Gaussian Splatting SLAM

Supplementary Material

6. Implementation Details

6.1. System Details and Hyperparameters

6.1.1 Tracking and Mapping (Sec. 3.3.1 and 3.3.3)

Learning Rates We use the Adam optimiser for both camera poses and Gaussian parameters optimisation. For camera poses, we used 0.003 for rotation and 0.001 for translation. For 3D Gaussians, we used the default learning parameters of the original Gaussian Splatting implementation [11], apart from in monocular setting where we increase the learning rate of the positions of the Gaussians $\boldsymbol{\mu}_W$ by a factor of 10.

Iteration numbers 100 tracking iterations are performed per frame for across all experiments. However, we terminate the iterations early if the magnitude of the pose update becomes less than 10^{-4}. For mapping, 150 iterations are used for the single-process implementation.

Loss Weights Given a depth observation, for tracking we minimise both photometric Eq. (7) and geometric residual Eq. (8) as:

$$\min_{\boldsymbol{T}_{CW} \in \boldsymbol{SE}(3)} \lambda_{pho} E_{pho} + (1 - \lambda_{pho}) E_{geo} \,, \tag{12}$$

and similarly, for mapping we modify Eq. (11) to:

$$\min_{\substack{\boldsymbol{T}^k_{CW} \in \boldsymbol{SE}(3), \mathcal{G}, \\ \forall k \in \mathcal{W}}} \sum_{\forall k \in \mathcal{W}} (\lambda_{pho} E^k_{pho} + (1 - \lambda_{pho}) E^k_{geo}) + \lambda_{iso} E_{iso} \,. \tag{13}$$

We set $\lambda_{pho} = 0.9$ for all RGB-D experiments, and $\lambda_{iso} = 10$ for both monocular and RGB-D experiments.

6.1.2 Keyframing (Sec. 3.3.2)

Gaussian Covisibility Check (Sec. 3.3.2) As described in Sec. 3.3.2, keyframe selection is based on the covisibility of the Gaussians. Between two keyframes i, j, we define the covisibility using the Intersection of Union (IOU) and Overlap Coefficient (OC):

$$IOU_{cov}(i,j) = \frac{|\mathcal{G}^v_i \cap \mathcal{G}^v_j|}{|\mathcal{G}^v_i \cup \mathcal{G}^v_j|} \,, \tag{14}$$

$$OC_{cov}(i,j) = \frac{|\mathcal{G}^v_i \cap \mathcal{G}^v_j|}{\min(|\mathcal{G}^v_i|, |\mathcal{G}^v_j|)} \,, \tag{15}$$

where $\mathcal{G}^v_i$ is the Gaussians visible in keyframe i, based on visibility check described in Section 3.3.2, Gaussian Covisibility. A keyframe i is added to the keyframe window $\mathcal{W}_k$ if given last keyframe j, $IOU_{cov}(i,j) < kf_{cov}$ or if the relative translation $t_{ij} > kf_m \hat{D}_i$, where $\hat{D}_i$ is the median depth of frame i. For Replica $kf_{cov} = 0.95, kf_m = 0.04$ and for TUM $kf_{cov} = 0.90, kf_m = 0.08$. We remove the registered keyframe j in $\mathcal{W}_k$ if the $OC_{cov}(i,j) < kf_c$, where keyframe i is the latest added keyframe. For both Replica and TUM, we set the cutoff to $kf_c = 0.3$. We set the size of the keyframe window to be for Replica, $|\mathcal{W}_k| = 10$, and for TUM, $|\mathcal{W}_k| = 8$.

Gaussian Insertion and Pruning (Sec. 3.3.2) As we optimise the positions of Gaussians and prune geometrically unstable Gaussians, we do not require any strong prior such as depth observation for Gaussian initialisation. When **inserting** new Gaussians in a monocular setting, we randomly sample the Gaussians position $\boldsymbol{\mu}_W$ using rendered depth D. Since the estimated depth may sometimes be incorrect, we account for this by initialising the Gaussians with some variance. For a pixel p where the rendered depth $\mathcal{D}_p$ exists, we sample the depth from $\mathcal{N}(\mathcal{D}_p, 0.2\sigma_D)$. Otherwise, for unobserved regions, we initialise the Gaussians by sampling from $\mathcal{N}(\hat{D}, 0.5\sigma_D)$, where $\hat{D}$ is the median of D. For **pruning**, as described in Section 3.3.2, we perform visibility-based pruning, where if new Gaussians inserted within the last 3 keyframes are not observed by at least 3 other frames, they are pruned. We only perform visibility-based pruning once the keyframe window $\mathcal{W}_k$ is full. Additionally, we prune all Gaussians with opacity of less than 0.7.

7. Evaluation details

7.1. Camera Tracking Accuracy (Table 1 and Table 2)

7.1.1 Evaluation Metric

We measured the keyframe absolute trajectory error (ATE) RMSE. For monocular evaluation, we perform scale alignment between the estimated scale-free and ground-truth trajectories. For RGB-D evaluation, we only align the estimated trajectory and ground truth without scale adjustment.

7.1.2 Baseline Results

Table 1 Numbers for monocular DROID-SLAM [38] and ORB-SLAM [21] is taken from [14]. We have lo-

cally run DSO [5], DepthCov [4] and DROID-VO [38] – which is DROID-SLAM without loop closure and global bundle adjustment. For the RGB-D case, numbers for NICE-SLAM [48], DI-Fusion [8], Vox-Fusion [45], Point-SLAM [29] are taken from Point-SLAM [29], and numbers for iMAP [35], BAD-SLAM [31], Kintinous [42], ORB-SLAM [21] are from iMAP [35], and ald all the other baselines: ESLAM [9], Co-SLAM [41] are from each individual papers.

