
COS3D: Collaborative Open-Vocabulary 3D Segmentation

Runsong Zhu¹ Ka-Hei Hui² Zhengzhe Liu³ Qianyi Wu⁴ Weiliang Tang¹
Shi Qiu¹ Pheng-Ann Heng¹ Chi-Wing Fu¹
¹ The Chinese University of Hong Kong ² Autodesk AI Lab
³ Lingnan University ⁴ Monash University

In the supplementary material, we provide more experimental results in Sec. A, more technical details in Sec. B and discussions on potential extensions in Sec. C. For more qualitative results, please refer to our video.

A Experiments results

A.1 More visual results on LeRF and ScanNetv2

We provide additional comparisons, including the state-of-the-art baselines and two implementations of our COS3D method, on LeRF [1] dataset and ScanNetv2 [2] dataset, as shown in Fig.1, Fig.2, Fig.3, Fig.4, and Fig. 5. In particular, in addition to common category queries, we test physical properties and affordances as queries (*i.e.*, “fleece” and “drinkable” in Fig. 1, and “sleep” and “music” in Fig. 4). Moreover, for the LeRF dataset, we perform comparisons with multiple views in Fig. 3 and the supplementary video to verify the view consistency of our methods. All results demonstrate that our method consistently and significantly outperforms existing approaches, providing accurate text-aligned segmentation with fewer artifacts.

A.2 Comparisons with InstanceGaussian [3] and Dr. Splat [4] on ScanNetv2.

Considering the inconsistent evaluation protocols and input (*i.e.*, segmentation mask) in these baselines, we ran the official codes of InstanceGaussian [3] and Dr. Splat [4] on ScanNetv2 using the same input and evaluation protocol as ours, following OpenGaussian, to ensure fair comparisons. The results presented in Tab. 1 demonstrate that our methods outperforms Instance Gaussian and Dr.Splat in all metrics.

Table 1: Comparison with the latest baselines on ScanNetv2. Our * are sourced from Table 2 in our main paper.

Methods	19 classes		15 classes		10 classes	
	mIoU \uparrow	mAcc. \uparrow	mIoU \uparrow	mAcc. \uparrow	mIoU \uparrow	mAcc. \uparrow
InstanceGaussian [3]	30.21	47.88	33.77	51.63	42.32	59.78
Dr. Splat [4]	23.35	40.17	30.06	49.13	38.44	55.33
Ours (*)	32.47	49.05	35.95	54.35	44.32	63.66

A.3 Analysis between kernel regression and the MLP counterpart.

Both kernel regression and the MLP counterpart are means to achieve our instance-to-language feature mapping function, and the improved performance achieved by these two strategies consistently demonstrates the effectiveness of our feature mapping design. Moreover, we provide a discussion from the perspectives of accuracy and memory efficiency:

Algorithm 1 The adaptive language-to-instance prompt refinement with non-redundant access

Input: $\{g_i\}$ - Gaussian point, $\{o_i\}$ - opacity of Gaussian point, $\{R_i\}$ - relevance of Gaussian point, $\{\mathbf{I}_i\}$ - instance feature, d_I - calculated mean distance of instance features, τ - relevance threshold.

Output: Accurate Gaussians point set \mathcal{S}_t corresponding to the query text.

```
1:  $\{\mathcal{T}\} = [0.8 \cdot d_I, 0.6 \cdot d_I, 0.4 \cdot d_I, 0.2 \cdot d_I]$   $\triangleright$  Threshold list: balancing completeness and precision
2:  $\mathcal{S}_t = \text{empty}()$   $\triangleright$  Initialize target set
3:  $\mathcal{S} = \text{high-relevance-point}(\{R_i\}, \{g_i\}, \tau)$   $\triangleright$  Extracting high relevance Gaussian points as prompt
4:  $\mathcal{S} = \text{sorted}(\mathcal{S})$   $\triangleright$  Descending order based on the relevance score
5:  $\mathcal{V} = \text{empty}()$   $\triangleright$  Visited set to avoid redundant access
6: for  $g'$  in  $\mathcal{S}$  do
7:   if  $g' \in \mathcal{V}$  then  $\triangleright$  Check if already visited
8:     continue  $\triangleright$  Skip to the next point
9:   end if
10:  for  $\mathcal{T}$  in  $\{\mathcal{T}\}$  do
11:     $\mathcal{S}_{g'} = \text{neighboring}(\mathbf{I}_{g'}, \{\mathbf{I}_i\}, \mathcal{T})$   $\triangleright$  Expansion region
12:     $R_{\mathcal{S}_{g'}} = \frac{\sum_{w \in \mathcal{S}_{g'}} o_w \cdot R_w}{\sum_{w \in \mathcal{S}_{g'}} o_w}$   $\triangleright$  Region-level relevance
13:    if  $R_{\mathcal{S}_{g'}} \geq \tau$  then
14:       $\mathcal{S}_t = \mathcal{S}_t \cup \mathcal{S}_{g'}$ 
15:      Break  $\triangleright$  Stop at the first valid threshold
16:    end if
17:  end for
18:   $\mathcal{V} = \mathcal{V} \cup \mathcal{S}_{g'}$   $\triangleright$  Mark all neighboring points as visited
19: end for
20: return  $\mathcal{S}_t$   $\triangleright$  Return the final target set
```

1) Accuracy: We observe that the kernel regression version tends to achieve better performance than the MLP counterpart in our experiments. We attribute this to the selection of discriminative instance features as the source domain, which makes the mapping learning process inherently an easy regression task, where the traditional kernel regression method is extremely suitable. Moreover, kernel regression benefits from its training-free property, bypassing the unstable optimization process required by the MLP counterpart.

