CAAI Transactions on Intelligence Technology

WILEY

The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

Geo2DGS-SLAM: Geometry-Driven SLAM With 2D Gaussian Splatting

Weiqing Yan[1] | Tianhao Xie[1] | Liang Liao[2] | Wujie Zhou[3] | Chang Tang[4]

[1]School of Computer and Control Engineering, Yantai University, Yantai, China | [2]Xidian University Hangzhou Institute of Technology, Hangzhou, China | [3]School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Zhejiang, China | [4]School of Software Engineering, Huazhong University of Science and Technology, Wuhan, China

Correspondence: Weiqing Yan (wqyan@tju.edu.cn) | Tianhao Xie (202400358015@s.ytu.edu.cn)

Received: 7 September 2025 | **Revised:** 25 May 2026 | **Accepted:** 9 September 2026

Keywords: 3D Gaussian Splatting | dense visual SLAM | indoor scene | surface reconstruction

ABSTRACT

Recent advances in SLAM systems based on 3D Gaussian Splatting have enabled dense and photorealistic scene reconstruction. However, 3DGS-based methods still suffer from limited geometric awareness due to the lack of normal rendering support, view inconsistency and weak depth constraints, ultimately hindering high-fidelity geometric reconstruction and robust pose tracking. In this paper, we present Geo2DGS-SLAM, a novel dense RGB-D SLAM framework that uses a geometry-aware 2D Gaussian representation for scene modelling, featuring a hybrid tracking framework, 2DGS-based submap construction and a mesh-binding submap merging strategy. Unlike prior approaches that rely solely on 3DGS appearance modelling, our 2DGS representation enables more accurate rendering of both depth and surface normals, facilitating high-quality and geometrically consistent reconstruction. To further enhance robustness and global consistency, the hybrid tracking framework integrates local bundle adjustment, loop closure and global bundle adjustment, which jointly mitigate accumulated drift and improve trajectory stability, particularly in long or challenging sequences. Extensive experiments on public indoor RGB-D datasets demonstrate that Geo2DGS-SLAM achieves competitive performance in geometry-aware mapping, rendering fidelity and pose estimation accuracy among recent Gaussian-based SLAM methods. Codes and additional experimental results will be available at https://github.com/Able1231/Geo2DGS-SLAM.

1 | Introduction

Simultaneous localisation and mapping (SLAM) [1] is a core technology in computer vision and robotics, enabling machines to build maps and localise themselves in unknown environments. Initially developed for robotics, SLAM is now widely used in areas such as augmented reality (AR), visual surveillance and medical applications. Modern SLAM systems [2] integrate data from cameras, LiDAR, inertial sensors and GPS. Although sonar and LiDAR offer high accuracy, their size and cost limit practical use. In contrast, monocular, stereo and RGB-D cameras are more affordable and capture rich information, making them popular in Visual SLAM for improved scene understanding and localisation. However, SLAM continues to encounter several critical challenges, such as ensuring global map alignment, achieving reliable camera pose estimation, producing detailed scene reconstructions, supporting efficient online processing, enabling inference in unexplored regions, scaling to large environments and coping with noisy sensor inputs.

SLAM methods have significantly evolved to accommodate these specific requirements. Early handcrafted algorithms [3–7] showed impressive efficiency in online processing and

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2026 The Author(s). *CAAI Transactions on Intelligence Technology* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology and Chongqing University of Technology.

demonstrated good scalability. Nonetheless, these methods encountered limitations when facing issues such as intense illumination changes, photometric variations and low-texture or dynamic scenes, resulting in degraded performance in challenging environments. Advanced techniques incorporating deep learning methods [8–10] have been crucial for improving localisation and mapping accuracy and reliability. By utilising the strong feature learning capacity of deep neural networks, these approaches enhance SLAM performance in complex and adverse environments. Nonetheless, the requirement for large-scale annotated datasets and precise supervisory signals limits the flexibility of these approaches when applied to unfamiliar or previously unobserved environments. In addition, traditional approaches along with those driven by deep learning techniques are often limited by their reliance on discrete surface formats, including point or surface clouds [11], voxel hashing [12], voxel grids [7] and octree structures [13]. These representations often suffer from low model density, reduced spatial detail and reconstruction artefacts. Accurately estimating the geometry of unobserved areas remains an unsolved challenge. Recent studies have shifted towards enhancing map fidelity and completeness. The introduction of neural 3D modelling techniques exemplified by Neural Radiance Fields (NeRF) [14], known for their high-quality novel view synthesis, has inspired a surge of dense neural SLAM approaches [15–22]. Despite notable advances in scene representation quality, these methods suffer from slow training, rendering speeds and an implicit scene representation that is difficult to manipulate or integrate into downstream tasks. Recently, 3D Gaussian Splatting (3DGS) [23] achieves rendering performance on par with NeRF while significantly accelerating both rendering and training by an order of magnitude. This representation is explicit and manipulable, making it a promising candidate for downstream tasks. SLAM methods that leverage 3DGS [24–28] have shown strong performance.

Despite recent progress in Gaussian-based SLAM, existing 3DGS methods [24–28] often suffer from inaccurate geometry, lack reliable normal rendering and rely heavily on frame-to-model tracking, which limits global consistency in long sequences or under noisy depth observations. To address these issues, recent work such as 2DGS-SLAM [29] replaces 3D Gaussian representations with 2D Gaussian Splatting (2DGS) [30], achieving improved geometric fidelity and more accurate surface-aware rendering. However, 2DGS-SLAM [29] continuously maintains a single global Gaussian map throughout optimisation, which introduces substantial computational overhead and limits scalability in long sequences. In addition, its loop relocalisation relies on re-rendered observations from the reconstructed map, making loop quality highly dependent on map quality and thus reducing robustness in challenging dynamic environments.

To address these limitations, this paper introduces a SLAM framework based on 2D Gaussian representations, designed to achieve high-accuracy scene reconstruction with RGB-D observations. Our approach improves geometric estimation by integrating geometric priors with learning-based dense SLAM, utilising 2DGS [30] as a concise map representation to achieve efficient and precise scene modelling. We construct the map using 2DGS [30] and use the existing map to guide camera pose optimisation. Different from 2DGS-SLAM [29], our method adopts an optimisation-based global pose graph framework for loop closure, decoupling relocalisation quality from rendering fidelity. Furthermore, we use an efficient submap-based representation with mesh-bounded merging, improving scalability while maintaining global consistency over long sequences. To further enhance robustness and consistency, we propose a hybrid tracking framework: camera poses are first refined through local bundle adjustment based on Gaussian rendering and then incorporated into a pose graph, where loop closure and global bundle adjustment are performed to ensure trajectory consistency across long sequences. This design tightly couples map training with tracking, enabling improved robustness, efficiency and accuracy. For an overview of our method, see Figure 1.

1. We propose a novel dense RGB-D SLAM system, Geo2DGS-SLAM, that utilises a geometry-aware 2D Gaussian representation for scene modelling. This approach improves mapping quality by enabling accurate depth and normal rendering during tracking, submap construction and merging, leading to more geometrically consistent surface reconstruction.
2. To address the inconsistency and instability of existing 3DGS-based tracking, we propose a hybrid framework combining geometry-aware 2DGS tracking with global optimisation. A rendering-based local bundle adjustment refines pose estimation and provides strong initialisation for pose graph construction, whereas pose graph optimisation with loop closure and global bundle adjustment enforces long-term consistency and corrects drift. This design mitigates pose errors, aligns submaps and yields more accurate and stable trajectories and reconstructions in long or challenging sequences.
3. To tackle visual artefacts caused by submap boundary overlaps and duplicated points, we introduce a submap merging strategy based on mesh binding. By leveraging mesh binding during the merging process, our method effectively preserves the local details of individual submaps while ensuring their seamless integration into the global map. Furthermore, the use of Gaussian regularisation and normal consistency loss refines the global reconstruction, reducing blending artefacts and producing a smoother and more consistent map.

2 | Related Work

2.1 | Dense Visual SLAM

Dense visual SLAM has undergone substantial advancements over the past decade. Traditional SLAM pipelines primarily relied on sparse feature-based methods, which, despite their computational efficiency, typically lacked the granularity required for high-quality surface reconstruction or comprehensive scene interpretation. To address this, dense SLAM methods began leveraging depth sensors to build complete 3D maps with geometric and photometric fidelity. One of the earliest milestones was KinectFusion [7], which utilised the truncated signed distance function (TSDF) [31] and frame-to-model alignment to enable real-time dense reconstruction of small-scale indoor environments.