Table 2 and 5 We took the numbers from Point-SLAM [29] paper.

Table 4 The numbers are from Co-SLAM [41] paper.

7.2. Rendering Performance (Table 5)

We provide the full detail of the rendering performance evaluation in Table 7.

In Table 5, we reported the photometric quality metrics (PSNR, SSIM and LPIPS) and rendering fps of our methods. We demonstrated that our rendering fps (769) is much higher than other existing methods (VoxFusion is the second best with 2.17fps). Here we describe the detail of how we measured the fps. The rendering time refers to the duration necessary for full-resolution rendering (1200×680 for the Replica sequence). For each method, we perform 100 renderings and report the average time taken per rendering. The reported rendering fps is found by taking 1 and dividing it by the average rendering time. We summarise the numbers in Table 8. Note that the "rendering fps" means the fps just for the forward rendering, which differs from the end-to-end system fps reported in Table 9 and 10.

7.3. The convergence basin analysis (Table 6 and Fig 5)

7.3.1 The detail of the benchmark Dataset

For convergence basin analysis, we create three datasets by rendering the synthetic Replica dataset. In addition to the qualitative visualisation in Figure 5, we report more detailed camera pose distributions in Figure 8. Figure 8 shows the camera view frustums of the test (red), training (yellow) and target (blue) views. As we mentioned in the main paper, we set the training view in the shape of a square with a width of 0.5m and test views are distributed with radii ranging from 0.2m to 1.2m, covering a larger area than the training views. We only apply displacements to the camera translation but not to the rotation. For each sequence, we use a total of 67 test views.

7.3.2 Training setup

For each method, the 3D representation is trained for 30000 iterations using the training views. Here, we detail the training setup of each of the methods:

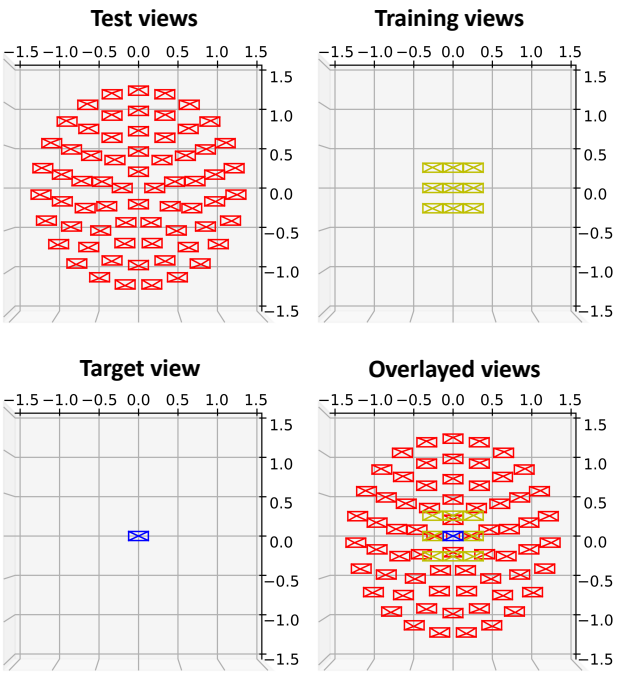


Figure 8. **2D Visualisation of the camera pose distributions used for convergence basin analysis in Figure 5.**

Ours We evaluated our method under two settings: "w/ depth" and "w/o depth", where we train the initial 3D Gaussian map $\mathcal{G}_{init}$ with and without depth supervision. In the "w/o depth" setting, the 3D Gaussians' positions are randomly initialised, and we minimise the monocular mapping cost Eq. (11) for the 3D Gaussian training, but keeping the camera poses fixed. Specifically, let $k \in \mathbb{N}$ be a number of training views and 3D Gaussians $\mathcal{G}$, we find $\mathcal{G}_{init}$ by:

$$\mathcal{G}_{init} = \arg\min_{\mathcal{G}} \sum_{\forall k \in \mathcal{W}} E^k_{pho} + \lambda_{iso} E_{iso} \, . \tag{16}$$

Note that training views' camera poses $\boldsymbol{T}^k_{CW}$ are fixed during the optimisation.

In the "w/ depth" setting, we train the Gaussian map by minimising the same cost function as our RGB-D SLAM system:

$$\mathcal{G}_{init} = \arg\min_{\mathcal{G}} \sum_{\forall k \in \mathcal{W}} (\lambda_{pho} E^k_{pho} + (1 - \lambda_{pho}) E^k_{geo}) + \lambda_{iso} E_{iso} \, , \tag{17}$$

where we use $\lambda_{pho} = 0.9$ and $\lambda_{iso} = 10$ for all the experiments

Baseline Methods For Hash Grid SDF, we trained the same network architecture as Co-SLAM [41]. For MLP

Method	Metric	room0	room1	room2	office0	office1	office2	office3	office4	Avg.	Rendering FPS
NICE-SLAM [48]	PSNR[dB] ↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42	0.54
	SSIM ↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809	
	LPIPS↓	0.33	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233	
Vox-Fusion [45]	PSNR[dB] ↑	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21	24.41	2.17
	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	
Point-SLAM [29]	PSNR[dB] ↑	32.40	34.08	35.5	38.26	39.16	33.99	33.48	33.49	35.17	1.33
	SSIM ↑	**0.974**	**0.977**	**0.982**	**0.983**	**0.986**	0.96	0.960	**0.979**	**0.975**	
	LPIPS↓	0.113	0.116	0.111	0.1	0.118	0.156	0.132	0.142	0.124	
Ours	PSNR[dB] ↑	**34.83**	**36.43**	**37.49**	**39.95**	**42.09**	**36.24**	**36.7**	**36.07**	**37.50**	**769**
	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**	

Table 7. **Rendering performance comparison of RGB-D SLAM methods on Replica.** Our method outperforms most of the rendering metrics compared to existing methods. Note that Point-SLAM uses sensor depth (ground-truth depth in Replica) to guide sampling along rays, which limits the rendering performance to existing views. The numbers for the baselines are taken from [29].

