SUPPLEMENTARY MATERIAL SPLAT-SLAM: GLOBALLY OPTIMIZED RGB-ONLY SLAM WITH 3D GAUSSIANS

Anonymous authors
Paper under double-blind review

This supplementary material accompanies the main paper and provides more details on the methodology and additional experimental results.

1 METHOD

We describe further details about our method that were left out from the main paper.

Comparison to Existing Works. To further clarify the differences between our method and existing 3DGS SLAM works, we classify each method in tab. 8 based on important characteristics. It shows that our work is the first to include loop closure, proxy depth, RGB-only and online 3D Gaussian deformations.

	RGB-only	Loop Closure	Proxy Depth	Online 3DGS Deformations
GS-SLAM Yan et al. (2023)	✗	✗	✓	✗
Gaussian-SLAM Yugay et al. (2023)	✗	✗	✓	✗
SplaTaM Keetha et al. (2023)	✗	✗	✓	✗
MonoGS Matsuki et al. (2023)	✓	✗	✗	✗
Photo-SLAM Huang et al. (2023)	✓	✓	✗	✗
Splat-SLAM (ours)	✓	✓	✓	✓

Table 8: **Method Classification.** We show that our method is the first to combine 3D Gaussian SLAM with loop closure, proxy depth and online 3D Gaussian map deformations in an RGB-only SLAM system.

Map Initialization. With map initialization, we refer to the process of anchoring new Gaussians during scene exploration. For every new keyframe to be mapped, we adopt the strategy that MonoGS Matsuki et al. (2023) uses in pure RGBD mode. It works by unprojecting the depth reading per pixel to 3D and then downsampling this point cloud by a factor θ. New Gaussians are then assigned their means as the point cloud. The rotations are initialized to identity, the opacity to 0.5 and the scales are initialized related to their distance to the nearest neighbor point in the point cloud.

Keyframe Selection and Local Windowing. As mentioned in the main paper, we adopt the keyframe selection strategy from MonoGS Matsuki et al. (2023). We describe this strategy in the following.

Keyframes are selected based on the covisibility of the Gaussians. Between two keyframes i and j, the covisibility is defined using the Intersection over Union (IOU) and Overlap Coefficient (OC):

$$\text{IOU}_{\text{cov}}(i,j) = \frac{|\mathcal{G}_v^i \cap \mathcal{G}_v^j|}{|\mathcal{G}_v^i \cup \mathcal{G}_v^j|}, \tag{16}$$

$$\text{OC}_{\text{cov}}(i,j) = \frac{|\mathcal{G}_v^i \cap \mathcal{G}_v^j|}{\min(|\mathcal{G}_v^i|, |\mathcal{G}_v^j|)}, \tag{17}$$

where $\mathcal{G}_v^i$ are the Gaussians visible in keyframe i, based on the following definition of visibility. A Gaussian is seen as visible from a camera pose if it is used in the rasterization pipeline when rendering and if the accumulated transmittance $\prod_{j=1}^{i-1}(1-\alpha_j)$ has not yet reached 0.5.

A keyframe i is added to the keyframe window KFs if, given the last keyframe j, $\text{IOU}_{\text{cov}}(i,j) < k_{\text{fcov}}$ or if the relative translation $t_{ij} > k_{\text{fm}}\hat{D}_i$, where $\hat{D}_i$ is the median depth of frame i. For Replica, $k_{\text{fcov}} = 0.95$, $k_{\text{fm}} = 0.04$ and for TUM and ScanNet, $k_{\text{fcov}} = 0.90$, $k_{\text{fm}} = 0.08$. The registered keyframe j in KFs is removed if $\text{OC}_{\text{cov}}(i,j) < k_{\text{fc}}$, where keyframe i is the latest added keyframe.

For all datasets, the cutoff is set to $k_{\text{fc}} = 0.3$. The size of the keyframe window is set to $|\text{KFs}| = 10$ for Replica and $|\text{KFs}| = 8$ for TUM and ScanNet.