However, voxel-based methods, such as KinectFusion, suffer from cubic memory growth as the scene scale increases, making

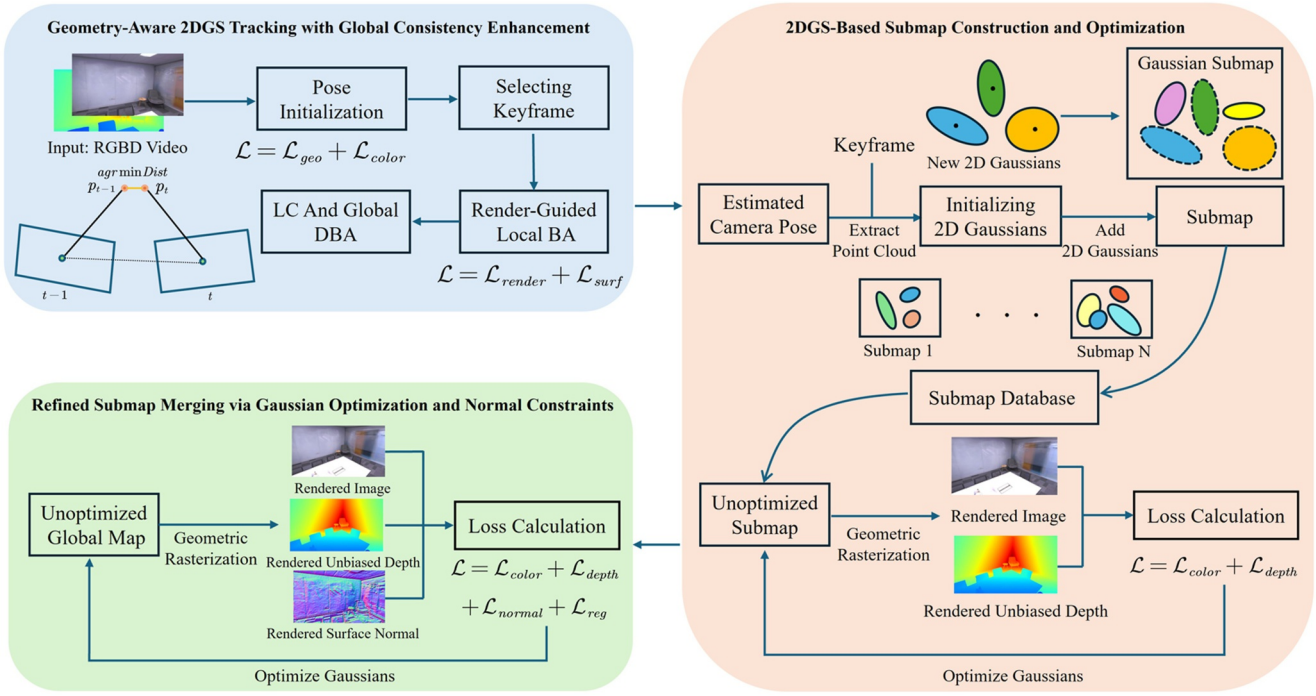


FIGURE 1 | The framework of Geo2DGS-SLAM. Camera poses are first estimated through coarse registration, local bundle adjustment and global pose graph optimisation. Based on the optimised poses, informative pixels from keyframes are back-projected to initialise 2D Gaussian representations in sparse regions. The generated Gaussians are jointly optimised within each submap, followed by mesh-bounded submap merging and global optimisation with geometric priors to improve reconstruction quality and global consistency.

them unsuitable for large or complex environments. To address these limitations, researchers have adopted voxel hashing strategies [12, 32] and hierarchical data structures, such as octrees [33, 34], to reduce memory consumption while maintaining reconstruction quality. For example, Dai et al. [35] proposed BundleFusion, and Yang et al. [36] introduced Vox-Fusion; both approaches significantly reduce memory usage through voxel-based strategies without sacrificing reconstruction accuracy. Similarly, Steinbrucker et al. [13] leveraged hierarchical memory management with octrees to achieve efficient large-scale reconstruction.

2.2 | NeRF-Based SLAM

Several systems have adapted implicit NeRF for online mapping and tracking. For example, iMAP [18] provides a scalable and memory-efficient map representation but suffers from catastrophic forgetting in large-scale scenes. NICE-SLAM [22] combines MLPs with a multi-resolution voxel grid to support large-scale reconstruction, whereas Vox-Fusion [36] uses octree-based extensions to enable dynamic map scalability. ESLAM [16] and Point-SLAM [17] enhance mapping capabilities through the use of triplanes and neural point clouds, respectively.

Despite these initial successes in integrating neural representations into SLAM systems, significant limitations remain. Constrained by the geometric expressiveness of implicit neural models, these methods generally exhibit low surface reconstruction accuracy, especially in scenes with complex structures or occlusions. The state-of-the-art method, Point-SLAM [17], improves geometric reconstruction quality via neural point clouds and achieves high-fidelity surface reconstruction across multiple datasets. However, it still relies on ground-truth depth maps to derive object surfaces, which limits its autonomy during the reconstruction phase.

Moreover, computational efficiency remains a key bottleneck preventing real-time deployment of NeRF-based SLAM systems. Yan et al. proposed MVoxTi-DNeRF [37], a multi-resolution voxel-based approach that reduces rendering and training costs to accelerate NeRF models. However, its performance in large-scale scenes still falls short of meeting the strict real-time requirements of SLAM. Therefore, improving both the real-time capability and geometric accuracy of NeRF-based SLAM remains a critical open challenge.

2.3 | 3D Gaussian Splatting Scene Representation

3D Gaussian Splatting (3DGS) [23] represents scenes using Gaussian primitives, enabling efficient, high-quality real-time rendering and reconstruction with strong compatibility for differentiable optimisation. Recent works have demonstrated that 3DGS can be effectively integrated into SLAM systems for photorealistic mapping and tracking. Huang et al. [25] introduced Photo-SLAM, which combines ORB-SLAM [4] for pose tracking with 3DGS for scene representation. Keetha et al. [26] developed SplaTAM, a dense RGB-D SLAM framework achieving high-accuracy tracking and reconstruction. Matsuki et al. [24] proposed MonoGS, a monocular SLAM system with a Gaussian rasteriser for pose optimisation. Yugay et al. [27] presented Gaussian-SLAM, which uses submap-based mapping for large-scale scenes, and Yan et al. [28] further introduced GS-SLAM with adaptive map expansion and bundle adjustment.

More recent methods explore improved geometry-aware and global consistency in Gaussian-SLAM. TVG-SLAM [38] and CG-SLAM [39], similar to our method, both consider geometry-aware tracking. Specifically, TVG-SLAM [38] introduces tri-view geometric constraints to improve pose estimation robustness, whereas CG-SLAM [39] models uncertainty through rendered depth residuals. However, these methods do not explicitly improve geometric consistency at the Gaussian representation level itself. In contrast, our method combines geometric constraints with the geometrically accurate 2DGS representation to further enhance geometric fidelity and tracking robustness. In addition, LoopSplat [40] and RobGS-SLAM [41] introduce loop closure to improve global consistency. Their loop closure frameworks are similar to that of 2DGS-SLAM [29], where relocalisation is achieved through re-rendering optimisation and therefore heavily depends on the quality of the reconstructed map. In contrast, our hybrid framework decouples loop closure quality from map rendering quality, resulting in improved robustness and stability in challenging scenarios. Meanwhile, RGB-only Gaussian-SLAM has also been studied. Splat-SLAM [42] and OpenGS-SLAM [43] investigate global optimisation and scalability under monocular or RGB-only settings, highlighting the challenges of maintaining robustness without ground-truth depth supervision.

3 | Preliminary on 2D Gaussian Splatting

3DGS represents the scene with semi-transparent ellipsoids, parameterised by mean, covariance and opacity:

$$\mathcal{G}(x) = \exp\left(-\frac{1}{2}(x - x_k)^\top \Sigma^{-1}(x - x_k)\right). \tag{1}$$

The covariance Σ is factorised into rotation R and scaling S. Its projection onto the image plane with world-to-camera matrix W and Jacobian J is:

$$\Sigma' = JW\Sigma W^\top J^\top. \tag{2}$$

After Gaussian depth sorting, pixel colour is computed by α blending:

$$C = \sum_{i\in\mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j), \tag{3}$$

where $\mathcal{N}$ denotes Gaussians along the ray.

Because 3DGS lacks view consistency, we adopt 2DGS [30], obtained by compressing one eigenvalue to zero. The 2D Gaussian is defined as:

$$\mathcal{G}(p) = \exp\left(-\frac{u(p)^2 + v(p)^2}{2}\right), \tag{4}$$

where $u(p)$ and $v(p)$ are local coordinates.

Splatting: A 2D Gaussian is defined by its centre p_k, tangent vectors t_u, t_v, scaling $S = [s_u, s_v]$ and rotation $R = [t_u, t_v, t_w]$ with $t_w = t_u \times t_v$. The transformation matrix is:

$$H = \begin{bmatrix} s_u t_u & s_v t_v & 0 & p_k \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} RS & p_k \\ 0 & 1 \end{bmatrix}. \tag{5}$$

With $W_{4\times4}$ denoting the world-to-image projection, the screen point $\mathbf{x}$ is:

$$\mathbf{x} = (xz, yz, z, 1) = WH(u, v, 1, 1)^\top. \tag{6}$$

Ray intersection: To determine the depth of the Gaussian, we need to compute where the viewing ray intersects the Gaussian. We determine this intersection using two planes. The pixel $\mathbf{x} = (x, y)$ is defined as the point where the x-plane and y-plane intersect. The x-plane is characterised by the normal vector $(-1, 0, 0)$ and an offset x, leading to its plane equation $h_x = (-1, 0, 0, x)^\top$. Similarly, the y-plane is defined with normal $(0, -1, 0)$ and offset y, giving $h_y = (0, -1, 0, y)^\top$. Next, we project these two planes onto the local coordinate system of the 2D Gaussian:

$$h_u = (WH)^\top h_x, \quad h_v = (WH)^\top h_y. \tag{7}$$

The local coordinate point of the 2D Gaussian is $(u, v, 1, 1)^\top$. The intersection of the two planes can be obtained using the following equation:

$$h_u(u, v, 1, 1)^\top = h_v(u, v, 1, 1)^\top = 0. \tag{8}$$

Therefore, the solution for the intersection point can be obtained as:

$$u(x) = \frac{h_u^2 h_v^4 - h_u^4 h_v^2}{h_u^1 h_v^2 - h_u^2 h_v^1} \quad v(x) = \frac{h_u^4 h_v^1 - h_u^1 h_v^4}{h_u^1 h_v^2 - h_u^2 h_v^1}, \tag{9}$$

where h_u^i h_v^i are the ith parameters of the plane vector.

