Supplementary Material for SGAD-SLAM: Splatting Gaussians at Adjusted Depth for Better Radiance Fields in RGBD SLAM

Pengchong Hu Zhizhong Han
Machine Perception Lab, Wayne State University, Detroit, USA
pchu@wayne.edu, h312h@wayne.edu

This supplementary material will include more details on the implementation and numerical results for each scene. Additionally, we also include more visual results.

1. Further Implementation Details

We implemented SGAD-SLAM in Python using the PyTorch framework, and ran all experiments on NVIDIA RTX4090 GPUs. In the mapping process, the number of mapping iterations is set to 100 for Replica [17], TUM-RGBD [18], and ScanNet [1], 500 for ScanNet++ [25]. During tracking, the downsampling ratio R is set to 10 for Replica [17] and ScanNet++ [25], 5 for TUM-RGBD [18] and ScanNet [1]. When calculating the covariance matrix c_i^j of a 3D point d_i^j from its K_c nearest neighbors, we choose $K_c = 10$ for Replica [17] and ScanNet++ [25], $K_c = 23$ for TUM-RGBD [18], and $K_c = 15$ for ScanNet [1]. To balance each term in the loss function Eq. 1 in the main paper, we set $\rho = 0.8$, $\tau = 0.2$, and $\sigma = 1.0$ for all datasets. At the beginning of mapping, we need to initialize Gaussians from the input RGBD images. The radius of each Gaussian, r, is initialized by utilizing the following formula as introduced in [8]:

$$r = \frac{D_{gt}}{F}, \tag{1}$$

where D_{gt} is the ground-truth depth, and F is the focal length. We set the learning rates as follows: $lr_{color} = 0.0025$ for color, $lr_{radius} = 0.005$ for radius, $lr_{opacity} = 0.05$ for opacity, and $lr_{offset} = 0.01$ for offset. During mapping, at least 40% of the mapping iterations will be spent on the current view. In camera tracking, we initialize camera pose for Generalized ICP (GICP) [16] by using the constant speed assumption on ScanNet [1]. Due to the high quality of RGBD images and smooth motion of the camera, we adopt the latest camera initialization strategy on Replica [17] and TUM-RGBD [18]. Note that ScanNet++ [25] is not specifically designed for SLAM tasks, and its DSLR-captured sequences include occasional sudden large motions. Therefore, following previous methods [26, 28], we utilize multi-scale RGBD odometry [13] to help the pose initialization and only employ the first 250 frames of each scene in evaluations, which present smoother trajectories.

2. More Results

In this section, to demonstrate the state-of-the-art performance of our method, we report more detailed results on each scene on Replica [17], TUM-RGBD [18], ScanNet [1], and ScanNet++ [25].

Datasets. Replica [17] is a synthetic dataset that provides high-fidelity 3D reconstructions of indoor scenes. For evaluation, we utilize the widely used RGB-D sequences from eight scenes captured by Sucar [19], which includes precise ground-truth trajectories. TUM-RGBD [18], ScanNet [1], and ScanNet++ [25] are real-world datasets, offering diverse and challenging environments for evaluations. The poses in TUM-RGBD were captured using an external motion capture system, whereas ScanNet derives its poses from BundleFusion [2]. Additionally, ScanNet++ employs laser scanning to register images and obtain accurate camera poses. Similar to previous methods [8, 9, 12, 22, 26, 28], to evaluate the superior performance of our method, we conduct experiments on eight scenes from Replica [17], three scenes from TUM-RGBD [18], six scenes from ScanNet [1], and five scenes from ScanNet++ [25] ((a) `b20a261fdf`, (b) `8b5caf3398`, (c) `fb05e13ad1`, (d) `2e74812d00`, (e) `281bc17764`).

Metrics. We evaluate both the accuracy of estimated camera poses at each frame and the rendering quality from both observed and unobserved viewpoints. To evaluate tracking performance, we employ the root mean square absolute trajectory error (ATE RMSE) [18] as a metric. To assess rendering performance, we measure PSNR, SSIM [21], and LPIPS [27]. Consistent with prior works [10, 15, 26, 28], all rendering metrics are computed by rendering full-resolution images along the estimated trajectory. Additionally, we reconstruct scene meshes using the marching cubes algorithm [11], following the approach in [15]. The reconstruction quality is assessed using the F1-score, the harmonic

mean of Precision (P) and Recall (R), with a distance threshold of 1 cm for all evaluations. Furthermore, we employ the depth L1 to evaluate the rendered mesh depth error at sampled novel views as in [29].

Numerical Results. To ensure statistical significance, all reported numerical results are averaged over five runs. For camera tracking performance, per-scene results are reported in Tab. 5 - Tab. 10 for Replica [17], TUM-RGBD [18], ScanNet [1], and ScanNet++ [25]. We present comparisons in rendering performance for each scene in Replica [17] in Tab. 1, in TUM-RGBD [18] in Tab. 3, in ScanNet [1] in Tab. 4, and ScanNet++ [25] in Tab. 11. Additionally, we report the novel view synthesis (NVS) results on ScanNet++, where the test views are distant from training views. To evaluate PSNR on each novel view, we follow previous methods [26, 28] to finetune the merged global map with $10K$ iterations and obtain the renderings in novel views. Our per scene results in Tab. 12 show that our method yields the best NVS performance.

Visual Results. Moreover, we provide more visual comparisons in rendering and reconstruction. In Replica [17], we report reconstruction visual comparisons in Fig. 3, and present each scene reconstruction comparisons in Tab. 2. Meanwhile, we show a visual comparison in scene reconstruction with camera tracking in Fig. 4 for Replica, where the error map is obtained from the reconstructed mesh for better visualization. Here we employ depth L1 and F1-score as metrics to evaluate the mesh obtained by marching cubes [11] following [15]. The comparisons show that our method can acquire more accurate reconstruction, although Point-SLAM [15] requires ground truth depth images as input to guide sampling when rendering, which is an unfair advantage over other NeRF-based methods. In TUM-RGBD [18], we provide more rendering and reconstruction results in Fig. 1 and Fig. 5 separately. Compared to the latest methods [3, 6, 8, 26, 29], our method shows superior rendering performance and reconstruction quality. In addition, we also present more rendering results and reconstruction results for ScanNet [1] in Fig. 2 and Fig. 6, and more rendering results for ScanNet++ [25] in Fig. 7. All of these visual comparisons clearly show our high fidelity rendering.

