Computational Visual Media
https://doi.org/10.26599/CVM.2025.9450513 Vol. 11, No. 6, December 2025, 1195–1208

Research Article

Gaussian-plus-SDF SLAM: High-fidelity 3D reconstruction at 150+ fps

Zhexi Peng[1,2], **Kun Zhou**[1,2] (✉), **and Tianjia Shao**[1,2]

© The Author(s) 2025.

Abstract While recent Gaussian-based SLAM methods achieve photorealistic reconstruction from RGB-D data, their computational performance remains a critical bottleneck. State-of-the-art techniques operate at less than 20 fps, significantly lagging behind geometry-based approaches like KinectFusion (hundreds of fps). This limitation stems from the heavy computational burden: modeling scenes requires numerous Gaussians and complex iterative optimization to fit RGB-D data; insufficient Gaussian counts or optimization iterations cause severe quality degradation. To address this, we propose a Gaussian-SDF hybrid representation, combining a colorized signed distance field (SDF) for smooth geometry and appearance with 3D Gaussians to capture underrepresented details. The SDF is efficiently constructed via RGB-D fusion (as in geometry-based methods), while Gaussians undergo iterative optimization. Our representation enables significant Gaussian reduction (50% fewer) by avoiding full-scene Gaussian modeling, and efficient Gaussian optimization (75% fewer iterations) through targeted appearance refinement. Building upon this representation, we develop GPS-SLAM (Gaussian-plus-SDF SLAM), a real-time 3D reconstruction system achieving over 150 fps on real-world Azure Kinect sequences, faster by an order-of-magnitude than state-of-the-art techniques while maintaining comparable reconstruction quality. The source code and data are available at https://gapszju.github.io/GPS-SLAM.

1 State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China. E-mail: Z. Peng, zhexipeng@zju.edu.cn; K. Zhou, kunzhou@zju.edu.cn (✉); T. Shao, tjshao@zju.edu.cn.

2 Hangzhou Research Institute of AI and Holographic Technology, Hangzhou 310015, China.

Manuscript received: 2025-07-31; accepted: 2025-09-16

Keywords 3D reconstruction; Gaussian splatting; SLAM; RGBD cameras; 3D scanning; signed distance fields

1 Introduction

Recent advances in Gaussian-based RGB-D SLAM have demonstrated impressive capabilities in jointly reconstructing high-quality geometry and photorealistic appearance [1–3]. However, achieving real-time performance remains a significant challenge. The core limitation stems from the computationally expensive online optimization of 3D Gaussians required by these methods. To accurately fit observations based on image loss, a large number of Gaussians must be optimized, a process typically demanding many iterations for convergence. For instance, RTG-SLAM [3], the state-of-the-art Gaussian-based RGB-D SLAM system, achieves real-time reconstruction at 17 fps (59 ms per frame), with Gaussian optimization alone consuming an average of 38 ms per frame. While reducing the number of optimization iterations improves speed, it inevitably compromises reconstruction quality. GS-ICP SLAM [4], for example, reports real-time speeds (e.g., 100 fps) by decoupling camera tracking and scene mapping into separate threads, where mapping operates slower than tracking. Consequently, reconstruction terminates prematurely upon tracking completion, before optimization converges, leading to significantly degraded reconstruction quality, as we report in Table 1 and Fig. 6.

Conversely, traditional geometry-based RGB-D SLAM methods—exemplified by KinectFusion [5] and its derivatives [6–9]—achieve real-time performance at hundreds of fps. These approaches represent scenes

清華大學出版社 Tsinghua University Press Available on IEEE Xplore®

Table 1 Comparison of runtime performance and rendering quality on the Replica, ScanNet++, and our indoor datasets. Our indoor dataset was scanned by ourselves with an Azure Kinect RGB-D camera, and contains five real-world indoor scenes. Our system runs at only 79 fps on ScanNet++ because of its much higher resolution (1752×1168, which is 2.2× that of standard RGB-D images). The numbers for memory indicate usage in MB; × means out of memory

