
Supplementary Materials:

Understanding while Exploring:

Semantics-driven Active Mapping

Anonymous Author(s)

Affiliation

Address

email

1 In this supplementary document, we provide a detailed outline structured as follows: Section S.1
2 offers additional implementation details of ActiveSGM. Section S.2 presents comprehensive ablation
3 studies on the key components influencing semantic segmentation performance. Section S.3 includes
4 extended quantitative and qualitative results, along with a runtime analysis. Section S.4 provides
5 justifications for the checklist items.

6 S.1 Implementation Details

7 **Hardware and Software.** We conducted the experiments on a server with 2 NVIDIA RTX A6000
8 GPUs and an Intel i9-10900X CPU with 20 cores. Our ActiveSGM is implemented with python
9 3.8 and CUDA 11.7. Please refer to Section S.4.5 for more information about baselines and other
10 packages we used. Our code will be released upon acceptance.

11 **OneFormer Finetuning Details.** Following the instructions from ActiveGAMER [3], we imple-
12 mented the geometry-based exploration criterion to construct our fine-tuning dataset. Beginning from
13 a random position, the agent performs 500 exploration steps, collecting 500 RGB-Semantic frame
14 pairs per scene. We then fine-tuned OneFormer [4] separately on the collected data from Replica and
15 MP3D, training for 3,000 steps per scene. The Replica dataset has 101 classes, while MP3D has 40.
16 The fine-tuning process follows the official OneFormer tutorial provided by Hugging Face (https://huggingface.co/docs/transformers/main/en/model_doc/oneformer). The novel trajec-
17 tories described in Table 1 of the main paper are used as the test set. These trajectories are distinct
18 from those used for fine-tuning. The train/test Top-1 accuracy is reported in Table S.1.

20 **Sparse Rendering.** We illustrate the semantic rendering process using our proposed sparse semantic
21 representation (with fewer classes) in Fig. S.1. The overall rendering process proceeds as follows:

```
22     For each tile:  
23         For each pixel in the tile:  
24             For each batch of Gaussians in the frustum:  
25                 Load batch to shared memory # fewer classes decreases loading time  
26                 For each Gaussian in the batch:  
27                     If pixel is affected:  
28                         Compute contribution (semantic, alpha)  
29                         Composite with alpha blending # sparse mode has fewer iters.  
30                         Early exit if opacity is sufficient  
31     Write final semantic
```

32 If our sparse representation is not used, each Gaussian stores a full probability distribution over all
33 classes, and alpha blending of semantic probabilities is performed by iterating over all classes:

```
34 For each Gaussian G_i in the batch:  
35     for idx in range(num_classes+1):  
36         P[idx] += prob[idx] * alpha[idx] * transmittance[idx]
```

where P is the rendered probability distribution of each pixel. This becomes increasingly inefficient when the number of classes is large and many probabilities are near zero. For instance, in the Replica dataset with 101 classes plus one unknown class, this results in 102 iterations per Gaussian.

In contrast, our sparse rendering strategy stores only the Top- k most probable classes per Gaussian ($k \ll \text{number of classes}$). During rasterization, alpha blending is performed only over these sparse indices:

```

42 For each Gaussian  $G_i$  in the batch:
43     indices = topk_indices in  $G_i$ 
44     for idx in indices:
45          $P[\text{idx}] += \text{prob}[\text{idx}] * \alpha[\text{idx}] * \text{transmittance}[\text{idx}]$ 

```

This reduces the number of memory accesses and blending operations without sacrificing semantic fidelity. By reducing the number of stored logits and accessed channels, our sparse representation speeds up both the memory workload, as more Gaussians can be loaded into shared memory, and the Gaussian processing loop, leading to faster semantic rendering. Please refer to Section S.3.2 for a quantitative runtime comparison.