Comparison with VTGS-SLAM. Since VTGS-SLAM [6] still relies on rendering for tracking, it requires complex strategies to balance the memory limit and the number of Gaussians, such as splitting a video into sections, adding dense Gaussians merely on the first frame in a section while incrementally adding sparse Gaussians on the others, and adopting different tracking strategies in the same section. Instead, we do not need section splitting, can use extremely dense Gaussians on each frame, and run appearance mapping on multiple frames in parallel, leading to much simpler tracking, better appearance modeling, and faster geometry mapping. Tab. 7 shows VTGS-SLAM fails in tracking with dense Gaussians on each frame like us.

Performance in Structureless Environments. We report an evaluation on `fr3/nostructure_texture_far` in Tab. 8, which is a scene with minimal geometric structure. Our point-to-surface metric and rendering-based initialization contribute to our superior performance in this challenging scenario.

3. More Analysis.

Our pixel-aligned Gaussians can enhance our ability to handle large scenes and the efficiency during mapping. For each frame, we only need to splat its Gaussians with the ones associated to its N neighboring frames, rather than all Gaussians in the scene as in previous methods [14, 26, 28]. This design not only saves time on every rendering by reducing the number of Gaussians, but also enables us to speed up the mapping by splatting Gaussians in each frame in parallel in some scenarios. Additionally, unlike previous methods that require a set of keyframes to maintain the Gaussians' consistency to all previous frames, we do not need to minimize rendering errors to all keyframes, but just the latest frame as shown in Eq. 1 in the main paper.

4. Limitations

In real-world applications, the high-quality depth image are often difficult to acquire, which increases the time cost in the mapping process and degrades the rendering and tracking performance, although our movable pixel-aligned Gaussians can mitigate the effect of noisy depth images.

5. Code

Please refer to our project page for code at https://machineperceptionlab.github.io/SGAD-SLAM-Project.

6. Video

Our accompanying video provides additional visualizations, including more comprehensive visual comparisons. Please watch our video for more details.

References

[1] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017. 1, 2, 4, 6, 10, 12

[2] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics*, 2017. 1

[3] Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu. Rgbd gs-icp slam, 2024. 2, 11, 12

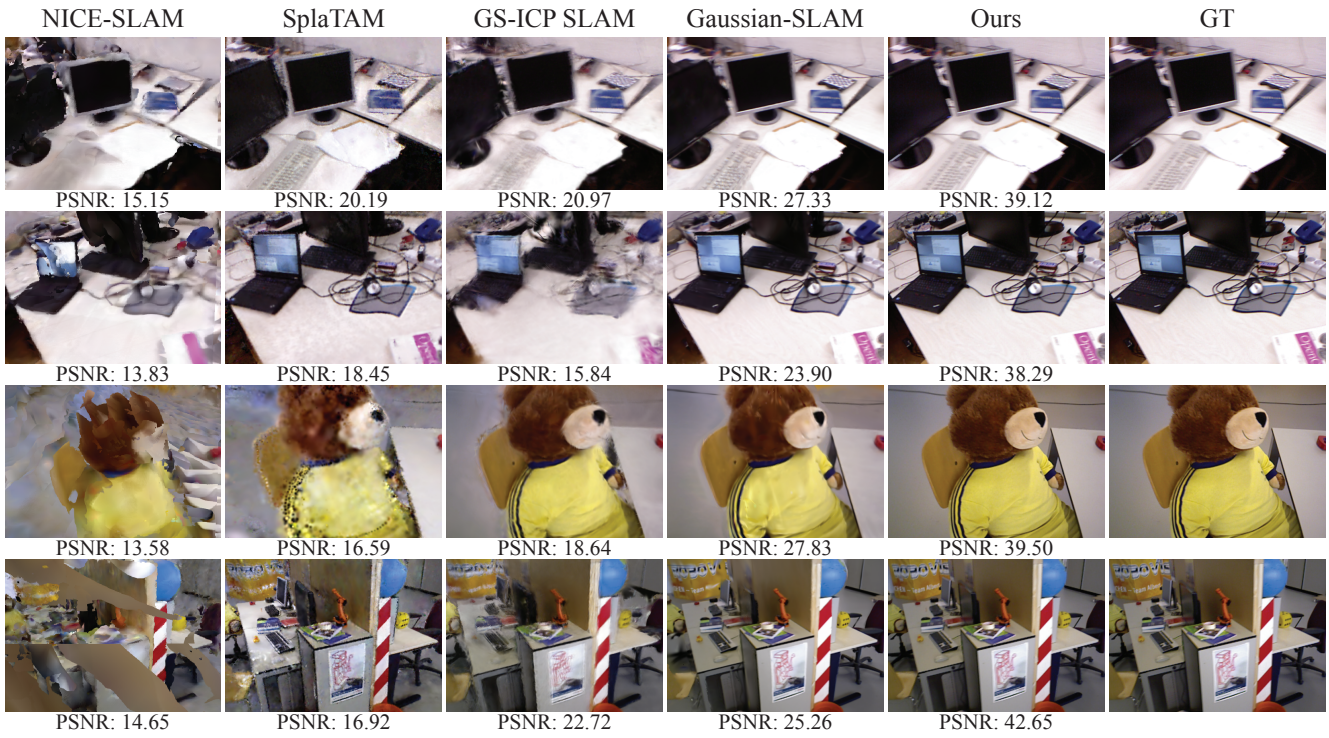

Figure 1. Visual comparisons in rendering on TUM-RGBD [18]. Please watch our video for more comparisons in rendering.

[4] Jiarui Hu, Xianhao Chen, Boyin Feng, Guanglin Li, Liangjing Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cg-slam: Efficient dense rgb-d slam in a consistent uncertainty-aware 3d gaussian field. *arXiv preprint arXiv:2403.16095*, 2024. 7, 11, 12

[5] Pengchong Hu and Zhizhong Han. Learning neural implicit through volume rendering with attentive depth fusion priors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 11

[6] Pengchong Hu and Zhizhong Han. Vtgaussian-slam: Rgbd slam for large scale scenes with splatting view-tied 3d gaussians. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. 2, 7, 8, 9, 10, 11, 12, 13

[7] M. M. Johari, C. Carta, and F. Fleuret. ESLAM: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 7, 8, 9, 10, 11, 13

[8] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2, 7, 8, 9, 10, 11, 12, 13

[9] Mingrui Li, Shuhong Liu, Heng Zhou, Guohao Zhu, Na Cheng, Tianchen Deng, and Hongyu Wang. Sgs-slam: Semantic gaussian splatting for neural dense slam, 2024. 1, 7

[10] Lorenzo Liso, Erik Sandström, Vladimir Yugay, Luc Van Gool, and Martin R Oswald. Loopy-slam: Dense neural slam with loop closures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20363–20373, 2024. 1, 7, 8, 9, 10, 11, 12, 13