Unbiased depth estimation: In 3DGS [23], the z-component of the Gaussian mean is used as depth, which leads to errors when the viewing ray intersects the Gaussian far from its mean. To mitigate this, 2DGS estimates depth via the Ray-Gaussian intersection, adjusting the mean z-value for unbiased estimation:

$$D = \alpha_i \cdot T_i \cdot (s_u q_u + s_v q_v + z), \tag{10}$$

where α_i is the opacity of the ith Gaussian, T_i is the pixel opacity, s is the intersection point in Equation (9) and $q_{1\times2}$ is derived from camera extrinsics and Gaussian parameters. Because each ray intersects only one Gaussian, the equation is solved once per ray, ensuring accuracy with real-time performance.

4 | Methods

4.1 | Geometry-Aware 2DGS Tracking With Global Consistency Enhancement

Existing GS-SLAM systems typically rely on single-view rendering-based optimisation for pose estimation, which suffers from limited geometric awareness and strong dependence on Gaussian rendering quality, leading to inaccurate tracking and poor robustness under challenging conditions. To address these issues, we propose a geometry-aware 2DGS tracking framework with global consistency enhancement. Specifically, we adopt a 2DGS representation to construct more geometrically faithful scene modelling and further introduce explicit bi-view geometric constraints to enforce cross-view consistency during tracking. To

address tracking errors and accumulated drift caused by imperfect Gaussian rendering, we design a hybrid optimisation framework that jointly leverages geometric constraints, loop closure and global bundle adjustment to reduce the strong dependence of tracking on rendering quality and maintain globally consistent trajectories and reconstructions in long and challenging sequences.

Pose initialisation: At frame t, given the colour point clouds of the current frame P_t and the previous frame P_{t-1}, the goal is to estimate the camera pose $T_{t-1\to t} \in \mathrm{SE}(3)$ that aligns them. Here, SE(3) denotes a 3D rigid transformation consisting of rotation and translation. This is formulated as:

$$E\left(T_{t-1\to t}\right) = \lambda E_{geo} + (1-\lambda)E_c, \tag{11}$$

where λ is a weight coefficient that balances the geometric and colour errors. E_{geo} represents the geometric error, defined as:

$$E_{geo} = \sum \left\| n_p \cdot \left(T_{t-1\to t} \cdot P_{t-1} - P_t\right)\right\|^2, \tag{12}$$

where n_p is the normal vector at point P_t and $T_{t-1\to t}$ is the transformation from the previous frame $t-1$ to the current frame t. The colour error E_c is defined as:

$$E_c = \sum \left\| C\left(T_{t-1\to t} \cdot P_{t-1}\right) - C(P_t)\right\|^2, \tag{13}$$

where $C(P)$ denotes the colour intensity of point P. For more details on the formulation and correspondence construction, refer to Ref. [44]. This initialisation provides a coarse initial pose, which serves as a good starting point for subsequent optimisation and enables local bundle adjustment to converge more efficiently.

Local bundle adjustment: After pose initialisation, we refine the camera pose using both re-rendering and 3D point alignment errors. Considering the difficulty in constraining the properties of Gaussians, the optimisation focusses solely on the camera pose, whereas the Gaussian attributes remain unchanged. For nonkeyframes, we optimise the pose using only the re-rendering error:

$$\mathcal{L}_{render} = \sum M_\alpha \cdot M_{in} \cdot \left(\lambda|I - I_1| + (1-\lambda)|D - D_1^*|\right), \tag{14}$$

where the soft alpha mask M_α is directly obtained from the alpha map rendered by the active submap. The pixel opacity is computed as $\alpha = \sum_{i\in\mathcal{N}} \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)$, where a higher α value indicates that more Gaussians contribute to the rendering of the pixel. To suppress unstable regions caused by sparsely distributed Gaussians, we apply a threshold of $\alpha > 0.6$ to form the mask M_a, effectively filtering out Gaussians in low-density areas. The error mask M_{in} filters out all pixels where the colour and depth errors exceed a frame-specific threshold. A rigid body transformation $T_{t-1\to t}$ is applied to deform the Gaussians, and the rendered colour image I_1 and depth image D_1^* from the deformed Gaussians are spatially aligned with the ground-truth colour image I and ground-truth depth image D. The enhanced depth D_1^* is obtained using the unbiased depth estimation described in Equation (10). λ is used to balance the weights of the colour and depth losses.

For keyframes, we jointly optimise the camera pose using both the re-rendering and 3D point alignment errors:

$$\mathcal{L}_{BA} = \mathcal{L}_{render} + \mathcal{L}_{surf}. \tag{15}$$

The surface consistency loss $\mathcal{L}_{surf}$ enforces alignment of 3D surface points across adjacent frames:

$$\mathcal{L}_{surf} = \sum_{\left(P_{t-1}, P_t^*\right)\in\mathcal{M}} \left\| P_{t-1} - T_t \cdot P_t^* \right\|^2, \tag{16}$$

where P_{t-1} is a 3D point in frame $t-1$ and $T_t \cdot P_t^*$ is the back-projection of P_t^* into world coordinates. Here, $T_t = T_{t-1} \cdot T_{t-1\to t}$, with $T_{t-1\to t}$ as the relative pose to be optimised. The correspondence set $\mathcal{M}$ is obtained by applying LightGlue [45] to extract 2D matches and back-projecting them into 3D with depth maps.

Loop closure and global bundle adjustment: To address trajectory drift, 2DGS-SLAM [29] adopts loop closure strategies. Nevertheless, they depend on the rendering loss defined in Equation (14) for loop relocalisation, making loop closure performance rely on rendering quality and limiting model generality. In contrast, our method uses optimisation-based global bundle adjustment combined with loop closure constraints to correct drift, realising map rectification without re-rendering the Gaussian map. Specifically, keyframe selection is performed based on the RAFT [46] operator: when the optical flow magnitude between the current frame and the most recent keyframe exceeds a predefined threshold, the current frame is designated as a keyframe and its RGB-D information is stored in a buffer to support subsequent keyframe graph construction.

During the construction of the keyframe graph (V, E), the poses predicted by Equation (15) are used as strong initial estimates for the keyframe nodes, whereas odometry edges connect adjacent keyframe nodes. For establishing loop edges, we adopt a high-covisibility-based loop detection strategy inspired by GO-SLAM [21]. When a loop edge is added to the keyframe graph, the system triggers a differentiable dense bundle adjustment (DBA) optimisation to jointly refine the poses of the keyframes. The optimisation follows the nonlinear least squares formulation proposed in DROID-SLAM [10], minimising the following objective function to achieve global consistency:

$$\arg\min \sum_{(i,j)\in\varepsilon} \left\| p_{ij}^* - \Pi_c\left(G_{ij} \circ \Pi_c^{-1}\left(p_i, d_i\right)\right)\right\|_{\Sigma_{ij}}^2, \tag{17}$$

where $(i,j) \in \varepsilon$ denotes an edge in the keyframe graph; Π_c and Π_c^{-1} represent the projection and back-projection functions, respectively; p_i is the pixel position back-projected from keyframe KF_i; G_{ij} denotes the pose transformation from keyframe KF_i to KF_j; p_{ij}^* and w_{ij} are the estimated optical flow and the corresponding confidence map, respectively; $\Sigma_{ij} = \mathrm{diag}\left(w_{ij}\right)$, and $\|\cdot\|_\Sigma$ denotes the weighted Mahalanobis distance, which weights the error term according to the confidence w_{ij}. For more optimisation details, refer to DROID-SLAM [10].

Unlike DROID-SLAM [10], which initialises poses with the identity matrix and relies on many interleaved local and global

BA iterations for convergence, our method provides reliable initial poses during keyframe graph construction. Thanks to the fast Gaussian-based tracking and these strong initialisations, the system requires even fewer global BA iterations to optimise odometry and loop edges, achieving higher accuracy with lower computational cost.

4.2 | 2DGS-Based Submap Construction and Optimisation

Similar to Gaussian-SLAM [27], we represent the scene using a collection of submaps, which alleviates catastrophic forgetting and reduces GPU memory usage by avoiding the storage of all Gaussian models simultaneously. However, their method relies on 3DGS for map representation, which suffers from limited geometric expressiveness and hampers the quality of surface reconstruction. To overcome this drawback, we adopt 2DGS as our map representation, allowing for more accurate rendering of depth and surface normals. This geometry-aware representation enhances our ability to capture fine geometric details within the scene.

Submap initialisation: The submap starts from the first frame and expands as new keyframes are input. As the explored area increases, new submaps are needed to cover the previously unobserved regions while avoiding storing all Gaussian models in GPU memory. We do not use a fixed interval to create new submaps (as in Ref. [35]) but instead use an initialisation strategy that depends on camera motion (as in ref. [47]). Specifically, when the translation relative to the starting frame of the active submap surpasses the threshold d_{thre}, or the Euler angle difference exceeds θ_{thre}, a new submap is initialised. At any given time, only the active submap is processed. This approach limits computational cost, ensuring that the optimisation process remains fast even as the larger scene is explored.

Submap construction: Each new keyframe may add 2D Gaussian models to the active submap to represent newly observed portions of the scene. After completing the pose estimation for the current keyframe, a dense point cloud is computed using the keyframe's RGB-D measurements. At the beginning of each submap, we uniformly sample N_{u} points from the keyframe's point cloud in areas with high colour gradients to add new 2D Gaussian models. For subsequent keyframes in the submap, we uniformly sample N_{k} points from regions with alpha values below a threshold for rendering. This ensures that the map extends into sparse regions covered by Gaussian models. New 2D Gaussian models are added using sampled points within the search radius P of the current submap that do not have neighbours.

