gaussians that are located sufficiently close to \mathbf{p} , as formulated in:

$$\hat{\mathbf{P}} = \sum_{i \in \mathcal{N}(\mathbf{p})} \hat{\mathbf{P}}_i, \quad (11)$$

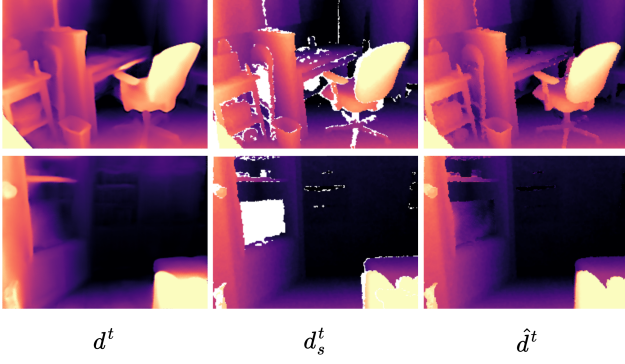


Figure 6. Depth visualizations with RGB-D inputs.

where $\mathcal{N}(\mathbf{p})$ denotes the set of neighboring Gaussians of point \mathbf{p} . Annotated point clouds are compared to the ground-truth semantic labels to evaluate the 3D semantic segmentation performance. Since EmbodiedSplat and all of the baselines exploit the camera poses given by the dataset, the resulting 3DGS is already aligned with the ground-truth point clouds which enables the proper aggregation of 3DGS to given 3D points via Eq. 10.

Using Ground-Truth Depth Maps. Embodied agent equipped with depth sensor can acquire RGB-D inputs instead of RGB alone. Therefore, we also evaluate the performance of our EmbodiedSplat with RGB-D instead of using depth predictions from the model (*cf.* *gray*-colored rows in Tab. 1-2). However, depths estimated from sensor often contains *holes* which represents missing or invalid depth values. These typically arise from the surfaces that are reflective, transparent, too dark, too thin, or outside the valid measurement range. Hence, we fill the missing regions of the sensor depth d_s^t with depth predictions d^t from our model, formulated as:

$$\hat{d}^t(i) = \begin{cases} d_s^t(i), & \text{if } d_s^t(i) > 1e-3 \text{ and } d_s^t(i) < 10, \\ d^t(i), & \text{otherwise} \end{cases} \quad (12)$$

where $i \in \{1, \dots, H \times W\}$. Fig. 6 shows the d^t , d_s^t and \hat{d}^t , respectively. Sensor depths d_s^t show large missing parts indicated with white, where those parts are filled by predicted depths, resulting in a smooth and complete depth maps \hat{d}^t .

A.2. EmbodiedSplat

The main objective of our EmbodiedSplat is to map the T number of posed streaming images into open-vocabulary 3DGS which supports diverse perception tasks such as 1) 3D semantic segmentation, 2) 2d-rendered semantic segmentation and 3) novel-view synthesis with depth rendering, while achieving near real time reconstruction speed. Here, we explain further details for inference pipeline of our EmbodiedSplat.

Warm-up Stage. Our EmbodiedSplat first collects the $N = 30$ number of images and reconstructs the initial semantic 3DGS as warm-up stage. The warm-up stage is performed

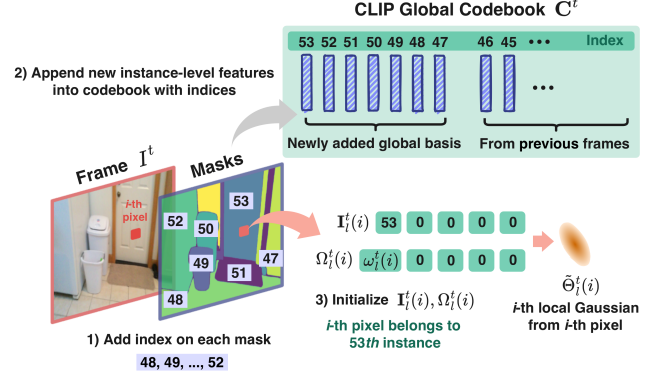


Figure 7. Toy example of sparse coefficient field initialization at i -th pixel. 1) We first add index to each instance-level mask. Index starts from 47 since the latest index in global codebook from previous time step is 46. 2) Instance-level features extracted from current frame I^t are appended into codebook with attached indices, producing updated codebook C^t . 3) We initialize the index cache $\mathbf{I}_i^t(i)$ and weight cache $\Omega_i^t(i)$ of i -th local Gaussian aligned with i -th pixel. Since i -th pixel belongs to instance ‘53’, index ‘53’ is added to first entry of $\mathbf{I}_i^t(i)$: $\mathbf{I}_i^t(i, 0) = 53$. Corresponding confidence value $\omega_i^t(i)$ is further inserted into first entry of $\Omega_i^t(i)$.

offline, requiring approximately 33–34 seconds in the EmbodiedSplat setting and 4–5 seconds in the EmbodiedSplat-fast setting. Constructed semantic 3DGS serves as starting point which is then progressively expanded in an online manner as the model continues to explore the scene.

Local Semantic Gaussians Field. Given the current frame I^t with time step t , we select the $N = 30$ past frames from the previous time steps $t - N$ to $t - 1$. N reference views and I^t are converted to the local semantic Gaussians field $\hat{\Theta}_l^t = \{\mu_l^t, \omega_l^t, \mathbf{f}_l^t, \mathbf{I}_l^t, \Omega_l^t, \hat{\mathbf{g}}_l^t\}$ with updated global codebook C^t through the process described below:

- **Local gaussian triplets $\{\mu_l^t, \omega_l^t, \mathbf{f}_l^t\}$:** We follow the FreeSplat++ to obtain the local gaussian triplets by leveraging the CNN-based network \mathcal{E} . Specifically, the backbone features of I^t and N reference views are extracted using a shared 2D backbone, after which a cost volume is constructed between them via plane sweep stereo [8, 19]. Obtained cost volume is then processed by UNet++ [52]-like decoder, outputting the depth map $d^t \in \mathbb{R}^{H \times W}$, pixel-wise Gaussian latents $\mathbf{f}_l^t \in \mathbb{R}^{H \times W \times D}$ and confidence scores $\omega_l^t \in \mathbb{R}^{H \times W}$ for each Gaussian latent. Finally, 2D pixels are unprojected to 3D space by using depth prediction d^t , resulting in 3D points $\mu_l^t \in \mathbb{R}^{H \times W \times 3}$. Here, $H \times W$ denotes the resolution of input images and D represents feature dimension of \mathbf{f}_l^t . Kindly refer to FreeSplat++ [41] for more details.
- **Sparse Coefficient Field $\{\mathbf{I}_l^t, \Omega_l^t\}$ and codebook C^t :** We lift the original 2D CLIP features to each Gaussian through our novel sparse coefficient field, preserving both memory efficiency and full semantic capability of CLIP. We provide the illustration of toy example for local sparse

coefficient field initialization in Fig. 7 for better understanding.

- **3D CLIP Features $\hat{\mathbf{g}}_l^t$:** Given the pixel-aligned Gaussian latents \mathbf{f}_l^t and pixel-wise CLIP features \mathbf{s}_l^t , semantic-aware Gaussian latents \mathbf{g}_l^t are obtained by doing $\mathbf{g}_l^t = \mathbf{f}_l^t + \text{proj}(\mathbf{s}_l^t)$, where MLP layer $\text{proj}(\cdot)$ is adopted to match the feature dimension between them. Then, $\mathbf{g}_l^t \in \mathbb{R}^{H \times W \times D}$ and 3D points $\mu_l^t \in \mathbb{R}^{H \times W \times 3}$ construct the local feature point clouds which is subsequently processed by 3D U-Net and memory-based adapter. Memory-based adapter retrieves the global Gaussian latents \mathbf{f}_g^{t-1} and their 3D coordinates μ_g^{t-1} from the previous time step, and selects the latents that are spatially close to the local feature point clouds in 3D space. While the 3D U-Net processes the local point clouds as input, selected global latents are injected to its intermediate layers. This design enables the network to aggregate geometric priors not only from the local point clouds of the current frame, but also from the previously reconstructed global scene. Resulting 3D features $\hat{\mathbf{g}}_l^t \in \mathbb{R}^{H \times W \times D^s}$ compensate the 3D geometric prior to 2D CLIP features, leading to the clear performance improvement in 3D scene understanding (cf. Tab. 3). Kindly refer to [46] for more details about memory-based adapter.

