Gaussian Splatting SLAM

Hidenobu Matsuki*, Riku Murai*, Paul H.J. Kelly and Andrew J. Davison

Abstract— **We present the first application of 3D Gaussian Splatting to incremental 3D reconstruction using a single moving monocular or RGB-D camera. Our Simultaneous Localisation and Mapping (SLAM) method, which runs live at 3fps, utilises Gaussians as the only 3D representation, unifying the required representation for accurate, efficient tracking, mapping, and high-quality rendering. Several innovations are required to continuously reconstruct 3D scenes with high fidelity from a live camera. First, to move beyond the original 3DGS algorithm, which requires accurate poses from an offline Structure from Motion (SfM) system, we formulate camera tracking for 3DGS using direct optimisation against the 3D Gaussians, and show that this enables fast and robust tracking with a wide basin of convergence. Second, by utilising the explicit nature of the Gaussians, we introduce geometric verification and regularisation to handle the ambiguities occurring in incremental 3D dense reconstruction. Finally, we introduce a full SLAM system which not only achieves state-of-the-art results in novel view synthesis and trajectory estimation, but also reconstruction of tiny and even transparent objects.**

The full version of this paper has been accepted to CVPR 2024. For complete technical details, please refer to our work [1].

I. Introduction

A long-term goal of online reconstruction with a single moving camera is near-photorealistic fidelity, which will surely allow new levels of performance in many areas of Spatial AI and robotics as well as opening up a whole range of new applications. While we increasingly see the benefit of applying powerful pre-trained priors to 3D reconstruction, a key avenue for progress is still the invention and development of core 3D representations with advantageous properties. While many "layered" SLAM systems exist which combine multiple representations, the most interesting advances are when a new unified dense representation can be used for all aspects of a system's operation: local representation of detail, large-scale geometric mapping and also camera tracking by direct alignment.

In this paper we present the first online visual SLAM system based solely on the 3D Gaussian Splatting (3DGS) representation [2] recently making a big impact in offline scene reconstruction. In 3DGS a scene is represented by a large number of Gaussian blobs with orientation, elongation, colour and opacity. Other previous world/map-centric scene representations used for visual SLAM include occupancy or Signed Distance Function (SDF) voxel grids [3]; meshes [4]; point or surfel clouds [5], [6]; and recently neural fields [7].

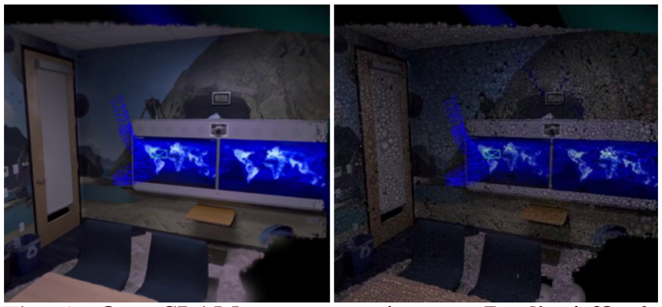
Fig. 1: **Our SLAM reconstruction on Replica/office0**. From a single monocular camera, we reconstruct a high fidelity *3D scene* live at 3fps. For every incoming RGB frame, 3D Gaussians are incrementally formed and optimised together with the camera poses. We show both the rasterised Gaussians (left) and Gaussians shaded to highlight the geometry (right).

Each of these has disadvantages: grids use significant memory and have bounded resolution, and even if octrees or hashing allow more efficiency they cannot be flexibly warped for large corrections [8], [9]; meshes require difficult, irregular topology to fuse new information; surfel clouds are discontinuous and difficult to fuse and optimise; and neural fields require expensive per-pixel raycasting to render. We show that 3DGS has none of these weaknesses. As a SLAM representation, it is most similar to point and surfel clouds, and inherits their efficiency, locality and ability to be easily warped or modified. However, it also represents geometry in a smooth, continuously differentiable way: a dense cloud of Gaussians merge together and jointly define a continuous volumetric function. And crucially, the design of modern graphics cards means that a large number of Gaussians can be efficiently rendered via "splatting" rasterisation, up to 200fps at 1080p. This rapid, differentiable rendering is integral to the tracking and map optimisation loops in our system.

Our contributions are as follows:

- The first near real-time SLAM system which works with a 3DGS as the only underlying scene representation.
- Novel techniques within the SLAM framework, including the analytic camera pose Jacobian, Gaussian shape regularisation and geometric verification.
- Extensive evaluations on a variety of datasets both for monocular and RGB-D settings, demonstrating competitive performance in real-world scenarios.