Submap optimisation: Whenever new 2D Gaussian models are added to the active submap, all 2D Gaussian models in that submap are jointly optimised within the submap over a predefined number of iterations to minimise the loss. We optimise the active submap to recover keyframe depth and colour. We perform optimisation by only focussing on the keyframes in the active submap and applying at least 40% of the iterations. The loss function is defined as:

$$\mathcal{L}_{mapping} = \mathcal{L}_c + \mathcal{L}_d^*, \tag{18}$$

where $\mathcal{L}_c$ is the colour loss and $\mathcal{L}_d^*$ is the enhanced depth loss.

The difference between the predicted image I_1 and the ground truth I is evaluated using the colour loss $\mathcal{L}_c$, formulated as a weighted combination of pixel-wise difference and SSIM:

$$\mathcal{L}_c = \lambda_c \cdot \mathcal{L}_1 + (1 - \lambda_c) \cdot \mathcal{L}_{\mathrm{SSIM}}, \tag{19}$$

where the parameter $\lambda_c = 0.8$ balances the contributions of the $\mathcal{L}_1$ loss and the $\mathcal{L}_{\mathrm{SSIM}}$.

$\mathcal{L}_d^*$ represents the enhanced depth loss between the predicted depth D_1^* and the ground-truth depth D:

$$\mathcal{L}_d^* = |D - D_1^*|, \tag{20}$$

where D_1^* denotes the enhanced depth estimated using unbiased depth estimation introduced in Equation (10).

To accelerate the optimisation process, we adopt a simplified appearance model by optimising only the zeroth-order spherical harmonic coefficients.

4.3 | Refined Submap Merging via Gaussian Optimisation and Normal Constraints

A naive merging strategy first aggregates all submap point clouds into a global map and then follows 3DGS [23] by randomly selecting frames for global optimisation. However, in long sequences, this introduces spatiotemporal inconsistencies and degrades local accuracy. To address this, we propose a surface-constrained Gaussian binding strategy. We first apply TSDF [31] fusion on 2DGS-rendered depth maps to reconstruct submap meshes and then bind Gaussians to the mesh vertices. During global optimisation, we freeze the x and y components of Gaussian means while refining the z component along with colour, orientation and shape. A Gaussian regularisation loss suppresses artefacts, and a normal loss enforces geometric consistency. This preserves fine local geometry while ensuring global structural alignment.

Regularisation: Large Gaussians may introduce artefacts due to perspective projection. To prevent this, we constrain Gaussian scales using a scale regularisation loss:

$$\mathcal{L}_{scale} = \lambda_{scale} \cdot \frac{\sum_{i,j} \left| eig_j\left(\mathcal{G}(p)_i\right) \right| \cdot \mathbf{1}\left(eig_j\left(\mathcal{G}(p)_i\right) > e_{\max}\right)}{N + 1}, \tag{21}$$

where λ_{scale} is the regularisation weight, $eig_j\left(\mathcal{G}(p)_i\right)$ is the jth eigenvalue of the ith Gaussian, $e_{\max}$ is the maximum allowed scale and N is the number of Gaussians exceeding this threshold. This loss penalises Gaussians that grow too large during training.

To prevent excessive stretching of Gaussians, previous work [27] used isotropic regularisation, which limits Gaussian diversity and reduces rendering quality. We instead use a scale ratio regularisation to allow moderate anisotropy while constraining extreme axis ratios:

$$\mathcal{L}_{ratio} = \lambda_{ratio} \cdot \frac{\sum_i \left| \frac{eig_{\max}(\mathcal{G}(p)_i)}{eig_{\min}(\mathcal{G}(p)_i)} \right| \cdot \mathbf{1}\left(\frac{eig_{\max}(\mathcal{G}(p)_i)}{eig_{\min}(\mathcal{G}(p)_i)} > r_{\max} \right)}{N+1}, \tag{22}$$

where λ_{ratio} is the weight, $eig_{\max}/eig_{\min}$ are the major/minor axes of the ith Gaussian, $r_{\max}$ is the maximum allowed axis ratio and N is the number exceeding it. The total regularisation loss is:

$$\mathcal{L}_{reg} = \mathcal{L}_{scale} + \mathcal{L}_{ratio}. \tag{23}$$

Normal loss: Depth-based optimisation can cause some 2D Gaussians to have incorrect orientations, for example, embedded inside surfaces, increasing depth errors. To address this, we align each 2D Gaussian normal with the object surface normal, improving depth accuracy and stability:

$$\mathcal{L}_n = \lambda_n \sum_i \omega_i (1 - n_i^T N), \tag{24}$$

where ω_i is the blending weight, n_i is the 2D Gaussian normal and N is the surface normal from the depth map. The surface normal is computed as:

$$N(x,y) = \frac{\nabla_x p_s \times \nabla_y p_s}{|\nabla_x p_s \times \nabla_y p_s|}, \tag{25}$$

where $\nabla_x p_s$ and $\nabla_y p_s$ are depth gradients along x and y.

Total loss: Finally, we optimise the global map by minimising the loss function:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_d^* + \mathcal{L}_n + \mathcal{L}_{reg}. \tag{26}$$

5 | Experiments

5.1 | Implementation Details and Parameter Setting

In our experiments, we carefully tuned various parameters to ensure the effective operation of our method. For different datasets, we adjusted the key parameter values as needed. Regarding the number of samples, for the Replica [48] dataset, we set N_u to 600,000, whereas for the TUM-RGB-D [49] and ScanNet [50] datasets, we set N_u to 100,000. The number of N_k was uniformly set to 50,000 across all datasets. Regarding the rendering loss in the tracking module, we set $\lambda = 0.95$ for the Replica dataset and $\lambda = 0.5$ for both the TUM and ScanNet datasets. For iteration settings, the first keyframe in a submap used 1000 iterations for Replica and 100 for TUM and ScanNet, whereas subsequent keyframes used 100 iterations for all datasets. During submap merging, 26,000 iterations were applied to optimise global colour and geometry, with regularisation weights λ_{reg} and λ_n set to 0.05, and the regularisation thresholds defined as the mean of all Gaussian scales and scale ratios. We initialise a new submap when camera motion exceeds predefined thresholds: $d_{thre} > 0.5m$ or $\theta_{thre} > 50°$, following Gaussian-SLAM [27]. To ensure effective optimisation, at least 40% of mapping iterations are assigned to newly added keyframes within each submap. Our keyframe selection and bundle adjustment procedures follow the principles of DROID-SLAM [10]. For loop closure detection, we adopt a GO-SLAM [21]-inspired approach: candidate loops are retrieved in the latent feature space and verified geometrically via pose graph optimisation. Once confirmed, the corresponding keyframes are linked, and the pose graph is jointly optimised with BA to propagate corrections along the trajectory. For scene meshing, for each keyframe along the estimated trajectory, we rendered its depth and colour and used a TSDF [31] fusion method with a voxel size of 1 cm to perform scene meshing, facilitating subsequent analysis and evaluation. All experiments in this paper were conducted on a GeForce RTX 3090 GPU.

5.2 | Evaluation

Datasets: We evaluated our method on the Replica dataset [48], which provides high-quality 3D indoor reconstructions with RGB-D trajectories, and on the real-world TUM-RGB-D [49] and ScanNet [50] datasets. TUM-RGB-D contains Kinect-captured indoor sequences, whereas ScanNet provides camera poses estimated with BundleFusion [35], enabling a fair evaluation of performance in realistic indoor scenarios.

Comparison methods: We compare the proposed Geo2DGS-SLAM method with the state-of-the-art dense neural RGB-D SLAM method, Point-SLAM [17]. In addition, we evaluate against concurrent Gaussian-based SLAM approaches, including SplaTAM [26], MonoGS [24], GS-SLAM [28] and Gaussian-SLAM [27], while also discussing the concurrent 2DGS-SLAM [29], which was developed independently and contemporaneously with our work.

Evaluation metrics: We evaluated performance using multiple metrics. Tracking accuracy is measured by Absolute Trajectory Error RMSE (ATE) [49], where lower is better. Rendering quality is assessed via PSNR, SSIM [51] and LPIPS [52] on full-resolution renderings along the estimated trajectory. Reconstruction is quantified on marching cubes meshes [53] using the F1-score, and depth accuracy on unobserved views is reported using the L1 metric [22].

Tracking: We report the tracking performance in Tables 1–3. Our method outperforms 3DGS-based approaches on both synthetic and real-world datasets. Compared with 2DGS-SLAM, our method achieves slightly lower performance on Replica but delivers comparable results on TUM-RGB-D and ScanNet. The main reason is that 2DGS-SLAM leverages MASt3R [56] to detect loop frames and perform registration, where loop poses are refined via re-rendering-based localisation, which shows better performance on datasets such as Replica due to its high rendering fidelity. However, this loop correction strategy relies on the quality of the reconstructed Gaussian map, and its effectiveness can degrade when the map is incomplete or noisy. In contrast, our method adopts an optimisation-based scheme that can accomplish loop alignment using only RGB-D images, without requiring the reconstructed map, which makes it more generally applicable. MonoGS mainly optimises Gaussian colour attributes without explicitly refining geometry or depth, so camera tracking primarily relies on photometric consistency. In contrast, our method jointly incorporates photometric and geometric constraints during pose optimisation. On the synthetic Off2 and Off3 datasets, the rendered appearances are highly clean and photorealistic, allowing reliable tracking based mainly on photometric consistency. Meanwhile, the reconstructed geometry in these sequences is less accurate, as indicated by the relatively high depth

TABLE 1 | Quantitative comparison of reconstruction/rendering/tracking performance on a replica dataset.