2) Memory efficiency: On the other hand, the MLP counterpart, consisting of small layers, offers better GPU memory efficiency. Specifically, the computational complexity of kernel regression depends on the size of the training data, resulting in higher memory requirements.

To illustrate this, we report the accuracy and GPU memory usage of kernel regression and MLP using an identical input batch size of 100K, as shown in Tab. 2

Table 2: Trade-off between accuracy and memory efficiency on LeRF.

Type	mIoU	mAcc	GPU Memory
MLP	49.75	70.60	819M
Kernel regression	50.76	72.08	6863M

A.4 Real-world robotic application

We verify the applicability of COS3D in the real-world robotic application. Inspired by recent progress in robotics [5–9], for a robotic command (*i.e.*, “Grasp the headphone and place it into the box”), we use a large vision-LLM model, *i.e.*, Qwen2-vl [10] to decompose the task and output the text of necessary objects or parts (*i.e.*, “box” and “headband”) that need to be located. Next, our COS3D is applied to infer the 3D Gaussian segmentation results according to the query text, which are further used to determine the grasping [11] and manipulation parameters [8, 9]. For the robotic application results, please refer to our video.

A.5 Hyperparameter analysis

Kernel regression. For kernel regression, the bandwidth hyperparameter (σ) controls the smoothness of the function. We set this hyperparameter to 0.1 across all experiments, and it performs well without requiring additional tuning. Additionally, we conducted an ablation study on σ , with the results presented in Tab. 3. Overall, the results remain stable under moderate changes to σ , and our method consistently outperforms the latest state-of-the-art approach, InstanceGaussian [3].

Adaptive refinement. There are two hyperparameters in the adaptive refinement algorithm: the region expansion threshold \mathcal{T} and the widely used relevance threshold τ .

- **The region expansion threshold \mathcal{T} :** Instead of using a fixed value for this hyperparameter, we implement a hyperparameter generation strategy as described in Algorithm 1. In this approach, the final expansion threshold is determined based on the calculated mean segmentation feature distances. Additionally, we conducted experiments with different expansion thresholds by applying permutations (ranging from -20% to 20%) to the mean feature distances. The results, shown in Tab. 4, demonstrate that COS3D achieves stable performance within a reasonable range of region threshold values and consistently outperforms the latest SOTA method (InstanceGaussian [12]).
- **The relevance threshold τ :** This hyperparameter has been widely used and proven to be relatively robust in existing methods [13, 14, 4, 15]. Therefore, we simply set it to a fixed value of 0.5 throughout our experiments.

Table 3: Quantitative comparisons for different σ values in kernel regression.

σ	mIoU	mAcc
0.05	51.00	71.38
0.08	50.55	70.59
0.10 (default)	50.76	72.08
0.15	50.55	71.23
0.20	50.58	71.25
InstanceGaussian [4]	45.30	58.44

Table 4: Quantitative comparisons of different region expansion threshold \mathcal{T} values. We conduct the experiments under different expansion thresholds by adding a moderate permutation (from -20% to 20%) to the mean feature distances.

Perturbation	mIoU	mAcc
-20%	49.96	72.44
-10%	49.90	72.44
0.0 (default)	50.76	72.08
+10%	49.68	71.28
+20%	49.87	72.18
InstanceGaussian [3]	45.30	58.44

B Implementation details

In COS3D, the proposed collaborative field utilizes 3D-GS as the underlying 3D representation and comprises an instance field and a language field. For 3D-GS, we use the official 3D Gaussian Splatting implementation from [16]. For the instance field, each Gaussian point is augmented with an instance feature vector, with the dimension set to 16. For the language field, each Gaussian point is augmented with a language feature vector, with the dimension set to 512. Additionally, to obtain the 2D segmentation, we use SAM ViT-H [17] for segmentation at the large granularity level, following the practice in [14]. To extract CLIP features, we use OpenCLIP ViT-B/16 [18].

B.1 Training details

During training, we pre-train the 3D-GS with a default of 30K training iterations to reconstruct the 3D scene as the 3D representation, following [13, 14, 4]. Then, for our collaborative field learning, we adopt a two-stage training solution. For stage one (instance filed learning), we employ the InfoNCE loss (Eq. 2 in main paper) with a default of 30K training iterations using the Adam optimizer with a learning rate of 0.0025. For stage two (instance-to-language mapping learning), we provide two implementation strategies: 1) MLPs version and 2) a kernel regression version.

1) For the MLPs version, the network consists of six layers with hidden sizes of (32, 64, 128, 256, 256) to map the input instance feature vector of 16 dimensions to a language feature vector of 512 dimensions. We train the MLPs using the L1 loss function, with the loss weight set to 1 and a default of 30K training iterations. Additionally, we use the Adam optimizer with a learning rate of 1e-3.

