RTG-SLAM: Real-time 3D Reconstruction at Scale using Gaussian Splatting
——— Supplementary Document ———

1 GAUSSIAN INITIALIZATION

Here we introduce how we compute the covariance matrix $\mathbf{\Sigma_u}$ for each newly added Gaussian for pixel $\mathbf{u}$ in detail. Each new Gaussian $G_\mathbf{u}$ is initialized as a flat circle disc. For opaque Gaussians, our goal is to cover the surface as much as possible without largely impacting existing ones. For this, we calculate the distance from the pixel's 3D position $\mathbf{V}_k^g(\mathbf{u})$ to its three nearest Gaussians $G_{1,2,3}$ in the scene, and initialize its scale based on the following formula:

$$\mathbf{s}_{\mathbf{u},1} = \sqrt{\frac{1}{3}\sum_{i=1}^{3}\Big(||\mathbf{V}_k^g(\mathbf{u}) - \mathbf{p}_i|| - 0.5(a_i + b_i)\Big)},$$
$$\mathbf{s}_{\mathbf{u},2} = \mathbf{s}_{\mathbf{u},1} \quad \mathbf{s}_{\mathbf{u},3} = 0.1\mathbf{s}_{\mathbf{u},1}. \tag{1}$$

Here a is the biggest eigenvalue of the Gaussian covariance matrix and b is the second largest eigenvalue. For the transparent Gaussian, the ratios of its three axes are also set to 1:1:0.1. However, in order to reduce the disruption of the rendering results on other pixels, its maximum scale is limited to 0.01 m. Finally, the Gaussian orientation $\mathbf{q_u}$ is initialized such that its shortest axis aligns with the pixel's global normal $\mathbf{N}_k^g(\mathbf{u})$. The above initialization method allows the newly added opaque Gaussians to cover the scene surface as much as possible, with little influence on the existing Gaussians in the scene $\mathcal{S}$.

2 GAUSSIAN OPTIMIZATION

Since we only optimize the unstable Gaussians, we use $\mathcal{S}_{unstable}$ to render a light transmission map $\hat{\mathbf{T}}unstable$ before optimization, and the loss L is only computed on the pixels:

$$M_{unstable} = \{\mathbf{u}_{unstable} | \hat{\mathbf{T}}_{unstable}(\mathbf{u}_{unstable}) < 1\}. \tag{2}$$

However, in order to achieve fast rendering and optimization, [Kerbl et al. 2023] adopts a tile-based rasterizer for Gaussian splatting: the image is divided into 16×16 tiles, and each Gaussian is assigned a key that combines view space depth and tile ID and then sorted. In fact, when there are just a few pixels within a tile that need to be rendered, instantiating all Gaussians on this tile is quite inefficient. Therefore, we discard those tiles where the number of pixels that need to be rendered is less than 50%. This strategy drastically reduces inefficient computation, and increases the overall speed of the optimization process.

3 IMPLEMENTATION DETAILS

We accelerate our system by implementing it in parallel with three threads: one thread for Gaussians optimization, one for front-end online tracking, and the other for back-end graph optimization. The Gaussians optimization and front-end online tracking are implemented by `python` using the pytorch framework and we write custom `CUDA` kernels for our rendering process and back propagation.

Author's address:

Statistic	corridor	storeroom	hotel room	home	office
Trajectory Length (m)	21.9	18.9	39.7	32.2	41.0
Scan Area (m^2)	43.4	44.3	56.3	69.8	100.2
Frame Number	4890	3310	4838	6130	6889

Table 1. Statistics of Azure dataset

The back-end optimization part is inherited from ORB SLAM2 [Mur-Artal and Tardós 2017] and implemented in `C++`. We also build an interactive viewer using the open-source SIBR [Bonopera et al. 2020; Kerbl et al. 2023] to visualize the SLAM process and the reconstructed model. In order to achieve free movement and scanning in the scene, we use a laptop with an `intel i7 10750-H` CPU and `nvidia 2070 GPU` connected to an Azure Kinect RGBD camera for data acquisition. The RGBD images captured by the camera are transmitted to the desktop computer through a wireless network and the desktop computer completes the SLAM computations, and then the results are sent back to our viewer on the laptop for visualization. In all our experiments, we sample 5% of pixels for the Gaussians adding. For small-scale synthetic dataset Replica [Straub et al. 2019], we set the optimization time window to 6 and optimize 50 iterations. For our large-scale real Azure dataset, we set the optimization time window to 8 and optimize 50 iterations. For TUM-RGBD [Sturm et al. 2012], we set the optimization time window to 4 and optimize 50 iterations. For ScanNet++ [Yeshwanth et al. 2023], we set the optimization time window to 3 and optimize 75 iterations due to the high-resolution (1752×1168) and sparse viewports. For learning rates, we set $lr_{position} = 0.001$, $lr_{SH0} = 0.0005$, $lr_\alpha = 0$, $lr_{scale} = 0.004$, $lr_{rotation} = 0.001$ on Replica and ScanNet++. And we set $lr_{position} = 0.001$, $lr_{SH0} = 0.001$, $lr_\alpha = 0$, $lr_{scale} = 0.002$, $lr_{rotation} = 0.001$ on our datset and TUM-RGBD. The learning rates of other SH coefficients is $0.05 \times lr_{SH0}$. For the confidence count threshold, we use $\delta_\eta = 100$ for synthetic Replica, $\delta_\eta = 200$ for TUM-RGBD and $\delta_\eta = 400$ for large-scale Azure and ScanNet++ scenes.