Method	Scene	Category	Metric	Rm0	Rm1	Rm2	Off0	Off1	Off2	Off3	Off4	Avg.
Point-SLAM [17]	NeRF	Reconstruction	Depth L1 [cm]↓	0.53	0.22	0.46	0.30	0.57	**0.49**	**0.51**	0.46	**0.44**
			F1 [%]↑	86.9	**92.3**	90.8	93.8	**91.6**	**89.0**	88.2	85.6	89.8
		Rendering	PSNR↑	32.40	34.08	35.50	38.26	39.16	33.99	33.48	33.49	35.17
			SSIM↑	0.974	0.977	0.982	0.983	0.986	0.960	0.960	0.979	0.975
			LPIPS↓	0.113	0.116	0.111	0.100	0.118	0.156	0.132	0.142	0.124
		Tracking	ATE↓	0.61	0.41	0.37	0.38	0.48	0.54	0.72	0.63	0.52
SplaTAM [26]	3DGS	Reconstruction	Depth L1 [cm]↓	0.43	0.38	0.54	0.44	0.66	1.05	1.60	0.68	0.72
			F1 [%]↑	89.3	88.2	88.0	91.7	90.0	85.1	77.1	80.1	86.1
		Rendering	PSNR↑	32.86	33.89	35.25	38.26	39.17	31.97	29.70	31.81	34.11
			SSIM↑	0.98	0.97	0.98	0.98	0.97	0.95	0.95	0.97	0.97
			LPIPS↓	0.07	0.1	0.09	0.09	0.10	0.12	0.15	0.10	0.10
		Tracking	ATE↓	0.31	0.40	0.29	0.47	0.27	0.29	0.32	0.55	0.36
MonoGS [24]	3DGS	Reconstruction	Depth L1 [cm]↓	3.24	3.38	5.02	0.83	5.82	7.09	5.24	4.69	4.41
			F1 [%]↑	33.45	28.36	25.73	82.26	21.33	22.65	25.90	26.39	33.27
		Rendering	PSNR↑	34.83	36.43	37.49	39.95	42.09	36.24	36.70	36.07	37.50
			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
		Tracking	ATE↓	0.33	0.22	0.29	0.36	0.19	0.25	0.12	0.81	0.32
GS-SLAM [28]	3DGS	Reconstruction	Depth L1 [cm]↓	1.31	0.82	1.26	0.81	0.96	1.41	1.53	1.08	1.16
			F1 [%]↑	62.89	79.85	66.84	80.03	81.55	65.95	59.17	64.98	70.15
		Rendering	PSNR↑	31.56	32.86	32.59	38.7	41.17	32.36	32.03	32.92	34.27
			SSIM↑	0.968	0.973	0.971	0.986	0.993	0.978	0.970	0.968	0.975
			LPIPS↓	0.094	0.075	0.093	0.050	0.033	0.094	0.110	0.112	0.082
		Tracking	ATE↓	0.48	0.53	0.33	0.52	0.41	0.59	0.46	0.70	0.50
Gaussian-SLAM [27]	3DGS	Reconstruction	Depth L1 [cm]↓	0.61	0.25	0.54	0.50	0.52	0.98	1.63	0.42	0.68
			F1 [%]↑	88.8	91.4	90.5	91.7	90.1	87.3	84.2	87.4	88.9
		Rendering	PSNR↑	30.45	31.72	33.96	37.77	40.66	32.64	33.83	34.82	34.50
			SSIM↑	0.954	0.955	0.977	0.985	0.992	0.974	0.982	0.982	0.975
			LPIPS↓	0.172	0.179	0.137	0.102	0.089	0.156	0.121	0.128	0.136
		Tracking	ATE↓	0.29	0.29	0.22	0.37	0.23	0.41	0.30	0.35	0.31
2DGS-SLAM[a] [29]	2DGS	Reconstruction	Depth L1 [cm]↓	**0.34**	0.21	0.43	**0.27**	0.41	1.08	0.67	**0.28**	0.46
			F1 [%]↑	**90.8**	91.6	90.6	93.1	90.1	87.0	87.6	87.5	89.7
		Rendering	PSNR↑	**35.63**	37.09	**38.47**	**43.14**	42.39	36.33	36.16	**38.80**	38.50
			SSIM↑	0.965	0.968	0.973	0.985	0.980	0.968	0.966	0.971	0.972
			LPIPS↓	**0.044**	**0.048**	**0.050**	**0.029**	**0.046**	**0.049**	0.046	**0.049**	**0.045**
		Tracking	ATE↓	**0.06**	**0.08**	**0.10**	**0.04**	**0.07**	**0.07**	**0.06**	**0.09**	**0.07**
Ours	2DGS	Reconstruction	Depth L1 [cm]↓	0.43	**0.19**	**0.38**	0.28	**0.33**	0.93	0.87	0.32	0.46
			F1 [%]↑	90.6	92.0	**91.1**	**94.0**	**91.6**	88.2	**88.6**	**88.7**	**90.6**
		Rendering	PSNR↑	35.29	**37.14**	37.72	42.27	**43.07**	**37.11**	**37.55**	38.11	**38.53**
			SSIM↑	**0.991**	**0.986**	**0.992**	**0.994**	**0.995**	**0.992**	**0.994**	**0.992**	**0.992**
			LPIPS↓	0.055	0.067	0.055	0.045	0.053	0.063	**0.041**	0.055	0.054
		Tracking	ATE↓	0.28	0.25	0.18	0.20	0.09	0.28	0.25	0.34	0.23

Note: Bold and underlined values indicate the best and second-best performance. It is worth noting that point-slam [17] uses ground-truth depth maps for derivation.
[a]Concurrent works. The reconstruction and rendering results are obtained after offline global optimisation.

reconstruction errors, which weakens the effectiveness of geometric constraints. As a result, MonoGS achieves better tracking performance on these sequences, especially Off3. In real-world datasets, however, factors such as sensor noise, motion blur, illumination changes and incomplete geometry reduce the reliability of appearance-only tracking. Under these conditions, the additional geometric constraints in our method improve tracking accuracy.

TABLE 2 | Quantitative comparison of rendering/tracking performance on the TUM-RGB-D dataset ('—' indicates nonreleased).

Rendered method	Metric	fr1/desk	fr2/xyz	fr3/office	Avg.
Point-SLAM [17]	PSNR↑	13.87	17.56	18.43	16.62
	SSIM↑	0.627	0.708	0.754	0.696
	LPIPS↓	0.544	0.585	0.448	0.526
	ATE↓	4.3	1.3	3.5	3.0
SplaTAM [26]	PSNR↑	22.00	**24.50**	21.90	22.80
	SSIM↑	0.857	**0.947**	0.876	0.893
	LPIPS↓	**0.232**	**0.100**	**0.202**	**0.178**
	ATE↓	3.4	1.2	5.2	3.3
MonoGS [24]	PSNR↑	19.69	19.21	20.71	19.87
	SSIM↑	0.732	0.721	0.777	0.743
	LPIPS↓	0.332	0.314	0.343	0.330
	ATE↓	3.4	1.4	3.5	2.8
Gaussian-SLAM [27]	PSNR↑	21.05	22.52	20.42	21.33
	SSIM↑	0.829	0.893	0.789	0.837
	LPIPS↓	0.306	0.228	0.356	0.297
	ATE↓	2.7	1.3	4.6	2.9
2DGS-SLAM[a] [29]	PSNR↑	—	—	—	—
	SSIM↑	—	—	—	—
	LPIPS↓	—	—	—	—
	ATE↓	1.84	1.16	1.97	1.66
Ours	PSNR↑	**23.72**	24.20	**23.97**	**23.96**
	SSIM↑	**0.897**	0.926	**0.879**	**0.901**
	LPIPS↓	0.277	0.216	0.247	0.247
	ATE↓	**1.7**	**0.9**	**1.6**	**1.4**
Mature method					
Kintinuous [54]	ATE↓	3.7	2.9	3.0	3.2
ElasticFusion [3]	ATE↓	2.0	1.1	1.7	1.6
DROID-SLAM [10]	ATE↓	1.8	0.9	2.8	1.8
ORB-SLAM2 [55]	ATE↓	**1.6**	**0.4**	**1.0**	**1.0**

Note: Bold and underlined values indicate the best and second-best performance.
[a]Concurrent works. The reconstruction and rendering results are obtained after offline global optimisation.

Reconstruction: We report reconstruction results in Table 1. Compared with SOTA methods, our approach achieves superior surface reconstruction by effectively merging fine-grained submap details into a globally consistent map, aided by expressive Gaussian representation and geometric priors. Unlike Point-SLAM [17], our method does not rely on ground-truth depth. Figure 2 shows lower and more uniform depth errors across scenes. Surface reconstructions on TUM and ScanNet (Figure 3) demonstrate accurate geometry: on TUM-desk, the computer is reconstructed with the highest completeness and walls appear smoother; on ScanNet-169, walls remain flat while other methods produce tilted or distorted surfaces; on ScanNet-181, tables and chairs are reconstructed more completely compared with the results from competing methods.

Rendering: Tables 1 to 3 report our rendering results. Our method outperforms 3DGS-based approaches on Replica and competes with 2DGS-SLAM. On TUM, it achieves the highest PSNR and SSIM. On ScanNet, our method attains the best overall rendering. To avoid submap-level overfitting, we evaluate rendering on the global map and re-evaluate Gaussian-SLAM [27] accordingly. Details are in Table 4.