2) For the Kernel regression version, we adopt the widely-used Nadaraya-Watson estimator [19], allowing us to directly construct the mapping functions without requiring training. Note that the standard Nadaraya-Watson estimator [19] considers all data points for calculation, resulting in high computational costs. In practice, we accelerate the process by dynamically ignoring points with larger distances and only considering the influence of the top 25 closest points based on the input features.

B.2 Inference details

During inference, for a given text query, we first: (1) generate the 3D relevance map and (2) perform an adaptive language-to-instance prompt refinement.

1) For the 3D relevance map generation, we use the CLIP [18] text encoder to process the query text and extract the corresponding language feature \mathbf{L}_{text} . Then, we compute the 3D point-level relevance R as: $R = \min_i \frac{\exp(\mathbf{L} \cdot \mathbf{L}_{\text{text}})}{\exp(\mathbf{L} \cdot \mathbf{L}_{\text{text}}) + \exp(\mathbf{L} \cdot \mathbf{L}_{\text{canon}}^i)}$, where $\mathbf{L}_{\text{canon}}$ is the CLIP embedding of a predefined canonical phrase [1] chosen from “object”, “things”, “stuff”, and “texture”.

2) For the adaptive language-to-instance prompt refinement, we provide detailed pseudo-code in Alg. 1 to better illustrate the process. As introduced in the main paper, our adaptive language-to-instance prompt refinement includes two key steps, *i.e.*, region expansion and adaptive filtering. Note that, directly expanding all prompt points requires computation and storage of a distance matrix of size $N' \times N$, where N' is the number of prompt points in \mathcal{S} , and N is the total number of Gaussian points in the scene. However, since $N' \times N$ can be extremely large (e.g., exceeding 1 billion), this approach leads to excessive computations and can cause inefficiencies or out-of-memory (OOM) issues. To address this, we optimize the process by reducing redundant computations. Specifically, we process regions in descending order of their center points’ relevance scores, ensuring that any prompt points already covered in previously calculated regions are excluded from further neighborhood expansions.

Hyperparameter generation: To automatically determine the hyperparameter \mathcal{T} for region expansion, we first compute the average feature distance d_I of the rendered instance features belonging to the same instance ID, based on the SAM mask within each view. To ensure that the expansion region is semantically and spatially coherent, the threshold \mathcal{T} should be smaller than the average distance d_I . Intuitively, \mathcal{T} can be determined by scaling d_I with a factor between 0 to 1. A larger factor (e.g., 0.8) results in broader expansion with complete segmentation but may also introduce undesired targets. In contrast, a smaller factor (e.g., 0.2) leads to more precise expansion but may result in incomplete segmentation. To address this trade-off, we generate a candidate threshold list: $[0.8 \cdot d_I, 0.6 \cdot d_I, 0.4 \cdot d_I, 0.2 \cdot d_I]$. Then, we iteratively test the candidate thresholds in descending order and select the largest threshold that satisfies the region-level relevance condition, as illustrated in Alg. 1.

B.3 Evaluation

We conduct the main experiments on two standard benchmarks, *i.e.*, the LeRF [1] dataset and the ScanNetv2 [2] dataset, to verify the effectiveness of our method.

For the LeRF [1] dataset, we follow the official 3D-aware evaluation protocol established by OpenGaussian [14] to report the metrics of mIoU and mAcc for comparison with baselines. Specifically,

we conduct experiments on its four scenes (*i.e.*, “figures”, “ramen”, “teatime”, and “waldo_kitchen”) with pairs of query text and 2D segmentation annotations released by Langsplat [13].

For the ScanNetv2 dataset, we follow the well-established evaluation protocol from OpenGaussian [14] to report the metrics of mIoU and mAcc for comparison. Specifically, we use the same 10 scenes (*i.e.*, scene0000_00, scene0062_00, scene0070_00, scene0097_00, scene0140_00, scene0200_00, scene0347_00, scene0400_00, scene0590_00, scene0645_00) as OpenGaussian [14] to conduct experiments. Moreover, the 19/15/10 categories defined by ScanNet are used for text query: 1) The 19 categories include: wall, floor, cabinet, bed, chair, sofa, table, door, window, bookshelf, picture, counter, desk, curtain, refrigerator, shower curtain, toilet, sink, and bathtub. 2) The 15 categories exclude: picture, refrigerator, shower curtain, and bathtub. 3) The 10 categories further exclude: cabinet, counter, desk, curtain, and sink.

C Future work

Potential extension to relational or multi-object queries. We plan to explore relational reasoning or multi-object queries in future work. Firstly, COS3D already provides a solid foundation. For instance, the recent study [20] demonstrates that reasoning 3D segmentation (e.g., querying “the red chair next to the table”) can be achieved to some extent by employing an LLM as an agent in combination with an OV3DS approach as a tool. Furthermore, incorporating reasoning abilities using advanced LLM models is a promising direction. Specifically, it is technically feasible to replace the CLIP [18] feature with the vision encoder feature from a V-LLM model, such as Qwen-vl2.5 [21]. However, how to effectively integrate the learned 3D Qwen feature fields with the LLM backbone to handle the reasoning queries and enhance segmentation quality remains an open question, as the relevance between the 3D Qwen feature fields and the text queries cannot be directly modeled using feature distance (e.g., cosine similarity) as in CLIP-style models (e.g., CLIP [18], SigLIP [22]). One possible solution is to render the 3D Qwen feature field into a 2D feature map based on a reference view, allowing it to be fed into the LLM backbone along with reasoning queries. It is an interesting research problem to design comprehension solutions that enable interaction between 3D LLM-style feature fields and reasoning queries.