Pruning and Densification We also follow Matsuki et al. (2023) when it comes to Gaussian pruning and densification. Pruning is done based on the visibility: if new Gaussians inserted within the last 3 keyframes are not visible by at least 3 other frames in the keyframe window KFs, they are removed. Visibility-based pruning is only done when the keyframe window KFs is full. Additionally, every 150 mapping iterations, Gaussians with opacity lower than 0.7 are removed globally. Also Gaussians which project in 2D with a too large scale are removed. Densification is done as in Kerbl et al. (2023), also at an interval of every 150 mapping iterations.

Final Refinement. Similar to GlORIE-SLAM Zhang et al. (2024), which performs a final refinement after the last final global BA at the end of the trajectory, we also perform a few refinement iterations after the last final global BA. Also MonoGS Matsuki et al. (2023) performs a set of final iterations at the end of the SLAM trajectory to refine the colors.

Our refinement strategy is straight forward. We disable pruning and densification of the Gaussians and perform a set of optimization iterations β using the same loss function as in the main paper, but only sampling random single frames per iteration.

Differences to GlORIE-SLAM. We briefly discuss some differences to GlORIE-SLAM Zhang et al. (2024) not covered in the main paper. GlORIE-SLAM uses an additional point cloud called P_d consisting of all inlier mullti-view depth maps unprojected into a point cloud. We found that this is not needed and it saves memory and compute to not use it. GlORIE-SLAM also re-anchors the neural points at the depth reading. We do not do this as the Gaussians do not necessarily lie on the surface exactly. Finally, GlORIE-SLAM requires input depth to guide the sampling of points to render color and depth. If the depth is noisy or if the map is used for tracking (*i.e.* frame-to-model tracking), the depth guiding strategy is not favorable as it leads to artifacts when sampling the wrong points (when noisy depth is encountered) and to a much smaller basin of convergence when tracking (because the rendering is conditioned on the current view point). With 3D Gaussians, we can avoid depth guidance during rendering.

2 More Experiments

To accompany the evaluations provided in the main paper, we provide further experiments in this section.

Implementation Details. As the point cloud downsampling factor, we use $\theta = 32$ for all frames but the first frame where $\theta = 16$ is used. We use $\beta = 2000$, the number of iterations for the final refinement optimization, on the Replica dataset and $\beta = 26000$ on the TUM-RGBD Sturm et al. (2012) and ScanNet Dai et al. (2017) datasets (same as MonoGS Matsuki et al. (2023)). We benchmark the runtime on an AMD Ryzen Threadripper Pro 3945WX 12-Cores with an NVIDIA GeForce RTX 3090 Ti with 24 GB of memory. For the remaining hyperparameters, we refer to MonoGS Matsuki et al. (2023) for the Gaussian mapping and GlORIE-SLAM for tracking Zhang et al. (2024).

A Note on Rendering and Runtime with MonoGS. By default, MonoGS Matsuki et al. (2023) does not evaluate the rendering error on the mapped keyframes nor implement the exposure compensation during rendering evaluation. To compare our results fairly to MonoGS, we implement these details and run the experiments with these settings enabled. Further, we report the runtime for MonoGS using a single process (same as us) compared to the reported number in the paper, which was using multiple processes at once.

A Note on Gaussian Deformation with Photo-SLAM. Though not fully clear from reading the paper, after discussing with the authors of Photo-SLAM Huang et al. (2023), we find that they do, in fact, not deform the Gaussians as a result of global BA or loop closure. They found this to be unstable in their experiments. This suggests that our deformation strategy is non-trivial.

Justification of Monocular Depth Estimator. There are already numerous monocular depth estimators, but most of them are limited by speed, memory or quality. We use Omnidata Eftekhar et al. (2021) since empirically we found it still provides the best trade-off between output performance and runtime. We also tested our system with Depth Anything Yang et al. (2024), but found that it was marginally worse in terms of the final reconstructed mesh accuracy.

2.1 Tracking on ScanNet and TUM-RGBD

We do not put the results on tracking for ScanNet and TUM-RGBD since we use the tracking framework from GlORIE-SLAM Zhang et al. (2024), but we provide the numbers here. Tab. 9 and tab. 10 show the tracking accuracy of the estimated trajectory on ScanNet Dai et al. (2017) and TUM-RGBD Sturm et al. (2012) respectively. Our method shows competitive results in every single scene and gives the best average value among the RGB and RGB-D methods.