Runtime and memory: The runtime and memory consumption are summarised in Table 5. Our method remains highly competitive in terms of mapping and tracking speed while maintaining the smallest Gaussian model size, with an average of only 18.5 M on the Replica dataset. The reported peak GPU memory usage already includes the global optimisation stage and remains competitive with 3DGS-based methods. Despite jointly optimising all 2D Gaussian submaps, the memory consumption remains moderate for two reasons. First, the submap fusion stage produces compact mesh representations, which reduce the number of required Gaussians. Second, 2DGS is more memory efficient than 3DGS because each Gaussian requires fewer parameters by omitting one covariance dimension. Therefore, the overall GPU memory usage remains relatively low even during global optimisation. Our modified 2DGS renderer achieves

TABLE 3 | Quantitative comparison of rendering/tracking performance on the SCANNET dataset.

Method	Metric	0000	0059	0106	0169	0181	0207	Avg.
Point-SLAM [17]	PSNR↑	21.3	19.48	16.8	18.53	22.27	20.56	19.82
	SSIM↑	0.806	0.765	0.676	0.686	0.823	0.75	0.751
	LPIPS↓	0.485	0.499	0.544	0.542	0.471	0.544	0.514
	ATE↓	10.24	7.81	8.65	22.16	14.77	9.54	12.19
SplaTAM [26]	PSNR↑	19.33	19.27	17.73	21.97	16.76	19.8	19.14
	SSIM↑	0.66	0.792	0.69	0.776	0.683	0.696	0.716
	LPIPS↓	0.438	**0.289**	0.376	**0.281**	0.420	**0.341**	**0.358**
	ATE↓	12.8	10.1	17.7	12.1	11.1	7.5	11.9
MonoGS [24]	PSNR↑	21.13	19.70	21.35	22.44	22.02	20.95	21.26
	SSIM↑	0.723	0.722	0.808	0.781	0.814	0.725	0.762
	LPIPS↓	0.448	0.436	**0.339**	0.362	0.432	0.459	0.412
	ATE↓	9.8	32.1	8.9	10.7	21.8	7.9	15.2
Gaussian-SLAM [27]	PSNR↑	20.74	19.95	20.99	22.16	21.98	22.77	21.43
	SSIM↑	0.703	0.726	0.776	0.754	0.759	0.774	0.749
	LPIPS↓	0.576	0.504	0.471	0.457	0.555	0.503	0.511
	ATE↓	24.75	8.63	11.27	14.59	18.70	14.36	15.38
2DGS-SLAM[a] [29]	PSNR↑	23.36	19.00	20.53	**24.67**	21.27	23.71	22.09
	SSIM↑	0.767	0.729	0.795	0.796	0.821	0.779	0.781
	LPIPS↓	0.440	0.444	0.357	0.362	0.485	0.425	0.418
	ATE↓	6.6	**6.9**	**7.1**	**6.5**	8.2	**6.0**	6.9
Ours	PSNR↑	**23.79**	**22.93**	**24.11**	24.57	**24.89**	**25.79**	**24.35**
	SSIM↑	**0.813**	**0.820**	**0.855**	**0.872**	**0.860**	**0.861**	**0.847**
	LPIPS↓	**0.397**	0.413	0.377	0.316	**0.411**	0.398	0.385
	ATE↓	**5.1**	7.4	**7.1**	7.2	**6.7**	6.3	**6.6**

Note: Bold and underlined values indicate the best and second-best performance.
[a]Concurrent works. The reconstruction and rendering results are obtained after offline global optimisation.

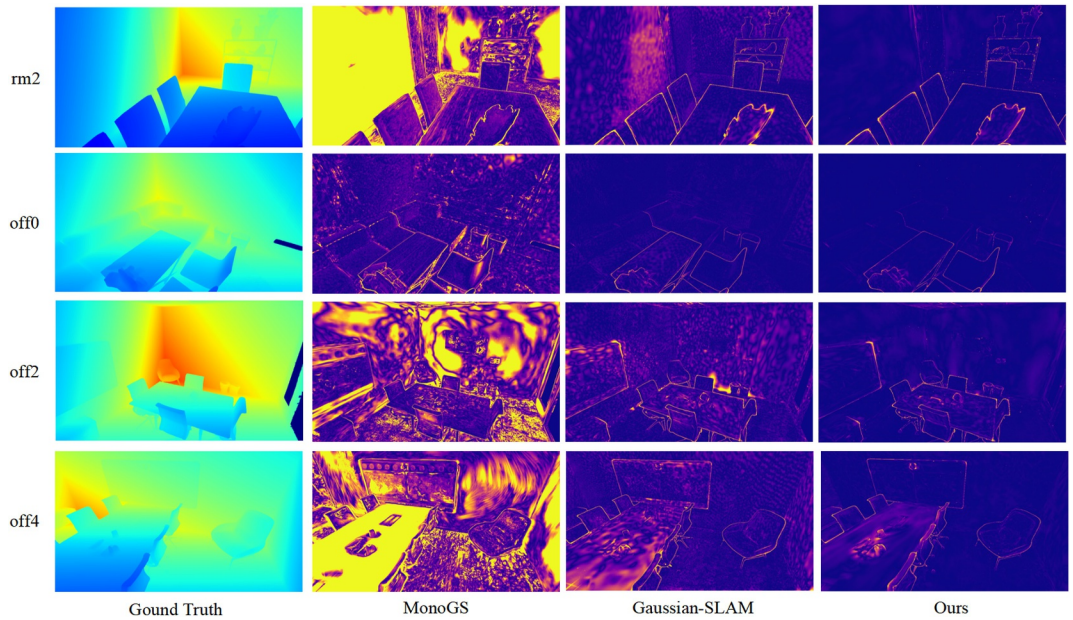


FIGURE 2 | Depth error comparison with different methods on Replica. Our method achieves more accurate depth estimation due to the geometrically faithful 2DGS representation. MonoGS lacks explicit geometric supervision, resulting in relatively large depth estimation errors.

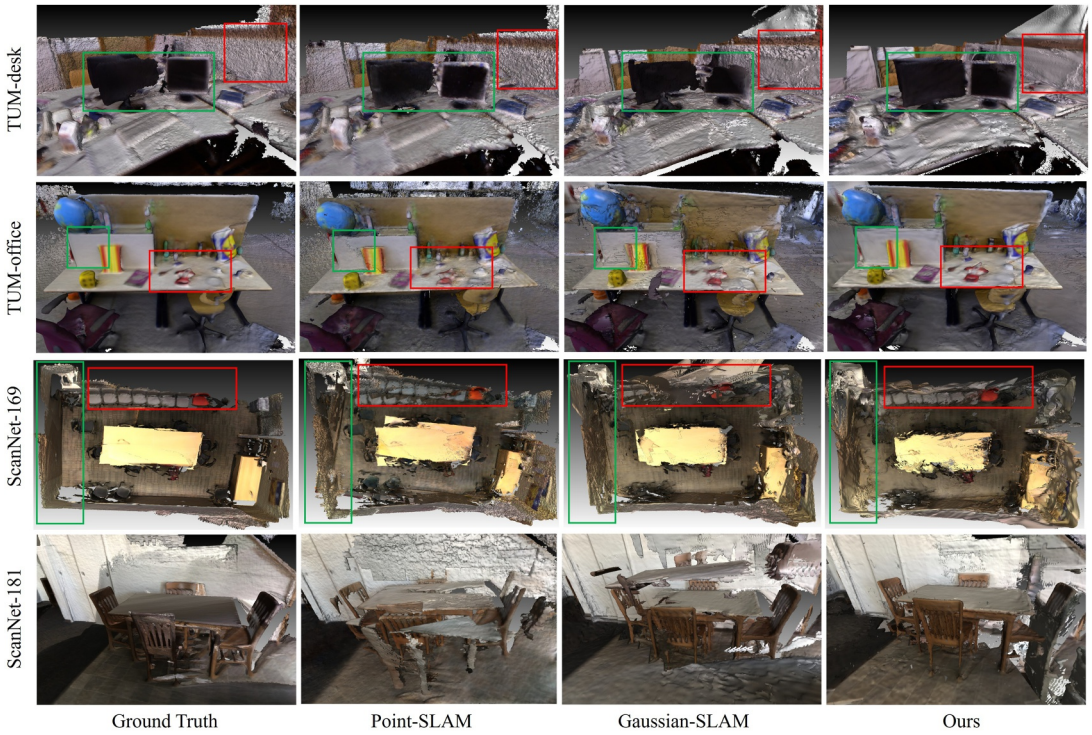


FIGURE 3 | Mesh quality comparison with the state-of-the-art surface reconstruction method. Our method achieves smoother and more structurally consistent surface reconstruction. Although Point-SLAM uses ground-truth depth for surface reconstruction, our method still achieves superior geometric accuracy.

TABLE 4 | Comparison of rendering quality before and after submap merging.

		Replica		TUM		ScanNet	
Method	**Metric**	**Before**	**After**	**Before**	**After**	**Before**	**After**
Gaussian-SLAM + point cloud fusion	PSNR↑	42.08	34.50	25.05	21.33	27.67	21.43
	SSIM↑	0.996	0.954	0.929	0.837	0.923	0.749
	LPIPS↓	0.018	0.172	0.168	0.297	0.248	0.511
Gaussian-SLAM + our merging	PSNR↑	42.08	36.32	25.05	22.54	27.67	22.14
	SSIM↑	0.996	0.979	0.929	0.857	0.923	0.783
	LPIPS↓	0.018	0.093	0.168	0.284	0.248	0.483
Our base model + point cloud fusion	PSNR↑	43.42	36.17	25.98	23.13	29.31	23.29
	SSIM↑	0.997	0.984	0.941	0.863	0.931	0.801
	LPIPS↓	0.014	0.083	0.137	0.273	0.219	0.413
Our full model	PSNR↑	43.42	38.53	25.98	23.96	29.31	24.35
	SSIM↑	0.997	0.992	0.941	0.901	0.931	0.847
	LPIPS↓	0.014	0.054	0.137	0.247	0.219	0.385

Note: 'Point cloud fusion' denotes stacked point clouds. 'Before' refers to metrics on individual submaps and 'after' refers to metrics on the global map.

approximately 1.53 ms per frame, whereas the standard 3DGS renderer runs at around 1.25 ms per frame. Although this makes our method slightly less efficient than standard 3DGS in terms of rendering speed, overall, our method achieves superior performance in memory usage and the other metrics reported in Table 5 compared with 3DGS.