Method	Rendering FPS ↑	Rendering time per image [s] ↓
NICE-SLAM [48]	0.54	1.85
Vox-Fusion [45]	2.17	0.46
Point-SLAM [29]	1.33	0.75
Ours	**769**	**0.0013**

Table 8. **Further detail of Rendering FPS and Rendering Time comparison based on Table 5**.

SDF, we trained the network of iMAP [35]. For both baselines, we supervised networks with the same loss functions as Co-SLAM, which are colour rendering loss L_{rgb}, depth rendering loss L_{depth}, SDF loss L_{fs}, free-space loss L_{fs}, and smoothness loss L_{smooth}. Please refer to the original Co-SLAM paper for the exact formulation (equation (6) - (9)). All the training hyperparameters (e.g. learning rate of the network, number of sampling points, loss weight) are the same as Co-SLAM's default configuration of the Replica dataset. While Co-SLAM stores training view information by downsampling the colour and depth images, we store the full pixel information because the number of training views is small.

7.3.3 Testing Setup

For testing, we localise the camera pose by minimising only the photometric error against the ground-truth colour image of the target view.

Ours Let the camera pose $\boldsymbol{T}_{CW} \in \boldsymbol{SE}(3)$ and initial 3D Gaussians $\mathcal{G}_{init}$, the localised camera pose $\boldsymbol{T}_{CW}^{est}$ is found by:

$$\boldsymbol{T}_{CW}^{est} = \underset{\boldsymbol{T}_{CW}}{\arg\min} \left\| I(\mathcal{G}_{init}, \boldsymbol{T}_{CW}) - \bar{I}_{target} \right\|_1 . \quad (18)$$

Note that $\mathcal{G}_{init}$ is fixed during the optimisation. We initialise $\boldsymbol{T}_{CW}$ at one of the test view's positions, and optimisation is performed for 1000 iterations. We perform this localisation process for all the test views and measure the success rate. Camera localisation is successful if the estimated pose converges to within 1cm of the target view within the 1000 iterations.

Method	Total Time [s]	FPS
Monocular	798.9	3.2
RGB-D	986.7	2.5

Table 9. **Performance Analysis using fr3/office.** Both monocular and RGB-D implementations use multiprocessing. We report **the total execution time of our system**, FPS computed by dividing the total number of processed frames by the total time.

Method	Total Time [s]	FPS
RGB-D	1111.1	1.8
RGB-D (sp)	1904.7	1.1

Table 10. **Performance Analysis using replica/office1.** RGB-D uses a multi-process implementation and RGB-D-sp is the single-process implementation. We report **the total execution time of our system**, FPS computed by dividing the total number of processed frames by the total time.

Baseline Methods For the baseline methods, the camera localisation is performed by minimising colour volume rendering loss L_{rgb}, while all the other trainable network parameters are fixed. The learning rates of the pose optimiser are also the same as Co-SLAM's default configuration of Replica dataset.

8. Further Ablation Analysis (Table 3)

8.1. Pruning Ablation (Monocular input)

In Table 8.1, we report the ablation study of our proposed Gaussian pruning, which prunes randomly initialised 3D

Input	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Mono	w/o pruning	78.2	4.5	57.0	46.6
	Ours	**3.78**	**4.60**	**3.50**	**3.96**

Table 11. **Pruning Ablation Study on TUM RGB-D dataset (Monocular Input).** Numbers are camera tracking error (ATE RMSE) in cm.

Input	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
RGB-D	w/o E_{iso}	1.60	**1.42**	**1.32**	**1.43**
	Ours	**1.50**	1.44	1.49	1.47

Table 12. **Isotropic Loss Ablation Study on TUM RGB-D dataset (RGB-D input).** Numbers are camera tracking error (ATE RMSE) in cm.

Method	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
w/o E_{iso}	**0.44**	0.86	**0.28**	0.75	0.99	0.36	0.28	2.6	0.82
Ours	**0.44**	**0.32**	0.31	**0.44**	**0.52**	**0.23**	**0.17**	**2.25**	**0.58**

Table 13. **Isotropic Loss Ablation Study on Replica dataset (RGB-D input).** Numbers are camera tracking error (ATE RMSE) in cm.

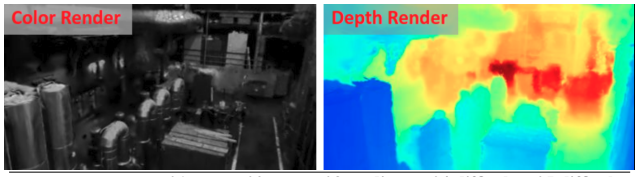
Gaussians effectively in a monocular SLAM setting. As the result shows, Gaussian pruning plays a significant role in enhancing camera tracking performance. This improvement is primarily because, without pruning, randomly initialised Gaussians persist in the 3D space, potentially leading to incorrect initial geometry for other views.