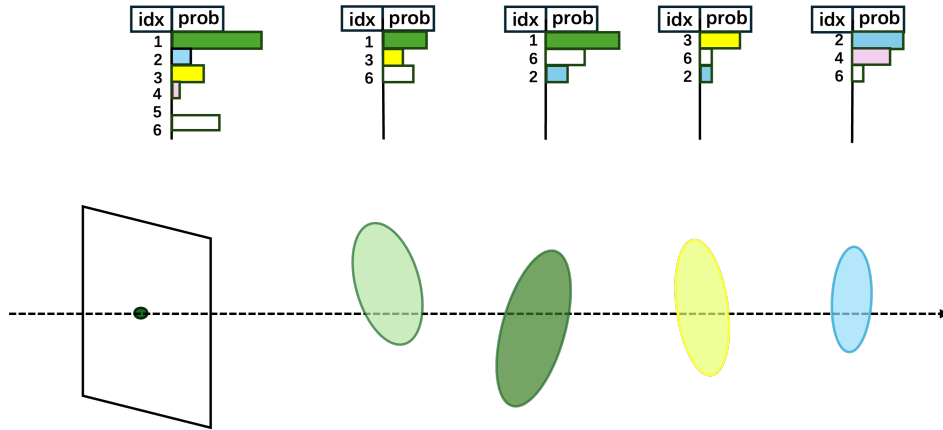


Figure S.1: **Visualization of Rendering Semantic Map with Sparse Semantic Vector.** Each Gaussian only stores indexes and probabilities of the top- k most probable categories, the semantic distribution of the given pixel is rendered following Eqn. (3) in the main paper.

Local Path Planner. We employed the Efficient Rapid-exploration Random Tree (RRT) proposed by NARUTO [5] for local path planning. Once the goal location is determined, we use an efficient RRT-based planner to find a path from the current state s_t to the goal s_g , using the Exploration Map to measure collision and reachability. (Specifically, the agent should only move within the free voxels defined by the Exploration Map. Additionally, we enforce a collision buffer of 20 cm, ensuring the agent avoids regions that are too close to surrounding surfaces.) To speed up planning in large-scale 3D environments, we enhance standard RRT by also attempting direct connections between samples and the goal. This greatly improves efficiency.

S.2 Ablation Studies.

We present the full ablation studies in Table S.2, consistent with the discussion in Section 4.3 of the main paper. We also compare KL divergence and Hellinger distance, finding that the Hellinger

Table S.1: **OneFormer Fintuneing Accuracy**

Dataset	Splits	Avg.	Of0	Of1	Of2	Of3	Of4	R0	R1	R2
Replica [1]	Train	97.31	98.75	98.67	99.17	97.35	96.45	98.13	98.83	91.15
	Test	89.12	89.07	71.84	92.76	93.18	91.54	87.37	92.25	94.96
MP3D [2]			GdvGF	gZ6f7	HxpKQ	pLe4w	YmJkq			
	Train	93.87	94.37	94.99	93.84	95.22	90.94			
	Test	89.77	93.68	92.58	91.21	89.52	81.86			

Table S.2: **Ablation of Semantic Components.** Experiments on *office0* and *room0* from Replica to evaluate the impact of the number of retained categories (*Top-k*) and the use of Hellinger distance (*H.D.*), KL-Divergence (*KL*) and cosine similarity (*Cos.*) in the semantic loss.

Top-k	H.D.	KL	Cos.	Avg. mIoU (%) ↑	Top-1 Acc (%) ↑	Top-3 Acc (%) ↑	mAP (%) ↑	F-1 (%) ↑
Top-5	✓		✓	83.06	95.66	99.68	94.79	74.24
Top-8	✓		✓	83.34	95.70	99.66	95.05	74.23
Top-16	✓		✓	84.08	95.68	99.73	94.92	74.73
Top-16	✓			82.26	95.57	99.61	94.21	74.10
Top-16			✓	82.70	95.62	99.73	94.40	72.43
Top-16		✓	✓	82.22	95.63	99.70	94.66	73.93
Top-16	✓		✓	84.08	95.68	99.73	94.92	74.73

Table S.3: **Semantic Segmentation on ReplicaSLAM.** Rendered semantics are evaluated on 4 scenes using the SGS-SLAM [6] protocol, which compares predictions to ground-truth categories visible per view. Our method uses semantic predictions from OneFormer and is evaluated on novel views.