Method	Metric	Replica	ScanNet++	Indoor
GauS-SLAM	FPS↑	0.97	×	0.83
	Memory↓	13,682	×	14,930
	PSNR↑	**39.96**	×	29.18
	SSIM↑	**0.980**	×	0.881
	LPIPS↓	**0.052**	×	0.277
RTG-SLAM	FPS↑	17.31	4.58	14.82
	Memory↓	**2882**	**5120**	**3319**
	PSNR↑	34.73	**29.08**	30.03
	SSIM↑	0.972	0.911	0.908
	LPIPS↓	0.119	0.209	0.264
GSFusion	FPS↑	14.16	4.26	15.33
	Memory↓	7977	6415	7817
	PSNR↑	35.12	28.84	**30.99**
	SSIM↑	0.953	0.897	**0.919**
	LPIPS↓	0.148	0.216	0.238
GS-ICP SLAM	FPS↑	166.39	×	114.18
	Memory↓	5893	×	10,251
	PSNR↑	33.15	×	27.05
	SSIM↑	0.934	×	0.874
	LPIPS↓	0.158	×	0.287
GPS-SLAM (ours)	FPS↑	**252.64**	**79.18**	**151.00**
	Memory↓	3979	8870	4098
	PSNR↑	37.24	28.97	30.19
	SSIM↑	0.960	**0.918**	0.907
	LPIPS↓	0.103	**0.206**	**0.230**

using volumetric truncated signed distance fields (SDF) and fuse each incoming depth and RGB frame directly into the SDF volume. The fusion process is highly efficient, averaging approximately 0.1 ms per frame. However, it is well known that this direct fusion strategy introduces significant color artifacts, including blur and texture holes resulting from depth occlusion or sensor limitations. Consequently, the photometric quality of the reconstructions produced by these methods is generally inferior to that achieved by Gaussian-based approaches.

In this paper, we introduce a novel method that combines the ultra-fast reconstruction speed of geometry-based SLAM with the photorealistic rendering quality of Gaussian-based SLAM. Our core contribution is a hybrid scene representation, Gaussian-plus-SDF, comprising a core SDF representation providing base geometry and color, and an optimizable Gaussian overlay dedicated to appearance refinement. The SDF serves as the foundational scene model, delivering 3D structure and initial color. The 3D Gaussians are then optimized via photometric loss to correct color inaccuracies within the SDF (e.g., blur, missing data) and to model high-frequency appearance details beyond the SDF's capacity. A key insight underpinning our approach is that the SDF already provides a geometrically coherent scene initialization. This fundamentally simplifies the role of the Gaussians: instead of modeling the entire scene geometry and appearance from scratch, their task is reduced to efficient appearance correction and enhancement. This shift yields two significant advantages: the number of required Gaussians is drastically reduced, and the optimization complexity is substantially lowered, enabling rapid convergence. Consequently, the computational overhead introduced by Gaussian optimization is minimized, while retaining the same efficient SDF fusion speed as KinectFusion [5, 9]. This synergy enables our method to achieve ultra-fast, high-fidelity reconstruction.

Technically, Gaussian-plus-SDF is a hybrid representation combining an SDF with a 3D Gaussian radiance field. The SDF volume stores truncated distance values and initial RGB colors on surface voxels, while a sparse set of 3D Gaussians distributed around the reconstructed surface forms a volumetric radiance field to correct residual color errors. The rendering pipeline operates in two stages. In the first stage, standard ray-casting is performed on the SDF volume to generate a surface depth map and an initial SDF-rendered color image. In the second stage, 3D Gaussians are rendered with depth-culling using the surface depth map. Their colors are accumulated per pixel into the color image via order-independent blending. This sorting-free Gaussian rendering significantly speeds up the forward rendering process [10]. More importantly, by shifting parallelization from the pixel level to Gaussian level, we avoid inefficient atomic operations during gradient accumulation in backpropagation, and achieve significant speedup in optimization.

We leveraging our Gaussian-SDF hybrid representation in GPS-SLAM (Gaussian-plus-SDF SLAM), an ultra-fast RGB-D SLAM system for real-time

3D reconstruction. Built upon the InfiniTAM framework [9, 11], it processes each input frame through three stages: estimating camera pose via SDF-based alignment, integrating depth and RGB data into the SDF volume, and optimizing 3D Gaussians via photometric loss minimization, dynamically managed by our high-efficiency Gaussian insertion and removal strategy. We have evaluated the system extensively on three benchmarks: Replica [12], TUM-RGBD [13], and ScanNet++ [14], plus diverse real-world scenes. Compared to state-of-the-art Gaussian-based RGB-D SLAM systems, our method achieves much faster reconstruction (150+ fps for real-world scenes and 250+ fps for Replica scenes), which is an order of magnitude faster than current best methods, while maintaining comparable high-quality reconstruction and photorealistic appearance.