[11] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987. 1, 2

[12] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. 2024. 1, 7

[13] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 143–152, 2017. 1

[14] Erik Sandström, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Luc Van Gool, Martin R Oswald, and Federico Tombari. Splat-slam: Globally optimized rgb-only slam with 3d gaussians. *arXiv preprint arXiv:2405.16544*, 2024. 2

[15] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 1, 2, 7, 8, 9, 10, 11, 12, 13

[16] Aleksandr V. Segal, Dirk Hähnel, and Sebastian Thrun. Generalized-icp. In *Robotics: Science and Systems*, 2009. 1

[17] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva,

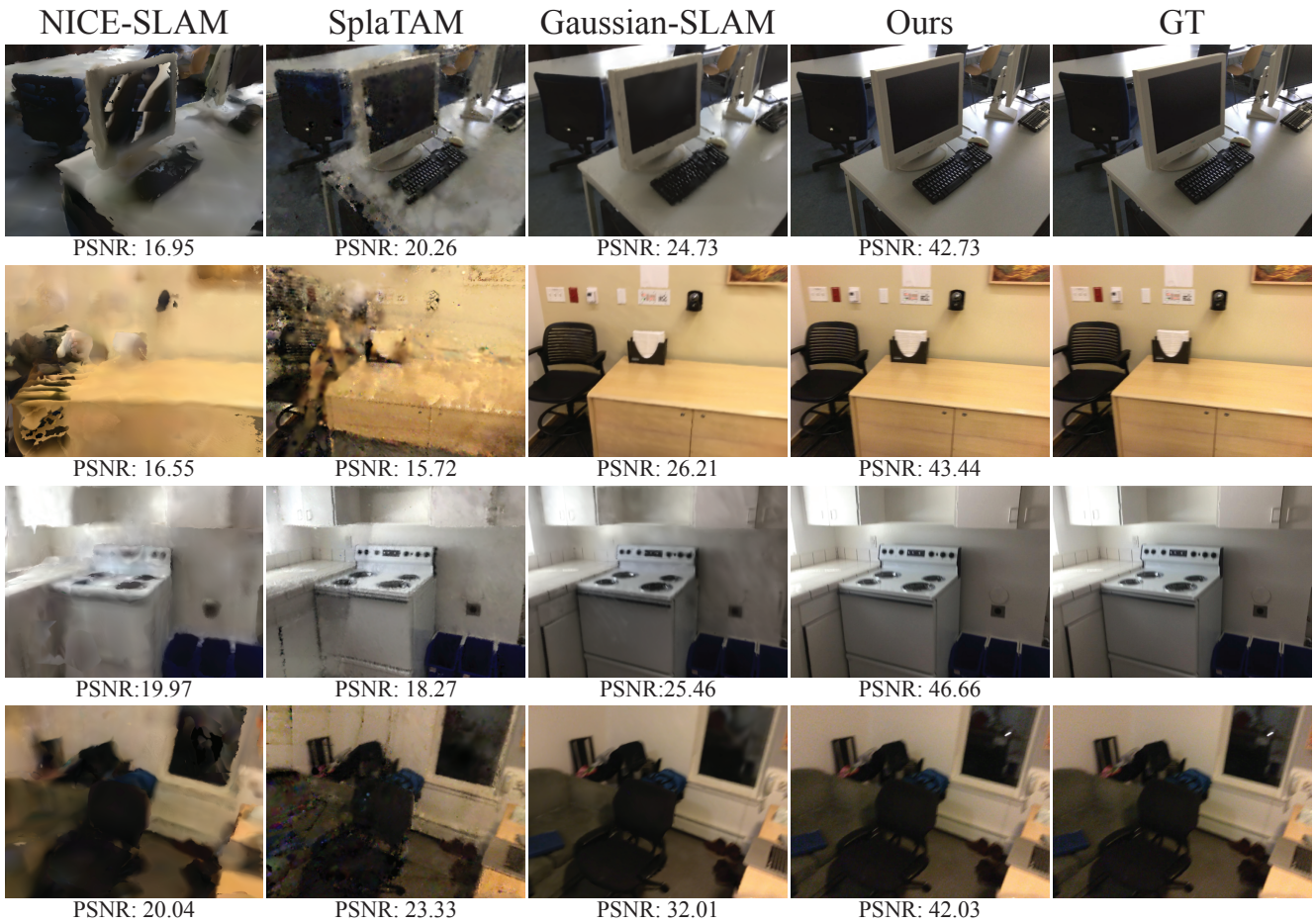

Figure 2. Visual comparisons in rendering on ScanNet [1]. Please watch our video for more comparisons in rendering.

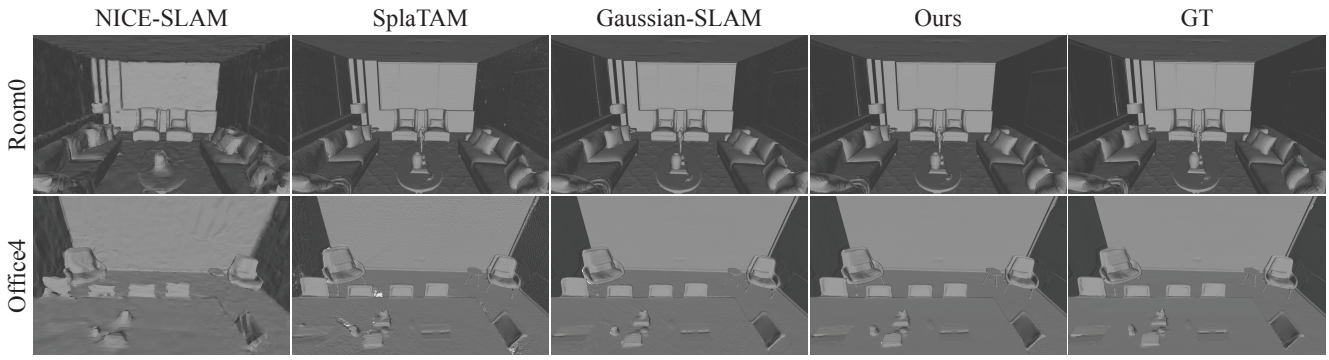

Figure 3. Visual comparisons in reconstruction on Replica [17].

Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard A. Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019. 1, 2, 4, 5, 7, 8, 11

[18] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012. 1, 2, 3, 5, 9, 11, 12

[19] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021. 1

[20] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-

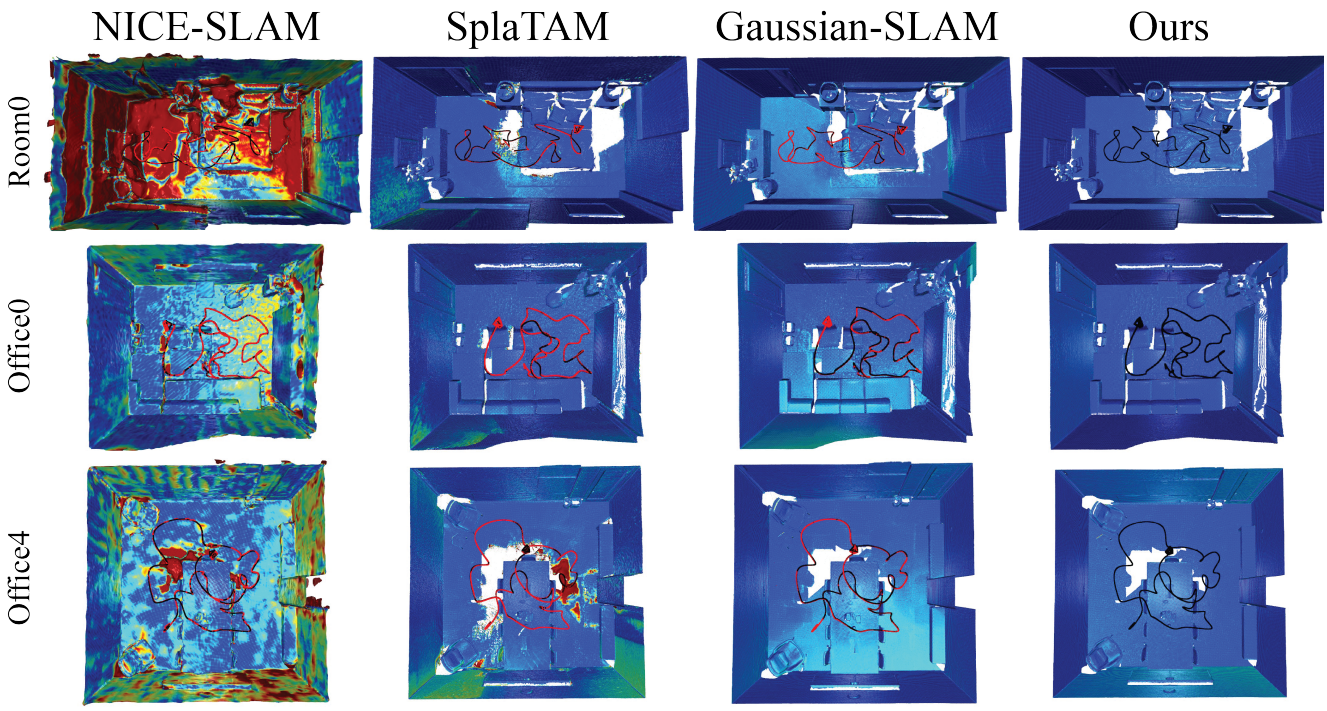

Figure 4. Visual comparisons in reconstruction and camera tracking on Replica [17]. Please watch our video for more comprehensive results.

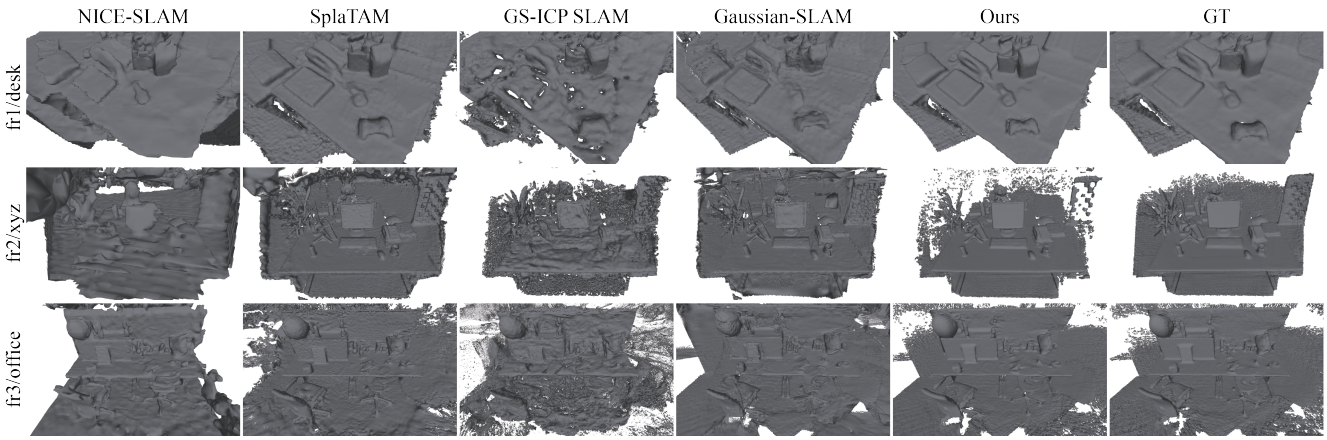

Figure 5. Visual comparisons in reconstruction on TUM-RGBD [18].

slam: Joint coordinate and sparse parametric encodings for neural real-time slam, 2023. 8

[21] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 1

[22] Jiaxin Wei and Stefan Leutenegger. Gsfusion: Online rgb-d mapping where gaussian splatting meets tsdf fusion, 2024. 1

[23] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*, 2024. 7, 8, 11

[24] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2022. 7, 8, 9, 10, 11, 12

[25] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 1, 2, 6, 13

[26] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-slam: Photo-realistic dense slam with gaus-