4 DATASET DETAILS

For ScanNet++, we select 4 subsets (8b5caf3398, 39f36da05b, b20a261fdf, f34d532901) for evaluation. The statistics of Azure dataset are listed in Table 1. Note that we don't have the ground truth for Azure dataset, so it is mainly used for qualitative demonstration (except the evaluation of time & memory performance).

5 MORE COMPARISON RESULTS

5.1 Quantitative results on different datasets

We add more evaluation on different datasets here. Please note that the low quality 3D model of TUM-RGBD makes it infeasible to evaluate geometry accuracy, as in previous papers. The cameras

Method	Acc.↓	Acc. Ratio↑	Comp.↓	Comp. Ratio↑
NICE-SLAM[2022]	2.84	84.44	2.31	84.97
Co-SLAM[2023]	2.33	88.89	1.63	89.94
ESLAM[2023]	1.47	91.44	**1.11**	**93.84**
Point-SLAM[2023]	**0.61**	**99.94**	2.42	86.85
SplaTAM[2023]	2.88	73.89	3.57	71.68
Ours	0.75	98.87	2.81	82.76

Table 2. Comparison of geometry accuracy on Replica.

Method	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
NICE-SLAM[2022]	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.06
Co-SLAM[2023]	0.69	0.59	0.73	0.87	0.47	2.16	1.30	0.62	0.93
ESLAM[2023]	0.66	0.62	0.55	0.44	0.43	0.50	0.66	0.53	0.55
Point-SLAM[2023]	0.54	0.43	0.34	0.36	0.45	0.44	0.63	0.72	0.49
SplaTAM[2023]	0.47	0.42	0.32	0.46	0.24	0.28	0.39	0.56	0.39
Ours	**0.20**	**0.18**	**0.13**	**0.22**	**0.12**	**0.22**	**0.20**	**0.19**	**0.18**

Table 3. Comparison of tracking accuracy (unit: *cm*) on Replica.

far apart in ScanNet++ result in tracking failure for classical, NeRF and our SLAM, so only geometry accuracy is evaluated. Our Azure dataset doesn't have the groundtruth camera or 3D model. The comparison of geometry accuracy on Replica is shown in Table 2. Similar to the paper, our geometry accuracy still outperforms the other methods except Point-SLAM which doesn't optimize point positions from the perfect depth. SplaTAM and ours degrade in completion because Gaussians cannot complete unscanned regions as NeRF. The comparison of tracking accuracy on Replica are shown in Table 3. Our method consistently outperforms the NeRF SLAM and the concurrent Gaussian SLAM method. We believe this is due to our back-end graph optimization based on ORB landmarks. Please note that in order to ensure fairness in comparison, although there is no noise in the depth images on Replica, we still used frame-to-model ICP, just without applying the bilateral filter to the depth input. We also report the time and memory performance on TUM-RGBD in Table 4. Our system achieves the highest scanning speed and lowest memory cost.

Finally we compare the rendering quality. The results of training view synthesis quality on Replica are reported in Table 5. Please note that this comparison is actually unfair as Point-SLAM [2023] takes the ground-truth depth maps as input to help sampling the 3D volume for rendering. In contrast, our method and SplaTAM [2023] do not require any auxiliary input. Even so, our method still achieves a rendering quality comparable with Point-SLAM and SplaTAM, and consistently outperforms the other NeRF SLAM methods. The quantitative comparison of novel-view synthesis on ScanNet++ testing views is reported in Table 6. Our method is comparable to SplaTAM and outperforms the other NeRF-SLAM methods.

