4DTAM: Non-Rigid Tracking and Mapping via Surface Gaussian Splatting

Supplementary Material

We encourage readers to watch the supplementary video for additional details and qualitative results.

1. Implementation Details

1.1. System Details and Hyper parameters

Non-Rigid SLAM: We set the learning weights as follows: $\lambda_p = 0.9$, $\lambda_g = 0.1$, $\lambda_{iso} = 10.0$ and $\lambda_n = 0.002$. For the ARAP regularization [2], we use a nearest neighbor count of 20, a radius of 0.05, and an exponential decay weight of 500. Keyframes are selected with $N = 1$. For the MLP, we use an 8-layer architecture with 256 neurons per layer. Frequency encoding is set to 1 for time and 4 for position. MLP is implemented with CUDA-optimized CutlassMLP in tiny-cuda-nn [4] for the fast optimization.

Static SLAM Ablation: We followed the same hyperparameters as MonoGS [3], but we use normal loss L_n with the weight $\lambda_n = 0.01$ for the entire mapping process and $\lambda_g = 0.5$ for the final refinement. For the Replica 3D reconstruction evaluation, we have used the script introduced in [5].

Offline Non-rigid RGB-D Reconstruction Ablation: Camera poses are provided by the dataset and remain fixed during training. For the MLP, we adopt the same architecture described in [9], consisting of an 8-layer network with 256 dimensions per layer, where a concatenated feature vector is input to the fourth layer. The positional encoding frequencies are set to 6 for time and 10 for position. Following the approach in [1, 7], we evaluate the geometric and appearance metrics against the input views and report the average values.

2. Camera Pose Jacobian

We provide the detail of the derivation of camera pose jacobian of 2D Gaussian Splatting in **??**.

We use the notation from [6]. Let $\boldsymbol{T} \in \boldsymbol{SE}(3)$ and $\boldsymbol{\tau} = (\boldsymbol{\rho}, \boldsymbol{\theta}) \in \mathfrak{se}(3)$, the left-side partial derivative on the manifold is defined as:

$$\frac{\mathcal{D}f(\boldsymbol{T})}{\mathcal{D}\boldsymbol{T}} \triangleq \lim_{\boldsymbol{\tau}\to 0} \frac{\mathrm{Log}(f(\mathrm{Exp}(\tau) \circ \boldsymbol{T}) \circ f(\boldsymbol{T})^{-1})}{\boldsymbol{\tau}} \tag{1}$$

Eq ??:

$$\begin{aligned} \boldsymbol{T} &= \mathrm{Exp}(\boldsymbol{\tau}) = \exp(\boldsymbol{\tau}^\wedge) \\ &= \exp\left(\sum_{j=1}^{6} \mathbf{E}_j \tau_j\right), \quad j = 1, \ldots, 6, \quad \boldsymbol{\tau} \in \mathbb{R}^6. \end{aligned} \tag{2}$$

where the matrices $\mathbf{E}_j \in \mathbb{R}^{4\times 4}$ are the $\boldsymbol{SE}(3)$ *group generators* and form a basis for $\mathfrak{se}(3)$:

$$\begin{aligned} \mathbf{E}_1 &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{E}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{E}_3 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{E}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{E}_5 &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{E}_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{3}$$

We get the partial derivative as follows:

$$\frac{\partial}{\partial \tau_j} \exp(\boldsymbol{\tau}^\wedge)\bigg|_{\boldsymbol{\tau}=0} = \mathbf{E}_j, \quad j = 1, \ldots, 6. \tag{4}$$

Therefore, the full derivative is given as:

$$\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{\tau}}\bigg|_{\boldsymbol{\tau}=0} = \boldsymbol{T} \frac{\partial\left(\sum_{j=1}^{6} \mathbf{E}_j \tau_j\right)}{\partial \boldsymbol{\tau}}\bigg|_{\boldsymbol{\tau}=0} \tag{5}$$

Since the meaningful elements of the camera $\boldsymbol{T}$ is 12 number variables, we stack the elements for 12×6 matrix and we obtain

$$\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{\tau}}\bigg|_{\boldsymbol{\tau}=0} = \begin{bmatrix} \mathbf{0} & -\mathbf{R}^\times_{:,1} \\ \mathbf{0} & -\mathbf{R}^\times_{:,2} \\ \mathbf{0} & -\mathbf{R}^\times_{:,3} \\ \boldsymbol{I} & -\mathbf{t}^\times \end{bmatrix}. \tag{6}$$

where $\mathbf{R} \in \boldsymbol{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$ denote the rotation and translation parts of $\boldsymbol{T}$.