Potential extension to online setting. Following recent approaches [13, 14, 4, 15], Our COS3D adopts the offline setting. It is an interesting direction to incorporate an online setting. To do so, one possible solution is to integrate our COS3D with existing online Gaussian reconstruction methods [23–26] to update our collaborative fields incrementally. We acknowledge that efficient online OV3DS requires additional research efforts, and we leave this for future work.

Potential extension to other 3D representations. In this paper, our COS3D adopts 3DGS as the advanced 3D representation. We believe our methods have the potential to generalize to other 3D representations, such as point clouds. For instance, recent advancements (e.g., VGGT [27]) can connect the original 2D image with 3D representations (e.g., point (cloud) maps), enabling the use of 2D foundation models to provide open-vocabulary segmentation for 3D point clouds. We leave further explorations in future work.

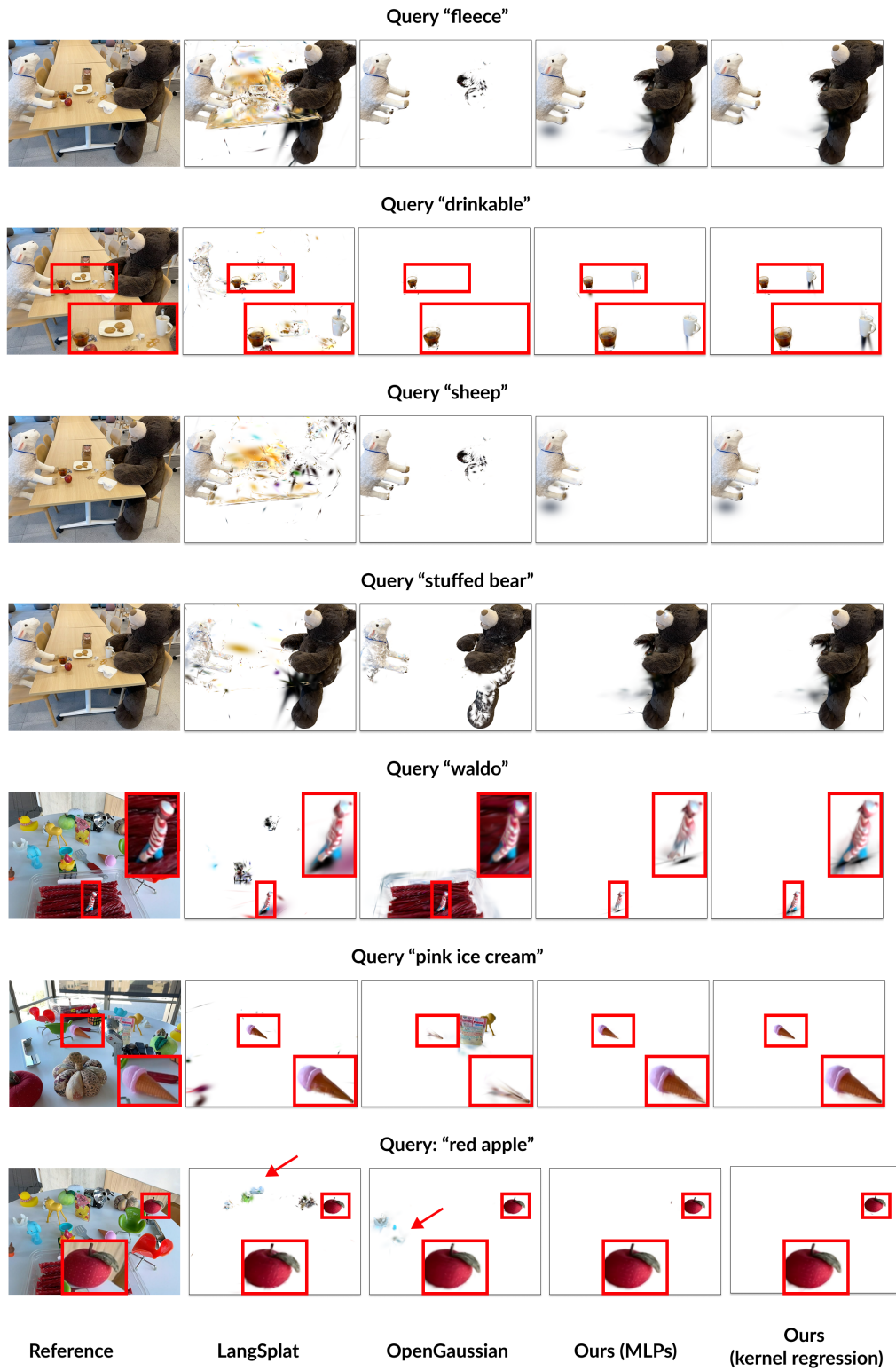


Figure 1: Open-vocabulary 3D Gaussian segmentation on the LeRF [1] dataset. In addition to common category text, we test physical properties and affordances as queries, *i.e.*, "fleece" and "drinkable".