Methods	Semantic	View	Avg. Steps ↓	Avg. mIoU (%) ↑	RO (%)	R1 (%)	R2 (%)	OfO (%)
NIDS-SLAM [7]	GT	Train	2000	82.37	82.45	84.08	76.99	85.94
DNS-SLAM [8]	GT	Train	2000	84.77	88.32	84.90	81.20	84.66
SNI-SLAM [9]	GT	Train	2000	87.41	88.42	87.43	86.16	87.63
SGS-SLAM [6]	GT	Train	2000	92.72	92.95	92.91	92.10	92.90
OneFormer [4]	GT	Novel	3000	65.41	69.06	65.71	67.01	59.85
Ours	Pred.	Novel	713	85.13	84.54	85.98	85.40	84.60

distance is a more effective choice. Notably, KL divergence can lead to gradient vanishing due to the instability of the logarithmic function, necessitating gradient norm clipping during training.

S.3 Additional Results

In this section, we present additional qualitative and quantitative results on both datasets.

S.3.1 Quantitative Results

Semantic Segmentation on ReplicaSLAM We evaluate on 4 scenes following the SGS-SLAM protocol [6], which compares rendered semantic masks to ground-truth labels visible in each view. The full results are shown in the Table S.3, and have been summarized in Table 1 yellow .

Semantic Segmentation on Replica (Novel View) To assess generalization, we generate new trajectories near the SLAM trajectories, following the instructions of SplatAM [10]. We present the complete results in Table S.4, as a supplement to Table 1 blue in the main paper.

Semantic Segmentation on MP3D We also evaluate the average IoU on five large indoor scenes from MP3D (see Table 1 red in the main paper). Table S.5 reports the IoU scores for six common categories, as well as the mean IoU across all 40 categories of our method (denoted as 'mpcat40'). The semantic ground-truth meshes provided by MP3D are noisier than the texture meshes, often containing floaters and missing regions. To ensure a fair comparison, we computed the L1 distance from each point in the semantic mesh to its nearest neighbor in the texture mesh, and filtered out all points with distances greater than 5 cm. Points in the texture meshes inherit the semantic label of the nearest neighboring point in the semantic mesh, if it is within 5 cm, otherwise their labels are set to *unknown*, and then they are used as ground truth in the evaluation. We show an example of the filtered mesh in Figure S.2.

3D Reconstruction and Novel View Synthesis. We evaluate the 3D reconstruction and novel view synthesis (NVS) performance of ActiveSGM on MP3D and Replica. The 3D reconstruction results are reported in Table S.6, while the NVS results are presented in Table S.7. Please refer to Section 4.2 for details on how the novel trajectories are generated. Overall, ActiveSGM achieves the best 3D reconstruction and NVS performance on MP3D and performs on par with the state-of-the-art method ActiveGAMER on Replica.

Table S.4: **Semantic Segmentation on Replica (Novel Views)**. We compare three settings: (1) SGS-SLAM retrained using OneFormer predictions, instead of ground-truth labels as used in Table 1 of the main paper—leads to a noticeable drop in performance; (2) Our method without active exploration, which demonstrates the advantage of the sparse semantic representation alone; (3) Our full pipeline with active exploration, which achieves better segmentation performance with fewer steps.