5.2 Comparison with more SLAM systems

We show more comparisons with ElasticFusion [Whelan et al. 2015], BundleFusion [Dai et al. 2017], and GO-SLAM [Zhang et al. 2023]. The tracking accuracy is evaluated on Replica and TUM-RGBD and the results are listed in Table 7. we also evaluate the geometric accuracy on Replica as shown in Table 8. Our method achieves comparable geometry accuracy as BundleFusion. GO-SLAM's completion numbers are worse than those of its paper due to considering unscanned regions for fair comparison.

Method	FPS↑	Memory (MB)↓
NICE-SLAM[2022]	0.06	9930
CO-SLAM[2023]	7.18	18607
ESLAM[2023]	0.30	18617
Point-SLAM[2023]	0.26	11000
SplaTAM[2023]	0.14	12100
Ours	**21.74**	**3563**

Table 4. Comparison of time and memory performance on TUM-RGBD.

Method	Metric	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
NICE-SLAM [2022]	PSNR↑	24.72	26.79	27.06	30.21	32.78	26.59	26.22	24.74	27.39
	SSIM↑	0.787	0.799	0.807	0.881	0.906	0.816	0.801	0.834	0.829
	LPIPS↓	0.431	0.372	0.329	0.322	0.275	0.321	0.288	0.333	0.334
Co-SLAM [2023]	PSNR↑	28.88	28.51	29.37	35.44	34.63	26.56	28.79	32.16	30.54
	SSIM↑	0.892	0.843	0.851	0.854	0.826	0.814	0.866	0.856	0.850
	LPIPS↓	0.213	0.205	0.215	0.177	0.181	0.172	0.163	0.176	0.188
ESLAM [2023]	PSNR↑	26.96	28.98	29.80	35.04	33.81	30.08	30.01	31.34	30.75
	SSIM↑	0.821	0.837	0.843	0.902	0.873	0.865	0.881	0.886	0.863
	LPIPS↓	0.171	0.173	0.187	0.172	0.181	0.186	0.172	0.174	0.177
Point-SLAM [2023]	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
SplaTAM [2023]	PSNR↑	32.31	33.36	34.78	38.16	38.49	31.66	29.24	31.54	33.69
	SSIM↑	**0.974**	0.966	**0.983**	0.982	0.980	0.962	0.948	0.946	0.968
	LPIPS↓	**0.072**	**0.101**	**0.073**	0.084	0.095	**0.102**	**0.123**	0.157	**0.101**
Ours	PSNR↑	31.56	**34.21**	**35.57**	**39.11**	**40.27**	33.54	32.76	**36.48**	**35.43**
	SSIM↑	0.967	**0.979**	0.981	**0.990**	**0.992**	**0.981**	**0.981**	**0.984**	**0.982**
	LPIPS↓	0.131	0.105	0.115	**0.068**	**0.075**	0.134	0.128	**0.117**	0.109

Table 5. Comparison of train view synthesis on Replica.

Method	PSNR↑	SSIM↑	LPIPS↓
NICE-SLAM[2022]	23.71	0.797	0.341
CO-SLAM[2023]	23.20	0.837	0.413
ESLAM[2023]	27.06	0.856	0.322
Point-SLAM[2023]	21.85	0.802	0.404
SplaTAM[2023]	**27.77**	0.864	**0.233**
Ours	27.27	**0.872**	0.295

Table 6. Comparison of novel view synthesis on ScanNet++.

6 MORE ABLATION STUDIES

6.1 Ablation study on sampled pixel number

Here we evaluate the influence of the number of sampled pixels for adding Gaussians. We set the sampling ratio to 5%, 10%, and 20% for each frame to reconstruct Replica office0 and reported the image quality metrics. The results suggest that even if we sample