II. Method

A. Gaussian Splatting

Our SLAM representation is 3DGS, mapping the scene with a set of anisotropic Gaussians $\mathcal{G}$. Each Gaussian $\mathcal{G}^i$

*First two authors contributed equally to this work. Research presented in this paper has been supported by Dyson Technology Ltd. Dyson Robotics Laboratory, Imperial College London, United Kingdom `h.matsuki20, riku.murai15@imperial.ac.uk`.

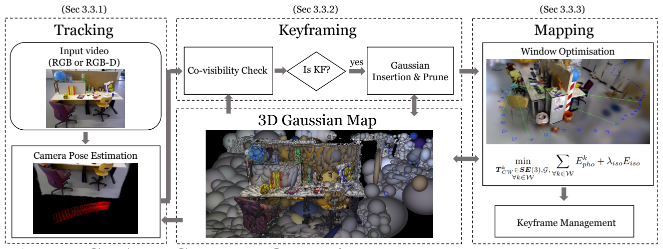
(Sec 3.3.1)
Tracking
Input video (RGB or RGB-D)
Camera Pose Estimation
(Sec 3.3.2)
Keyframing
Co-visibility Check
Is KF?
yes
Gaussian Insertion & Prune
3D Gaussian Map
(Sec 3.3.3)
Mapping
Window Optimisation
$\min_{T^k_{CW} \in SE(3), \mathcal{G}, \forall k \in \mathcal{W}} \sum_{\forall k \in \mathcal{W}} E^k_{pho} + \lambda_{iso} E_{iso}$
Keyframe Management

Fig. 2: **SLAM System Overview:** Our SLAM system uses 3D Gaussians as the only representation, unifying all components of SLAM, including tracking, mapping, keyframe management, and novel view synthesis.

contains optical properties: colour c^i and opacity α^i. For continuous 3D representation, the mean μ^i_W and covariance Σ^i_W, defined in the world coordinate, represent the Gaussian's position and its ellipsoidal shape. Since 3DGS uses volume rendering, explicit extraction of the surface is not required. Instead, by splatting and blending $\mathcal{N}$ Gaussians, a pixel colour $\mathcal{C}_p$ is synthesised:

$$\mathcal{C}_p = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \ . \tag{1}$$

3DGS performs rasterisation, iterating over the Gaussians rather than marching along the camera rays, and hence, free spaces are ignored during rendering. During rasterisation, the contributions of α are decayed via a Gaussian function, based on the 2D Gaussian formed by splatting a 3D Gaussian. The 3D Gaussians $\mathcal{N}(\mu_W, \Sigma_W)$ in world coordinates are related to the 2D Gaussians $\mathcal{N}(\mu_I, \Sigma_I)$ on the image plane through a projective transformation:

$$\mu_I = \pi(T_{CW} \cdot \mu_W) \ , \Sigma_I = \mathbf{J}\mathbf{W}\Sigma_W \mathbf{W}^T \mathbf{J}^T \ , \tag{2}$$

where π is the projection operation and $T_{CW} \in SE(3)$ is the camera pose of the viewpoint. $\mathbf{J}$ is the Jacobian of the linear approximation of the projective transformation and $\mathbf{W}$ is the rotational component of T_{CW}.

B. *Camera Pose Optimisation*

In order to avoid the overhead of automatic differentiation, 3DGS implements rasterisation with CUDA with derivatives for all parameters calculated explicitly. Since rasterisation is performance critical, we similarly derive the camera Jacobians explicitly. We use Lie algebra to derive the minimal Jacobians, ensuring that the dimensionality of the Jacobians matches the degrees of freedom, eliminating any redundant computations. The terms of Eq. (2) are differentiable with respect to the camera pose T_{CW}; using the chain rule:

$$\frac{\partial \mu_I}{\partial T_{CW}} = \frac{\partial \mu_I}{\partial \mu_C} \frac{\mathcal{D}\mu_C}{\mathcal{D}T_{CW}} \ , \tag{3}$$

$$\frac{\partial \Sigma_I}{\partial T_{CW}} = \frac{\partial \Sigma_I}{\partial \mathbf{J}} \frac{\partial \mathbf{J}}{\partial \mu_C} \frac{\mathcal{D}\mu_C}{\mathcal{D}T_{CW}} + \frac{\partial \Sigma_I}{\partial \mathbf{W}} \frac{\mathcal{D}\mathbf{W}}{\mathcal{D}T_{CW}} \ . \tag{4}$$