Methods	Metrics	Avg.	Of0	Of1	Of2	Of3	Of4	R0	R1	R2
OneFormer [4]	Steps ↓	-	-	-	-	-	-	-	-	-
	mIoU (%) ↑	66.05	62.73	55.67	66.38	70.03	69.81	62.16	74.19	67.43
	mAP (%) ↑	84.59	83.29	72.47	87.39	88.36	85.83	81.56	90.68	87.12
	F-1 (%) ↑	57.96	57.78	40.45	59.51	66.33	47.89	59.20	69.70	62.81
	Top-1 Acc (%) ↑	89.12	89.07	71.84	92.76	93.18	91.54	87.37	92.25	94.96
	Top-3 Acc (%) ↑	96.18	96.76	89.10	96.53	97.70	97.29	96.40	96.92	98.76
SGS-SLAM [6]	Steps ↓	2000	2000	2000	2000	2000	2000	2000	2000	2000
	mIoU (%) ↑	80.42	77.60	75.68	78.70	78.10	89.96	83.23	83.97	76.12
	mAP (%) ↑	89.94	86.37	84.67	88.63	90.98	96.22	92.10	93.80	86.73
	F-1 (%) ↑	18.70	18.35	15.06	19.03	17.68	18.02	25.28	18.47	17.69
	Top-1 Acc (%) ↑	94.42	92.68	90.06	93.52	93.42	98.14	97.16	96.71	93.64
	Top-3 Acc (%) ↑	95.53	93.39	90.90	94.54	96.64	98.70	98.00	97.35	94.68
Ours (Passive)	Steps ↓	2000	2000	2000	2000	2000	2000	2000	2000	2000
	mIoU (%) ↑	80.14	74.15	74.88	76.97	79.60	88.29	84.50	84.50	78.23
	mAP (%) ↑	90.09	88.91	84.86	86.27	89.12	94.86	93.78	93.78	89.13
	F-1 (%) ↑	67.81	64.54	53.49	72.42	64.26	66.48	70.18	80.09	71.03
	Top-1 Acc (%) ↑	94.05	89.80	89.69	94.99	93.83	97.60	95.65	95.65	95.16
	Top-3 Acc (%) ↑	96.82	95.00	91.71	96.58	98.71	99.45	98.14	98.14	96.85
Ours (Active)	Steps ↓	777	664	501	749	1175	941	1082	514	591
	mIoU (%) ↑	84.89	82.58	83.99	83.57	83.40	89.36	84.08	85.28	86.83
	mAP (%) ↑	94.39	94.66	91.93	92.86	93.65	96.35	95.19	94.93	95.55
	F-1 (%) ↑	77.56	73.81	72.53	79.57	75.95	76.80	75.65	83.85	82.33
	Top-1 Acc (%) ↑	96.62	94.55	96.07	98.39	94.82	97.75	96.80	96.18	98.40
	Top-3 Acc (%) ↑	99.52	99.76	99.01	99.77	99.51	99.58	99.69	99.05	99.81

Table S.5: **Semantic Segmentation on MP3D**.

Methods	Semantic	View	Avg. ↑	ceiling	appliances	sink	plant	counter	table	mpcat40
SSMI [11]	GT	Train	36.14	46.02	41.01	25.13	39.30	36.12	29.25	-
TARE [12]	GT	Train	31.70	42.01	36.86	23.86	32.51	31.70	23.27	-
Zhang et al. [13]	GT	Train	42.92	50.73	45.26	43.91	40.42	39.18	37.99	-
Ours	Pred.	Novel	65.58	70.31	76.95	69.36	73.60	14.03	69.89	55.77

89 S.3.2 Run Time Analysis

90 We conduct a runtime analysis using the *room0* scene from the Replica dataset to highlight the
91 efficiency of our sparse semantic representation and rendering strategy. The scene, measuring
92 $8\text{ m} \times 4.8\text{ m} \times 3\text{ m}$, is explored and mapped by *ActiveSGM* in 1082 steps over 48 minutes. During
93 the rendering of a semantic map with resolution ($340 \times 600 \times 102$), approximately 204k Gaussians are
94 involved in the rasterization process. Using a dense semantic representation—where each Gaussian



Figure S.2: **Filtered Semantic Mesh for MP3D**. We present both the original and the filtered semantic mesh from an MP3D scene. After filtering, most of the floaters—such as those highlighted in the yellow box—are successfully removed. The cleaned meshes are then used for semantic segmentation evaluation on MP3D.