Gaussians Fusion. Obtained local semantic Gaussians $\bar{\Theta}_l^t$ are then fused with the global Gaussians $\bar{\Theta}_g^{t-1}$ to produce the updated global set $\bar{\Theta}_g^t$ at step t . Gaussian fusion is applied only to valid Gaussians pairs between the local and global sets where the valid pairs are determined according to the rules described below.

For i -th local Gaussian $\bar{\Theta}_l^t(i)$ which is aligned with i -th pixel of current frame I^t , we first obtain a set of Gaussians S_i^t from global set $\bar{\Theta}_g^{t-1}$ whose 3D coordinates project onto pixel i in frame I^t . Subsequently, we search the valid match for $\bar{\Theta}_l^t(i)$ within the S_i^t based on the *broader fusion* technique proposed by [41]:

$$m_i = \begin{cases} \arg \min_{j \in S_i^t} d_g^t(j), & \text{if } d_l^t(i) - \min_{j \in S_i^t} d_g^t(j) > -\delta, \\ \emptyset, & \text{otherwise} \end{cases} \quad (13)$$

where δ is a threshold, $d_g^t(j)$ is the depth value of j -th global Gaussian in S_i^t . Similarly, $d_l^t(i)$ is the predicted depth value of i -th local Gaussian on frame I^t . Local Gaussians which have no valid match (\emptyset) are directly appended to the global set without modification. In contrast, resulting valid Gaussians pairs $(i, m_i) \in \mathcal{P}^t$ are fused according to the following fusion rule:

$$\mu_g^t(m_i) = \frac{\omega_l^t(i) \mu_l^t(i) + \omega_g^{t-1}(m_i) \mu_g^{t-1}(m_i)}{\omega_l^t(i) + \omega_g^{t-1}(m_i)}, \quad (14a)$$

$$\omega_g^t(m_i) = \omega_l^t(i) + \omega_g^{t-1}(m_i), \quad (14b)$$

$$\mathbf{f}_g^t(m_i) = \text{GRU}(\mathbf{f}_l^t(i), \mathbf{f}_g^{t-1}(m_i)), \quad (14c)$$

$$\hat{\mathbf{g}}_g^t(m_i) = \text{GRU}(\hat{\mathbf{g}}_l^t(i), \hat{\mathbf{g}}_g^{t-1}(m_i)), \quad (14d)$$

$$\mathbf{I}_g^t(m_i), \Omega_g^t(m_i) = \mathcal{F}(\mathbf{I}_l^t(i), \Omega_g^{t-1}(m_i)). \quad (14e)$$

Here, $\mathcal{F}(\cdot, \cdot)$ denotes our proposed online fusion algorithm for sparse coefficient field which is described in Algorithm. 1. Next, we provide the further details about our sparse coefficient field with its online fusion algorithm $\mathcal{F}(\cdot, \cdot)$.

Motivation of Sparse Coefficient Field. At the beginning of Sec. ??, we introduce a naive approach for lifting the pixel-level 2D CLIP features to each Gaussian. Eq. 3 subsequently fuses the local CLIP features with paired global features based on the confidence-weighted average. It can be rewritten as:

$$\mathbf{s}_g^t(m_i) = \alpha \cdot \mathbf{s}_l^t(i) + (1 - \alpha) \cdot \mathbf{s}_g^{t-1}(m_i), \quad (15)$$

where $\alpha = \frac{\omega_l^t(i)}{\omega_l^t(i) + \omega_g^{t-1}(m_i)}$ denotes the coefficient of linear combination. During the exploration from step 1 to T , the 2D CLIP features of m_i -th global Gaussian may be produced by fusing the $k \geq 1$ number of local features by repeating the weighted-sum of Eq. 15 for $k - 1$ times. For example, if $k = 3$, the final CLIP features $\mathbf{s}_g^T(m_i)$ of m_i -th global Gaussian can be formulated as:

$$\begin{aligned} \mathbf{s}_g^T(m_i) &= (1 - \alpha_2)((1 - \alpha_1) \cdot \mathbf{s}_l(m_i, 0) + \alpha_1 \cdot \mathbf{s}_l(m_i, 1)) \\ &\quad + \alpha_2 \cdot \mathbf{s}_l(m_i, 2) \\ &= (1 - \alpha_2)(1 - \alpha_1) \cdot \mathbf{s}_l(m_i, 0) + (1 - \alpha_2)\alpha_1 \cdot \mathbf{s}_l(m_i, 1) \\ &\quad + \alpha_2 \cdot \mathbf{s}_l(m_i, 2) \\ &= \sum_{j=0}^{k-1} \beta_j \cdot \mathbf{s}_l(m_i, j), \quad k=3, \end{aligned} \quad (16)$$

where $\mathbf{s}_l(m_i)$ is the set of k CLIP features collected across k different views to produce $\mathbf{s}_g^T(m_i)$. α_i denotes the coefficient of i -th fusion operation from Eq. 15. Eq. 16 shows that $\mathbf{s}_g^T(m_i)$ can be represented as linear combination of $\mathbf{s}_l(m_i)$, where $\sum_{j=0}^{k-1} \beta_j = 1$. Here, $\mathbf{s}_l(m_i)$ serves as local basis for $\mathbf{s}_g^T(m_i)$ and β denotes coefficients for each local basis. Note that Eq. 16 can be generalized to any number of k .

The main idea of our sparse coefficient field $\{\mathbf{I}, \Omega\}$ is to leverage the weighted combination of local basis in Eq. 16 to store per-Gaussian CLIP features in memory-efficient manner. **Pixel-level to Instance-level:** We replace the *pixel-level* local basis $\mathbf{s}_l(m_i)$ with *instance-level* representations and allow every Gaussians to share the same global basis dictionary \mathbf{C} through lookup indices. Specifically, index cache \mathbf{I} stores the indices where they are linked with corresponding instance-level CLIP features stacked in the global codebook \mathbf{C} . These instance features serve as global basis function for semantic features of each Gaussian. Weight cache Ω further stores the coefficients β to retrieve the contribution of each basis. Finally, the original CLIP features $\mathbf{s}_g^T(m_i)$ can be restored by performing linear combination with sparse coefficient field via Eq. 4.

Online fusion of Sparse Coefficient Field. The number of local basis k in Eq. 16 may increase if exploration continues with collecting more views while the coefficients list β may

be also updated based on the confidence-weighted average of Eq. 15. Our online fusion Algorithm. 1 tracks the growth of local basis $s_g^T(m_i)$ and updates of coefficients β by using our sparse coefficient field along the exploration. To aid the understanding of our online update process, Fig. 9 presents a toy example that illustrates Algorithm. 1. It continuously collects the new evidence from incoming view by accumulating the index of new local basis into global index cache. Coefficients for each local basis in weight cache Ω are also updated based on the confidence-weighted average. To limit the maximum number of local basis for each Gaussian, Algorithm. 1 only keeps the top $L - 1$ basis with the highest coefficients. It removes the basis with low confidence scores, thereby sharpening the semantic Gaussian representations and preserving the memory efficiency.

Post Refinement. After processing all of the T images, we perform *floaters removal* proposed by [41] as a post refinement. It effectively removes the floater which improves the rendering quality while only taking 2-3 seconds in both EmbodiedSplat and EmbodiedSplat-fast.

A.3. EmbodiedSplat-fast.

EmbodiedSplat-fast is introduced as a faster and lighter variant of our EmbodiedSplat to satisfy the near real-time per-frame processing time. Three modifications are adopted to EmbodiedSplat: 1) Replacing the heavy 2D VLM into real-time models, 2) removing the 3D CLIP features to improve the inference speed and 3) proposing the efficient 3D search strategy to obtain per-Gaussian cosine similarities faster. Since EmbodiedSplat-fast doesn't use 3D modules such as 3D U-Net and memory adapter, and relies solely on 2D CLIP features with the sparse coefficient field, it can be directly built on top of pretrained feed-forward 3DGS [41] without any additional training. This training-free approach allows the direct combination with various types of 2D VLMs, highlighting the broad applicability of EmbodiedSplat-fast.

Codebook-based Cosine Similarity. In EmbodiedSplat-fast, we further introduce the efficient inference strategy to compute the per-Gaussian cosine similarities. The main idea is to precompute the cosine similarities between instance-level CLIP features stored in codebook C^T and text prompts, and then reuse these values to obtain the per-Gaussian cosine similarities through the linear combination derived from sparse coefficient field (*cf.* Eq. 8). Since the number of features stored in global codebook is much smaller than the total number of Gaussians (*cf.* Tab. 14), it significantly improves the 3D search latency. Specifically, our codebook-based cosine similarity results in $O(KD + M(L - 1))$ complexity while naive computation of per-Gaussian cosine similarities incurs $O(MD)$, where K denotes the codebook size, M is the number of Gaussians, D is the CLIP dimension and L is the cache size. We show

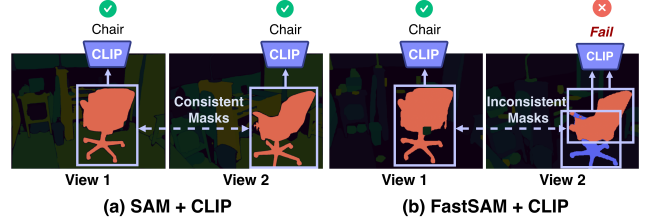


Figure 8. Multi-view inconsistent masks from FastSAM [49].