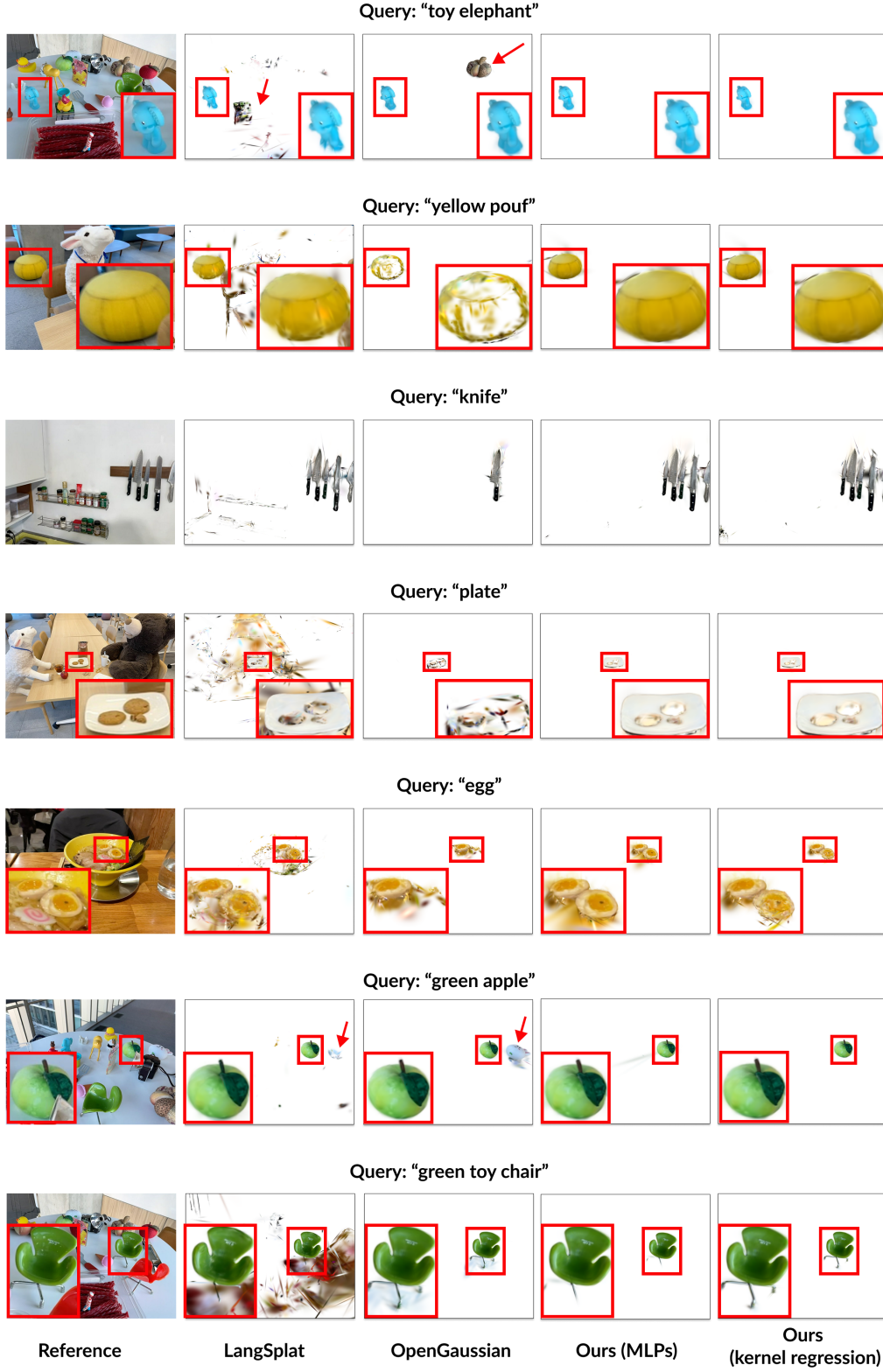


Figure 2: Open-vocabulary 3D Gaussian segmentation on the LeRF [1] dataset. Our method outperforms previous open-sourced SOTA methods (*i.e.*, LangSplat, OpenGaussian) in accurately identifying the 3D objects corresponding to text queries with fewer artifacts.