Method	0000	0059	0106	0169	0181	0207	**Avg.-6**	0054	0233	**Avg.-8**
RGB-D Input										
NICE-SLAM Zhu et al. (2022)	12.0	14.0	7.9	10.9	13.4	6.2	10.7	20.9	9.0	11.8
Co-SLAM Wang et al. (2023)	7.1	11.1	9.4	**5.9**	11.8	7.1	8.7	-	-	-
ESLAM Mahdi Johari et al. (2022)	7.3	8.5	7.5	6.5	9.0	**5.7**	**7.4**	36.3	**4.3**	10.6
MonoGSMatsuki et al. (2023)	16.1	**6.4**	8.1	8.7	26.4	9.2	12.5	20.6	13.1	13.6
RGB Input										
MonoGSMatsuki et al. (2023)	149.2	96.8	155.5	140.3	92.6	101.9	122.7	206.4	89.1	129.0
GO-SLAM Zhang et al. (2023b)	5.9	8.3	8.1	8.4	8.3	6.9	7.7	13.3	5.3	8.1
HI-SLAMZhang et al. (2023a)	6.4	7.2	**6.5**	8.5	**7.6**	8.4	**7.4**	-	-	-
Q-SLAM* Peng et al. (2024)	5.8	8.5	8.4	8.7	8.8	-	-	12.6	5.3	-
GlORIE-SLAM* Zhang et al. (2024)	**5.5**	9.1	7.0	8.2	8.3	7.5	7.6	**9.4**	5.1	**7.5**
Ours	**5.5**	9.1	7.0	8.2	8.3	7.5	7.6	**9.4**	5.1	**7.5**

Table 9: **Tracking Accuracy ATE RMSE [cm] ↓ on ScanNet Dai et al. (2017).** Our method equals to GlORIE-SLAM Zhang et al. (2024), giving the average lowest trajectory error. Results for the RGB-D methods are from Liso et al. (2024). Note that all methods with a * are concurrent works.

Method	f1/desk	f2/xyz	f3/off	**Avg.-3**	f1/desk2	f1/room	**Avg.-5**
RGB-D Input							
SplaTAM Keetha et al. (2023)	3.4	1.2	5.2	3.3	6.5	11.1	5.5
GS-SLAM* Yan et al. (2023)	1.5	1.6	1.7	1.6	-	-	-
GO-SLAM Zhang et al. (2023b)	1.5	0.6	1.3	**1.1**	-	4.7	-
MonoGS Matsuki et al. (2023)	1.4	1.4	1.5	1.5	5.1	6.3	3.1
RGB Input							
MonoGS Matsuki et al. (2023)	3.8	5.2	2.9	4.0	75.7	76.6	32.8
Photo-SLAM Huang et al. (2023)	1.5	1.0	1.3	1.3	-	-	-
DIM-SLAM Li et al. (2023)	2.0	0.6	2.3	1.6	-	-	-
GO-SLAM Zhang et al. (2023b)	1.6	0.6	1.5	1.2	2.8	5.2	2.3
MoD-SLAM* Zhou et al. (2024)	1.5	0.7	**1.1**	**1.1**	-	-	-
Q-SLAM* Peng et al. (2024)	**1.3**	0.9	-	-	**2.3**	4.9	-
GlORIE-SLAM* Zhang et al. (2024)	1.6	**0.2**	1.4	**1.1**	2.8	**4.2**	**2.1**
Ours	1.6	**0.2**	1.4	**1.1**	2.8	**4.2**	**2.1**

Table 10: **Tracking Accuracy ATE RMSE [cm] ↓ on TUM-RGBD Sturm et al. (2012).** Our method equals to GlORIE-SLAM Zhang et al. (2024), giving the average lowest trajectory error. Note that all methods with a * are concurrent works.

2.2 Full Evaluations Data

In tab. 11, tab. 12 and tab. 13, we provide the full per scene results on all commonly reported metrics on Replica Straub et al. (2019), TUM-RGBD Sturm et al. (2012) and ScanNet Dai et al. (2017).

The reconstruction results are only measured on Replica since the other two datasets are real world datasets which lack quality ground truth meshes.