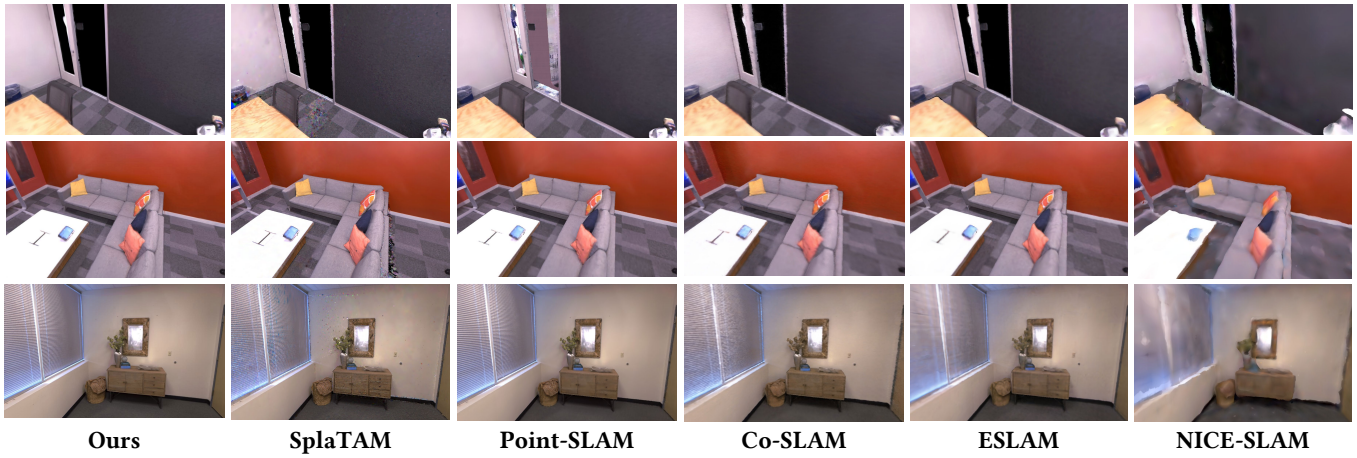

Fig. 1. Comparison of novel view synthesis on Replica.

Method	Replica	TUM-RGBD
ElasticFusion[2015]	1.13	2.07
BundleFusion[2017]	0.46	1.63
GO-SLAM[2023]	0.37	1.28
Ours	**0.18**	**1.06**

Table 7. Comparison of tracking accuracy (unit: *cm*) with more SLAM systems.

Method	Acc.↓	Acc. Ratio↑	Comp.↓	Comp. Ratio↑
ElasticFusion[2015]	1.13	96.33	4.43	75.25
BundleFusion[2017]	0.77	**99.88**	5.35	76.69
GO-SLAM[2023]	2.51	76.93	5.11	65.10
Ours	**0.75**	98.87	**2.81**	**82.76**

Table 8. Comparison of geometry accuracy on Replica with more SLAM systems.

Sample ratio	PSNR↑	SSIM↑	LPIPS↓
5%	39.01	0.965	0.072
10%	**39.63**	0.970	0.051
20%	39.31	**0.972**	**0.044**

Table 9. Ablation study on sampled pixel number.

a small number of images for reconstruction, the image quality is not significantly affected.

6.2 Ablation study on stable/unstable Gaussians

We test the impact of our proposed stable/unstable Gaussians on time performance. We report the optimization time per iteration, for optimizing all Gaussians using the whole image, optimizing only unstable Gaussians using the whole image, and optimizing only unstable Gaussians using only the pixels covered by them. As seen in Table 10, our strategy greatly improves the optimization speed.

Scene	Storeroom	Hotel room	Home
$\mathcal{S}$	9.8ms	8.4ms	8.9ms
$\mathcal{S}_{unstable}$, all pixels	7.4ms	6.5ms	6.4ms
$\mathcal{S}_{unstable}$, unstable pixels	**5.2ms**	**4.7ms**	**4.3ms**

Table 10. Ablation study on stable/unstable Gaussians.

Method	Acc.↓	Acc. Ratio↑	Comp.↓	Comp. Ratio↑	ATE (cm)↓	Gaussian Number
Alpha-blending	2.48	70.54	3.32	75.01	1.24	468916
Ours	**0.75**	**98.87**	**2.81**	**82.76**	**0.18**	**431692**

Table 11. Ablation study on depth rendering.

6.3 Ablation study on depth rendering

Our depth blending is tightly coupled with our Gaussian adding and state management, so in the paper we show that alpha blending yields much more Gaussians through the comparison with SplaTAM which uses alpha blending (987524 vs 7155880). Here for better ablation study, we first use our depth blending to determine the adding of opaque/transparent Gaussians as well as the states, and then replace our depth blending with alpha blending for optimization. The results are listed in Table 11. We can see with similar Gaussian numbers, our depth blending outperforms alpha blending in terms of geometry accuracy and tracking accuracy.

6.4 Ablation study on confidence count threshold

The ablation study on confidence count threshold δ_η on Replica is shown in Table 12. As δ_η increases, the Gaussians will be in the unstable state for a longer time, resulting in a slower speed. On the other hand, if δ_η is small, the Gaussians may not be fully optimized, yielding reduced rendering quality.

δ_η	PSNR↑	FPS↑
50	33.63	**18.21**
100	**35.43**	17.31
200	35.12	16.59
400	35.37	15.49

Table 12. Ablation study on confidence count threshold.