2D VLM Configurations	Contextual Information	Speed
SAM [23, 32] + CLIP [33]	✗	23220 ms
FastSAM [49] + CLIP [33]	✗	31.5 ms
FastSAM [49] + OpenSeg [15] (Ours)	✓	991.3 ms
FastSAM [49] + Mask-Adapter [28] (Ours)	✓	43.3 ms

Table 8. Comparisons on different 2D VLM configurations.

the complexity comparison between these two approaches:

$$\begin{aligned} O(KD + M(L - 1)) &\approx O(KD + M) \\ &\ll O(MD + M) \approx O(M(D + 1)) \approx O(MD), \end{aligned} \quad (17)$$

where $K \ll M$ and $L = 6$. Tab. 4 further reports the real inference time comparisons on NVIDIA 6000 Ada GPU.

B. Discussions.

In this section, we supplement additional discussions: Sec. B.1 explores the diverse configurations of 2D VLM in the embodied scenarios, justifying our choice of 2D models within EmbodiedSplat framework. Sec. B.2 explains additional works that are closely related to our study but are not included in the main paper. Finally, Sec. B.3 discusses the limitation of our EmbodiedSplat.

B.1. 2D VLM

Here, we discuss the motivation behind our choice of a 2D VLM. Most of the existing semantic 3DGS [6, 21, 26, 32, 43] exploit the combination of SAM [23] and CLIP [33] to extract the open-vocabulary cues from 2D images. Specifically, LangSplat [32] customizes the original SAM to output three levels of masks with different granularity by leveraging the SAM's inherent property of producing coarse-to-fine segmentations. After obtaining the instance-level masks, corresponding image patches are cropped and subsequently fed into CLIP to generate open-vocabulary features. Following works [6, 21, 26, 43] adopt the same strategy by simply adopting the same customized SAM from [32]. However, this simple combination of SAM and CLIP has two limitations in the embodied scenarios: 1) Customized SAM [32] + CLIP incurs prolonged inference time (23.22 seconds per image) which hinders the near real-time capability of the model. Main bottleneck arises from the heavy post-processing of customized SAM to obtain clean object-level masks for the entire image. 2) Feeding the cropped image patches to CLIP ignores the background part of the

objects which is crucial to understand the contextual information.

First limitation can be easily addressed by simply adopting the real-time 2D models such as FastSAM [49]. However, combining the FastSAM with CLIP further exacerbates the second issue. We empirically find that FastSAM frequently fails to generate masks with a consistent level of granularity; for example, it may produce an object-level mask in one view while generating part-level masks in another view with same object. Since the cropped patches of part-level masks lack the surrounding regions necessary to preserve object-level semantics, CLIP often produces incorrect semantic predictions for these patches which is illustrated in Fig. 8. To alleviate this issue, we adopt the combination of FastSAM and pixel-level CLIP models such as OpenSeg [15] and LSeg [25]. Because these models operate on the full image, each pixel-level CLIP feature inherently preserves global contextual information. The resulting per-pixel features are then pooled using the masks produced by FastSAM to obtain instance-level representations. To further improve the inference speed, we adopt MaskAdpater [28] into EmbodiedSplat-*fast*, which pools the instance-level CLIP features from MaskCLIP [50]-based architecture. Tab. 8 summarizes the comparisons among different 2D VLM configurations. Furthermore, Tab. 10 and Sec. C.1 examines the performance of diverse 2D VLM within our EmbodiedSplat framework.

B.2. More Baselines.

We discuss more recent baselines or concurrent works in semantic 3DGS which are closely related to our study but not mentioned in the main paper. Tab. 9 shows the overall comparison between our EmbodiedSplat and the additional baselines which are described next.

3D methods. In the main paper, we adopt clustering-based methods [26, 43] and feature-lifting approaches [6, 21] as a 3D baseline that supports direct 3d referring. Here, we introduce more baselines which fall in this 3D category. VoteSplat [20] is another clustering-based method where it groups the gaussians by exploiting Hough voting algorithm to reduce the training cost. Since their code is not publicly available, we do not compare the performance with VoteSplat. LUDVIG [29] is another recent work in feature-lifting approach, where they directly lift the pixel-wise CLIP features into per-scene optimized 3DGS without feature distillation process. Specifically, they collect the multiple CLIP features for each Gaussian across multi-view images and aggregate them based on the rendering weights obtained from rasterization function. CF³ [24] follows the LUDVIG to directly bind the 2D features into 3DGS while more focusing on reducing the number of Gaussians. Since the way they lift the features are highly overlapped with Occam’s LGS [6], we do not include LUDVIG and CF³ in Tab. 1

Method	Venue	Generalizable	Online	Whole-Scene
VoteSplat [20]	ICCV’25	✗	✗	✓
LUDVIG [29]	ICCV’25	✗	✗	✓
CF ³ [24]	ICCV’25	✗	✗	✓
LSM [12]	NeurIPS’24	✓	✗	✗
OVGaussian [4]	arXiv’24	✓	✗	✗
SLGaussian [3]	arXiv’24	✓	✗	✗
GSemSplat [40]	arXiv’24	✓	✗	✗
SemanticSplat [27]	arXiv’25	✓	✗	✗
UniForward [39]	arXiv’25	✓	✗	✗
Gen-LangSplat [35]	arXiv’25	✓	✗	✗
SIU3R [44]	NeurIPS’25	✓	✗	✗
EA3D [51]	NeurIPS’25	✗	✓	✓
EmbodiedSplat (<i>Ours</i>)	-	✓	✓	✓

Table 9. Comparison between EmbodiedSplat and additional baselines.

of the main paper. However, Tab. 10 provides the further comparison with them on 3D semantic segmentation.

Feed-forward semantic 3DGS. LSM [12] is a pioneering work that introduces semantic feed-forward 3DGS. They add an additional semantic head on the feed-forward 3DGS. 2D CLIP features obtained from multi-view images are then distilled into feed-forward model through semantic head via 2D rendering function. Although effective, LSM only performs with only two or a few input views, lacking the capability to understand the whole scene. Furthermore, LSM focuses on understanding the scene by rendering the 2D feature maps rather than directly referring the 3D Gaussians. Since our study focuses on whole-scene understanding with direct 3D inference which is crucial in embodied scenarios, we do not include LSM as baseline in Tab. 1. Several literatures [3, 4, 27, 35, 39, 40, 44] follow the similar framework with LSM, proposing diverse variants of semantic feed-forward 3DGS. However, 1) all of them do not provide the open-sourced code except for SIU3R [44]. 2) They don’t address the whole-scene semantic reconstruction. 3) Finally, they only discuss the offline setting, solely relying on pre-collected multi-view images which deviates from embodied scenarios.