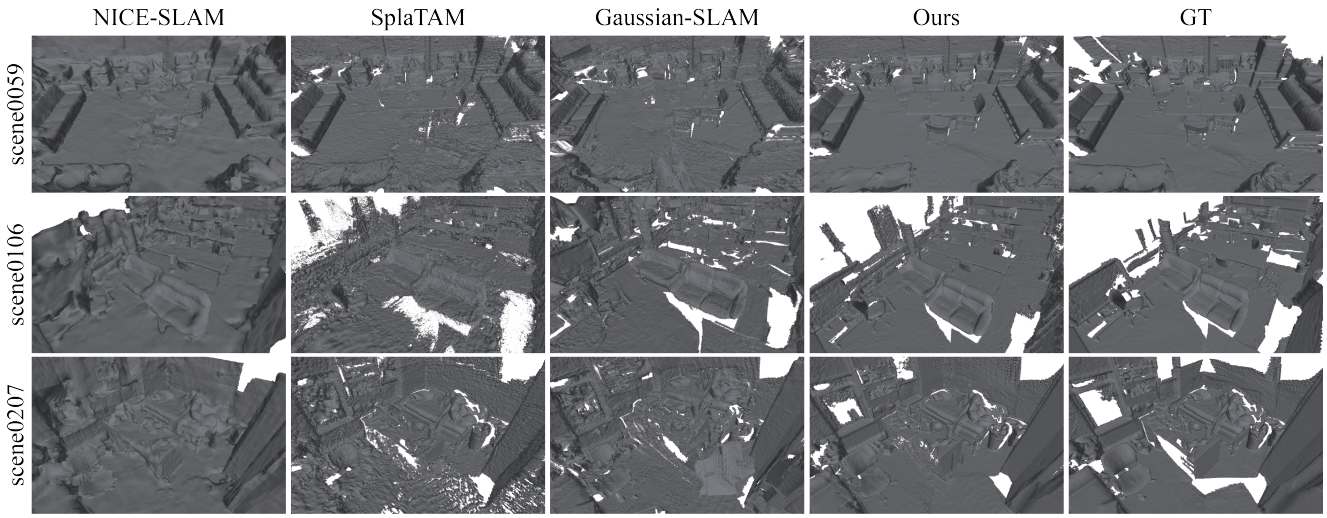

Figure 6. Visual comparisons in reconstruction on ScanNet [1].

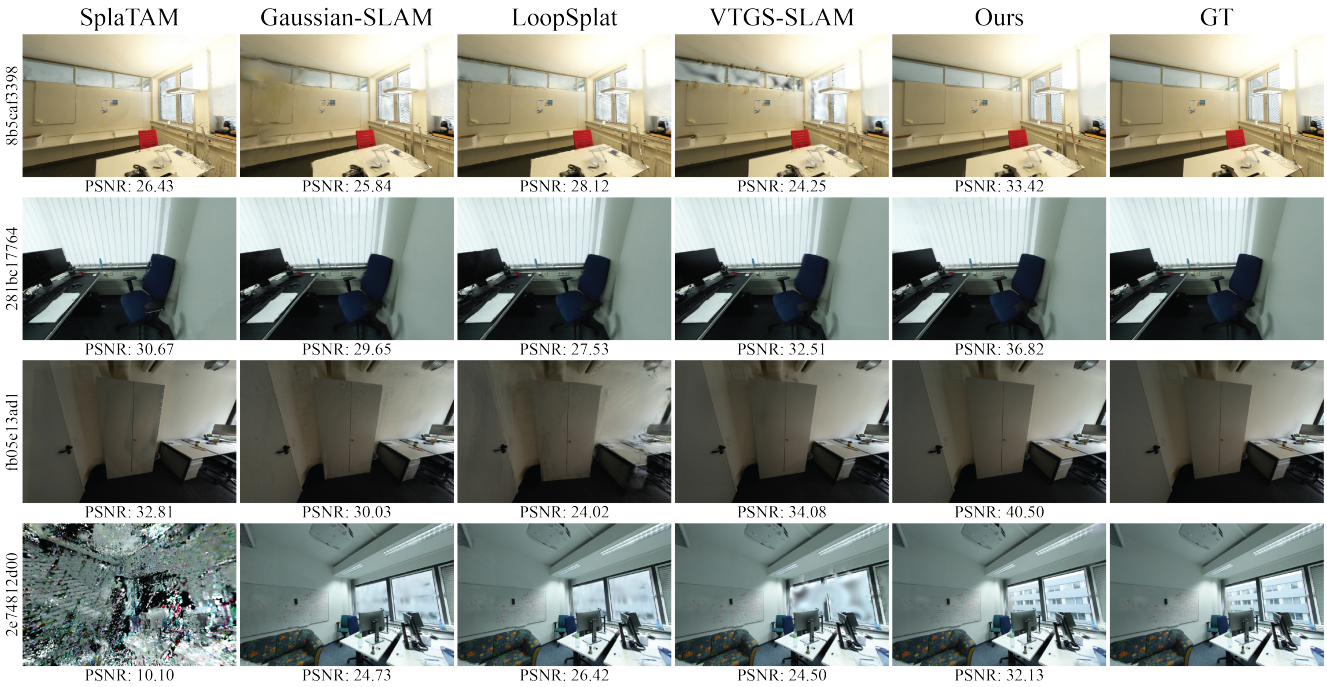

Figure 7. Visual comparisons in training views rendering on ScanNet++ [25]. Please watch our video for more comparisons in rendering.

sian splatting, 2023. 1, 2, 7, 8, 9, 10, 11, 12, 13

[27] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. 1

[28] Liyuan Zhu, Yue Li, Erik Sandström, Shengyu Huang, Konrad Schindler, and Iro Armeni. Loopsplat: Loop closure by registering 3d gaussian splats, 2024. 1, 2, 7, 8, 9, 10, 11, 12, 13

[29] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2, 7, 8, 9, 10, 11, 12

Table 1. Rendering performance comparisons in PSNR ↑, SSIM ↑, and LPIPS ↓ on Replica [17]. ∗ indicates methods relying on pre-trained data-driven priors.