Method	Replica	TUM-RGBD fr1_desk	TUM-RGBD fr2_xyz
ElasticFusion without backend	0.68	2.93	1.32
ElasticFusion	1.13	2.53	1.17
Ours without backend	0.22	5.39	1.63
Ours	**0.18**	**1.66**	**0.38**

Table 13. Ablation study on backend pose optimization in terms of tracking accuracy .

Method	Acc.↓	Acc. Ratio↑	Comp.↓	Comp. Ratio↑
ElasticFusion without backend	1.05	98.32	4.33	77.69
ElasticFusion	1.13	96.33	4.43	75.25
Ours without backend	0.95	97.57	**2.74**	**83.17**
Ours	**0.75**	**98.87**	2.81	82.76

Table 14. Ablation study on backend pose optimization in terms of geometry accuracy .

Color	PSNR↑	SSIM↑	LPIPS↓
RGB	33.90	0.951	0.113
SH	**35.43**	**0.982**	**0.109**

Table 15. Ablation study on SH coefficients in terms of training view synthesis.

6.5 Ablation study on backend pose optimization

Here we evaluate the effect of backend pose optimization. We test the tracking accuracy on Replica and TUM-RGBD, and report the results in Table 13. We also test the geometry accuracy on Replica and report the results in Table 14. On high-quality images on Replica, we achieve relatively high accuracy using only the frontend ICP. However, on low-quality images on TUM-RGBD, we rely more on the ORB backend, because the ICP may be performed on the Gaussians still under optimization.

6.6 Ablation study on SH coefficients

Here we report the rendering quality using SHs and RGB colors. We report the results of training view synthesis on Replica in Table 15. We also report the results of novel view synthesis on ScanNet++ in Table 16. We can notice that using SH coefficients has better rendering quality.

Color	PSNR↑	SSIM↑	LPIPS↓
RGB	25.67	0.855	0.304
SH	**27.27**	**0.874**	**0.291**

Table 16. Ablation study on SH coefficients in terms of novel view synthesis .

Frame ID	Ours	Ours without outlier pruning
1000#	**151947**	211390
2000#	**268625**	382736
3000#	**507987**	797242
4000#	**675818**	1073366

Table 17. Ablation study on outlier pruning.

6.7 Ablation study on outlier pruning

Here we evaluate the influence of our outlier pruning strategy. We report the number of Gaussians every 1000 frames in the Azure hotel room scene in Table 17 and the results show that the Gaussian number will increase significantly without outlier pruning.

REFERENCES

Sebastien Bonopera, Jerome Esnault, Siddhant Prakash, Simon Rodriguez, Theo Thonat, Mehdi Benadel, Gaurav Chaurasia, Julien Philip, and George Drettakis. 2020. sibr: A System for Image Based Rendering. https://gitlab.inria.fr/sibr/sibr_core

Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. 2017. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics 2017 (TOG)* (2017).

Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. 2023. ESLAM: Efficient Dense SLAM System Based on Hybrid Representation of Signed Distance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 17408–17419. https://doi.org/10.1109/CVPR52729.2023.01670

Nikhil Varma Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian A. Scherer, Deva Ramanan, and Jonathon Luiten. 2023. SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM. *CoRR* abs/2312.02126 (2023). https://doi.org/10.48550/ARXIV.2312.02126 arXiv:2312.02126

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (2023), 139:1–139:14. https://doi.org/10.1145/3592433

Raúl Mur-Artal and Juan D. Tardós. 2017. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics* 33, 5 (2017), 1255–1262. https://doi.org/10.1109/TRO.2017.2705103

Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. 2023. Point-SLAM: Dense Neural Point Cloud-based SLAM. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

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, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. 2019. The Replica Dataset: A Digital Replica of Indoor Spaces. *arXiv preprint arXiv:1906.05797* (2019).

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 573–580. https://doi.org/10.1109/IROS.2012.6385773

Hengyi Wang, Jingwen Wang, and Lourdes Agapito. 2023. Co-SLAM: Joint Coordinate and Sparse Parametric Encodings for Neural Real-Time SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 13293–13302. https://doi.org/10.1109/CVPR52729.2023.01277

Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. 2015. ElasticFusion: Dense SLAM Without A Pose Graph. In *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, Lydia E. Kavraki, David Hsu, and Jonas Buchli (Eds.). https://doi.org/10.15607/RSS.2015.XI.001

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. 2023. ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. 2023. GO-SLAM: Global Optimization for Consistent 3D Instant Reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. 2022. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 12776–12786. https://doi.org/10.1109/CVPR52688.2022.01245