8.2. Isotropic Loss Ablation (RGB-D input)

Table 12 and 13 report the ablation study of the effect of isotropic loss E_{iso} for RGB-D input. In TUM, as Table 12 shows, isotropic regularisation does not improve the performance but only shows a marginal difference. However, for Replica, as summarised in Table 13, isotropic loss significantly improves camera tracking performance. Even with the depth measurement, since rasterisation does not consider the elongation along the viewing axis. Isotropic regularisation is required to prevent the Gaussians from over-stretching, especially for textureless regions, which are common in Replica.

8.3. Effect of Spherical Harmonics (SH)

While we disabled SHs in the main paper for simplicity, here we report the ablation study of the effect of SHs. The 3DGS paper [11] shows that addition of SH leads to small improvements in rendering metrics, and we have found similar improvement with SH enabled in our system (Tab.15a). We did not observe a significant change in runtime with SH enabled, but it notably increases Gaussian map size and hence GPU memory usage. Though an analytical Jacobian propagates the gradients from SH to camera poses, ATE marginally gets worse when SH is enabled (Tab. 16), as SH may incorrectly explain non-view directional effects caused by the camera motion, degrading the trajectory estimate.



	01-easy	02-easy	03-medium	04-difficult	05-difficult
Point-SLAM [29]	-	-	-	-	-
Ours	0.121	0.141	2.197	4.515	3.190
Vins-Fusion [28]	0.540	0.460	0.330	0.780	0.500
SVO [6]	0.040	0.070	0.270	0.170	0.120
ORB-SLAM3 [1]	**0.029**	**0.019**	**0.024**	**0.085**	**0.052**

Table 14. ATE RMSE (meter) on EuRoC Machine Hall with Stereo Depth. Baseline numbers are from [1]. The third best result is highlighted with a dash line.

8.4. Mapping Performance with ORB-SLAM

One of the most straightforward approaches for real-time operation is to combine an existing tracking system and 3DGS. In particular, frame-based SLAM methods have been well-studied for years and is capable of providing reliable tracking. In this section, we compare our unified 3DGS-based method to the combined approach. We have run RGB-D ORB-SLAM to recover the poses and train 3DGS with the poses and sensor depth of the keyframes, equivalent to performing offline splatting. Though ORB-SLAM is best in terms of ATE (Tab.1 main), we find no significant difference across the rendering metrics (Tab.15b). SH is omitted in the synthetic Replica dataset as it contains no view-directional effects. While using an off-the-shelf tracker with a 3DGS mapper is possible, this work has focused on the value of the 3DGS throughout the entire algorithms, which is unexplored and therefore provides new insights. Further performance improvement of the unified approach will be an interesting future work.

8.5. Large-scale Scenes with Stereo Inputs:

This work focuses on pioneering 3DGS-based SLAM for live operation in small-scale scenes. However, we tested our method on the large-scale EuRoC Machine Hall dataset with depth from stereo (Tab.14). Fig.1 is a qualitative reconstruction result from our system. Our method is competitive in "easy" sequences, although performance drops for more difficult, longer sequences. Note that Point-SLAM [29] fails on all sequences in this dataset (over 3m ATE within the first 500 frames). In future work, we expect to improve our method by incorporating loop closure. In principle, loop closure will be easier to incorporate compared to other representations such as voxel grids (where feature allocations are fixed), via a method similar to surfel-based approaches like ElasticFusion [43].

	Method	TUM PSNR ↑	TUM SSIM ↑	TUM LPIPS ↓	Replica PSNR ↑	Replica SSIM ↑	Replica LPIPS ↓
(a)	Ours (w/o SH)	21.89	0.733	0.327	38.94	0.968	0.0703
	Ours (w. SH)	24.37	0.804	0.225	-	-	-
	Point-SLAM	21.39	0.727	0.463	24.37	0.840	0.185
(b)	ORB+GS (w/o SH)	25.12	0.837	0.161	37.11	0.964	0.040
	ORB+GS (w.SH)	25.44	0.842	0.146	-	-	-

Table 15. Mean Rendering metrics for TUM and Replica (RGBD).

Memory Usage for RGB-D SLAM					ATE RMSE	
Ours (w/o SH)	Ours (w. SH)	Point-SLAM	ORB+GS (w/o SH)	ORB+GS (w. SH)	Ours (w/o SH)	Ours (w. SH)
3.97MB	11.47MB	38.0MB	45.97MB	186.5MB	1.47cm	1.56cm

Table 16. Mean Memory and ATE metrics for TUM (RGBD).

8.6. Memory Consumption and Frame Rate (Table. 4)

8.6.1 Memory Analysis

In memory consumption analysis, for Table. 4, we measure the final size of the created Gaussians. The memory footprint of our system is lower than the original Gaussian Splatting, which uses approximately 300-700MB for the standard novel view synthesis benchmark dataset [11], as we only maintain well-constrained Gaussians via pruning and do not store the spherical harmonics.

8.6.2 Timing Analysis

To analyse the processing time of our monocular/RGB-D SLAM system, we measure the total time required to process all frames in the TUM-RGBD fr3/office dataset. This approach assesses the performance of our system as a whole, rather than isolating individual components. By adopting this approach, we gain a more realistic understanding of the system's true performance which better reflects the real-world operating conditions, as it avoids the assumption of an idealised, sequential interleaving of the tracking and mapping processes. As shown in Table 10, our system operates at 3.2 FPS with monocular and 2.5 FPS with depth. The FPS is found by dividing the number of processed frames by the total time. We conducted a similar analysis with the Replica dataset office2. Here, we compare the RGB-D method with and without multiprocessing. As expected, single-process implementation takes longer as it performs more mapping iterations.