Method	Metric	Rm0	Rm1	Rm2	Off0	Off1	Off2	Off3	Off4	Avg.
Neural Implicit Fields										
NICE-SLAM [29]	PSNR↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42
	SSIM↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809
	LPIPS↓	0.330	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233
Vox-Fusion [24]	PSNR↑	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21	24.41
	SSIM↑	0.683	0.751	0.798	0.857	0.876	0.794	0.803	0.847	0.801
	LPIPS↓	0.303	0.269	0.234	0.241	0.184	0.243	0.213	0.199	0.236
ESLAM [7]	PSNR↑	25.25	27.39	28.09	30.33	27.04	27.99	29.27	29.15	28.06
	SSIM↑	0.874	0.89	0.935	0.934	0.910	0.942	0.953	0.948	0.923
	LPIPS↓	0.315	0.296	0.245	0.213	0.254	0.238	0.186	0.210	0.245
Point-SLAM [15]	PSNR↑	32.40	34.08	35.50	38.26	39.16	33.99	33.48	33.49	35.17
	SSIM↑	0.974	0.977	0.982	0.983	0.986	0.960	0.960	0.979	0.975
	LPIPS↓	0.113	0.116	0.111	0.100	0.118	0.156	0.132	0.142	0.124
Loopy-SLAM∗ [10]	PSNR↑	-	-	-	-	-	-	-	-	35.47
	SSIM↑	-	-	-	-	-	-	-	-	0.981
	LPIPS↓	-	-	-	-	-	-	-	-	0.109
3D Gaussian Splatting										
SplaTAM [8]	PSNR↑	32.86	33.89	35.25	38.26	39.17	31.97	29.70	31.81	34.11
	SSIM↑	0.98	0.97	0.98	0.98	0.98	0.97	0.95	0.95	0.97
	LPIPS↓	0.07	0.10	0.08	0.09	0.09	0.10	0.12	0.15	0.10
SGS-SLAM [9]	PSNR↑	32.50	34.25	35.10	38.54	39.20	32.90	32.05	32.75	34.66
	SSIM↑	0.976	0.978	0.981	0.984	0.980	0.967	0.966	0.949	0.973
	LPIPS↓	0.070	0.094	0.070	0.086	0.087	0.101	0.115	0.148	0.096
GS-SLAM [23]	PSNR↑	31.56	32.86	32.59	38.70	41.17	32.36	32.03	32.92	34.27
	SSIM↑	0.968	0.973	0.971	0.986	0.993	0.978	0.970	0.968	0.975
	LPIPS↓	0.094	0.075	0.093	0.050	0.033	0.094	0.110	0.112	0.082
MonoGS [12]	PSNR↑	34.83	36.43	37.49	39.95	42.09	36.24	36.70	36.07	37.50
	SSIM↑	0.954	0.959	0.965	0.971	0.977	0.964	0.963	0.957	0.960
	LPIPS↓	0.068	0.076	0.075	0.072	0.055	0.078	0.065	0.099	0.070
Gaussian-SLAM [26]	PSNR↑	38.88	41.80	42.44	46.40	45.29	40.10	39.06	42.65	42.08
	SSIM↑	0.993	0.996	0.996	0.998	0.997	0.997	0.997	0.997	0.996
	LPIPS↓	0.017	0.018	0.019	0.015	0.016	0.020	0.020	0.020	0.018
VTGS-SLAM [6]	PSNR↑	39.95	43.06	43.13	46.88	47.20	42.14	40.99	43.35	43.34
	SSIM↑	0.992	0.996	0.996	0.998	0.997	0.996	0.996	0.996	0.996
	LPIPS↓	**0.014**	**0.013**	**0.014**	**0.009**	**0.009**	**0.012**	**0.013**	**0.015**	**0.012**
LoopSplat∗ [28]	PSNR↑	33.07	35.32	36.16	40.82	40.21	34.67	35.67	37.10	36.63
	SSIM↑	0.973	0.978	0.985	0.992	0.990	0.985	0.990	0.989	0.985
	LPIPS↓	0.116	0.122	0.111	0.085	0.123	0.140	0.096	0.106	0.112
CG-SLAM∗ [4]	PSNR↑	33.27	-	-	-	-	-	34.60	-	-
	SSIM↑	-	-	-	-	-	-	-	-	-
	LPIPS↓	-	-	-	-	-	-	-	-	-
Ours	PSNR↑	**40.85**	**43.94**	**44.52**	**48.55**	**48.41**	**44.35**	**42.62**	**45.71**	**44.87**
	SSIM↑	**0.997**	**0.998**	**0.998**	**0.999**	**0.998**	**0.998**	**0.998**	**0.998**	**0.998**
	LPIPS↓	0.020	0.020	0.022	0.018	0.024	0.017	0.022	0.022	0.021

Table 2. Reconstruction performance comparison in Depth L1 [cm]↓ and F1 [%] ↑ on Replica [17]. ∗ indicates methods relying on pre-trained data-driven priors.

Method	Metric	Rm0	Rm1	Rm2	Off0	Off1	Off2	Off3	Off4	Avg.
Neural Implicit Fields										
NICE-SLAM [29]	Depth L1 [cm]↓	1.81	1.44	2.04	1.39	1.76	8.33	4.99	2.01	2.97
	F1 [%] ↑	45.0	44.8	43.6	50.0	51.9	39.2	39.9	36.5	43.9
Vox-Fusion [24]	Depth L1 [cm]↓	1.09	1.90	2.21	2.32	3.40	4.19	2.96	1.61	2.46
	F1 [%] ↑	69.9	34.4	59.7	46.5	40.8	51.0	64.6	50.7	52.2
ESLAM [7]	Depth L1 [cm]↓	0.97	1.07	1.28	0.86	1.26	1.71	1.43	1.06	1.18
	F1 [%] ↑	81.0	82.2	83.9	78.4	75.5	77.1	75.5	79.1	79.1
Co-SLAM [20]	Depth L1 [cm]↓	0.99	0.82	2.28	1.24	1.61	7.70	4.65	1.43	2.59
	F1 [%] ↑	77.7	74.2	69.3	75.2	75.2	54.3	56.8	75.3	69.7
Point-SLAM [15]	Depth L1 [cm]↓	0.53	0.22	0.46	0.30	0.57	0.49	0.51	0.46	0.44
	F1 [%] ↑	86.9	**92.3**	90.8	93.8	**91.6**	89.0	88.2	85.6	89.8
Loopy-SLAM∗ [10]	Depth L1 [cm]↓	0.30	0.20	0.42	0.23	0.46	0.60	0.37	0.24	0.35
	F1 [%] ↑	91.6	92.4	90.6	93.9	91.6	88.5	89.0	88.7	90.8
3D Gaussian Splatting										
SplaTAM [8]	Depth L1 [cm]↓	0.43	0.38	0.54	0.44	0.66	1.05	1.60	0.68	0.72
	F1 [%] ↑	89.3	88.2	88.0	91.7	90.0	85.1	77.1	80.1	86.1
GS-SLAM [23]	Depth L1 [cm]↓	1.31	0.82	1.26	0.81	0.96	1.41	1.53	1.08	1.16
	F1 [%] ↑	62.9	79.9	66.8	80.0	81.6	66.0	59.2	65.0	70.2
Gaussian-SLAM [26]	Depth L1 [cm]↓	0.61	0.25	0.54	0.50	0.52	0.98	1.63	0.42	0.68
	F1 [%] ↑	88.8	91.4	90.5	91.7	90.1	87.3	84.2	87.4	88.9
VTGS-SLAM [6]	Depth L1 [cm]↓	0.48	0.28	0.61	0.41	0.48	0.62	0.86	0.53	0.53
	F1 [%] ↑	90.7	91.7	90.7	93.0	90.8	88.3	87.5	87.0	90.0
LoopSplat∗ [28]	Depth L1 [cm]↓	0.39	0.23	0.52	0.32	0.51	0.63	1.09	0.40	0.51
	F1 [%] ↑	90.6	91.9	91.1	93.3	90.4	88.9	88.7	88.3	90.4
Ours	Depth L1 [cm]↓	**0.27**	**0.17**	**0.36**	**0.22**	**0.38**	**0.37**	**0.45**	**0.21**	**0.30**
	F1 [%] ↑	**91.6**	**92.3**	**91.4**	**93.9**	91.2	**89.3**	**88.9**	**88.7**	**90.9**

Table 3. Rendering performance comparison in PSNR ↑, SSIM ↑, and LPIPS ↓ on TUM-RGBD [18]. ∗ indicates methods relying on pre-trained data-driven priors.