SLAM + Semantic 3DGS. In contrast to aforementioned baselines, this category addresses the online reconstruction of semantic 3DGS by exploiting the SLAM pipeline. Online-LangSplat [22] falls within this category where it combines the language embeddings into MonoGS [30] which is 3DGS-based SLAM. EA3D [51] further proposes the advanced online framework based on HiCOM [14] which is the 4DGS-based SLAM. They improve the multi-view consistency among the 2D feature maps by exploiting the matching distributions between two adjacent frames. However, Online-LangSplat and EA3D both still require the per-scene optimization, which prevents them from generalizing to novel scenes in near real time. Furthermore, they distill the feature of 2D models into 3DGS via rendering function, inherently limiting the framework from supporting direct 3D referring. Note that the code of EA3D is not

Method	Inputs	ScanNet [9]						ScanNet200 [34]		Online / Offline
		10 classes		15 classes		19 classes		70 classes		
		mIoU	mACC	mIoU	mACC	mIoU	mACC	mIoU	mACC	
Occam’s LGS [6]	RGB	42.14	70.28	35.04	63.71	30.49	57.91	20.32	40.49	Offline
Dr. Splat [21]	RGB	39.21	66.66	31.84	60.58	28.38	55.85	19.29	33.84	Offline
LUDVIG [29]	RGB	41.11	68.34	33.73	62.90	29.34	56.98	21.23	39.87	Offline
CF ³ [24]	RGB	38.14	65.13	30.13	59.13	26.34	52.43	18.23	30.11	Offline
EmbodiedSplat (SAM [23, 32] + CLIP [33])	RGB	45.56	75.13	37.90	66.82	33.12	59.03	21.98	41.11	Online
EmbodiedSplat (FastSAM [49] + LSeg [25])	RGB	57.48	77.63	51.84	68.78	42.23	58.12	23.13	35.18	Online
EmbodiedSplat (FastSAM [49] + OpenSeg [15])	RGB	49.81	76.13	49.23	75.47	46.22	70.37	31.16	48.38	Online
EmbodiedSAM [45]	RGB-D	52.13	78.13	50.98	77.81	48.11	71.45	33.11	48.14	Online
OpenScene [31]	Point Cloud, RGB-D	54.56	80.45	53.74	79.85	50.71	72.75	33.84	50.06	Offline
EmbodiedSplat (FastSAM [49] + OpenSeg [15])	RGB-D	57.41	82.45	55.18	80.27	52.12	75.66	34.75	52.36	Online

Table 10. Additional comparisons on 3D Semantic Segmentation in ScanNet [9] and ScanNet200 [34] datasets.

publicly available yet.

B.3. Limitations

Our EmbodiedSplat is built on top of pretrained FreeSplat++ [41]. Hence, it inherits the limitation of FreeSplat++: when the feed-forward 3DGS fails to reconstruct the scene accurately, the resulting semantic Gaussian field becomes correspondingly noisy. We provide several examples where EmbodiedSplat fails to build clean semantic Gaussians due to the inaccurate 3DGS reconstruction.

Out-of-Distribution Scenarios. This is shown in Tab. 2 of the main paper where the EmbodiedSplat trained in ScanNet [9] dataset fails to outperform the baselines in Replica [38] due to the huge domain gap between real-world scenes and synthetic scenes. Since feed-forward 3DGS is overfitted to the real-world domain, it fails to perform accurate 3D reconstruction in the synthetic domain. Inaccurate 3DGS reconstruction leads to noisy lifting of semantic features to each Gaussian, finally resulting in low 3D segmentation performance.

Inaccurate Depth Estimation. We discuss the inaccurate depth estimation case in Sec. ?? of the main paper. If the model faces difficult regions for depth estimation such as *ceilings* or *transparent backgrounds*, and these cases are largely absent from the training dataset, feed-forward 3DGS tends to fail in producing high quality depth predictions. This is shown in the performance drop of Tab. 2 when the model is trained on ScanNet [9] and evaluated on ScanNet++ [47]. Since *ceiling* parts are largely absent in the multi-view images of ScanNet but frequently appear in ScanNet++, model trained on ScanNet tend to generate noisy depth maps for these regions during evaluation on ScanNet++. Inaccurate depth maps lead to noisy point clouds, which in turn degrade the quality of feature aggregation performed by the 3D U-Net and the memory-based

adapter of EmbodiedSplat. If the agent is equipped with depth sensors, this issue can be largely mitigated.

C. Additional Experiments.

Understanding the 3D scene with direct 3D referring is crucial in embodied scenarios for the faster inference and better spatial comprehension. Hence, main paper focuses on 3D Semantic Segmentation by annotating the point clouds without rendering the 2D feature maps. However, as we show in Fig. 1, our EmbodiedSplat supports diverse perception tasks such as 2D-rendered segmentation and novel-view synthesis. In this section, we present more various experiments that are not included in the main paper: Sec. C.1 presents comparisons against a broader set of baselines and further evaluates EmbodiedSplat with diverse 2D models on 3D semantic segmentation. Sec. C.2 explores the comparisons on 2D-rendered segmentation. Sec. C.3 conducts the experiments on novel-view synthesis in RGB space. Sec. C.4 provides deeper ablations on memory efficiency of our sparse coefficient field. Finally, Sec. C.5 provides more qualitative results of our EmbodiedSplat and EmbodiedSplat-*fast*.

C.1. 3D Semantic Segmentation

Here, we provide additional comparisons on 3D semantic segmentation with broader set of baselines and explores diverse 2D models within the EmbodiedSplat framework. Experimental setting is kept identical with the main paper and experiment is conducted on ScanNet [9] and ScanNet200 [34] datasets with varying number of classes: 10, 15, 19 and 70 classes.

Comparisons with 3DGS methods. 1st-7th rows of Tab. 10 presents the additional comparisons with semantic 3DGS which support direct 3D referring. Specifically, LUDVIG [29] and CF³ [24] are further added as a recent baselines. The 5th-7th rows of Tab. 10 further demon-

Method	Search Domain	ScanNet [9]					
		10 classes		15 classes		19 classes	
		mIoU	mACC	mIoU	mACC	mIoU	mACC
LangSplat [32]	2D	45.83	73.12	42.89	69.34	44.15	70.45
Occam's LGS [21]	3D	41.13	73.34	40.21	66.82	39.11	64.34
Dr. Splat [21]		40.11	71.62	38.45	66.11	39.67	65.72
EmbodiedSplat	3D	47.44	76.95	44.11	70.12	43.75	68.16

Table 11. Quantitative results on 2D-rendered semantic segmentation in ScanNet [9] dataset.

strate that EmbodiedSplat performs well across diverse 2D models, indicating that its compatibility is not restricted to any specific 2D VLM. It is worth noting that widely used SAM+CLIP combination is not suitable for embodied scenarios that require fast inference, as discussed in Sec. B.1. Hence, we adopt FastSam [49] with pixel-level CLIP model [33] to extract semantic cues from 2D images.

Comparisons with point-cloud methods. We further compare EmbodiedSplat with the point-cloud understanding methods in 8th-10th rows of Tab. 10. Specifically, we adopt OpenScene [31] as offline method and EmbodiedSAM [45] as online method. Since they take RGB-D as inputs, we feed same RGB-D into EmbodiedSplat for the fair evaluation. Our EmbodiedSplat outperforms both methods by exploiting the 3DGS representation which enables smooth feature aggregation onto 3D points using the Mahalanobis distance defined in Eq. 10.

C.2. 2D-rendered Semantic Segmentation

Here, we explore the 2D-rendered semantic segmentation with our EmbodiedSplat.

Rendering 2D Feature Map with EmbodiedSplat. Rendering the 2D feature maps from 3D Gaussians with high dimensional features incurs huge computational overhead. Hence, we adopt alternative approach which does not require direct rendering of Gaussian features. For pixel j of the target view, we obtain the list of the rendering weights for every Gaussians. Specifically, rendering weight can be easily obtained by leveraging rasterization function of Eq. 1 where $T_i \tilde{\alpha}_i$ denotes the rendering weight of i -th Gaussian. We keep top 5 Gaussians with the highest weight and compute the cosine similarities for each selected Gaussian with given text classes. Finally, cosine similarities of every 5 Gaussians are linearly combined using their corresponding rendering weights, finally outputting the cost map for pixel j . Note that rendering weights are renormalized to sum to 1 before performing the linear combination.

We compute two cost maps by using 2D CLIP features and 3D CLIP features, and ensemble them via Eq. 6 to obtain final costmap in 2D domain.

Experimental Settings. We evaluate the 2D-rendered semantic segmentation on interpolated novel views of ScanNet [9] dataset with 10, 15 and 19 classes. Testing scenes

Method	iPSNR \uparrow	ePSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	$\delta < 1.1 \uparrow$	Type
pixelSplat [2]	15.54	13.47	0.557	0.608	0.023	Offline
MVSplat [5]	16.51	13.67	0.591	0.541	0.323	Offline
PixelGaussian [13]	16.33	13.40	0.601	0.549	0.282	Offline
FreeSplat++ [41]	23.29	19.44	0.771	0.320	0.904	Offline
EmbodiedSplat	22.78	19.14	0.738	0.367	0.885	Online

Table 12. Whole Scene Reconstruction results on ScanNet [9]. iPSNR and ePSNR denotes PSNR on the interpolated views and extrapolated views, respectively.

Method	iPSNR \uparrow	ePSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	$\delta < 1.1 \uparrow$	Type
pixelSplat [2]	10.70	10.37	0.497	0.663	0.000	Offline
MVSplat [5]	11.10	10.62	0.497	0.648	0.028	Offline
PixelGaussian [13]	10.78	10.44	0.529	0.639	0.012	Offline
FreeSplat++ [41]	22.63	19.51	0.829	0.261	0.890	Offline
EmbodiedSplat	21.54	18.62	0.780	0.330	0.925	Online

Table 13. Whole Scene Reconstruction results on ScanNet++ [47]. iPSNR and ePSNR denotes PSNR on the interpolated views and extrapolated views, respectively.

are identical with 3D semantic segmentation setting.