Table S.6: **3D Reconstruction Results on Replica and MP3D.** Overall, our method achieves the best performance on MP3D and ranks second on Replica, delivering higher reconstruction accuracy and improved scene completeness compared to prior approaches. Notably, ours is the only method that incorporates semantic information into the exploration criterion, whereas all other baselines rely on geometry-based strategies.

Methods	Dataset	Acc. (cm) ↓	Comp. (cm) ↓	Comp. Ratio (%) ↑
NARUTO[5]	Replica	1.61	1.66	97.20
ActiveGAMER [3]	Replica	1.16	1.56	96.50
Ours	Replica	1.19	1.59	96.68
FBE [14]	MP3D	/	9.78	71.18
UPEN [15]	MP3D	/	10.60	69.06
OccAnt [16]	MP3D	/	9.40	71.72
ANM [17]	MP3D	7.80	9.11	73.15
NARUTO[5]	MP3D	6.31	3.00	90.18
ActiveGAMER [3]	MP3D	1.66	2.30	95.32
Ours	MP3D	1.56	1.77	97.35

Table S.7: **Novel View Rendering Performance on Replica and MP3D.** We report the average rendering metrics across scenes for each method. Our approach delivers consistently strong performance in terms of PSNR, SSIM, LPIPS, and L1 depth error, achieving comparable or better results than baselines, ranking as the second-best on Replica and the best on MP3D. Notably, our method is the only one that also addresses semantic segmentation.

Method	Dataset	PSNR ↑	SSIM ↑	LPIPS ↓	L1-D ↓
SplaTAM [10]	Replica	29.08	0.95	0.14	1.38
SGS-SLAM [6]	Replica	27.14	0.94	0.16	7.09
NARUTO [5]	Replica	26.01	0.89	0.41	9.54
ActiveGAMER [3]	Replica	32.02	0.97	0.11	1.12
Ours	Replica	30.61	0.96	0.14	1.36
NARUTO [5]	MP3D	20.52	0.72	0.58	7.95
ActiveGAMER [3]	MP3D	24.76	0.90	0.25	4.83
Ours	MP3D	26.15	0.92	0.26	3.76

95 carries a full 102-class probability distribution—the rendering takes 61 ms. In contrast, our sparse
 96 semantic representation significantly reduces computation, requiring only 3.1 ms to render the same
 97 map. This improvement stems from the reduced number of active channels during rendering and more
 98 importantly from the reduced amount of data transfers on the GPU, showcasing the effectiveness of
 99 our sparse approach for real-time semantic mapping.

100 S.3.3 Qualitative Results

101 We also preset the top-down view visualization of the 8 scenes from Replica in Figure S.3 and 5
 102 scenes from MP3D in Figure S.4, please zoom in to see more details.

103 S.4 Checklist

104 S.4.1 Limitations

105 Please see Section 5 in the main paper. Due to space constraints in the main paper, we summarize
 106 below some additional assumptions made in this work:

- 107 • **Perfect Localization:** Since this work focuses on active mapping, we assume the robot’s
 108 pose is known throughout the process. In real-world deployments, a separate localization or
 109 tracking module would be required.
- 110 • **Perfect Execution:** We assume the robot can perfectly follow the planned trajectory to
 111 reach the selected next-best-view. In practice, navigation errors should be considered for
 112 deployment.
- 113 • **Semantic Segmentation Model:** Our system depends on an external semantic segmentation
 114 model (OneFormer in our case) to generate semantic predictions. If a stronger model is used,