We take the derivatives on the manifold to derive minimal parameterisation. Borrowing the notation from [10], let $T \in SE(3)$ and $\tau \in \mathfrak{se}(3)$. We define the partial derivative on the manifold as:

$$\frac{\mathcal{D}f(T)}{\mathcal{D}T} \triangleq \lim_{\tau \to 0} \frac{\mathrm{Log}(f(\mathrm{Exp}(\tau) \circ T) \circ f(T)^{-1})}{\tau} \ , \tag{5}$$

where $\circ$ is a group composition, and Exp, Log are the exponential and logarithmic mappings between Lie algebra and Lie Group. With this, we derive the following:

$$\frac{\mathcal{D}\mu_C}{\mathcal{D}T_{CW}} = \begin{bmatrix} I & -\mu_C^{\times} \end{bmatrix} , \frac{\mathcal{D}\mathbf{W}}{\mathcal{D}T_{CW}} = \begin{bmatrix} \mathbf{0} & -\mathbf{W}^{\times}_{:,1} \\ \mathbf{0} & -\mathbf{W}^{\times}_{:,2} \\ \mathbf{0} & -\mathbf{W}^{\times}_{:,3} \end{bmatrix} \ , \tag{6}$$

where $^{\times}$ denotes the skew symmetric matrix of a 3D vector, and $\mathbf{W}_{:,i}$ refers to the ith column of the matrix.

C. *SLAM*

In this section, we present details of full SLAM framework. The overview of the system is summarised in Fig. 2.

1) Tracking: In tracking only the current camera pose is optimised, without updates to the map representation. In the monocular case, we minimise the following photometric residual:

$$E_{pho} = \| I(\mathcal{G}, T_{CW}) - \bar{I} \|_1 \ , \tag{7}$$

where $I(\mathcal{G}, T_{CW})$ renders the Gaussians $\mathcal{G}$ from T_{CW}, and $\bar{I}$ is an observed image.

We further optimise affine brightness parameters for varying exposure. When depth observations are available, we define the geometric residual as:

$$E_{geo} = \| D(\mathcal{G}, T_{CW}) - \bar{D} \|_1 \ , \tag{8}$$

where $D(\mathcal{G}, T_{CW})$ is depth rasterisation and $\bar{D}$ is the observed depth. Rather than simply using the depth measurements to initialise the Gaussians, we minimise both photometric and geometric residuals: $\lambda_{pho} E_{pho} + (1 - \lambda_{pho}) E_{geo}$, where λ_{pho} is a hyperparameter.

As in Eq. (1), per-pixel depth is rasterised by alpha-blending:

$$\mathcal{D}_p = \sum_{i \in \mathcal{N}} z_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \ , \tag{9}$$

where z_i is the distance to the mean μ_W of Gaussian i along the camera ray. We derive analytical Jacobians for the camera pose optimisation in a similar manner to Eq. (3), (4).

2) Keyframing:

Since using all the images from a video stream to jointly optimise the Gaussians and camera poses online is infeasible, we maintain a small window $\mathcal{W}_k$ consisting of carefully selected keyframes based on inter-frame covisibility. Ideal keyframe management will select non-redundant keyframes observing the same area, spanning a wide baseline to provide better multiview constraints.

a) Selection and Management: Every tracked frame is checked for keyframe registration based on our simple yet effective criteria. We measure the covisibility by measuring the intersection over union of the observed Gaussians between the current frame i and the last keyframe j. If the covisibility drops below a threshold, or if the relative translation t_{ij} is large with respect to the median depth, frame i is registered as a keyframe. For efficiency, we maintain only a small number of keyframes in the current window $\mathcal{W}_k$ following the keyframe management heuristics of DSO [11]. The main difference is that a keyframe is removed from the current window if the overlap coefficient with the latest keyframe drops below a threshold.

b) Gaussian Covisibility: An accurate estimate of covisibility simplifies keyframe selection and management. 3DGS respects visibility ordering since the 3D Gaussians are sorted along the camera ray. This property is desirable for covisibility estimation as occlusions are handled by design. A Gaussian is marked to be visible from a view if used in the rasterisation and if the ray's accumulated α has not yet reached 0.5. This enables our estimated covisibility to handle occlusions without requiring additional heuristics.