5.3 | Ablation Study

This section presents a set of experiments aimed at exploring the fundamental components and influencing factors of our method. The aim is to comprehend the underlying mechanisms and demonstrate the efficacy of our approach.

Trajectory accuracy: We evaluate the impact of Loop Closure and Global Bundle Adjustment on tracking consistency, especially for long sequences or low-quality depth data. Table 6 shows that adding LC and Global BA to the baseline ($\mathcal{L}_{BA}$) substantially improves accuracy, particularly on ScanNet. Figure 4 presents relative trajectories on scene 181, with our method achieving the lowest error (0.195). Figure 5 illustrates how LC and Global BA correct map distortions. Table 7

TABLE 5 | All metric comparisons (averaged over the replica scene), including tracking, reconstruction, rendering, runtime and memory usage (— means the method is not open-sourced).

Method	LC	Tracking/ frame(s)	Mapping/ frame(s)	Global BA/time (s)	EmbeddingSize (MiB)	Peak GPU use (GiB)	Depth L1↓	F1↑	PSNR↑	SSIM↑	LPIPS↓	ATE↓
Point-SLAM [17]	×	2.78	10.63	0	27.2	**7.7**	**0.44**	89.8	35.17	0.975	0.124	0.52
SplaTAM [26]	×	4.32	7.82	0	308.3	18.5	0.72	86.1	34.11	0.97	0.10	0.36
MonoGS [24]	×	2.14	2.32	0	22.4	22.1	4.41	33.27	34.26	0.961	0.086	0.32
GS-SLAM [28]	×	—	—	0	—	—	1.16	70.15	34.27	0.975	0.082	0.50
Gaussian-SLAM [27]	×	**1.23**	**1.31**	0	48.2	15.0	0.68	88.9	34.50	0.975	0.136	0.31
2DGS-SLAM [29]	✓	—	—	—	—	—	0.46	89.7	38.50	0.972	**0.045**	**0.07**
Ours	✓	1.79	1.68	0.6	**18.5**	8.2	0.46	**90.6**	**38.53**	**0.992**	0.054	0.23

Note: The bold values indicate the best performance.

TABLE 6 | Ablation study of loop closure and global BA on tracking accuracy evaluated by ATE RMSE (lower is better) across SCANNET scenes.

	0000	0059	0106	0169	0181	0207	Avg.
$\mathcal{L}_{BA}$ w/o $\mathcal{L}_{surf}$	24.1	10.2	10.9	18.8	17.9	13.8	16.0
$\mathcal{L}_{BA}$	23.8	9.7	10.7	18.4	17.5	13.6	15.6
$\mathcal{L}_{BA}$ + LC	8.7	7.8	8.5	9.2	8.3	8.6	8.5
$\mathcal{L}_{BA}$ + global BA	8.9	9.1	9.2	9.9	8.5	9.3	9.2
Full model	**5.1**	**7.4**	**7.1**	**7.2**	**6.7**	**6.3**	**6.6**



FIGURE 5 | Comparison of tracking quality via mesh visualisation. The global optimisation effectively corrects the Gaussian map and resolves structural misalignment in the reconstructed scene.

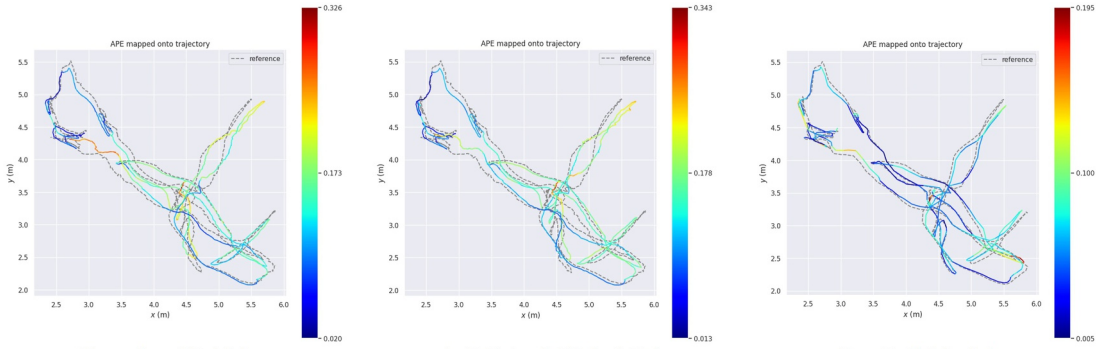


FIGURE 4 | Ablation study on tracking trajectories. Visualisation of tracking trajectories on the ScanNet-181 sequence demonstrates that the global optimisation effectively eliminates large accumulated drift.

compares iteration counts, showing a good balance between computational cost and accuracy.

Component analysis: We evaluate the contribution of each module in Table 8. Removing unbiased depth estimation degrades both rendering and reconstruction. Excluding Gaussian regularisation lowers rendering quality with little effect on reconstruction, as $\mathcal{L}_{reg}$ mainly stabilises Gaussian shapes during long-term optimisation and suppresses rendering artefacts caused by oversized or elongated Gaussians. $\mathcal{L}_n$ improves geometric reconstruction by enhancing local surface smoothness and orientation consistency, especially at the boundaries after submap merging. Table 4 shows that naive point cloud stacking harms global maps, whereas our submap merging preserves local details. Figure 6 illustrates that intersection depth produces smoother, more coherent surfaces than centre depth, and Figure 7 presents a visualisation comparison of Gaussian regularisation.

TABLE 7 | Comparison of optimisation iterations and tracking accuracy on SCANNET-181.

Metric	GO-SLAM	SplaTAM	Gaussian-SLAM	Ours
Render-based tracking iterations (per frame)	0	200	200	100
Global BA iterations (per sequence)	300	0	0	30
ATE ↓	8.3	11.1	18.7	6.7

TABLE 8 | Contribution of different modules to scene rendering and reconstruction quality (averaged over replica scenes).

Method	PSNR↑	SSIM↑	LPIPS↓	F1↑	Depth L1↓
W/o $\mathcal{L}_d^*$	38.26	0.990	0.058	89.5	0.61
W/o $\mathcal{L}_{reg}$	38.34	0.991	0.057	90.6	0.46
W/o $\mathcal{L}_n$	**38.56**	**0.992**	**0.053**	90.1	0.49
Ours	38.53	**0.992**	0.054	**90.6**	**0.46**

Note: The bold values indicate the best performance.

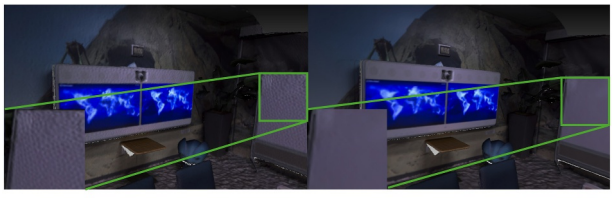


FIGURE 6 | Comparison of depth estimation quality via mesh visualisation. The intersection-based depth estimation used in our method produces smoother and more geometrically coherent surfaces.

5.4 | Limitations

Although we have introduced a globally consistent Gaussian-based SLAM system built upon 2DGS representation, the current framework still faces limitations in dynamic environments, where tracking performance can degrade due to moving objects. On the other hand, the current framework is also limited by the quality of the depth sensor, which can affect tracking stability and reconstruction accuracy in challenging scenarios. In addition, in large-scale scenarios, high-fidelity meshes often introduce significant storage overhead, and extending mesh binding to such settings requires a careful trade-off between reconstruction accuracy and memory efficiency.

6 | Conclusion

In this paper, we propose Geo2DGS-SLAM, a novel dense RGB-D SLAM framework that introduces a geometry-aware 2D Gaussian representation for unified scene modelling, addressing the limitations of 3DGS in geometric consistency, depth accuracy and tracking robustness. Unlike prior methods that focus solely on appearance modelling, our system supports view-consistent rendering of depth and normals, enabling high-fidelity and geometrically consistent scene reconstruction. To address the weak geometric awareness and instability of existing 3DGS-based tracking methods, we design a hybrid tracking framework that combines geometry-aware 2DGS tracking with optimisation-based consistency enhancement. By decoupling tracking quality from reconstruction quality, our framework reduces the dependence of tracking on map quality and significantly improves pose estimation accuracy and trajectory stability. Furthermore, we introduce a mesh-binding submap fusion strategy that preserves local details while ensuring smooth and coherent global integration. Experiments on public indoor RGB-D datasets demonstrate that Geo2DGS-SLAM achieves competitive performance with existing dense SLAM methods in reconstruction quality, rendering fidelity and tracking robustness.