2 Related work

2.1 Classical RGB-D volumetric SLAM

Extensive research has been conducted on online 3D reconstruction with RGB-D cameras. Following the advent of KinectFusion [5], volumetric mapping utilizing SDFs has emerged as a popular reconstruction representation. The original volumetric mapping was limited to small scenes due to its large memory overhead. To address this issue, octree-based surface representations [15–17] have been proposed as alternatives to uniform voxel grids to reduce memory cost and computation time. On the other hand, Ref. [7] utilizes a spatial hash table to represent scenes. InfiniTAM redesigns the data structure to allow for much faster read and write operations, achieving hundreds of fps. However, these existing methods focus on geometry reconstruction but struggle with realistic texture due to simple color averaging in the voxels. Our method addresses this problem by extending InfiniTAM with radiance fields, achieving photorealistic rendering during ultra-fast reconstruction.

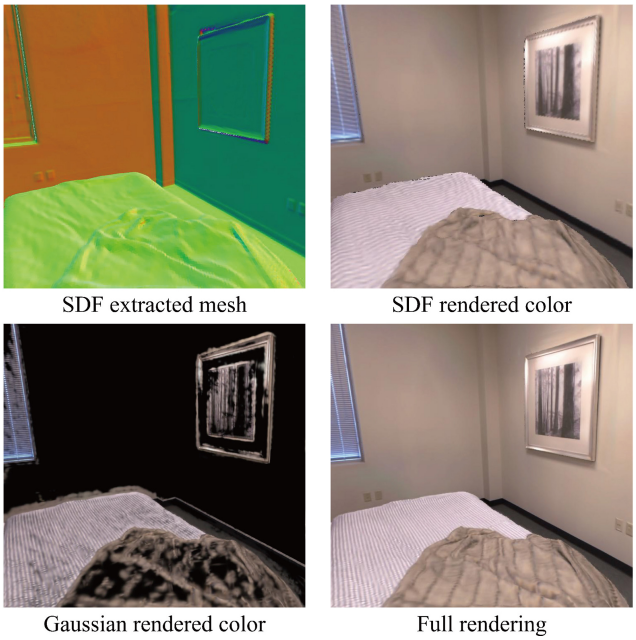

Fig. 1 In our Gaussian-plus-SDF representation, the SDF provides 3D structure and initial color, while the 3D Gaussians are optimized to correct residual color errors. This allows us to achieve high-fidelity reconstruction with a small number of Gaussians, enabling ultra-fast scene reconstruction.

2.2 NeRF-based RGB-D dense SLAM

Following the breakthrough success of neural radiance fields (NeRFs) [18], researchers have increasingly explored the integration of NeRFs into RGB-D dense SLAM systems. A pioneering effort in this direction is iMap [19], which introduced a NeRF SLAM method employing a single MLP for scene representation. Subsequent advances include NICE-SLAM [20], which utilizes hierarchical feature grids and pre-trained MLPs for scene decoding. Vox-Fusion [21] adopts a voxel-based neural implicit surface representation, efficiently stored using octrees. ESLAM [22] and Co-SLAM [23] leverage multi-resolution feature grids and hash grids, respectively, for scene representation. Currently, the state-of-the-art NeRF SLAM system, HS-SLAM [24], uses a hybrid encoding network that combines hash grids, tri-planes, and one-blob to improve the completeness and smoothness of reconstruction. On the other hand, Point-SLAM [25] and Loopy-SLAM [26] employ neural point embedding for high-quality dense reconstruction. Despite achieving high visual quality, these methods are limited by the computational burden of volume rendering, hindering real-time performance and increasing memory consumption.

2.3 Gaussian-based RGB-D dense SLAM

3D Gaussians [27] achieve high-quality novel view synthesis and real-time rendering through differentiable rasterization. This has led to research into employing 3D Gaussians as a map representation within SLAM systems [1, 28–30]. RTG-SLAM proposes a compact Gaussian representation, enabling

清華大學出版社 Tsinghua University Press Available on IEEE Xplore®

reconstruction speeds of around 15 fps for real scenes. However, this still falls short of the speeds of traditional SLAM methods. GS-ICP SLAM achieves a reconstruction speed of 100 fps by completely decoupling Gaussian optimization from camera tracking. However, it is not a purely real-time system because it does not optimize the newly captured frame. GauS-SLAM [2] delivers superior tracking precision and rendering fidelity based on 2D Gaussian surfels, but the reconstruction speed is quite slow. GSFusion [31] leverages octree-based SDF to reconstruct geometry, while Gaussians are used solely to represent texture. This approach significantly reduces the number of Gaussians required, but the complex Gaussian initialization mechanism limits its runtime performance and results in poor rendering quality.