c) Gaussian Insertion and Pruning: At every keyframe, new Gaussians are inserted into the scene to capture newly visible scene elements and to refine the fine details. When depth measurements are available, Gaussian means μ_W are initialised by back-projecting the depth. In the monocular case, we render the depth at the current frame. For pixels with depth estimates, μ_W are initialised around those depths with low variance; for pixels without the depth estimates, we initialise μ_W around the median depth of the rendered image with high variance. In the monocular case, the positions of many newly inserted Gaussians are incorrect. While the majority will quickly vanish during optimisation as they violate multiview consistency, we further prune the excess Gaussians by checking the visibility amongst the current window $\mathscr{W}_k$. If the Gaussians inserted within the last 3 keyframes are unobserved by at least 3 other frames, we prune them out as they are geometrically unstable.

3) Mapping: The purpose of mapping is to maintain a coherent 3D structure and to optimise the newly inserted Gaussians. During mapping, the keyframes in $\mathscr{W}_k$ are used to reconstruct currently visible regions. Additionally, two random past keyframes $\mathscr{W}_r$ are selected per iteration to avoid forgetting the global map. Rasterisation of 3DGS imposes no constraint on the Gaussians along the viewing ray direction, even with a depth observation. This is not a problem when sufficient carefully selected viewpoints are provided (e.g. in the novel view synthesis case); however, in continuous SLAM this causes many artefacts, making tracking challenging. We therefore introduce an isotropic regularisation:

$$E_{iso} = \sum_{i=1}^{|\mathscr{G}|} \|\mathbf{s}_i - \tilde{\mathbf{s}}_i \cdot \mathbf{1}\|_1 \tag{10}$$

to penalise the scaling parameters $\mathbf{s}_i$ (i.e. stretch of the ellipsoid) by its difference to the mean $\tilde{\mathbf{s}}_i$. Let the union of the keyframes in the current window and the randomly selected one be $\mathscr{W} = \mathscr{W}_k \cup \mathscr{W}_r$. For mapping, we solve the following problem:

$$\min_{\substack{T_{CW}^k \in SE(3), \mathscr{G}, \\ \forall k \in \mathscr{W}}} \sum_{\forall k \in \mathscr{W}} E_{pho}^k + \lambda_{iso} E_{iso} \ . \tag{11}$$

If depth observations are available, as in tracking, geometric residuals Eq. (8) are added to the optimisation problem.

III. Evaluation

A. Experimental Setup

a) Datasets: For our quantitative analysis, we evaluate our method on the TUM RGB-D dataset [12] (3 sequences) and the Replica dataset [13] (8 sequences), following the evaluation in [7]. For qualitative results, we use self-captured real-world sequences recorded by Intel Realsense d455. Since the Replica dataset is designed for RGB-D SLAM evaluation, it contains challenging purely rotational camera motions. We hence use the Replica dataset for RGB-D evaluation only. The TUM RGB-D dataset is used for both monocular and RGB-D evaluation.

b) Implementation Details: We run our SLAM on a desktop with Intel Core i9 12900K 3.50GHz and a single NVIDIA GeForce RTX 4090. We present results from our multi-process implementation aimed at real-time applications. For a fair comparison with other methods on Replica, we additionally report result for single-process implementation which performs more mapping iterations. As with 3DGS, time-critical rasterisation and gradient computation are implemented using CUDA. The rest of the SLAM pipeline is developed with PyTorch. Details of hyperparameters are provided in the supplementary material.

c) Metrics: For camera tracking accuracy, we report the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE) of the keyframes. We report the average across three runs for all our evaluations. In the tables, the best result is in bold, and the second best is underlined.

B. Quantitative Evaluation

a) Camera Tracking Accuracy: Table I shows the tracking results on the TUM RGB-D dataset. In the monocular setting, our method surpasses other baselines without requiring any deep priors. Furthermore, our performance is comparable to systems which perform explicit loop closure. This clearly highlights that there still remains potential for enhancing the tracking of monocular SLAM by exploring fundamental SLAM representations.

Our RGB-D method shows better performance than any other baseline method. Notably, our system surpasses ORB-SLAM in the fr1 sequences, bridging the gap between Map-centric SLAM and the state-of-the-art sparse frame-centric methods. Our system demonstrates strong performance on real-world data, as our system flexibly handles real sensor noise by direct optimisation of the Gaussian positions against information from every pixel.