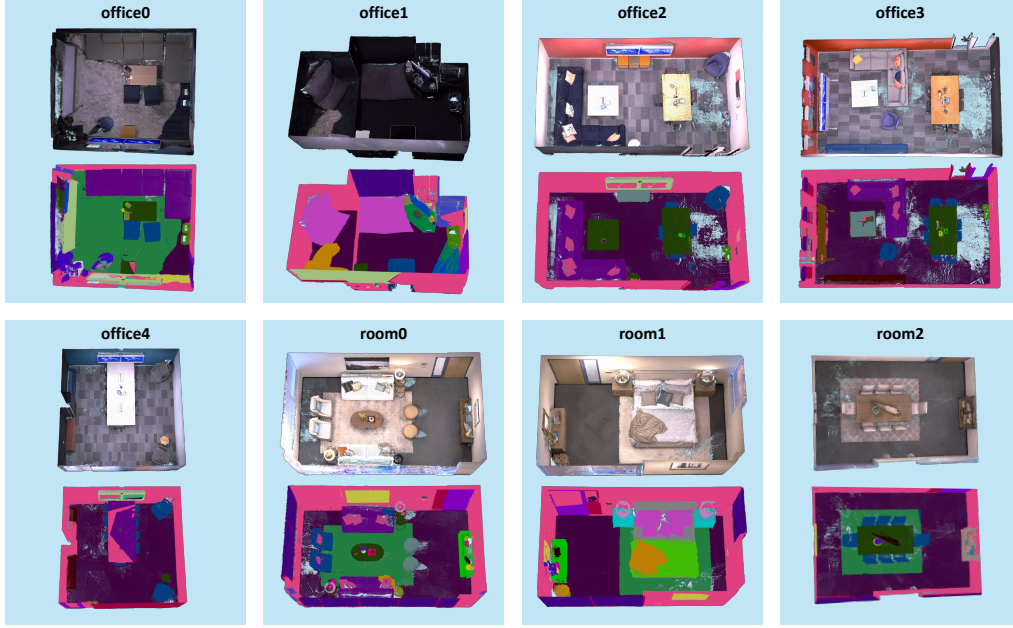


Figure S.3: **RGB and Semantic Reconstruction for Replica.**

the performance of our pipeline can improve. Conversely, if the model has limited accuracy or generalization ability, it may degrade the quality of the semantic map. Fine-tuning is recommended for adapting to new domains.

S.4.2 Experimental result reproducibility

We provide sufficient implementation details in Section S.1 in the supplement to make results reproducible. We build upon open source software, and we plan to release ours if the paper is accepted.

S.4.3 Experimental setting/details

Please refer to Section 4.1 in the main paper.

S.4.4 Experiments compute resources

Please see the *Hardware* paragraph in Section S.1 in the supplement, and also refer to Sec S.3.2 for run time analysis.

S.4.5 Licenses for existing assets

Datasets. In this paper, we conduct experiments on the following publicly available datasets. We list the URLs, license information, and citation for each dataset below.

1. Replica Dataset [1]

- URL: <https://github.com/facebookresearch/Replica-Dataset>
- License: Research or Education only. (<https://github.com/facebookresearch/Replica-Dataset/blob/main/LICENSE>)

2. Matterport3D Dataset [2]

- URL: <https://niessner.github.io/Matterport/>
- License: Non-commercial (https://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf)