2.4 Other work

Our work is also related to the offline 3D reconstruction approach GSDF [32], which introduces a dual-branch architecture combining neural SDF and 3DGS to enhance reconstruction and rendering. While their SDF and 3DGS in GSDF are two independent representations, our Gaussian-plus-SDF is a hybrid representation in which the SDF represents the 3D scene with smooth geometry and initial colors, and the 3D Gaussians correct color errors. Our representation is inspired by the most recent work GESs (Gaussian-enhanced surfels) [10], which employs a bi-scale approach for radiance field rendering, using surfels for coarse scene representation and 3D Gaussians to enhance fine-scale details, achieving 3.6× the rendering speed of 3DGS. However, GESs is also designed for offline reconstruction and is difficult to apply to online reconstruction. In contrast, our representation can be optimized very quickly and enables real-time online reconstruction.

3 Method

An overview of our SLAM pipeline is provided in Fig. 2. In Section 3.1, we first introduce our Gaussian-plus-SDF representation, and the corresponding rendering process (see Fig. 3). Next, in Section 3.2, we describe in detail the entire online reconstruction process based on the Gaussian-plus-SDF representation.

3.1 Gaussian-plus-SDF

Gaussian-plus-SDF is a hybrid representation of the signed distance field and the radiance field. It uses an SDF volume to store SDF values and RGB colors on voxels, representing the 3D scene surfaces with

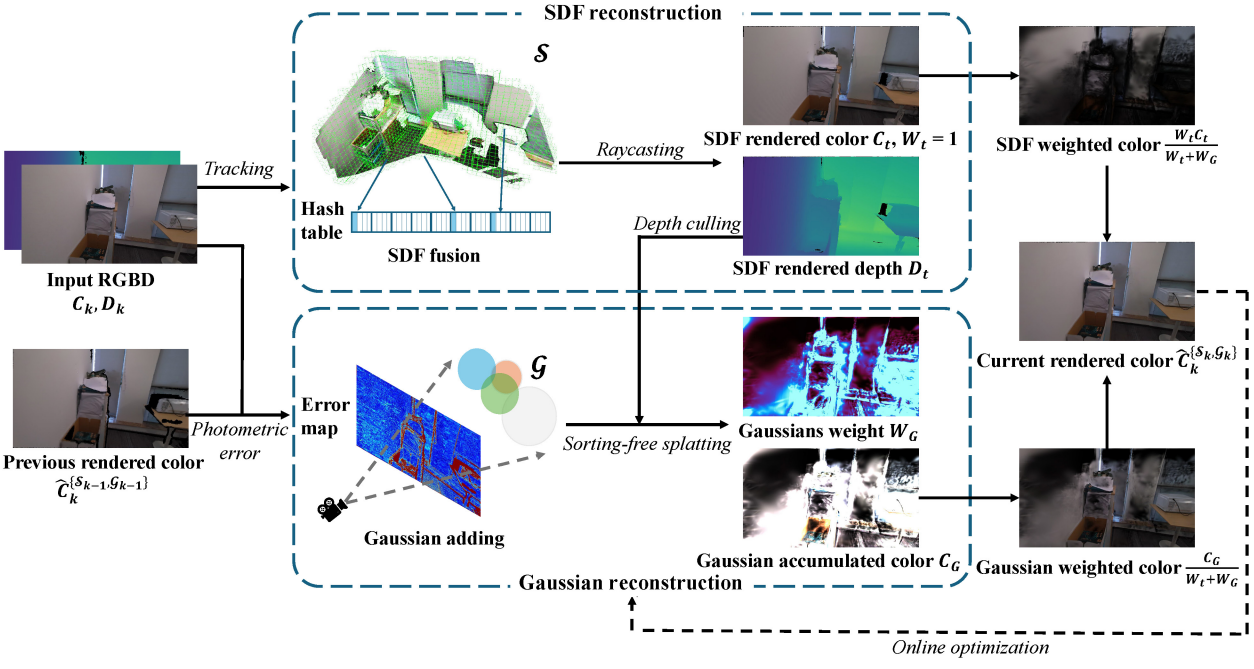

Fig. 2 Overview of our GPS-SLAM system. Given an RGB-D image as input, standard SDF fusion is performed to update the SDF and color values in a global hash table. Then, we sample pixels within regions exhibiting significant color errors and add Gaussians to these locations. Online optimization for 3D Gaussians is executed based on our Gaussian-plus-SDF rendering.