Experimental Results. Tab. 11 exhibits the performance on 2D-rendered segmentation performance on ScanNet dataset. Our EmbodiedSplat shows the comparable performance with 2D-specific method such as LangSplat [32] even though it is not optimized to specific scene.

C.3. Novel View Synthesis

In this section, we evaluate the rendering quality of EmbodiedSplat in the RGB space.

Experimental Settings. Given the constructed whole-scene 3DGS, we render the interpolated and extrapolated novel views respectively to evaluate the novel view synthesis, following [41]. PSNR, SSIM [42] and LPIPS [48] are adopted as rendering metric. We further evaluate geometric accuracy by reporting the depth quality. Specifically, threshold tolerance $\delta < 1.1$ on depth difference between rendered depth and ground-truth depth is adopted as metric. We conduct the experiments on ScanNet [9] and ScanNet++ [47] datasets.

Baselines. We compare the rendering quality of our EmbodiedSplat with representative feed-forward 3DGS works: pixelSplat [2], MVSplat [5], PixelGaussian [13] and FreeSplat++ [41].

Experimental Results. Tab. 12 and Tab. 13 show that our EmbodiedSplat successfully adapt the inference pipeline of FreeSplat++ into online setting, where it shows the comparable performance to original FreeSplat++ which leverages the entire set of pre-collected images in offline setting. Hence, it inherits the superior performance of FreeSplat++ compared to the previous feed-forward 3DGS models [2, 5, 13] in whole-scene reconstruction setting.

Scene	Gaussians Num	Codebook Size	Total Size (MB)	Compression Ratio
scene0000_01	3.2M	8.7K	148	× 63 efficient
scene0046_00	2.4M	5.3K	106	× 65 efficient
scene0079_00	1.5M	2.8K	64	× 67 efficient
scene0158_00	1.1M	1.8K	48	× 68 efficient
scene0316_00	0.6M	0.6K	23	× 70 efficient
scene0389_00	1.9M	2.8K	82	× 69 efficient
scene0406_00	0.9M	2.1K	41	× 65 efficient
scene0521_00	1.1M	2.0K	49	× 67 efficient
scene0553_00	0.7M	0.8K	31	× 70 efficient
scene0616_00	2.3M	3.4K	98	× 68 efficient
Average	1.57M	3.0K	69	× 67 efficient

Table 14. Ablations on memory efficiency of sparse coefficient field in ScanNet [9] dataset.

C.4. Ablations on Memory Compression Rate

In this section, we further explore the memory efficiency of our proposed sparse coefficient field with CLIP global codebook.

Experimental Settings. We report the number of generated Gaussians and the number of CLIP features stored in the CLIP global codebook for each testing scene. Based on this, we estimate the total memory size of semantic Gaussians stored by sparse coefficient field. Specifically, we combine the size of the CLIP codebook with the sizes of the index and weight caches attached to each Gaussian. Finally, we report the memory compression ratio gained from our sparse coefficient field compared to naively storing every per-gaussian original CLIP features with 768 dimension.

Observations. Tab. 14 shows that the number of CLIP features stored in codebook is far smaller than the total number of Gaussians, yielding an average 67× improvement in memory efficiency compared to storing per-Gaussian original CLIP features. Our sparse coefficient field is highly practical since it doesn’t require any pretraining stage or per-scene optimization compared to the existing memory compression method such as Auto-encoder [32], PQ index [21] and per-scene optimized codebook [26, 43]. Instead, the sparse coefficient field is constructed on the fly alongside the semantic 3DGS reconstruction and supports real-time online updates through Algorithm. 1 (cf. Fig. 9), making it highly suitable for online settings.

C.5. Qualitative Results

In this section, we explain additional qualitative results of our EmbodiedSplat and EmbodiedSplat-fast.

Qualitative results on 3D Semantic Segmentation. Fig. 10 and Fig. 11 present the additional qualitative comparison on 3D semantic segmentation. Our EmbodiedSplat and EmbodiedSplat-fast output more clear segmentation mask with more accurate semantic classification compared to the 3D baselines [6, 21, 26, 43].

Qualitative results on 2D-rendered object search. Fig. 12

showcases the additional visualizations for 2D-rendered object search with our EmbodiedSplat. Text queries “*stool*” and “*book*” are given to each row. Our model outputs multi-view consistent segmentation results by exploiting the 3DGS representation.

Qualitative results on novel-view synthesis. Fig. 13 exhibits the novel-view synthesis results and rendered depths of our EmbodiedSplat. It supports novel-view rendering with high fidelity across the entire scene.

Video visualization. Video visualizations in project website demonstrates the online reconstruction process of semantic 3DGS with our EmbodiedSplat-fast in Bird’s-Eye View. It shows following key aspects of our framework: **1) Near real-time reconstruction:** EmbodiedSplat shows 5-6 FPS of per-frame processing time where it can effectively synchronize the semantic reconstruction process to its online exploration. **2) Online 3D perception with free-form language:** Our EmbodiedSplat-fast can localize the 3D objects based on the free-form language along its exploration. For example, video shows that EmbodiedSplat-fast progressively detects the “*guitar*” which is related to the given text prompt “*I wanna hear the music*”. Interestingly, our model supports the semantic refinement with re-exploration where it corrects the wrong semantic by exploring the same regions and collecting more views, as shown in the video. Our online fusion algorithm of sparse coefficient field enables this refinement since it accumulates the new evidences from the incoming images into the index and weight cache. Furthermore, it always keeps the top 5 entries with the highest confidence scores at each fusion step, effectively filtering out low-quality semantics signals along the exploration. Video further showcases several 3D localization examples using free-form languages. For instance, EmbodiedSplat-fast localizes “*chair*”, “*sofa*” and “*stool*” together when queried with the text prompt, “*where can I sit?*”. **3) Supporting diverse 3D perception tasks:** We also visualize the rendered RGB images along the camera trajectory as well as 2D-rendered PCA visualizations based on the CLIP features of each Gaussian. It demonstrates that our framework supports diverse perception tasks such as RGB reconstruction and semantic understanding in both 2D and 3D modalities.

References

- [1] Kingma DP Ba J Adam et al. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 1412(6), 2014. 1
- [2] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024. 7
- [3] Kangjie Chen, BingQuan Dai, Minghan Qin, Dongbin