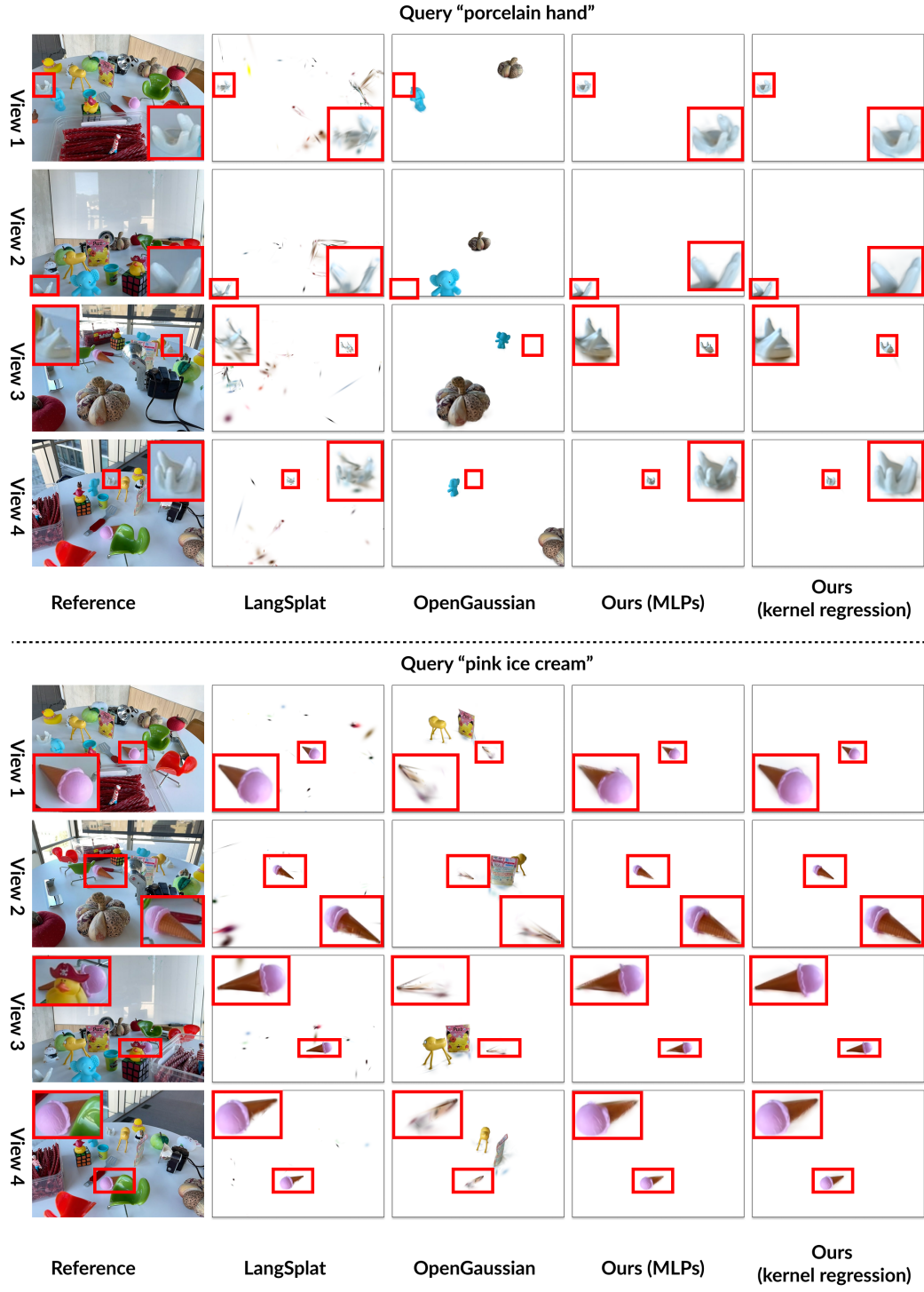


Figure 3: Multi-view visual comparisons on the LeRF [1] dataset.