Eq ??:

$$\begin{aligned} \frac{\partial \mathbf{n}_c}{\partial \boldsymbol{\tau}}\bigg|_{\boldsymbol{\tau}=0} = \frac{\mathcal{D}\mathbf{n}_c}{\mathcal{D}\boldsymbol{T}_{CW}} &= \lim_{\boldsymbol{\tau}\to 0} \frac{\mathrm{Exp}(\boldsymbol{\tau})\mathbf{n}_c - \mathbf{n}_c}{\boldsymbol{\tau}} && (7) \\ &= \lim_{\boldsymbol{\tau}\to 0} \frac{(\boldsymbol{I} + \boldsymbol{\tau}^\wedge) \cdot \mathbf{n}_c - \mathbf{n}_c}{\boldsymbol{\tau}} && (8) \\ &= \lim_{\boldsymbol{\tau}\to 0} \frac{\boldsymbol{\tau}^\wedge \cdot \mathbf{n}_c}{\boldsymbol{\tau}} && (9) \\ &= \lim_{\boldsymbol{\tau}\to 0} \frac{\boldsymbol{\theta}^\times \mathbf{n}_c + \boldsymbol{\rho}}{\boldsymbol{\tau}} && (10) \\ &= \lim_{\boldsymbol{\tau}\to 0} \frac{-\mathbf{n}_c^\times \boldsymbol{\theta} + \boldsymbol{\rho}}{\boldsymbol{\tau}} && (11) \\ &= \begin{bmatrix} \boldsymbol{I} & -\mathbf{n}_c^\times \end{bmatrix} && (12) \end{aligned}$$

3. Sim4D Training/Test Views

We define the training and test views on a sphere, with its center representing the target object. In spherical coordinates (r, θ, ϕ), we set $r = 2.0$. The training view is sampled from two arcs on the sphere's surface, defined by $\theta \in [-10°, 10°]$ and $\phi \in [-10°, 10°]$. The test views are sampled from a circle on the sphere's surface that pass through four key points: $(\theta, \phi) = (5°, 0°)$, $(0°, 5°)$, $(-5°, 0°)$, and $(0°, -5°)$. These points are chosen to ensure uniform sampling around the target object while maintaining a clear separation between the training and test views.

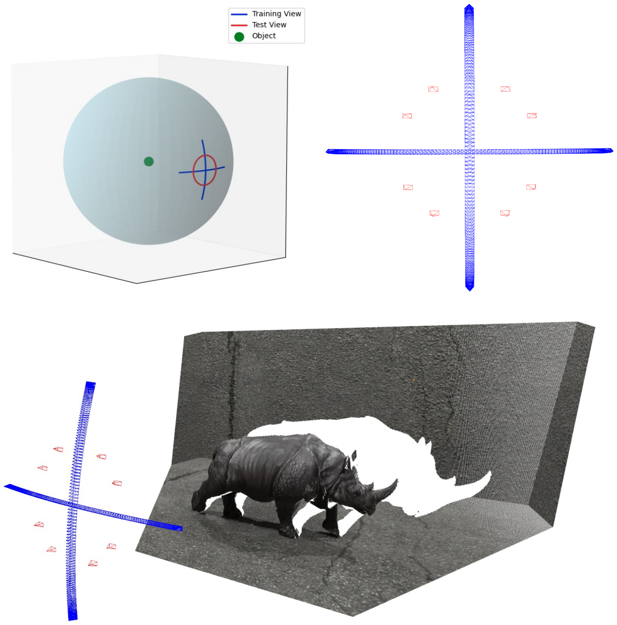

Figure 1. **Training and Test Views on the Sim4D Dataset: Blue** indicates training views, and **Red** indicates test views. Views are sampled (top right) from an arc on an object-centered sphere (top left) for dynamic scene reconstruction (bottom).

4. Further Ablation Analysis

4.1. Normal Rigidity Loss

Table 1 presents the quantitative results demonstrating the effect of the normal rigidity loss defined in Equation **??**. The normal rigidity loss improves the overall geometric metrics, such as camera ATE and L1 Depth, for the benchmark sequences by preserving the local geometric consistency of 2D Gaussians.

	ATE RMSE	L1 Depth	PSNR	SSIM	LPIPS
Ours full	**0.28**	**1.71**	28.47	0.820	**0.12**
w/o L_{ARAP_n}	0.52	2.00	**29.04**	**0.853**	0.13

Table 1. **Ablation Study on L_{ARAP_n}.** We report the average number of Sim4D dataset.

4.2. Monocular Depth Prior

While our method was primarily tested with RGB-D camera input, we conducted an ablation study using depth input from the state-of-the-art monocular prediction network [8], as shown in Table 5. The results demonstrate performance competitive with SurfelWarp, highlighting the potential for purely monocular non-rigid SLAM.

4.3. Static SLAM Ablation Analysis

Replica: Table 4 shows the photometric rendering performance analysis on the Replica dataset. The results demonstrate that the 2DGS-based SLAM approach offers an advantage in achieving accurate appearance reconstruction.

TUM: Table 2 presents the full ablation analysis on the TUM dataset. The 2DGS-based approach maintains competitive ATE and appearance metrics while achieving significantly better geometric rendering accuracy, as reflected in the Depth L1 error.