Method	**seq1**	**seq2**	**seq3**	**Avg.**
Neural SDF (Hash Grid)	0.13	0.15	0.16	0.14
Neural SDF (MLP)	0.40	0.38	0.22	0.33
Ours w/o depth	<u>0.82</u>	<u>0.91</u>	**0.65**	<u>0.79</u>
Ours w/ depth	**0.83**	**1.0**	**0.65**	**0.82**

TABLE II: **Camera convergence analysis.**

b) Convergence Basin Analysis: We conducted a convergence funnel analysis, an evaluation methodology proposed in [24] and used in [25]. Here, we train a 3D representation (e.g. 3DGS) using 9 fixed views arranged in a square. We set the viewpoint in the middle of the square to

Input	Loop-closure	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
90Monocular	w/o	DSO [11]	22.4	**1.10**	9.50	11.0
		DROID-VO [14]	5.20	10.7	7.30	7.73
		DepthCov [15]	5.60	1.20	68.8	25.2
		Ours	**4.15**	4.79	**4.39**	**4.44**
	w/	DROID-SLAM [14]	**1.80**	**0.50**	2.80	1.70
		ORB-SLAM2 [16]	2.00	0.60	**2.30**	**1.60**
90RGB-D	w/o	iMAP [7]	4.90	2.00	5.80	4.23
		NICE-SLAM [17]	4.26	6.19	6.87	5.77
		DI-Fusion [18]	4.40	2.00	5.80	4.07
		Vox-Fusion [19]	3.52	1.49	26.01	10.34
		ESLAM [20]	2.47	**1.11**	2.42	2.00
		Co-SLAM [21]	2.40	1.70	2.40	2.17
		Point-SLAM [22]	4.34	1.31	3.48	3.04
		Ours	**1.52**	1.58	**1.65**	**1.58**
	w/	BAD-SLAM [6]	1.70	1.10	1.70	1.50
		Kintinous [23]	3.70	2.90	3.00	3.20
		ORB-SLAM2 [16]	**1.60**	**0.40**	**1.00**	**1.00**

TABLE I: **Camera tracking result on TUM for monocular and RGB-D.** ATE RMSE in cm is reported. We divide systems into with and without explicit loop closures. In both monocular and RGB-D cases, we achieve state-of-the-art performance.

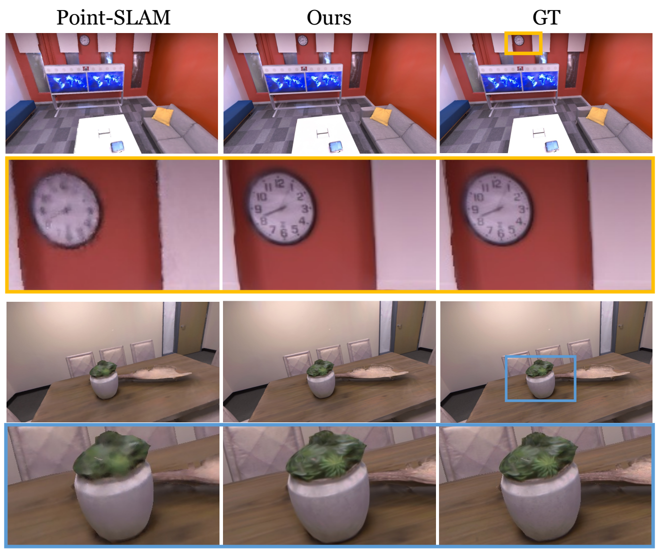

Fig. 3: **Rendering examples on Replica.** Due to the stochastic nature of ray sampling, Point-SLAM struggle with rendering fine details.

be the target view. As shown in Fig 4, we uniformly sample a position, creating a funnel. From the sampled position, given the RGB image of the target view, we perform camera pose optimisation for 1000 iterations. The optimisation is successful if it converges to within 1cm of the target view within the fixed iterations. We compare our Gaussian approach with Co-SLAM [21]'s network (Hash Grid SDF) and iMAP's [7] network with Co-SLAM's SDF loss for further geometric accuracy (MLP Neural SDF). We render the training views using a synthetic Replica dataset and create three sequences for testing (seq1, seq2 and seq3). The width of the square formed by the training view is 0.5m, and the test cameras are distributed with radii ranging from 0.2m to 1.2m, covering a larger area than the training view. When training the map, the three methods— Ours w/depth, Hash Grid SDF, and MLP SDF—use RGB-D images, whereas Ours w/o depth utilises only colour images. Fig. 4 shows the qualitative results and