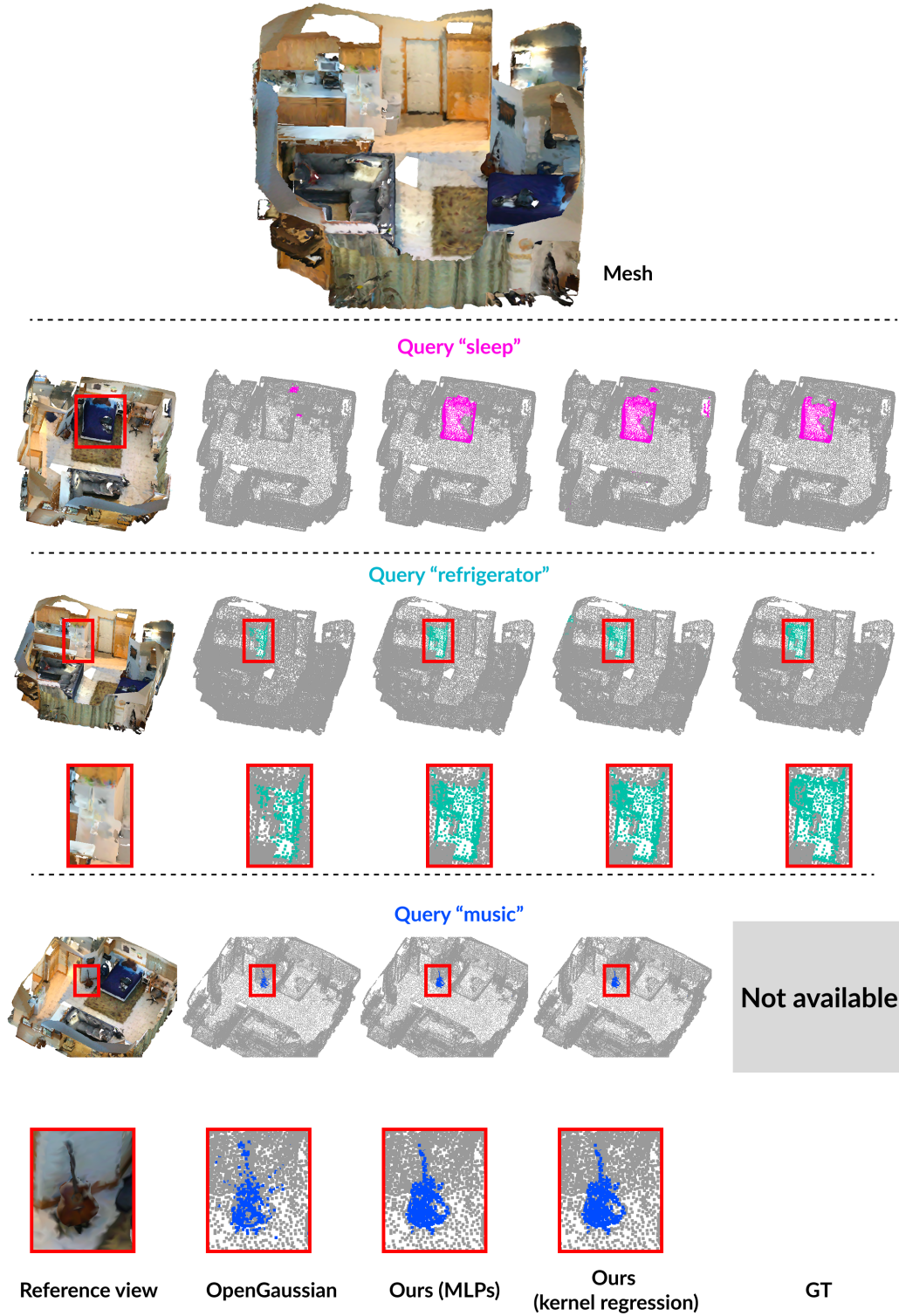


Figure 4: Open-vocabulary Gaussian segmentation on ScanNetv2 [2] dataset. Here, we test affordances as queries, *i.e.*, “sleep” and “music”. Both implementations (*i.e.*, kernel regression and MLPs) of our method outperform the previous open-sourced SOTA approach (*i.e.*, OpenGaussian) in accurately identifying 3D objects for various text queries.