We show the trajectory accuracy measurement of both keyframes and the full trajectory, which is obtained by first linear interpolation between keyframes and using optical flow to refine. The accuracy of these two trajectories are similar. In the main paper, the data we report is always measured on the full trajectory.

2.3 Influence of Monocular Depth

While we show that the monocular depth improves the geometric estimation capability of our framework, it may still be erroneous. To better understand the accuracy of the monocular depth, we

		Metric	R-0	R-1	R-2	O-0	O-1	O-2	O-3	O-4	Avg.
Reconstruction		Render Depth L1 ↓	2.90	2.16	2.18	2.44	1.97	2.46	2.62	2.53	**2.41**
		Accuracy ↓	1.99	1.91	2.06	3.96	2.03	3.45	2.15	1.89	**2.43**
		Completion ↓	3.78	3.38	3.34	2.75	3.33	4.36	3.96	4.25	**3.64**
		Comp. Rat. ↑	85.47	86.88	86.12	87.32	85.17	81.37	82.25	82.95	**84.69**
Rendering	Keyframes	PSNR ↑	32.25	34.31	35.95	40.81	40.64	35.19	35.03	37.40	**36.45**
		SSIM ↑	0.91	0.93	0.95	0.98	0.97	0.96	0.95	0.98	**0.95**
		LPIPS ↓	0.10	0.09	0.06	0.05	0.05	0.07	0.06	0.04	**0.06**
Tracking	Keyframes Trajectory	ATE RMSE ↓	0.29	0.38	0.24	0.27	0.35	0.34	0.42	0.43	**0.34**
	Full Trajectory	ATE RMSE ↓	0.29	0.33	0.25	0.29	0.35	0.34	0.42	0.43	**0.34**
Number of Gaussians	1000x		116	116	91	76	66	134	114	106	**102**

Table 11: **Full Evaluation on Replica Straub et al. (2019).** We show the ATE RMSE [cm] evaluation on the keyframes as well as on the full trajectory.

		Metric	f1/desk	f1/desk2	f1/room	f2/xyz	f3/office	Avg.
Rendering	Keyframes	PSNR ↑	25.61	23.98	24.06	29.53	26.05	**25.85**
		SSIM ↑	0.84	0.81	0.80	0.90	0.84	**0.84**
		LPIPS ↓	0.18	0.23	0.24	0.08	0.20	**0.19**
Depth Rendering	Keyframes	Depth L1↓ [cm]	8.05	15.70	15.05	14.53	25.59	**15.78**
Tracking	Key Frames Trajectory	ATE RMSE ↓	1.92	3.05	4.43	0.23	1.41	**2.21**
	Full Trajectory	ATE RMSE ↓	1.65	2.79	4.16	0.22	1.44	**2.05**
Number of Gaussians	1000x		88	78	211	173	114	**133**

Table 12: **Full Evaluation on TUM-RGBD Sturm et al. (2012).**

		Metric	0000	0054	0059	0106	0169	0181	0207	0233	Avg.
Rendering	Keyframes	PSNR↑	28.68	30.21	27.69	27.70	31.14	31.15	30.49	27.48	**29.32**
		SSIM ↑	0.83	0.85	0.87	0.86	0.87	0.84	0.84	0.78	**0.84**
		LPIPS ↓	0.19	0.22	0.15	0.18	0.15	0.23	0.19	0.22	**0.19**
Depth Rendering	Keyframes	Depth L1↓ [cm]	8.24	18.24	13.39	23.5	11.49	18.35	13.78	10.19	**11.37**
Tracking	Key Frames Trajectory	ATE RMSE ↓	5.66	9.17	9.48	7.03	8.72	8.42	7.47	4.97	**7.61**
	Full Trajectory	ATE RMSE ↓	5.57	9.50	9.11	7.09	8.26	8.39	7.53	5.17	**7.58**
Number of Gaussians	1000x		144	157	84	108	52	127	121	191	**123**

Table 13: **Full Evaluation on ScanNet Dai et al. (2017).**

replace it with the ground truth sensor depth instead. This experiment acts as the upper bound of our method if the monocular depth is perfect. The experiments are done on Replica Straub et al. (2019) and are shown in tab. 14. Compared with the standard setting with the monocular depth, the ground truth depth setting gives improvements on both reconstruction and rendering quality, which reveals that our method still has potential to achieve better mapping results once better monocular depth is available. Since our method does not require further training or fine-tuning for the monocular depth, it is quite easy to just replace the current off-the-shelf monocular depth estimator with a better one.