9. Camera Pose Jacobian

Use of 3D Gaussian as a primitive and performing camera pose optimisation is discussed in [12]; however, the method assumes a smaller number of Gaussians and is based on ray-intersection not splatting; hence, is not applicable to 3DGS. While many applications of 3DGS exist, for example, dynamic tracking and 4D scene representation [16, 44], they assume offline application and require accurate camera position. In contrast, we perform camera pose optimisation by deriving the minimal analytical Jacobians on Lie group, and for completeness, we provide the derivation of the Jacobians presented in Eq. (6).

$$\frac{\mathcal{D}\boldsymbol{\mu}_C}{\mathcal{D}\boldsymbol{T}_{CW}} = \lim_{\tau\to 0}\frac{\mathrm{Exp}(\tau)\cdot\boldsymbol{\mu}_C - \boldsymbol{\mu}_C}{\tau} \tag{19}$$

$$= \lim_{\tau\to 0}\frac{(\boldsymbol{I}+\tau^\wedge)\cdot\boldsymbol{\mu}_C - \boldsymbol{\mu}_C}{\tau} \tag{20}$$

$$= \lim_{\tau\to 0}\frac{\tau^\wedge\cdot\boldsymbol{\mu}_C}{\tau} \tag{21}$$

$$= \lim_{\tau\to 0}\frac{\theta^\times\boldsymbol{\mu}_C + \rho}{\tau} \tag{22}$$

$$= \lim_{\tau\to 0}\frac{-\boldsymbol{\mu}_C^\times\theta + \rho}{\tau} \tag{23}$$

$$= \begin{bmatrix}\boldsymbol{I} & -\boldsymbol{\mu}_C^\times\end{bmatrix} \tag{24}$$

where $\boldsymbol{T}\cdot\mathbf{x}$ is the group action of $\boldsymbol{T}\in\boldsymbol{SE}(3)$ on $\mathbf{x}\in\mathbb{R}^3$.

Simiarly, we compute the Jacobian with respect to $\mathbf{W}$. Since the translational component is not involved, we only consider the rotational part $\boldsymbol{R}_{CW}$ of $\boldsymbol{T}_{CW}$.

$$\frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\boldsymbol{R}_{CW}} = \lim_{\theta\to 0}\frac{\mathrm{Exp}(\theta)\circ\mathbf{W} - \mathbf{W}}{\theta} \tag{25}$$

$$= \lim_{\theta\to 0}\frac{(\boldsymbol{I}+\theta^\wedge)\circ\mathbf{W} - \mathbf{W}}{\theta} \tag{26}$$

$$= \lim_{\theta\to 0}\frac{\theta^\wedge}{\theta}\circ\mathbf{W} \tag{27}$$

$$= \lim_{\theta\to 0}\frac{\theta^\times}{\theta}\circ\mathbf{W} \tag{28}$$

Since skew symmetric matrix is:

$$\theta^\times = \begin{bmatrix}0 & -\theta_z & \theta_y\\ \theta_z & 0 & -\theta_x\\ -\theta_y & \theta_x & 0\end{bmatrix} \tag{29}$$

The partial derivative of one of the component (e.g. θ_x) is:

$$\frac{\partial\theta^\times}{\partial\theta_x} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{bmatrix} = \mathbf{e}_1^\times \tag{30}$$

where $\mathbf{e}_1 = [1,0,0]^\top, \mathbf{e}_2 = [0,1,0]^\top, \mathbf{e}_3 = [0,0,1]^\top$.

$$\frac{\partial\mathbf{W}}{\partial\theta_x} = \mathbf{e}_1^\times\mathbf{W} = \begin{bmatrix}\mathbf{0}_{1\times 3}\\ -\mathbf{W}_{3,:}\\ \mathbf{W}_{2,:}\end{bmatrix} \tag{31}$$

$$\frac{\partial\mathbf{W}}{\partial\theta_y} = \mathbf{e}_2^\times\mathbf{W} = \begin{bmatrix}\mathbf{W}_{3,:}\\ \mathbf{0}_{1\times 3}\\ -\mathbf{W}_{1,:}\end{bmatrix} \tag{32}$$

$$\frac{\partial\mathbf{W}}{\partial\theta_z} = \mathbf{e}_3^\times\mathbf{W} = \begin{bmatrix}-\mathbf{W}_{2,:}\\ \mathbf{W}_{1,:}\\ \mathbf{0}_{1\times 3}\end{bmatrix} \tag{33}$$

where $\mathbf{W}_{i,:}$ refers to the ith row of the matrix. After column-wise vectorisation of Eq. (31), (32), (33), and stacking horizontally we get:

$$\frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\boldsymbol{R}_{CW}} = \begin{bmatrix} -\mathbf{W}_{:,1}^{\times} \\ -\mathbf{W}_{:,2}^{\times} \\ -\mathbf{W}_{:,3}^{\times} \end{bmatrix} , \tag{34}$$

where $\mathbf{W}_{:,i}$ refers to the ith column of the matrix. Since the translational part is all zeros, with this we get Eq. (6).

10. Additional Qualitative Results

We urge readers to view our supplementary video for convincing qualitative results. In Fig. 9 - Fig. 16, we further show additional qualitative results. We visually compare other state-of-the-art SLAM methods using differentiable rendering (Point-SLAM [29] and ESLAM [9]).