Method	**Metric**	**fr1**	**fr2**	**fr3**
MonoGS	ATE RMSE [cm] ↓	**1.50**	1.44	**1.49**
	Depth L1 [cm] ↓	6.2	13.0	13.0
	PSNR [dB] ↑	23.5	**24.65**	**25.09**
	SSIM ↑	0.775	0.785	**0.842**
	LPIPS ↓	0.26 1	**0.201**	**0.200**
MonoGS-2D	ATE RMSE [cm] ↓	1.58	**1.2**	1.83
	Depth L1 [cm] ↓	**3.0**	**2.3**	**4.3**
	PSNR [dB] ↑	**23.63**	24.47	24.05
	SSIM ↑	**0.782**	**0.79**	0.826
	LPIPS ↓	**0.251**	0.228	0.223

Table 2. **Static SLAM Ablation on TUM Dataset.** Comparison of ATE RMSE, Depth L1, and Rendering Performance Metrics.

Memory Analysis Table 3 presents the average memory usage on the TUM dataset sequences. Due to the geometrically accurate alignment, 2D Gaussians require fewer primitives to represent the scene, resulting in reduced memory consumption.

Memory Usage [MB]	
MonoGS-2D	MonoGS
2.73MB	3.97MB

Table 3. **Memory Analysis on TUM RGB-D dataset.**

4.4. Offline Non-Rigid RGB-D Reconstruction Ablation

Table 6 provides the full evaluation details of the offline non-rigid RGB-D reconstruction ablation analysis.

References

[1] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural surface reconstruction of dynamic scenes with monocular rgb-d camera. In *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 1

	Metric	room0	room1	room2	office0	office1	office2	office3	avg
MonoGS	PSNR [dB] ↑	34.83	36.43	37.49	39.95	42.09	36.24	36.70	37.50
	SSIM ↑	0.954	0.959	0.9665	0.971	0.977	0.964	0.963	0.96
	LPIPS ↓	0.068	0.076	0.075	0.072	0.055	0.078	0.065	0.07
MonoGS-2D	PSNR [dB] ↑	**36.21**	**37.81**	**38.7**	**43.45**	**43.8**	**37.48**	**37.43**	**39.14**
	SSIM ↑	**0.966**	**0.969**	**0.9737**	**0.985**	**0.984**	**0.972**	**0.971**	**0.975**
	LPIPS ↓	**0.04**	**0.042**	**0.044**	**0.025**	**0.029**	**0.04**	**0.039**	**0.038**

Table 4. Static SLAM Ablation: Rendering Performance Metrics [5] on Replica Dataset

Method	Category	Metric	curtain	flag	mercedes	modular_vehicle	rhino	shoe_rack	water_effect	wave_toy
Ours (Monocular)	Trajectory	ATE RMSE[cm]↓	6.23	16.29	4.90	1.86	3.17	8.02	5.52	7.21
	Geometry	L1 Depth[cm]↓	74.2	155	59.2	38.0	37.7	89.8	72.4	80.8
	Appearance	PSNR [dB] ↑	17.73	16.22	20.72	26.28	21.48	17.49	18.86	17.98
		SSIM ↑	0.461	0.455	0.636	0.578	0253	0.448	0.390	0.441
		LPIPS ↓	0.297	0.517	0.282	0.380	0.339	0.391	0.258	0.281

Table 5. **Non-rigid SLAM Evaluation on Sim4D Dataset with Monocular Depth Prior.**

[2] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *3DV*, 2024. 1

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

[4] Thomas Müller. tiny-cuda-nn, 2021. 1

[5] 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. 1, 3

[6] J. Solà, J. Deray, and D. Atchuthan. A micro Lie theory for state estimation in robotics. *arXiv:1812.01537*, 2018. 1

[7] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Morpheus: Neural dynamic 360deg surface reconstruction from monocular rgb-d video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20965–20976, 2024. 1, 4

[8] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision, 2024. 2

[9] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. 2024. 1

		KillingFusion			DeepDeform			iPhone		
		frog	duck	snoopy	seq002	seq004	seq028	teddy	mochi	haru
Morpheus [7]	Depth L1 [cm]	4.37	3.01	2.30	2.08	1.24	2.26	5.40	0.31	1.63
	PSNR [dB] ↑	27.2	28.17	25.73	27.21	26.94	26.30	23.40	28.12	24.34
	SSIM ↑	0.802	0.716	0.779	0.809	0.823	0.795	0.237	0.623	0.510
	LPIPS ↓	0.31	0.419	0.483	0.301	0.428	0.397	0.776	0.55	0.564
Ours	Depth L1 [cm]	0.65	1.91	12.1	0.78	1.07	1.30	0.32	0.22	0.12
	PSNR [dB] ↑	33.72	32.75	26.95	24.36	24.13	24.02	23.89	36.15	22.60
	SSIM ↑	0.941	0.949	0.899	0.897	0.897	0.902	0.739	0.926	0.690
	LPIPS ↓	0.063	0.073	0.257	0.245	0.313	0.241	0.259	0.131	0.391

Table 6. **Offline RGB-D Reconstruction Results**