2.4 Impact of Deformation

During runtime, we deform the 3D Gaussian map to account for adjustments to poses and depth that have already been integrated into the existing map. An alternative to performing the deformation is to solely rely on optimization to resolve the new map. We conduct two experiments to show the benefit of performing the deformation, especially when it comes to rendering accuracy. In tab. 15, we vary the number of final refinement iterations and evaluate the rendering depth L1 and PSNR on the Replica `office 0` scene. We find that utilizing online 3D Gaussian deformations yields better rendering and depth L1 accuracy regardless of the number of iterations. In tab. 16 we conduct the same experiment, but over a set of scenes on ScanNet. We find that on average, by enabling the

	Metric	R-0	R-1	R-2	O-0	O-1	O-2	O-3	O-4	Avg.
Recon-struction	Render Depth L1 ↓	2.38	1.31	1.73	1.15	1.60	1.29	5.71	1.93	**2.14**
	Accuracy ↓	1.29	0.91	1.05	1.22	0.83	0.96	1.24	1.07	**1.07**
	Completion ↓	3.43	2.83	2.66	1.50	2.46	3.57	3.46	3.61	**2.94**
	Comp. Rat. ↑	86.61	88.69	88.70	93.44	89.09	85.20	84.60	85.32	**87.71**
Rendering	PSNR ↑	35.66	37.65	38.87	43.95	43.28	37.93	37.41	39.88	**39.33**
	SSIM ↑	0.96	0.96	0.97	0.99	0.98	0.96	0.96	0.98	**0.97**
	LPIPS ↓	0.04	0.05	0.03	0.02	0.02	0.06	0.04	0.03	**0.04**
Tracking	ATE RMSE ↓	0.29	0.38	0.24	0.28	0.39	0.35	0.45	0.40	**0.35**

Table 14: **Full Evaluations on Replica Straub et al. (2019) with ground truth depth.** Both reconstruction and rendering results improve significantly with the ground truth depth, suggesting that our method is bounded by the quality of current day monocular depth estimation. Since we do not require any extra training or fine-tuning of the monocular depth estimator, it is easy to plug in a better estimator once available. Tracking performance does not change much.

deformation, we achieve higher rendering accuracy and lower depth L1 error. The improvement is, however, more significant when it comes to the rendering accuracy.

Nbr of Final Iterations β		Metric	0K	0.5K	1K	2K
Reconstruction	W/O Deform	Render Depth L1 ↓	8.84	3.49	2.64	2.6
	W Deform	Render Depth L1 ↓	**6.55**	**2.37**	**2.34**	**2.40**
Rendering	W/O Deform	PSNR ↑	22.86	34.30	37.66	37.86
	W Deform	PSNR ↑	**30.50**	**39.87**	**40.59**	**41.20**

Table 15: **Gaussian Deformation Ablation on Replica Straub et al. (2019) `office 0`.**

		Metric	0000	0054	0059	0106	0169	0181	0207	Avg.
Rendering	W/O Deform	PSNR↑	25.15	28.39	**27.77**	25.25	29.41	30.38	29.30	27.95
	W Deform	PSNR↑	**28.68**	**30.21**	27.69	**27.70**	**31.14**	**31.15**	**30.49**	**29.58**
Depth Rendering	W/O Deform	L1↓ [cm]	**7.86**	22.81	**10.51**	24.19	11.54	18.48	**13.66**	15.58
	W Deform	L1↓ [cm]	8.24	**18.24**	13.39	**23.5**	**11.49**	**18.35**	13.78	**15.28**

Table 16: **Gaussian Deformation Ablation on ScanNet Dai et al. (2017).**

2.5 Final Refinement Iterations

After the final global BA step, we perform a final refinement, similar to MonoGSMatsuki et al. (2023), but include the geometric depth loss as well and do not only refine with a color loss. We ablate the influence on the results by varying the number of iterations of the final refinement in tab. 17. We find that the rendering accuracy increases monotonically with the number of iterations while the geometric accuracy decreases with more than 2K iterations. We believe this to be a result of fitting to the noisy monocular depth. We choose to use 2K iterations since this provides the best trade-off between rendering and geometric accuracy. 2K iterations takes around 15 seconds on our benchmark hardware which consists of an AMD Ryzen Threadripper Pro 3945WX 12-Cores with an NVIDIA GeForce RTX 3090 Ti with 24 GB of memory.