11. Limitation of this work

Although our novel Gaussian Splatting SLAM shows competitive performance on experimental results, the method also has several limitations.

- Currently, the proposed method is tested only on small room-scale scenes. For larger real-world scenes, the trajectory drift is inevitable. This could be addressed by integrating a loop closure module into our existing pipeline.
- Although we achieve interactive live operation, hard real-time operation on the benchmark dataset (30 fps on TUM sequences) is not achieved in this work. To improve speed, exploring a second-order optimiser would be an interesting direction.

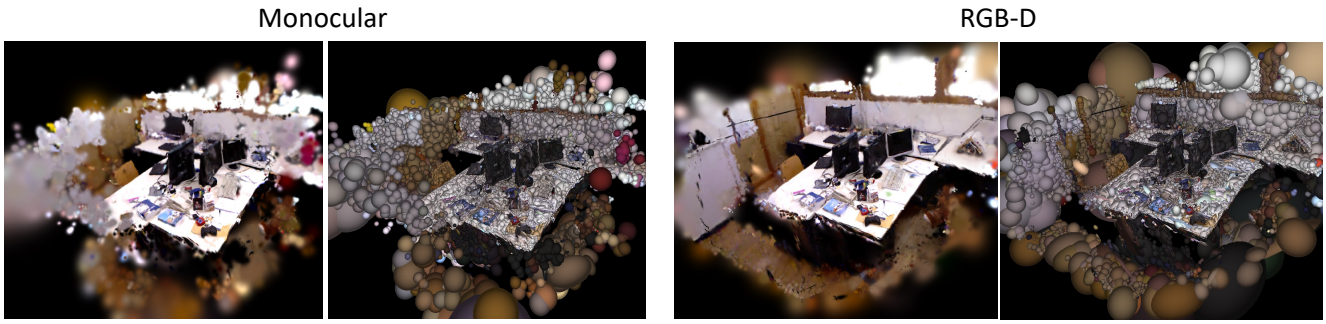


Figure 9. **Novel view rendering and Gaussian visualizations on TUM fr1/desk**

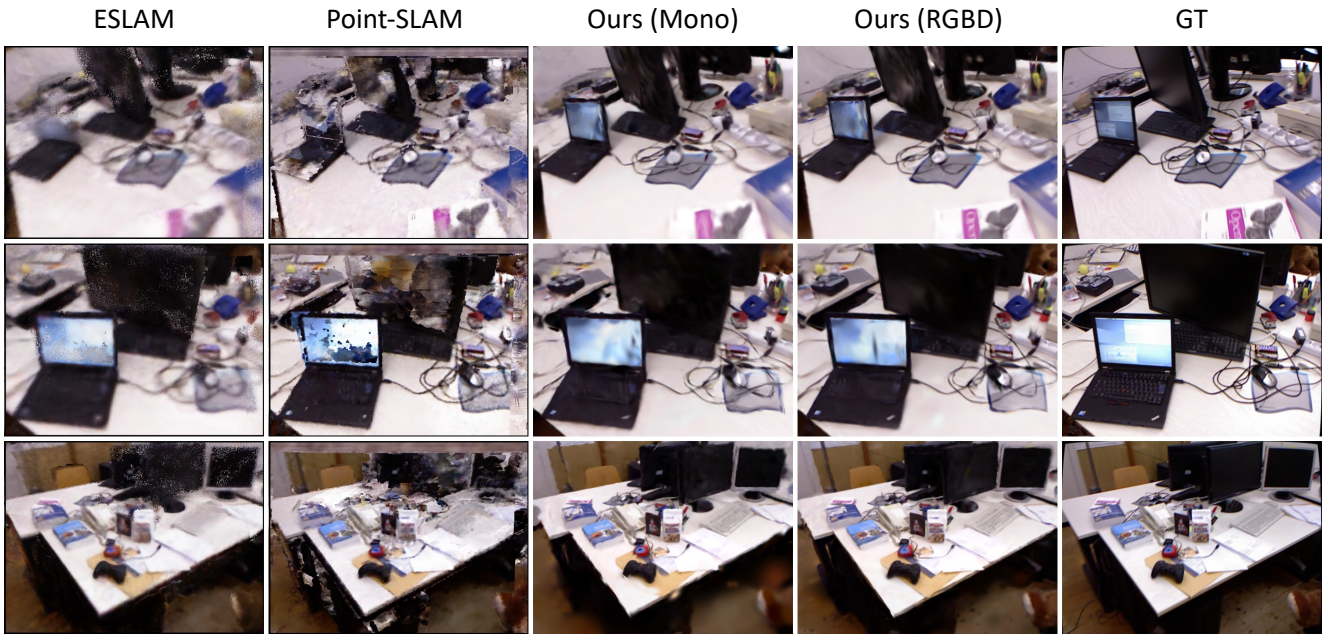


Figure 10. **Rendering comparison on TUM fr1/desk**

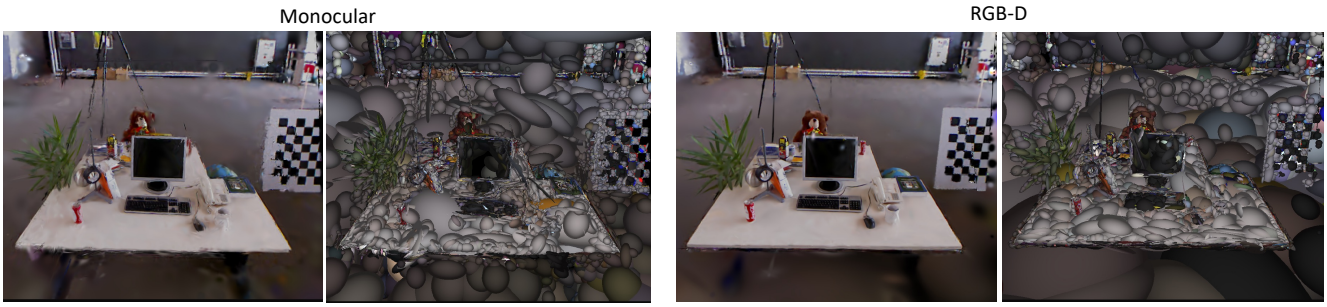


Figure 11. **Novel view rendering and Gaussian visualizations on TUM fr2/xyz**

Figure 12. **Rendering comparison on TUM fr2/xyz**

Figure 13. **Novel view rendering and Gaussian visualizations on TUM fr3/office**