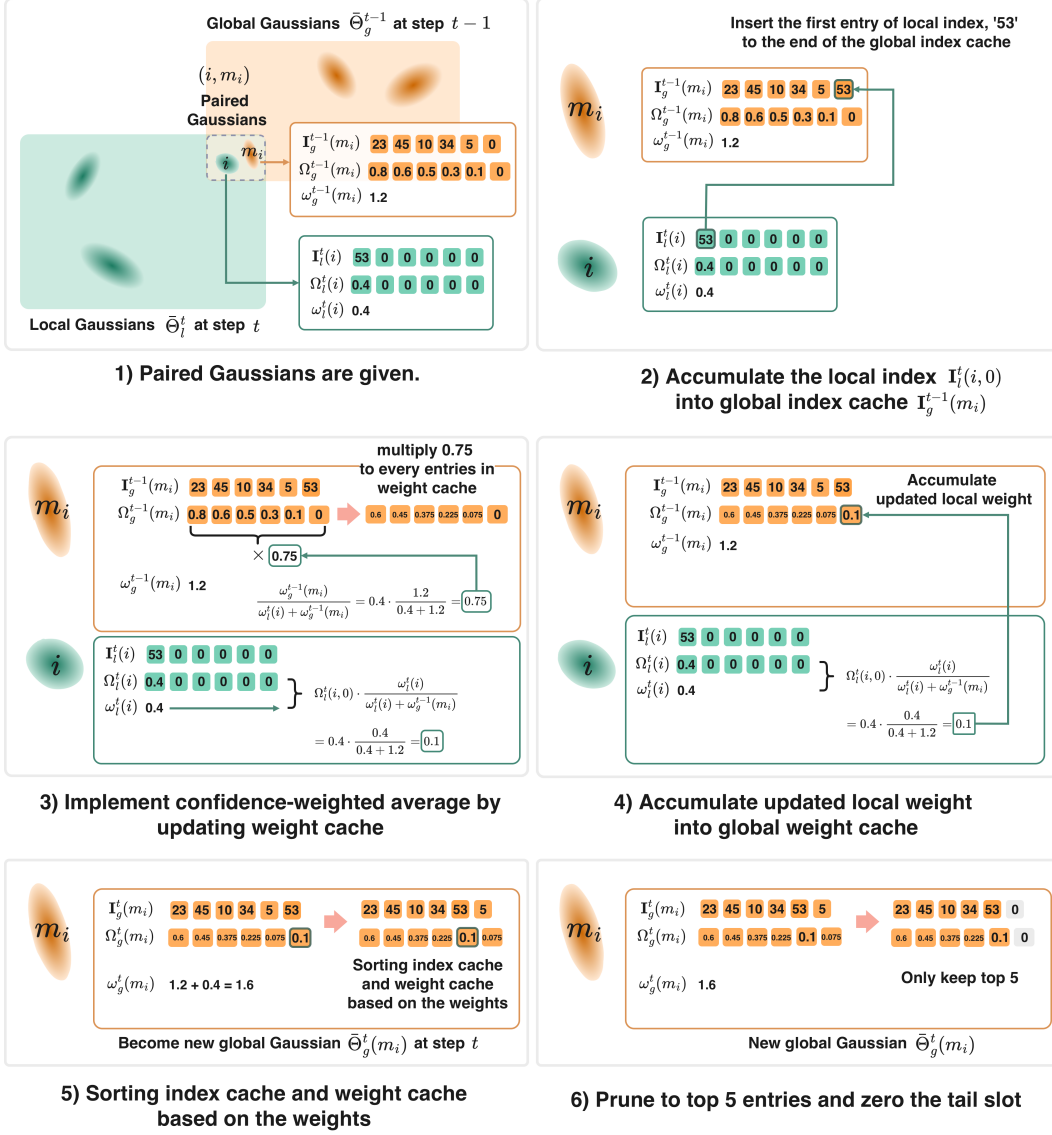


Figure 9. Toy example of online fusion algorithm with sparse coefficient field. 1) Sparse coefficient field of paired Gaussians (i, m_i) are fused by our online fusion Algorithm. 1. The local Gaussians (Green) which do not have valid match with global Gaussians are just appended to the global set without update. 2) **Line 1 of Algorithm. 1:** First entry of local index, $I_i^t(i, 0)$ is inserted to the last entry of global index cache $I_g^{t-1}(m_i, -1)$. In the above example, index 53 is appended to $I_g^{t-1}(m_i)$, such that $I_g^{t-1}(m_i, -1) \leftarrow 53$. **3-4) Lines 3-4 of Algorithm. 1:** Both the local weight cache $\Omega_i^t(i)$ and global weight cache $\Omega_g^{t-1}(m_i)$ are updated based on the confidence-weighted average. Specifically, all of the entries in $\Omega_g^{t-1}(m_i)$ are multiplied by 0.75, while the first entry of local weight cache $\Omega_i^t(i, 0)$ is scaled by 0.25. Scaled local weight value $0.25 \cdot \Omega_i^t(i, 0)$ is then inserted to the last entry of global weight cache, such that: $\Omega_g^{t-1}(m_i, -1) \leftarrow 0.25 \cdot \Omega_i^t(i, 0)$. Since both the index cache and weight cache of local Gaussian $\bar{\Theta}_i^t(i)$ are incorporated into global gaussian $\bar{\Theta}_g^{t-1}(m_i)$ from the previous stages, $\bar{\Theta}_g^{t-1}(m_i)$ becomes new m_i -th global Gaussian at step t : $\bar{\Theta}_g^t(m_i) \leftarrow \bar{\Theta}_g^{t-1}(m_i)$. i -th local Gaussian is simply discarded. **5-6) Lines 5-6 of Algorithm. 1:** We sort both the global weight cache and index cache based on the weight values $\Omega_g^t(m_i)$ in descending order. Then, we keep first $L-1 = 5$ entries and discard the last values by overwriting them with zero: $\Omega_g^t(m_i, -1) \leftarrow 0$, $I_g^t(m_i, -1) \leftarrow 0$. This keeps each cache size fixed at L along the exploration, effectively improving the memory efficiency.

Zhang, Peihao Li, Yingshuang Zou, and Haoqian Wang. Sl-gaussian: Fast language gaussian splatting in sparse views. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 3047–3056, 2025. 5

[4] Runnan Chen, Xiangyu Sun, Zhaoqing Wang, Youquan Liu,

Jiepeng Wang, Lingdong Kong, Jiankang Deng, Mingming Gong, Liang Pan, Wenping Wang, et al. Ovgaussian: Generalizable 3d gaussian segmentation with open vocabularies. *arXiv preprint arXiv:2501.00326*, 2024. 5

[5] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang,

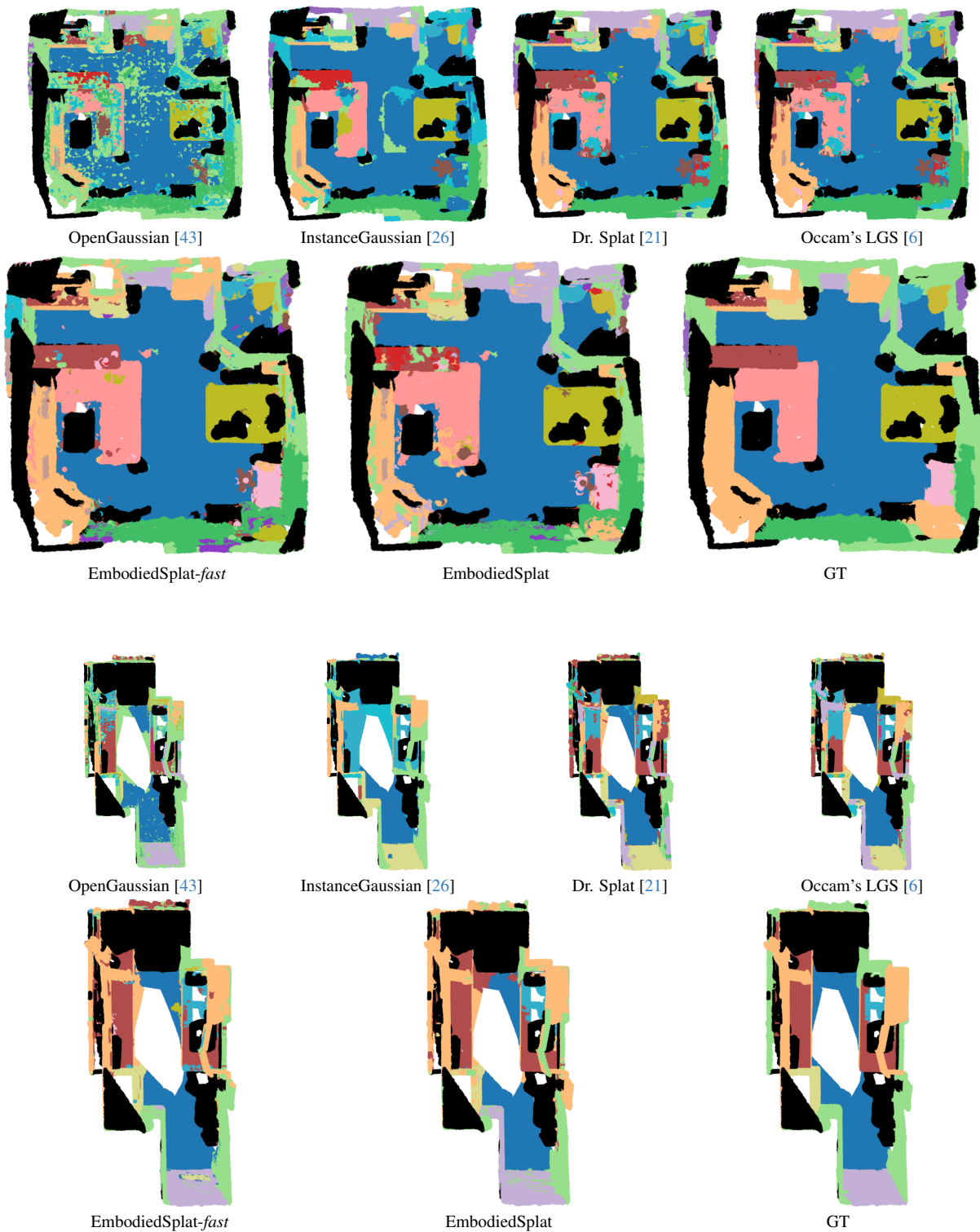


Figure 10. More qualitative comparisons on 3D semantic segmentation - (1)

Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer*

Vision, pages 370–386. Springer, 2024. 7

[6] Jiahuan Cheng, Jan-Nico Zaeche, Luc Van Gool, and Danda Pani Paudel. Occam's lgs: An efficient ap-