Nbr of Final Iterations β		Metric	2K	5K	10K	26K
Reconstruction		Render Depth L1 ↓	**2.36**	2.45	2.51	2.59
		Accuracy ↓	**2.46**	2.66	2.84	3.02
		Completion ↓	3.60	**3.61**	3.59	3.60
		Comp. Rat. ↑	**84.87**	84.71	84.80	84.77
Rendering	Keyframes	PSNR ↑	36.77	37.80	38.41	**38.95**

Table 17: **Final Refinement Iterations Ablation on Replica Straub et al. (2019).** The results are averaged over the 8 scenes.

2.6 Impact of Downsampling Factor

During mapping, the point cloud formed from unprojecting the depth input is downsampled to avoid adding redundant Gaussians to the scene representation. We investigate the impact of using stronger versus weaker downsampling in tab. 18 where we also compare to the sensitivity of MonoGSMatsuki et al. (2023) with respect to the same parameter. Tab. 18 shows that both systems are not very sensitive to the model compression as a result of a larger downsampling factor θ. When both systems use the same number of Gaussians on average ($\theta = 32$ for MonoGS and $\theta = 64$ for our method), we find that our method performs significantly better in terms of depth rerendering and photometric accuracy. For all results in the main paper, we use $\theta = 32$.

Downsampling Factor θ		Metric	16	32	64
Reconstruction	Ours	Render Depth L1 $\downarrow$	2.38	2.40	2.46
	MonoGS Matsuki et al. (2023)		33.43	28.47	28.09
Rendering	Ours	PSNR $\uparrow$	36.63	36.45	36.31
	MonoGS Matsuki et al. (2023)		31.17	30.87	29.64
Number of Gaussians	Ours	1000x$\downarrow$	141	102	83
	MonoGS Matsuki et al. (2023)		97	83	73

Table 18: **Downsampling Factor θ Ablation on Replica Straub et al. (2019).** The results are averaged over the 8 scenes.

2.7 Runtime Evaluation

To be consistent with the keyframe selection hyperparameters of MonoGS Matsuki et al. (2023), we report on the same parameters as MonoGS uses by default. In practice, this means that few keyframes from the tracking system (determined via mean optical flow thresholding) are actually filtered out and not mapped. In tab. 19, we show that by altering the hyperparamters, we can speed up the system during runtime, while still rendering and reconstructing the scene well. Note that we evaluate the rendering performance on the same set of views for all runs. We benchmark the runtime on an AMD Ryzen Threadripper Pro 3945WX 12-Cores with an NVIDIA GeForce RTX 3090 Ti with 24 GB of memory. We note that we currently do not leverage multiprocessing to the amount possible in practice *i.e.* currently we first do tracking and then mapping *i.e.* there is no simultaneous tracking and mapping. This is, however, straightforward to include, which should further speed up the runtime.

k_{fcov}, k_{fm}		0.95, 0.04	0.90, 0.08	0.85, 0.08	0.80, 0.12	0.70, 0.16	0.60, 0.20	0.50, 0.30
Reconstruction	Render Depth L1 $\downarrow$	**2.90**	2.94	2.97	3.08	3.37	3.53	4.78
	Accuracy $\downarrow$	1.99	**1.94**	2.06	2.04	2.54	3.20	6.20
	Completion $\downarrow$	3.78	**3.76**	3.79	3.77	3.86	3.93	5.23
	Comp. Rat. $\uparrow$	85.47	**85.58**	85.39	85.53	85.03	84.33	80.38
Rendering	PSNR $\uparrow$	**32.25**	31.65	31.31	30.59	30.12	29.25	27.59
Runtime	FPS $\uparrow$	1.24	1.45	1.62	2.02	2.50	3.03	**3.67**

Table 19: **Keyframe Hyperparameter Search on Replica Straub et al. (2019) `room 0`.** By changing the keyframe selection hyperparameters, we can speed up our runtime without impacting reconstruction and rendering too much. We evaluate the rendering performance on the same set of frames for all runs. In comparison, with the default $k_{\text{fcov}} = 0.95$, $k_{\text{fm}} = 0.04$, MonoGS Matsuki et al. (2023) yields PSNR: 26.12 and render depth L1: 17.38 cm.