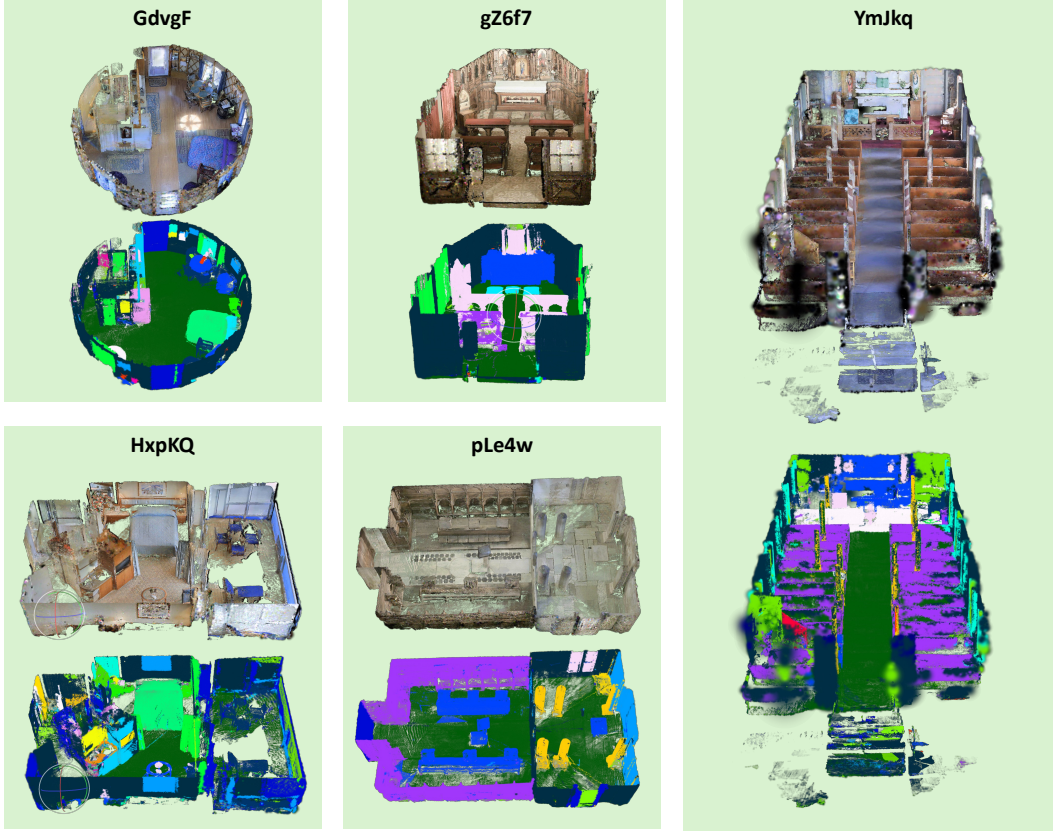


Figure S.4: **RGB and Semantic Reconstruction for MP3D.**

Software. We use Habitat-Sim as our simulation environment and develop a custom sparse rasterization CUDA toolkit based on 3D Gaussian Splatting. For mapping, we adopt SplaTAM as the backbone and fine-tune OneFormer to serve as our semantic camera. During evaluation, we also implement SGS-SLAM for comparative analysis. The source code for these components is available at:

1. **Habitat-Sim** [18]

- URL: <https://github.com/facebookresearch/habitat-sim.git>
- License: MIT

2. **3D Gaussian Splatting (3DGS)** [19]

- URL: <https://github.com/graphdeco-inria/gaussian-splatting.git>
- License: Custom (<https://github.com/graphdeco-inria/gaussian-splatting?tab=License-1-ov-file#readme>)

3. **SplaTAM** [10]

- URL: <https://github.com/spla-tam/SplaTAM.git>
- License: BSD-3-Clause

4. **SGS-SLAM** [6]

- URL: <https://github.com/ShuhongLL/SGS-SLAM.git>
- License: BSD-3-Clause

References

- [1] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, “The Replica Dataset: A Digital Replica of Indoor Spaces,” *arXiv preprint arXiv:1906.05797*, 2019.