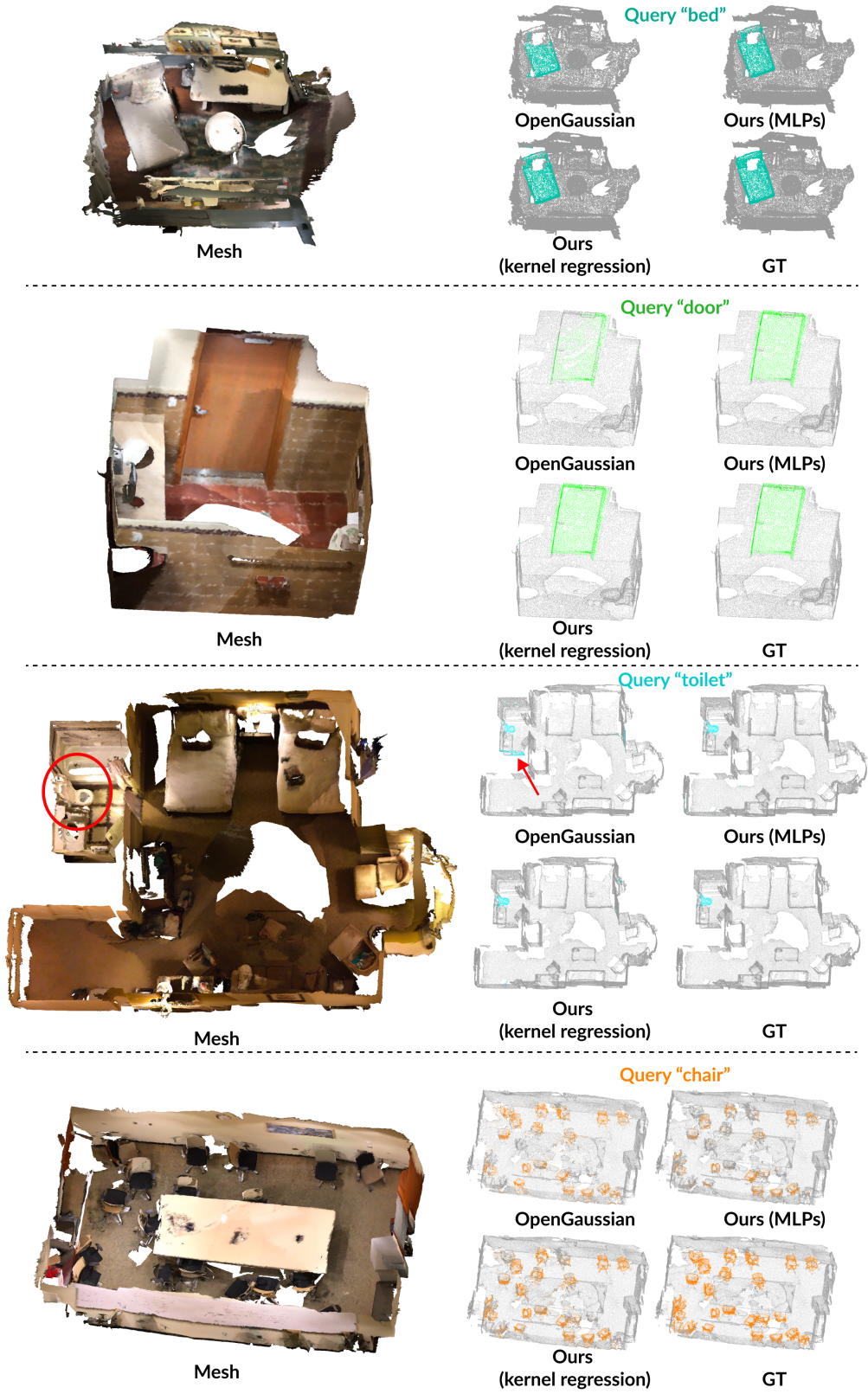


Figure 5: Open-vocabulary Gaussian segmentation on ScanNetv2 [2] dataset. Both implementations (*i.e.*, kernel regression and MLPs) of our method outperform the previous open-sourced SOTA approach (*i.e.*, OpenGaussian) in accurately identifying 3D objects for various text queries.

References

- [1] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, “LERF: Language embedded radiance fields,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19729–19739, 2023.
- [2] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “ScanNet: Richly-annotated 3D reconstructions of indoor scenes,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839, 2017.
- [3] H. Li, Y. Wu, J. Meng, Q. Gao, Z. Zhang, R. Wang, and J. Zhang, “Instancegaussian: Appearance-semantic joint gaussian representation for 3d instance-level perception,” *arXiv preprint arXiv:2411.19235*, 2024.
- [4] K. Jun-Seong, G. Kim, K. Yu-Ji, Y.-C. F. Wang, J. Choe, and T.-H. Oh, “Dr. splat: Directly referring 3d gaussian splatting via direct language embedding registration,” *arXiv preprint arXiv:2502.16652*, 2025.
- [5] A. Rashid, S. Sharma, C. M. Kim, J. Kerr, L. Y. Chen, A. Kanazawa, and K. Goldberg, “Language embedded radiance fields for zero-shot task-oriented grasping,” in *7th Annual Conference on Robot Learning*, 2023.
- [6] Y. Zheng, X. Chen, Y. Zheng, S. Gu, R. Yang, B. Jin, P. Li, C. Zhong, Z. Wang, L. Liu, *et al.*, “Gaussiangrasper: 3d language gaussian splatting for open-vocabulary robotic grasping,” *IEEE Robotics and Automation Letters*, 2024.
- [7] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, “Voxposer: Composable 3d value maps for robotic manipulation with language models,” *arXiv preprint arXiv:2307.05973*, 2023.
- [8] W. Tang, J.-H. Pan, Y.-H. Liu, M. Tomizuka, L. E. Li, C.-W. Fu, and M. Ding, “Geomanip: Geometric constraints as general interfaces for robot manipulation,” *arXiv preprint arXiv:2501.09783*, 2025.
- [9] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, “Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation,” *arXiv preprint arXiv:2409.01652*, 2024.
- [10] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, *et al.*, “Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution,” *arXiv preprint arXiv:2409.12191*, 2024.
- [11] A. Mousavian, C. Eppner, and D. Fox, “6-dof graspnet: Variational grasp generation for object manipulation,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2901–2910, 2019.
- [12] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *ACM transactions on graphics (TOG)*, vol. 41, no. 4, pp. 1–15, 2022.
- [13] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, “LangSplat: 3D language Gaussian splatting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20051–20060, 2024.
- [14] Y. Wu, J. Meng, H. Li, C. Wu, Y. Shi, X. Cheng, C. Zhao, H. Feng, E. Ding, J. Wang, *et al.*, “Opegaussian: Towards point-level 3d gaussian-based open vocabulary understanding,” *arXiv preprint arXiv:2406.02058*, 2024.
- [15] J.-C. Shi, M. Wang, H.-B. Duan, and S.-H. Guan, “Language embedded 3D Gaussians for open-vocabulary scene understanding,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5333–5343, June 2024.
- [16] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian splatting for real-time radiance field rendering,” *ACM Transactions on Graphics*, vol. 42, no. 4, 2023.
- [17] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollar, and R. Girshick, “Segment anything,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, October 2023.
- [18] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.
- [19] E. A. Nadaraya, “On estimating regression,” *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

- [20] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, *et al.*, “Grounding dino: Marrying dino with grounded pre-training for open-set object detection,” in *European Conference on Computer Vision*, pp. 38–55, Springer, 2024.
- [21] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, *et al.*, “Qwen2.5-vl technical report,” *arXiv preprint arXiv:2502.13923*, 2025.
- [22] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, “Sigmoid loss for language image pre-training,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.
- [23] C. Cheng, S. Yu, Z. Wang, Y. Zhou, and H. Wang, “Outdoor monocular slam with global scale-consistent 3d gaussian pointmaps,” *arXiv preprint arXiv:2507.03737*, 2025.
- [24] S. Yu, C. Cheng, Y. Zhou, X. Yang, and H. Wang, “Rgb-only gaussian splatting slam for unbounded outdoor scenes,” *arXiv preprint arXiv:2502.15633*, 2025.
- [25] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, “Gaussian-slam: Photo-realistic dense slam with gaussian splatting,” *arXiv preprint arXiv:2312.10070*, 2023.
- [26] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, “Gaussian splatting slam,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18039–18048, 2024.
- [27] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, “Vggt: Visual geometry grounded transformer,” in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5294–5306, 2025.