Method	Metric	fr1/desk	fr2/xyz	fr3/office	Avg.
Neural Implicit Fields					
NICE-SLAM [29]	PSNR↑	13.83	17.87	12.89	14.86
	SSIM↑	0.569	0.718	0.554	0.614
	LPIPS↓	0.482	0.344	0.498	0.441
Vox-Fusion [24]	PSNR↑	15.79	16.32	17.27	16.46
	SSIM↑	0.647	0.706	0.677	0.677
	LPIPS↓	0.523	0.433	0.456	0.471
ESLAM [7]	PSNR↑	11.29	17.46	17.02	15.26
	SSIM↑	0.666	0.310	0.457	0.478
	LPIPS↓	0.358	0.698	0.652	0.569
Point-SLAM [15]	PSNR↑	13.87	17.56	18.43	16.62
	SSIM↑	0.627	0.708	0.754	0.696
	LPIPS↓	0.544	0.585	0.448	0.526
Loopy-SLAM∗ [10]	PSNR↑	-	-	-	12.94
	SSIM↑	-	-	-	0.489
	LPIPS↓	-	-	-	0.645
3D Gaussian Splatting					
SplaTAM [8]	PSNR↑	22.00	24.50	21.90	22.80
	SSIM↑	0.857	0.947	0.876	0.893
	LPIPS↓	0.232	0.100	0.202	0.178
Gaussian-SLAM [26]	PSNR↑	24.01	25.02	26.13	25.05
	SSIM↑	0.924	0.924	0.939	0.929
	LPIPS↓	0.178	0.186	0.141	0.168
VTGS-SLAM [6]	PSNR↑	27.09	33.01	30.50	30.20
	SSIM↑	0.959	0.982	0.974	0.972
	LPIPS↓	0.085	0.038	0.063	0.062
LoopSplat∗ [28]	PSNR↑	22.03	22.68	23.47	22.72
	SSIM↑	0.849	0.892	0.879	0.873
	LPIPS↓	0.307	0.217	0.253	0.259
Ours	PSNR↑	**39.21**	**36.74**	**39.86**	**38.60**
	SSIM↑	**0.998**	**0.996**	**0.997**	**0.997**
	LPIPS↓	**0.009**	**0.014**	**0.012**	**0.012**

Table 4. Rendering performance comparison in PSNR ↑, SSIM ↑, and LPIPS ↓ on ScanNet [1]. ∗ indicates methods relying on pre-trained data-driven priors.

Method	Metric	0000	0059	0106	0169	0181	0207	Avg.
Neural Implicit Fields								
NICE-SLAM [29]	PSNR↑	18.71	16.55	17.29	18.75	15.56	18.38	17.54
	SSIM↑	0.641	0.605	0.646	0.629	0.562	0.646	0.621
	LPIPS↓	0.561	0.534	0.510	0.534	0.602	0.552	0.548
Vox-Fusion [24]	PSNR↑	19.06	16.38	18.46	18.69	16.75	19.66	18.17
	SSIM↑	0.662	0.615	0.753	0.650	0.666	0.696	0.673
	LPIPS↓	0.515	0.528	0.439	0.513	0.532	0.500	0.504
ESLAM [7]	PSNR↑	15.70	14.48	15.44	14.56	14.22	17.32	15.29
	SSIM↑	0.687	0.632	0.628	0.656	0.696	0.653	0.658
	LPIPS↓	0.449	0.450	0.529	0.486	0.482	0.534	0.488
Point-SLAM [15]	PSNR↑	21.30	19.48	16.80	18.53	22.27	20.56	19.82
	SSIM↑	0.806	0.765	0.676	0.686	0.823	0.750	0.751
	LPIPS↓	0.485	0.499	0.544	0.542	0.471	0.544	0.514
LoopySLAM∗ [10]	PSNR↑	-	-	-	-	-	-	15.23
	SSIM↑	-	-	-	-	-	-	0.629
	LPIPS↓	-	-	-	-	-	-	0.671
3D Gaussian Splatting								
SplaTAM [8]	PSNR↑	19.33	19.27	17.73	21.97	16.76	19.8	19.14
	SSIM↑	0.660	0.792	0.690	0.776	0.683	0.696	0.716
	LPIPS↓	0.438	0.289	0.376	0.281	0.420	0.341	0.358
Gaussian-SLAM [26]	PSNR↑	28.54	26.21	26.26	28.60	27.79	28.63	27.70
	SSIM↑	0.926	0.934	0.926	0.917	0.922	0.914	0.923
	LPIPS↓	0.271	0.211	0.217	0.226	0.277	0.288	0.248
VTGS-SLAM [6]	PSNR↑	31.51	30.60	31.27	32.02	29.60	31.58	31.10
	SSIM↑	0.957	0.974	0.975	0.962	0.954	0.946	0.961
	LPIPS↓	0.131	0.080	0.074	0.091	0.145	0.124	0.108
LoopSplat∗ [28]	PSNR↑	24.99	23.23	23.35	26.80	24.82	26.33	24.92
	SSIM↑	0.840	0.831	0.846	0.877	0.824	0.854	0.845
	LPIPS↓	0.450	0.400	0.409	0.346	0.514	0.430	0.425
Ours	PSNR↑	**40.85**	**41.10**	**42.91**	**42.76**	**43.51**	**42.71**	**42.31**
	SSIM↑	**0.996**	**0.997**	**0.998**	**0.997**	**0.997**	**0.996**	**0.997**
	LPIPS↓	**0.056**	**0.051**	**0.041**	**0.041**	**0.057**	**0.046**	**0.049**

Table 5. Tracking performance comparisons in ATE RMSE ↓ [cm] on Replica [17]. ∗ methods relying on pre-trained data-driven priors.