清華大學出版社 Tsinghua University Press Available on IEEE Xplore®

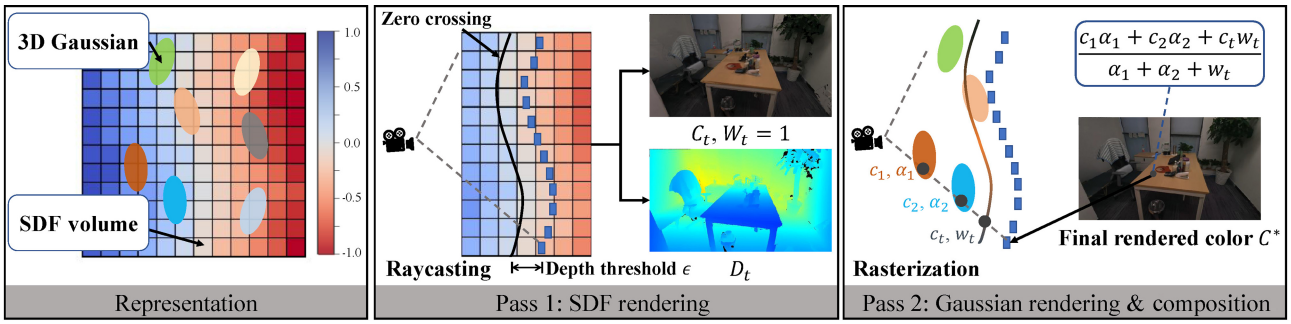

Fig. 3 Gaussian-plus-SDF representation and rendering process. Our representation consists of an SDF volume $\mathcal{S}$ and a set of 3D Gaussians $\mathcal{G} = \{\boldsymbol{p}_i, \sigma_i, \boldsymbol{r}_i, \boldsymbol{s}_i, \mathbf{SH}_i\}_{i=1}^{M}$, with properties: position $\boldsymbol{p}_i$, maximum opacity σ_i, scaling $\boldsymbol{s}_i$, rotation $\boldsymbol{r}_i$, and SH coefficients $\mathbf{SH}_i$. The rendering process consists of two passes. Firstly, standard per-pixel ray-casting is performed on the SDF volume to obtain a color map C_t and depth map D_t. Secondly, we splat the Gaussians to the screen, and their colors and weights are blended together independently of order with depth testing based on the SDF-rendered depth map D_t. A small positive threshold ϵ is used to prevent incorrect truncation. The accumulated Gaussian color and SDF-rendered color are combined as a weighted average to get the final image.

initial colors. A set of 3D Gaussians surrounding the surface forms a volumetric radiance field to correct residual color errors.

Following KinectFusion, the SDF volume $\mathcal{S}$ is defined as a discretization of the SDF with a specified resolution of voxels, where each voxel $\boldsymbol{p}$ stores its truncated signed distance to the scene surface $d(\boldsymbol{p})$ and its RGB color $\boldsymbol{c}(\boldsymbol{p})$. The 3D Gaussians $\mathcal{G}$ are defined as $\mathcal{G} = \{\boldsymbol{p}_i, \sigma_i, \boldsymbol{r}_i, \boldsymbol{s}_i, \mathbf{SH}_i\}_{i=1}^{M}$, with properties: position $\boldsymbol{p}_i$, maximum opacity σ_i, scaling $\boldsymbol{s}_i$, rotation $\boldsymbol{r}_i$ and spherical harmonic (SH) coefficients $\mathbf{SH}_i$, following Ref. [27].

The rendering of Gaussian-plus-SDF consists of two passes. In the first pass, standard per-pixel ray-casting is performed on the SDF volume [7, 9], where the ray for each pixel marches through the SDF voxels to find the zero crossing, as in Ref. [9]. When the zero-crossing point for a pixel $\boldsymbol{u}$ is found, we obtain the 3D surface vertex $\boldsymbol{V}^g(\boldsymbol{u})$ in the world coordinate system. Afterwards, we compute the RGB value $\boldsymbol{C}_t(\boldsymbol{u})$ for each pixel by linearly interpolating the colors of the eight neighboring voxels of $\boldsymbol{V}^g(\boldsymbol{u})$. The depth value $D_t(\boldsymbol{u})$ of the pixel is also obtained by projecting $\boldsymbol{V}^g(\boldsymbol{u})$ onto the image plane. In this way, the SDF-rendered image $\boldsymbol{C}_t$ and the depth map D_t are available after the first rendering pass.