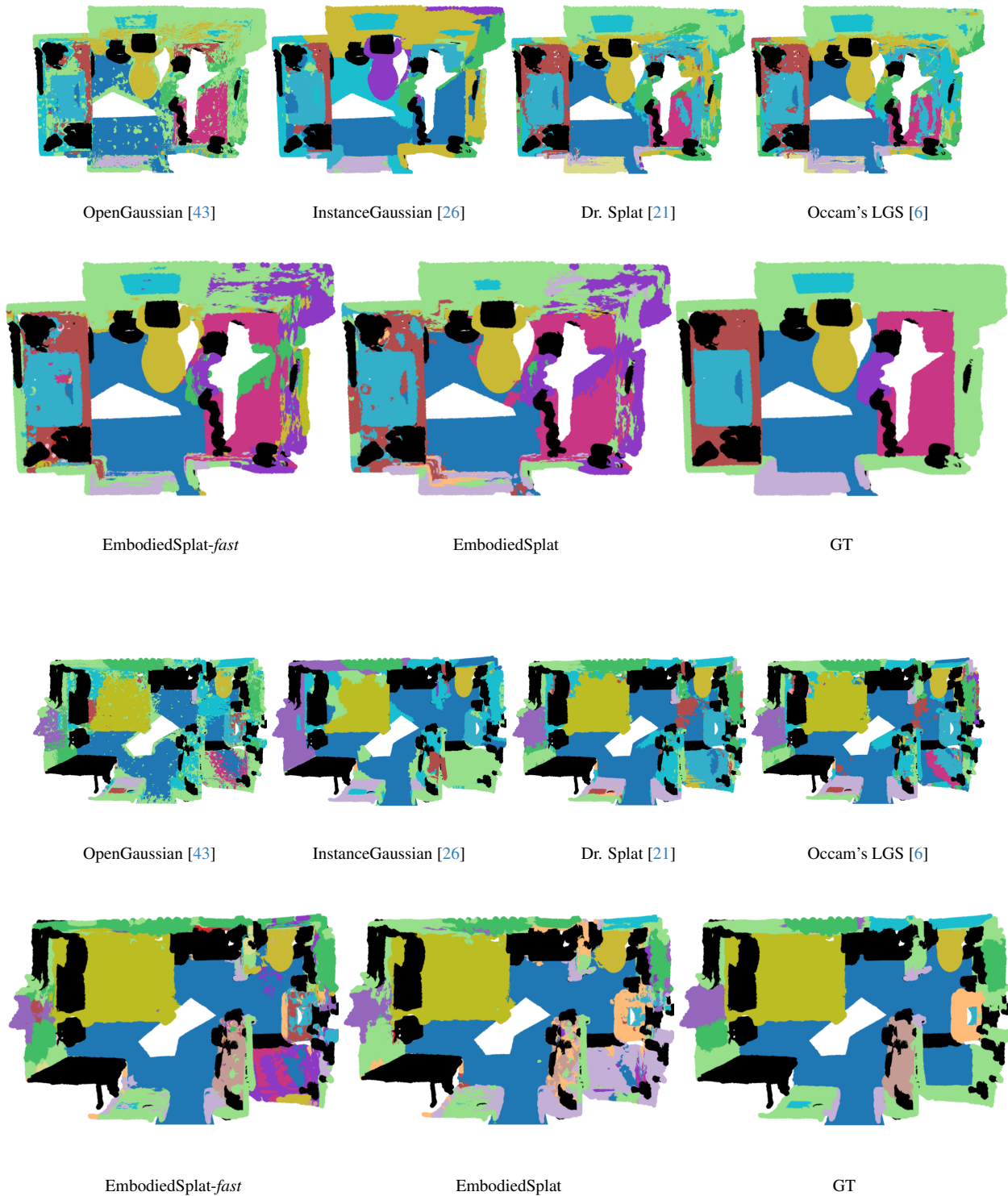


Figure 11. More qualitative comparisons on 3D semantic segmentaiton - (2)

proach for language gaussian splatting. *arXiv preprint arXiv:2412.01807*, 2024. 1, 4, 5, 6, 8, 10, 11

[7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d

spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084,

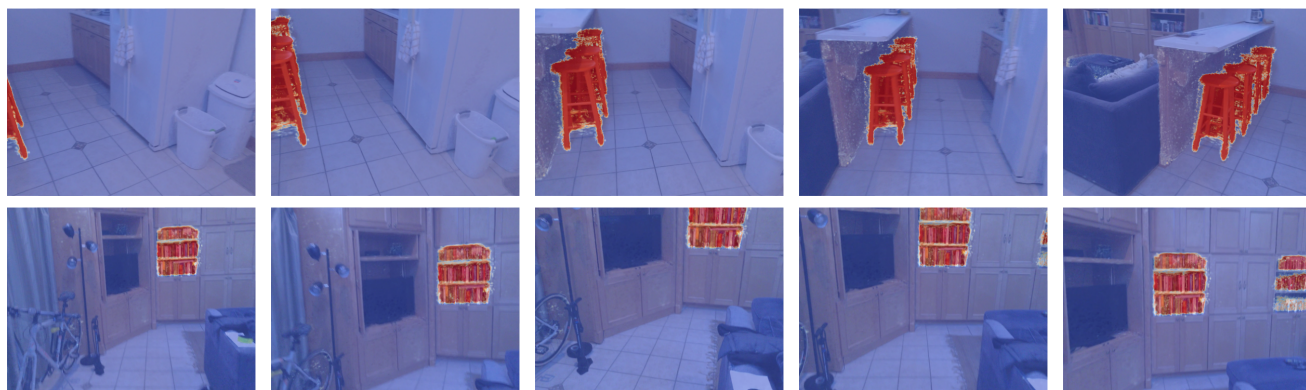


Figure 12. Qualitative results on 2D-rendered object search of our EmbodiedSplat.

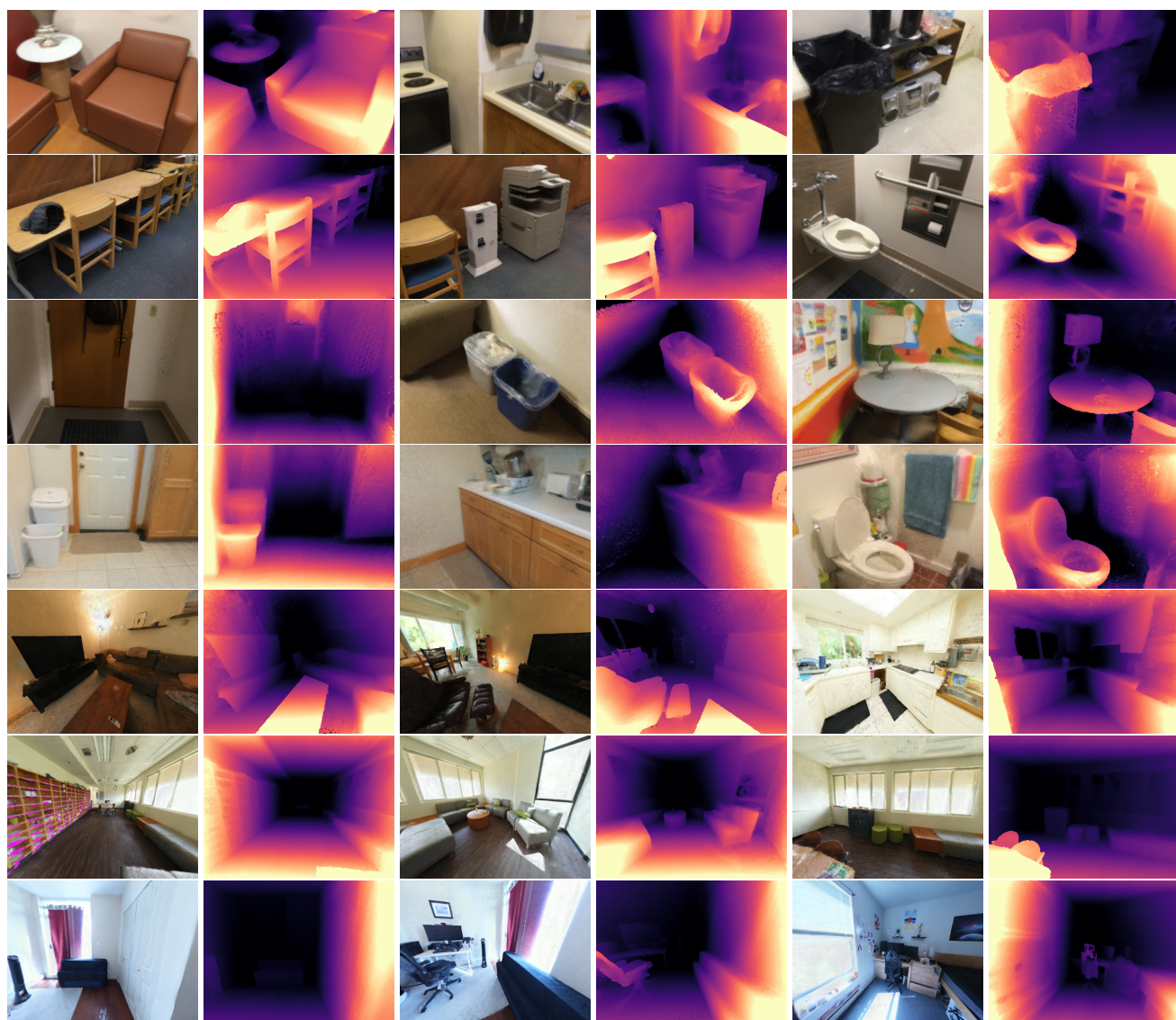


Figure 13. Qualitative results on novel-view synthesis and depth rendering of our EmbodiedSplat.