- 159 [2] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang,
160 “Matterport3D: Learning from RGB-D Data in Indoor Environments,” in *International Conference on 3D*
161 *Vision (3DV)*, 2017.
- 162 [3] L. Chen, H. Zhan, K. Chen, X. Xu, Q. Yan, C. Cai, and Y. Xu, “ActiveGAMER: Active GAussian Mapping
163 through Efficient Rendering,” in *CVPR*, 2025.
- 164 [4] J. Jain, J. Li, M. Chiu, A. Hassani, N. Orlov, and H. Shi, “OneFormer: One Transformer to Rule Universal
165 Image Segmentation,” in *CVPR*, 2023.
- 166 [5] Z. Feng, H. Zhan, Z. Chen, Q. Yan, X. Xu, C. Cai, B. Li, Q. Zhu, and Y. Xu, “NARUTO: Neural Active
167 Reconstruction from Uncertain Target Observations,” in *CVPR*, 2024, pp. 21 572–21 583.
- 168 [6] M. Li, S. Liu, H. Zhou, G. Zhu, N. Cheng, T. Deng, and H. Wang, “SGS-SLAM: Semantic Gaussian
169 Splatting for Neural Dense SLAM,” in *ECCV*, 2024, pp. 163–179.
- 170 [7] Y. Haghighi, S. Kumar, J.-P. Thiran, and L. Van Gool, “Neural Implicit Dense Semantic SLAM,” *arXiv*
171 *preprint arXiv:2304.14560*, 2023.
- 172 [8] K. Li, M. Niemeyer, N. Navab, and F. Tombari, “DNS-SLAM: Dense Neural Semantic-Informed SLAM,”
173 in *IROS*, 2024, pp. 7839–7846.
- 174 [9] S. Zhu, G. Wang, H. Blum, J. Liu, L. Song, M. Pollefeys, and H. Wang, “SNI-SLAM: Semantic Neural
175 Implicit Slam,” in *CVPR*, 2024, pp. 21 167–21 177.
- 176 [10] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, “SplaTAM:
177 Splat Track & Map 3D Gaussians for Dense RGB-D SLAM,” in *CVPR*, 2024, pp. 21 357–21 366.
- 178 [11] A. Asgharivaskasi and N. Atanasov, “Semantic OcTree Mapping and Shannon Mutual Information Compu-
179 tation for Robot Exploration,” *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1910–1928, 2023.
- 180 [12] C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, “Representation granularity enables time-efficient
181 autonomous exploration in large, complex worlds,” *Science Robotics*, vol. 8, no. 80, 2023.
- 182 [13] R. Zhang, H. M. Bong, and G. Beltrame, “Active Semantic Mapping and Pose Graph Spectral Analysis for
183 Robot Exploration,” in *IROS*, 2024, pp. 13 787–13 794.
- 184 [14] B. Yamauchi, “A Frontier-based Approach for Autonomous Exploration,” in *Proceedings 1997 IEEE*
185 *International Symposium on Computational Intelligence in Robotics and Automation CIRA’97. Towards*
186 *New Computational Principles for Robotics and Automation*. IEEE, 1997, pp. 146–151.
- 187 [15] G. Georgakis, B. Bucher, A. Arapin, K. Schmeckpeper, N. Matni, and K. Daniilidis, “Uncertainty-driven
188 planner for exploration and navigation,” in *2022 International Conference on Robotics and Automation*
189 *(ICRA)*. IEEE, 2022, pp. 11 295–11 302.
- 190 [16] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, “Occupancy anticipation for efficient exploration and
191 navigation,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28,*
192 *2020, Proceedings, Part V 16*. Springer, 2020, pp. 400–418.
- 193 [17] Z. Yan, H. Yang, and H. Zha, “Active neural mapping,” in *Proceedings of the IEEE/CVF International*
194 *Conference on Computer Vision*, 2023, pp. 10 981–10 992.
- 195 [18] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik
196 *et al.*, “Habitat: A Platform for Embodied AI Research,” in *ICCV*, 2019, pp. 9339–9347.
- 197 [19] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian Splatting for Real-Time Radiance
198 Field Rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.