Method	Rm0	Rm1	Rm2	Off0	Off1	Off2	Off3	Off4	Avg.
Neural Implicit Fields									
NICE-SLAM [29]	1.69	2.04	1.55	0.99	0.90	1.39	3.97	3.08	1.95
DF-Prior [5]	1.39	1.55	2.60	1.09	1.23	1.61	3.61	1.42	1.81
Vox-Fusion [24]	0.27	1.33	0.47	0.70	1.11	0.46	0.26	0.58	0.65
ESLAM [7]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63	0.63
Point-SLAM [15]	0.61	0.41	0.37	0.38	0.48	0.54	0.72	0.63	0.52
Loopy-SLAM∗ [10]	0.24	0.24	0.28	0.26	0.40	0.29	0.22	0.35	0.29
3D Gaussian Splatting									
SplaTAM [8]	0.31	0.40	0.29	0.47	0.27	0.29	0.32	0.55	0.36
GS-SLAM [23]	0.48	0.53	0.33	0.52	0.41	0.59	0.46	0.70	0.50
Gaussian-SLAM [26]	0.29	0.29	0.22	0.37	0.23	0.41	0.30	0.35	0.31
VTGS-SLAM [6]	0.22	0.26	0.19	0.28	0.26	0.34	0.25	0.43	0.28
GS-ICP SLAM [3]	**0.15**	**0.16**	0.11	0.18	**0.12**	0.17	**0.16**	0.21	**0.16**
LoopSplat∗ [28]	0.28	0.22	0.17	0.22	0.16	0.49	0.20	0.30	0.26
CG-SLAM∗ [4]	0.29	0.27	0.25	0.33	0.14	0.28	0.31	0.29	0.27
Ours	**0.15**	0.17	**0.10**	**0.16**	**0.12**	**0.16**	0.25	**0.20**	**0.16**

Table 6. Tracking performance in ATE RMSE ↓ [cm] on TUM-RGBD [18]. ∗ methods using pre-trained data-driven priors.

Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Neural Implicit Fields				
NICE-SLAM [29]	4.3	31.7	3.9	13.3
Vox-Fusion [24]	3.5	1.5	26.0	10.3
Point-SLAM [15]	4.3	1.3	3.5	3.0
Loopy-SLAM∗ [10]	3.8	1.6	3.4	2.9
3D Gaussian Splatting				
SplaTAM [8]	3.4	1.2	5.2	3.3
GS-SLAM [23]	3.3	1.3	6.6	3.7
Gaussian-SLAM [26]	2.6	1.3	4.6	2.9
VTGS-SLAM [6]	2.4	**1.1**	4.4	2.6
GS-ICP SLAM [3]	2.7	1.8	2.7	2.4
LoopSplat∗ [28]	2.1	1.6	3.2	2.3
CG-SLAM∗ [4]	2.4	1.2	2.5	2.0
Ours	**2.2**	1.7	**2.0**	**2.0**

Table 7. Tracking performance in ATE RMSE ↓ [cm] on TUM-RGBD [18]. ∗ indicates VTGS-SLAM [6] treats each frame as a section, initializing dense Gaussians on each frame similar to our approach.

Method	fr1/desk	fr2/xyz	fr3/office	Avg.
VTGS-SLAM [6]	382.4	3462.5	400.5	1415.1
Ours	**2.2**	**1.7**	**2.0**	**2.0**

Table 8. Tracking performance in ATE RMSE ↓ [cm] on fr3/nostructure_texture_far in TUM-RGBD [18]. ∗ methods using pre-trained data-driven priors.

Method	SplaTAM [8]	LoopSplat∗ [28]	VTGS-SLAM [6]	GS-ICP SLAM [3]	Ours(w/o init.)	Ours(w/ init.)
ATE RMSE↓[cm]	11.3	7.3	9.7	115.8	122.4	**4.7**

Table 9. Tracking performance in ATE RMSE ↓ [cm] on ScanNet [1]. ∗ methods using pre-trained data-driven priors.

Method	0000	0059	0106	0169	0181	0207	Avg.
Neural Implicit Fields							
NICE-SLAM [29]	12.0	14.0	7.9	10.9	13.4	6.2	10.7
Vox-Fusion [24]	68.8	24.2	8.4	27.3	23.3	9.4	26.9
Point-SLAM [15]	**10.2**	7.8	8.7	22.2	14.8	9.5	12.2
Loopy-SLAM∗ [10]	4.2	7.5	8.3	7.5	10.6	7.9	7.7
3D Gaussian Splatting							
SplaTAM [8]	12.8	10.1	17.7	12.1	11.1	7.5	11.9
Gaussian-SLAM [26]	24.8	8.6	11.3	14.6	18.7	14.4	15.4
VTGS-SLAM [6]	17.8	8.7	11.8	10.5	10.6	8.6	11.3
LoopSplat∗ [28]	6.2	7.1	7.4	10.6	8.5	6.6	7.7
CG-SLAM∗ [4]	7.1	7.5	8.9	8.2	11.6	5.3	8.1
Ours	11.9	**6.4**	**5.3**	**8.5**	**10.3**	**4.7**	**7.9**

Table 10. Tracking performance in ATE RMSE ↓ [cm] on ScanNet++ [25]. ∗ methods relying on pre-trained data-driven priors.

Method	a	b	c	d	e	Avg.
Neural Implicit Fields						
Point-SLAM [15]	246.16	632.99	830.79	271.42	574.86	511.24
ESLAM [7]	25.15	2.15	27.02	20.89	35.47	22.14
Loopy-SLAM∗ [10]	-	-	25.16	234.25	81.48	113.63
3D Gaussian Splatting						
SplaTAM [8]	1.50	**0.57**	0.31	443.10	1.58	89.41
Gaussian-SLAM [26]	1.37	5.97	2.70	2.35	1.02	2.68
VTGS-SLAM [6]	2.80	1.50	1.00	1.20	1.30	1.60
LoopSplat∗ [28]	1.14	3.16	3.16	1.68	0.91	2.05
Ours(w/o Initialization)	5.57	16.70	1.70	4.50	4.20	6.50
Ours	**0.80**	0.71	**0.05**	**0.63**	**0.74**	**0.59**

Table 11. Rendering performance comparison in PSNR ↑ on ScanNet++ [25]. ∗ indicates methods relying on pre-trained data-driven priors.

Method	a	b	c	d	e	Avg.
3D Gaussian Splatting						
SplaTAM [8]	28.02	27.93	29.48	19.65	28.48	26.71
Gaussian-SLAM [26]	30.06	30.02	31.15	28.75	31.94	30.38
VTGS-SLAM [6]	32.84	31.02	32.44	31.43	33.38	32.22
LoopSplat∗ [28]	30.15	30.08	30.04	28.94	31.78	30.20
Ours	**35.95**	**34.84**	**35.81**	**35.71**	**41.32**	**36.73**

Table 12. Novel View Synthesis performance comparison in PSNR ↑ on ScanNet++ [25]. ∗ indicates methods relying on pre-trained data-driven priors. We calculate PSNR including all pixels, regardless of whether they have a valid depth input.

Method	a	b	c	d	e	Avg.
3D Gaussian Splatting						
SplaTAM [8]	23.95	22.66	13.95	8.47	20.06	17.82
Gaussian-SLAM [26]	26.66	24.42	15.01	18.35	21.91	21.27
VTGS-SLAM [6]	25.55	24.25	**16.94**	18.59	**21.95**	21.46
LoopSplat∗ [28]	25.60	23.65	15.87	18.86	22.51	21.30
Ours	**26.81**	**26.79**	15.38	**20.89**	21.71	**22.32**