2019. 1

- [8] Robert T Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE computer society conference on computer vision and pattern recognition*, pages 358–363. Ieee, 1996. 2
- [9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 1, 6, 7, 8
- [10] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000. 1
- [11] Arda Duzceker, Silvano Galliani, Christoph Vogel, Pablo Speciale, Mihai Dusmanu, and Marc Pollefeys. Deepvideomvs: Multi-view stereo on video with recurrent spatio-temporal fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15324–15333, 2021. 1
- [12] Zhiwen Fan, Jian Zhang, Wenyan Cong, Peihao Wang, Renjie Li, Kairun Wen, Shijie Zhou, Achuta Kadambi, Zhangyang Wang, Danfei Xu, et al. Large spatial model: End-to-end unposed images to semantic 3d. *Advances in neural information processing systems*, 37:40212–40229, 2024. 5
- [13] Xin Fei, Wenzhao Zheng, Yueqi Duan, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Jiwen Lu. Pixelgaussian: Generalizable 3d gaussian reconstruction from arbitrary views. *arXiv preprint arXiv:2410.18979*, 2024. 7
- [14] Qiankun Gao, Jiarui Meng, Chengxiang Wen, Jie Chen, and Jian Zhang. Hicom: Hierarchical coherent motion for dynamic streamable scenes with 3d gaussian splatting. *Advances in Neural Information Processing Systems*, 37: 80609–80633, 2024. 5
- [15] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. Scaling open-vocabulary image segmentation with image-level labels. In *European conference on computer vision*, pages 540–557. Springer, 2022. 4, 5, 6
- [16] Yuxin Hou, Juho Kannala, and Arno Solin. Multi-view stereo by temporal nonparametric fusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2651–2660, 2019. 1
- [17] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Gaussianformer: Scene as gaussians for vision-based 3d semantic occupancy prediction. In *European Conference on Computer Vision*, pages 376–393. Springer, 2024. 1
- [18] Yuanhui Huang, Amonnut Thammatadatrakoon, Wenzhao Zheng, Yunpeng Zhang, Dalong Du, and Jiwen Lu. Gaussianformer-2: Probabilistic gaussian superposition for efficient 3d occupancy prediction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27477–27486, 2025. 1
- [19] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. Dpsnet: End-to-end deep plane sweep stereo. *arXiv preprint arXiv:1905.00538*, 2019. 2
- [20] Minchao Jiang, Shunyu Jia, Jiaming Gu, Xiaoyuan Lu, Guangming Zhu, Anqi Dong, and Liang Zhang. Votesplat: Hough voting gaussian splatting for 3d scene understanding. *arXiv preprint arXiv:2506.22799*, 2025. 5
- [21] 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*, pages 14137–14146, 2025. 1, 4, 5, 6, 7, 8, 10, 11
- [22] Saimouli Katragadda, Cho-Ying Wu, Yuliang Guo, Xinyu Huang, Guoquan Huang, and Liu Ren. Online language splatting. *arXiv preprint arXiv:2503.09447*, 2025. 1, 5
- [23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 4, 6
- [24] Hyunjoon Lee, Joonkyu Min, and Jaesik Park. Cf3: Compact and fast 3d feature fields. *arXiv preprint arXiv:2508.05254*, 2025. 5, 6
- [25] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022. 5, 6
- [26] Haijie Li, Yanmin Wu, Jiarui Meng, Qiankun Gao, Zhiyao Zhang, Ronggang Wang, and Jian Zhang. Instancegaussian: Appearance-semantic joint gaussian representation for 3d instance-level perception. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14078–14088, 2025. 1, 4, 5, 8, 10, 11
- [27] Qijing Li, Jingxiang Sun, Liang An, Zhaoqi Su, Hongwen Zhang, and Yebin Liu. Semanticsplat: Feed-forward 3d scene understanding with language-aware gaussian fields. *arXiv preprint arXiv:2506.09565*, 2025. 5
- [28] Yongkang Li, Tianheng Cheng, Bin Feng, Wenyu Liu, and Xinggang Wang. Mask-adapter: The devil is in the masks for open-vocabulary segmentation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14998–15008, 2025. 4, 5
- [29] Juliette Marrie, Romain Ménégaux, 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*, pages 7440–7450, 2025. 5, 6
- [30] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024. 5
- [31] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023. 6, 7
- [32] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024. 1, 4, 6, 7, 8
- [33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry,

- Amanda Askill, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 4, 6, 7
- [34] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *European conference on computer vision*, pages 125–141. Springer, 2022. 6
- [35] Pranav Saxena. Gen-langsplat: Generalized language gaussian splatting with pre-trained feature compression. *arXiv preprint arXiv:2510.22930*, 2025. 5
- [36] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. Simplerecon: 3d reconstruction without 3d convolutions. In *European Conference on Computer Vision*, pages 1–19. Springer, 2022. 1
- [37] Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. Language embedded 3d gaussians for open-vocabulary scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5333–5343, 2024. 1
- [38] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 1, 6
- [39] Qijian Tian, Xin Tan, Jingyu Gong, Yuan Xie, and Lizhuang Ma. Uniforward: Unified 3d scene and semantic field reconstruction via feed-forward gaussian splatting from only sparse-view images. *arXiv preprint arXiv:2506.09378*, 2025. 5
- [40] Xingrui Wang, Cuiling Lan, Hanxin Zhu, Zhibo Chen, and Yan Lu. Gsemplat: Generalizable semantic 3d gaussian splatting from uncalibrated image pairs. *arXiv preprint arXiv:2412.16932*, 2024. 5
- [41] Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. Freesplat++: Generalizable 3d gaussian splatting for efficient indoor scene reconstruction. *arXiv preprint arXiv:2503.22986*, 2025. 1, 2, 3, 4, 6, 7
- [42] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7
- [43] Yanmin Wu, Jiarui Meng, Haijie Li, Chenming Wu, Yahao Shi, Xinhua Cheng, Chen Zhao, Haocheng Feng, Errui Ding, Jingdong Wang, et al. Opengaussian: Towards point-level 3d gaussian-based open vocabulary understanding. *Advances in Neural Information Processing Systems*, 37:19114–19138, 2024. 1, 4, 5, 8, 10, 11
- [44] Qi Xu, Dongxu Wei, Lingzhe Zhao, Wenpu Li, Zhangchi Huang, Shunping Ji, and Peidong Liu. Siu3r: Simultaneous scene understanding and 3d reconstruction beyond feature alignment. *arXiv preprint arXiv:2507.02705*, 2025. 5
- [45] Xiuwei Xu, Huangxing Chen, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Embodiedsam: Online segment any 3d thing in real time. *arXiv preprint arXiv:2408.11811*, 2024. 6, 7
- [46] Xiuwei Xu, Chong Xia, Ziwei Wang, Linqing Zhao, Yueqi Duan, Jie Zhou, and Jiwen Lu. Memory-based adapters for online 3d scene perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21604–21613, 2024. 1, 3
- [47] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023. 1, 6, 7
- [48] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7
- [49] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023. 4, 5, 6, 7
- [50] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In *European Conference on Computer Vision (ECCV)*, 2022. 5
- [51] Xiaoyu Zhou, Jingqi Wang, Yuang Jia, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Ea3d: Online open-world 3d object extraction from streaming videos. *arXiv preprint arXiv:2510.25146*, 2025. 5
- [52] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *International workshop on deep learning in medical image analysis*, pages 3–11. Springer, 2018. 2