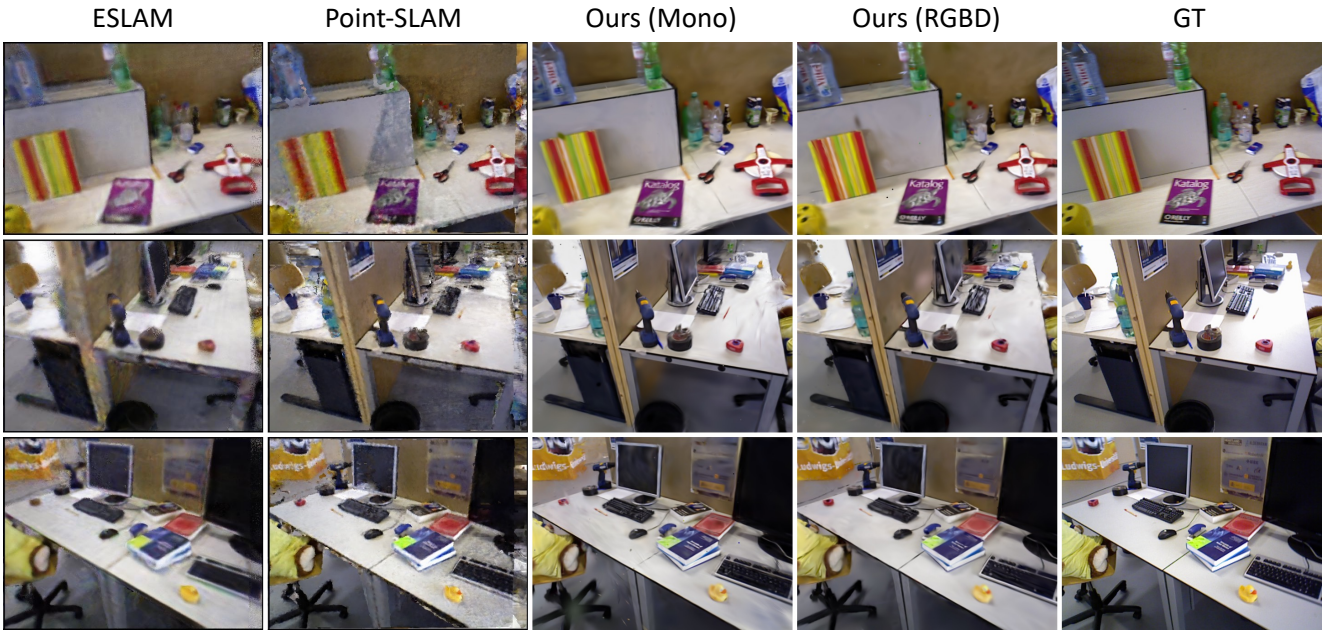


Figure 14. **Rendering comparison on TUM fr3/office**

Figure 15. **Novel view rendering and Gaussian visualizations on Replica**

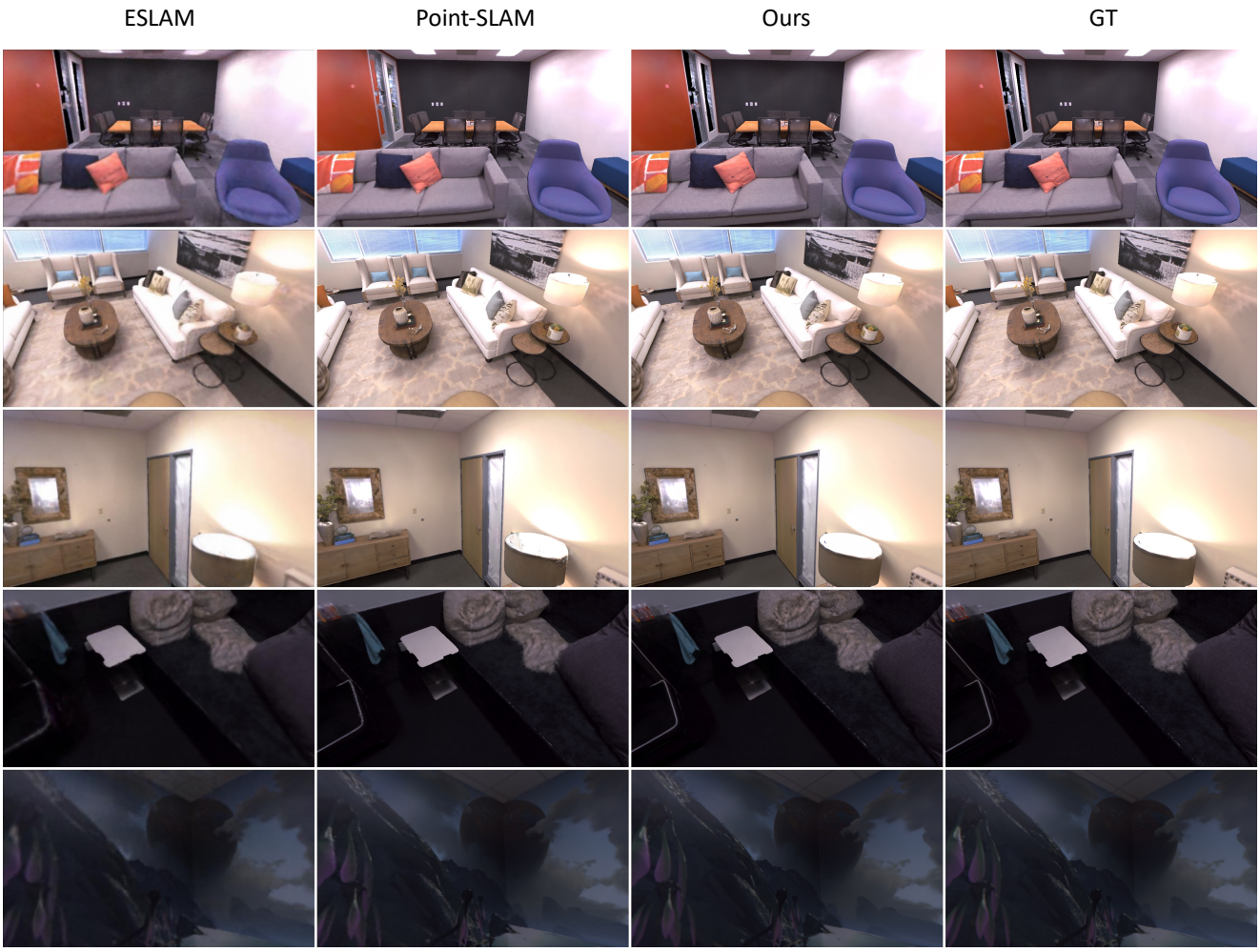


Figure 16. **Rendering comparison on Replica**

References

[1] Carlos Campos, Richard Elvira, Juan J. Gómez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics (T-RO)*, 37(6):1874–1890, 2021.

[2] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison. Deepfactors: Real-time probabilistic dense monocular SLAM. *IEEE Robotics and Automation Letters (RAL)*, 5(2): 721–728, 2020.

[3] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics (TOG)*, 36(3):24:1–24:18, 2017.

[4] Eric Dexheimer and Andrew J. Davison. Learning a Depth Covariance Function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[5] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[6] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[8] Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[9] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[10] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *Proc. of Joint 3DIM/3DPVT Conference (3DV)*, 2013.

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 2023.

[12] Leonid Keselman and Martial Hebert. Approximate differentiable rendering with algebraic surfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[14] 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 (ICLR)*, 2023.

[15] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[16] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *3DV*, 2024.

[17] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[18] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[19] N. J. Mitra, N. Gelfand, H. Pottmann, and L. J. Guibas. Registration of Point Cloud Data from a Geometric Optimization Perspective. In *Proceedings of the Symposium on Geometry Processing*, 2004.

[20] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 2022.

[21] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics (T-RO)*, 33(5): 1255–1262, 2017.

[22] R. Mur-Artal, J. M. M Montiel, and J. D. Tardós. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics (T-RO)*, 31(5):1147–1163, 2015.

[23] R. A. Newcombe. *Dense Visual SLAM*. PhD thesis, Imperial College London, 2012.

[24] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[25] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. In *Proceedings of SIGGRAPH*, 2013.