2.8 Additional Qualitative Reconstructions

In fig. 6 we show additional qualitative results from the Replica dataset on normal shaded meshes.

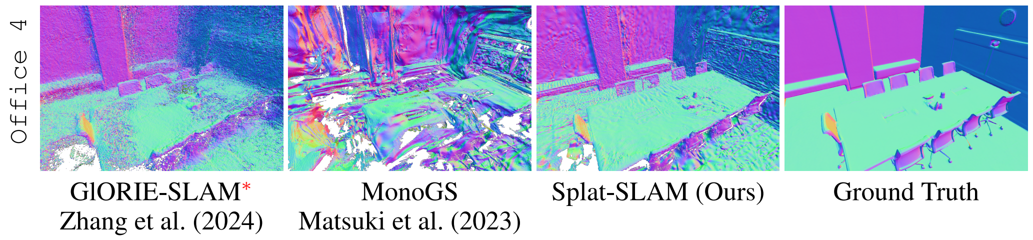

Figure 6: **Reconstruction Results on Replica Straub et al. (2019).** Our method improves upon the geometric accuracy compared to existing works, when observing the normal shaded meshes. In particular, GlORIE-SLAM suffers from floating point artifacts. MonoGS suffers badly from a lack of proxy depth, despite multiview optimization.

REFERENCES

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE/CVF, 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.261. URL `http://arxiv.org/abs/1702.04405`.

Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10786–10796, 2021.

Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. *arXiv preprint arXiv:2311.16728*, 2023.

Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track and map 3d gaussians for dense rgb-d slam. *arXiv preprint*, 2023.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

Heng Li, Xiaodong Gu, Weihao Yuan, Luwei Yang, Zilong Dong, and Ping Tan. Dense rgb slam with neural implicit maps. In *Proceedings of the International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=QUK1ExlbbA`.

Lorenzo Liso, Erik Sandström, Vladimir Yugay, Luc Van Gool, and Martin R Oswald. Loopy-slam: Dense neural slam with loop closures. *arXiv preprint arXiv:2402.09944*, 2024.

Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. *arXiv e-prints*, pp. arXiv–2211, 2022.

Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *arXiv preprint arXiv:2312.06741*, 2023.

Chensheng Peng, Chenfeng Xu, Yue Wang, Mingyu Ding, Heng Yang, Masayoshi Tomizuka, Kurt Keutzer, Marco Pavone, and Wei Zhan. Q-slam: Quadric representations for monocular slam. *arXiv preprint arXiv:2403.08125*, 2024.

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2012. ISBN 978-1-4673-1736-8. doi: 10.1109/IROS.2012.6385773. URL `http://ieeexplore.ieee.org/document/6385773/`.

Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13293–13302, June 2023.

Chi Yan, Delin Qu, Dong Wang, Dan Xu, Zhigang Wang, Bin Zhao, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. *arXiv preprint arXiv:2311.11700*, 2023.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv preprint arXiv:2401.10891*, 2024.

Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting, 2023.

Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R Oswald. Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam. *arXiv preprint arXiv:2403.19549*, 2024.

Wei Zhang, Tiecheng Sun, Sen Wang, Qing Cheng, and Norbert Haala. Hi-slam: Monocular real-time dense mapping with hybrid implicit fields. *IEEE Robotics and Automation Letters*, 2023a.

Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3727–3737, 2023b.

Heng Zhou, Zhetao Guo, Shuhong Liu, Lechen Zhang, Qihao Wang, Yuxiang Ren, and Mingrui Li. Mod-slam: Monocular dense mapping for unbounded 3d scene reconstruction, 2024.

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/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12786–12796, 2022.