Our current framework mainly focusses on static scene reconstruction and localisation; handling dynamic environments remains an important future direction. In the future, it would be highly valuable to leverage large 3D foundation models to infer submaps from dynamic environments, globally align these dynamic submaps through global optimisation and further incor-

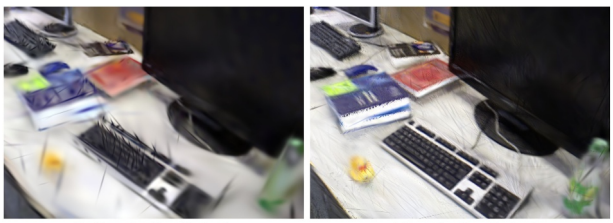


FIGURE 7 | Visualisation of Gaussian regularisation. Gaussian regularisation effectively suppresses spike artefacts caused by elongated Gaussians and perspective blur caused by oversized Gaussians.

porate motion masks for consistent 4D reconstruction, ultimately enabling an end-to-end dynamic SLAM framework.

Funding

This work was supported by the Shandong Provincial Natural Science Foundation (Project No. ZR2024MF060), the National Natural Science Foundation of China (Project Nos. 62471426 and 62576317), the Youth Innovation Team Project of Higher Education Institutions in Shandong Province (Project No. 2022KJ268) and the Science and Technology Innovation Development Plan Projects of Yantai (Project No. 2023JC YJ046).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are openly available in the following repositories: Replica at https://github.com/facebookresearch/Replica-Dataset [48], TUM-RGB-D at https://cvg.cit.tum.de/data/datasets/rgbd-dataset [49] and ScanNet at http://www.scan-net.org [50].

References

1. I. A. Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A Survey of State-of-the-Art on Visual SLAM," *Expert Systems with Applications* 205 (2022): 117734, https://doi.org/10.1016/j.eswa.2022.117734.

2. J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual Simultaneous Localization and Mapping: A Survey," *Artificial Intelligence Review* 43, no. 1 (2015): 55–81, https://doi.org/10.1007/s10462-012-9365-8.

3. T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense SLAM Without a Pose Graph," *Robotics: Science and Systems* 11 (2015): 3, https://doi.org/10.15607/RSS.2015.XI.001.

4. R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics* 31, no. 5 (2015): 1147–1163, https://doi.org/10.1109/tro.2015.2463671.

5. R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2013), 1352–1359.

6. R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-Time," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2011), 2320–2327.

7. R. A. Newcombe, S. Izadi, O. Hilliges, et al., "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality* (IEEE, 2011), 127–136.

8. K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-Time Dense Monocular SLAM With Learned Depth Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2017), 6243–6252.

9. Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari, "Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments," *IEEE Robotics and Automation Letters* 5, no. 4 (2020): 6583–6590, https://doi.org/10.1109/lra.2020.3015456.

10. Z. Teed and J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," *Advances in Neural Information Processing Systems* 34 (2021): 16558–16569, https://doi.org/10.48550/arXiv.2108.10869.

11. T. Schops, T. Sattler, and M. Pollefeys, "BAD SLAM: Bundle Adjusted Direct RGB-D SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2019), 134–144.

12. M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-Time 3D Reconstruction at Scale Using Voxel Hashing," *ACM Transactions on Graphics* 32, no. 6 (2013): 1–11, https://doi.org/10.1145/2508363.2508374.

13. F. Steinbrucker, C. Kerl, and D. Cremers, "Large-Scale Multi-Resolution Surface Reconstruction From RGB-D Sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, (2013), 3264–3271.

14. B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Communications of the ACM* 65, no. 1 (2021): 99–106, https://doi.org/10.1145/3503250.

15. 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/CVF Conference on Computer Vision and Pattern Recognition*, (2021), 8932–8941.

16. 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/CVF Conference on Computer Vision and Pattern Recognition*, (2023), 17408–17419.

17. E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, "Point-Slam: Dense Neural Point Cloud-Based SLAM," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, (2023), 18433–18444.

18. E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "iMAP: Implicit Mapping and Positioning in Real-Time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, (2021), 6229–6238.

19. Y. Tang, J. Zhang, Z. Yu, H. Wang, and K. Xu, "MIPS-Fusion: Multi-Implicit-Submaps for Scalable and Robust Online Neural RGB-D Reconstruction," *ACM Transactions on Graphics* 42, no. 6 (2023): 1–16, https://doi.org/10.1145/3618363.

20. H. Wang, J. Wang, and L. 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*, (2023), 13293–13302.

21. Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, "Go-SLAM: Global Optimization for Consistent 3D Instant Reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, (2023), 3727–3737.

22. Z. Zhu, S. Peng, V. Larsson, et al., "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2022), 12786–12796.

23. B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics* 42, no. 4 (2023): 139, https://doi.org/10.1145/3592433.

24. H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, "Gaussian Splatting SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2024), 18039–18048.

25. H. Huang, L. Li, H. Cheng, and S.-K. Yeung, "Photo-SLAM: Real-Time Simultaneous Localization and Photorealistic Mapping for Monocular Stereo and RGB-D Cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2024), 21584–21593.

26. N. Keetha, J. Karhade, K. M. Jatavallabhula, et al., "SplaTAM: Splat Track & Map 3D Gaussians for Dense RGB-D SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2024), 21357–21366.

27. V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, "Gaussian-SLAM: Photo-Realistic Dense SLAM With Gaussian Splatting," *arXiv preprint arXiv:2312.10070* (2023).

28. C. Yan, D. Qu, D. Xu, et al., "GS-SLAM: Dense Visual SLAM With 3D Gaussian Splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2024), 19595–19604.

29. X. Zhong, Y. Pan, L. Jin, M. Popović, J. Behley, and C. Stachniss, "Globally Consistent RGB-D SLAM With 2D Gaussian Splatting," arXiv preprint arXiv:2506.00970 (2025).

30. B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2D Gaussian Splatting for Geometrically Accurate Radiance Fields," in *ACM Transactions on Graphics (TOG)*, (2024), 1–11.

31. B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models From Range Images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, (1996), 303–312.

32. H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board Mav Planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2017), 1366–1373.

33. J. Chen, D. Bautembach, and S. Izadi, "Scalable Real-Time Volumetric Surface Reconstruction," *ACM Transactions on Graphics* 32, no. 4 (2013): 113, https://doi.org/10.1145/2461912.2461940.

34. L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural Sparse Voxel Fields," *Advances in Neural Information Processing Systems* 33 (2020): 15651–15663, https://doi.org/10.48550/arXiv.2007.11571.

35. A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration," *ACM Transactions on Graphics* 36, no. 4 (2017): 1, https://doi.org/10.1145/3072959.3126814.

36. 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 *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (IEEE, 2022), 499–507.

37. W. Yan, Y. Chen, W. Zhou, and R. Cong, "MVoxTi-DNeRF: Explicit Multi-Scale Voxel Interpolation and Temporal Encoding Network for Efficient Dynamic Neural Radiance Field," *IEEE Transactions on Automation Science and Engineering* 22 (2025): 5096–5107, https://doi.org/10.1109/tase.2024.3416533.

38. Z. Tan, X. Chen, L. Feng, et al., "TVG-SLAM: Robust Gaussian Splatting SLAM With Tri-View Geometric Constraints," *IEEE Robotics and Automation Letters* 11, no. 2 (2025): 1314–1321, https://doi.org/10.1109/lra.2025.3641103.

39. J. Hu, X. Chen, B. Feng, et al., "CG-SLAM: Efficient Dense RGB-D SLAM in a Consistent Uncertainty-Aware 3D Gaussian Field," in *European Conference on Computer Vision* (Springer, 2024), 93–112.

40. L. Zhu, Y. Li, E. Sandström, S. Huang, K. Schindler, and I. Armeni, "LoopSplat: Loop Closure by Registering 3D Gaussian Splats," in *2025 International Conference on 3D Vision (3DV)* (IEEE, 2025), 156–167.

41. Z. Zhu, Y. Fang, X. Li, et al., "Robust Gaussian Splatting SLAM by Leveraging Loop Closure," arXiv preprint arXiv:2409.20111 (2024).

42. E. Sandström, G. Zhang, K. Tateno, et al., "Splat-SLAM: Globally Optimized RGB-Only SLAM With 3D Gaussians," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2025), 1680–1691.

43. S. Yu, C. Cheng, Y. Zhou, X. Yang, and H. Wang, "RGB-Only Gaussian Splatting SLAM for Unbounded Outdoor Scenes," in *2025 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2025), 11068–11074.

44. J. Park, Q.-Y. Zhou, and V. Koltun, "Colored Point Cloud Registration Revisited," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2017), 143–152.

45. P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, "LightGlue: Local Feature Matching at Light Speed," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, (2023), 17627–17638.

46. Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," in *European Conference on Computer Vision* (Springer, 2020), 402–419.

47. Y.-P. Cao, L. Kobbelt, and S.-M. Hu, "Real-Time High-Accuracy Three-Dimensional Reconstruction With Consumer RGB-D Cameras," *ACM Transactions on Graphics* 37, no. 5 (2018): 1–16, https://doi.org/10.1145/3182157.

48. J. Straub, T. Whelan, L. Ma, et al., "The Replica Dataset: A Digital Replica of Indoor Spaces," arXiv preprint arXiv:1906.05797 (2019).

49. J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2012), 573–580.

50. A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2017), 5828–5839.

51. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing* 13, no. 4 (2004): 600–612, https://doi.org/10.1109/tip.2003.819861.

52. R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2018), 586–595.

53. W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *Seminal Graphics: Pioneering Efforts that Shaped the Field*, (1998), 347–353.

54. T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially Extended KinectFusion," in *Proceedings of RSS'12 Workshop on RGB-D: Advanced Reasoning With Depth Cameras*, (2012).

55. 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* 33, no. 5 (2017): 1255–1262, https://doi.org/10.1109/tro.2017.2705103.

56. V. Leroy, Y. Cabon, and J. Revaud, "Grounding Image Matching in 3D With MASt3R," in *European Conference on Computer Vision* (Springer, 2024), 71–91.