[27] Victor Adrian Prisacariu, Olaf Kähler, Ming-Ming Cheng, Carl Yuheng Ren, Julien P. C. Valentin, Philip H. S. Torr, Ian D. Reid, and David W. Murray. A framework for the volumetric integration of depth images. *CoRR*, abs/1410.0925, 2014.

[28] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors, 2019.

[29] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023.

[30] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. Surfelmeshing: Online surfel-based mesh reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020.

[31] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[32] J. Solà, J. Deray, and D. Atchuthan. A micro Lie theory for state estimation in robotics. *arXiv:1812.01537*, 2018.

[33] 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. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[34] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012.

[35] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[36] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[37] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[38] Zachary Teed and Jia Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. In *Neural Information Processing Systems (NIPS)*, 2021.

[39] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul HJ Kelly, and Stefan Leutenegger. Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters (RAL)*, 2018.

[40] Angtian Wang, Peng Wang, Jian Sun, Adam Kortylewski, and Alan Yuille. Voge: a differentiable volume renderer using gaussian ellipsoids for analysis-by-synthesis. 2022.

[41] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[42] T. Whelan, M. Kaess, H. Johannsson, M. F. Fallon, J. J. Leonard, and J. B. McDonald. Real-time large scale dense RGB-D SLAM with volumetric fusion. *International Journal of Robotics Research (IJRR)*, 34(4-5):598–626, 2015.

[43] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

[44] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[45] 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 *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2022.

[46] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[47] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[48] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[49] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. *International Conference on 3D Vision (3DV)*, 2024.

[50] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002.