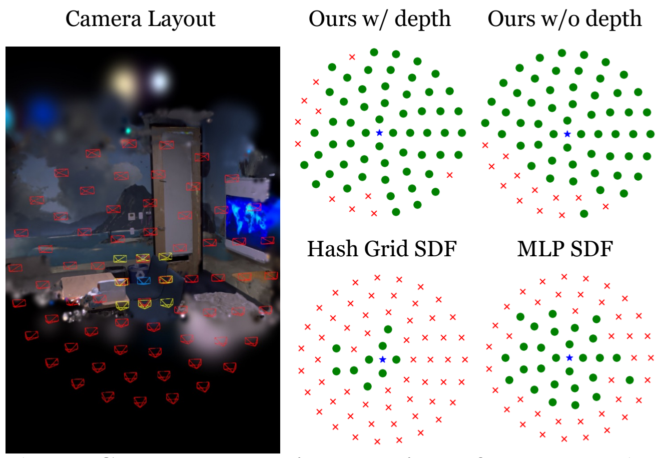

Fig. 4: **Convergence basin analysis**: **Left:** 3D Gaussian reconstructed using the training views (Yellow) and visualisation of the test poses (Red). **Right:** Visualisation of convergence basin of our method (top, with and without depth for training) and other representations (bottom). The green circle marks successful convergence, and the red cross marks failure.

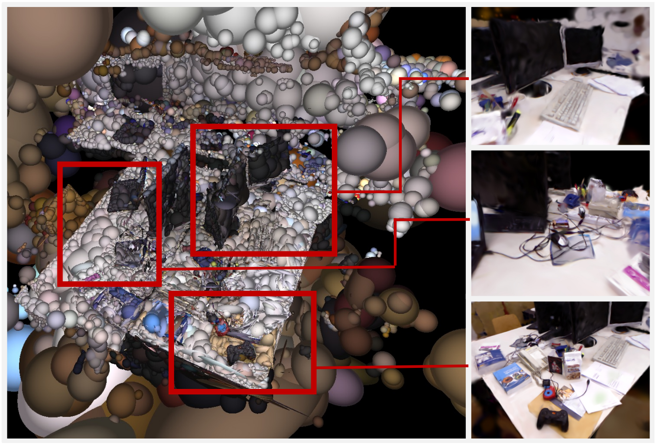
Fig. 5: **Monocular SLAM result on fr1/desk sequence:** We show the reconstructed 3D Gaussian maps (Left) and novel view synthesis result (Right).

Table II reports the success rate. For both with and without depth for training, our method shows better convergence. Unlike hashing and positional encoding which can lead to signal conflict, anisotropic Gaussians form a smooth gradient in 3D space, increasing the convergence basin.

IV. Conclusion

We have proposed the first SLAM method using 3D Gaussians as a SLAM representation. Via efficient volume rendering, our system significantly advances the fidelity and diversity of object materials a live SLAM system can capture. Our system achieves state-of-the-art performance across benchmarks for both monocular and RGB-D cases. Interesting directions for future research are the integration of loop closure for handling large-scale scenes and extraction of geometry such as surface normal as Gaussians do not explicitly represent surface.

References

[1] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison, "Gaussian Splatting SLAM," 2024.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (TOG)*, 2023.

[3] 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.

[4] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: Online surfel-based mesh reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020.

[5] 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.

[6] T. Schöps, T. Sattler, and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] 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.

[8] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. Kelly, and S. Leutenegger, "Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters (RAL)*, 2018.

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

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

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

[12] 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.

[13] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[14] Z. Teed and J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," in *Neural Information Processing Systems (NIPS)*, 2021.

[15] E. Dexheimer and A. J. Davison, "Learning a Depth Covariance Function," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[16] 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)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[17] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[18] J. Huang, S.-S. Huang, H. Song, and S.-M. Hu, "Di-fusion: Online implicit 3d reconstruction with deep priors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[19] X. Yang, H. Li, H. Zhai, Y. Ming, Y. Liu, and G. 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.

[20] 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.

[21] H. Wang, J. Wang, and L. 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.

[22] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, "Point-slam: Dense neural point cloud-based slam," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023.

[23] 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)*, vol. 34, no. 4-5, pp. 598–626, 2015.

[24] 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.

[25] R. A. Newcombe, "Dense Visual SLAM," Ph.D. dissertation, Imperial College London, 2012.