In the second pass, the Gaussians are splatted onto the screen, with their colors and weights accumulated for each pixel in an order-independent way. We conduct depth culling on the Gaussians during Gaussian accumulation, using the SDF-rendered depth map D_t. Gaussians whose centers are occluded by the scene surface are ignored during rendering. Specifically, the accumulated Gaussian color and weight for a pixel $\hat{\boldsymbol{x}}$ are defined as

$$\boldsymbol{C}_G(\hat{\boldsymbol{x}}) = \sum_{i=1}^{K} [\mathbb{1}(d_i < D_t(\hat{\boldsymbol{x}}) + \epsilon)] \boldsymbol{c}_i \alpha_i(\hat{\boldsymbol{x}}) \tag{1}$$

$$W_G(\hat{\boldsymbol{x}}) = \sum_{i=1}^{K} [\mathbb{1}(d_i < D_t(\hat{\boldsymbol{x}}) + \epsilon)] \alpha_i(\hat{\boldsymbol{x}}) \tag{2}$$

$$\alpha_i(\hat{\boldsymbol{x}}) = \sigma_i \exp\left(-\frac{(\hat{\boldsymbol{x}} - \hat{\boldsymbol{p}}_i)^{\mathrm{T}} \boldsymbol{\Sigma}_{2\mathrm{D}}^{-1} (\hat{\boldsymbol{x}} - \hat{\boldsymbol{p}}_i)}{2} \right) \tag{3}$$

where $D_t(\hat{\boldsymbol{x}})$ is the depth of pixel $\hat{\boldsymbol{x}}$ in the SDF-rendered depth map D_t, and d_i is the depth of the Gaussian's center. $\mathbb{1}(\cdot)$ is the indicator function, and ϵ is a small positive threshold introduced to prevent 3D Gaussians distributed close to the surface from being wrongly truncated. $\boldsymbol{\Sigma}_{2\mathrm{D}}$ is the covariance matrix of projected Gaussians in pixel space. $\hat{\boldsymbol{p}}_i$ is the projected Gaussian center, and $\boldsymbol{c}_i$ is the Gaussian color in the viewing direction corresponding to the image. $\alpha_i(\hat{\boldsymbol{x}})$ is clamped to 0 if $\alpha_i(\hat{\boldsymbol{x}}) < 1/255$, as in 3DGS [27].

After the two-pass rendering, we obtain the final rendered image $\boldsymbol{C}^*$ as a weighted combination of the SDF-rendered image and the Gaussian rendered image

$$\boldsymbol{C}^* = \frac{\boldsymbol{C}_t W_t + \boldsymbol{C}_G}{W_t + W_G} \tag{4}$$

Here, $W_t = 1$ is the fixed weight for the SDF color.

The Gaussian colors are accumulated and normalized by weights, so the rendering process of Gaussian-plus-SDF is entirely sorting-free, which also helps GPS-SLAM reach ultra-fast performance. First, in forward rendering, the computation bottleneck of Gaussian sorting in 3DGS is bypassed, boosting the rendering speed. More importantly, during backpropagation in the online optimization in SLAM,

清華大學出版社 Tsinghua University Press Available on IEEE Xplore®

3DGS requires the launching of threads per-pixel and traversing Gaussians in a back-to-front order to compute gradients. This results in slow gradient accumulation for each Gaussian because of the inefficiency of atomic addition. In contrast, we estimate the affected range for each Gaussian based on its pixel-space radius and launch threads per-Gaussian to directly accumulate gradients within each thread, significantly improving computational speed. To address the varying sizes of different Gaussians, we implement a carefully designed thread scheduling mechanism to achieve optimal thread utilization. Further implementation details can be found in Appendix B.

3.2 Online reconstruction process

As Fig. 2 shows, our online reconstruction system has two parts. In the SDF reconstruction part, we perform camera tracking and update the signed distance field, acquiring the 3D structure and initial colors. In the Gaussian reconstruction part, we conduct online Gaussian insertion, Gaussian optimization, and Gaussian removal, obtaining high-fidelity scene appearance with details.

3.2.1 SDF reconstruction

SDF fusion. Given the k-th frame of an RGB-D video stream (i.e., an RGB image $\boldsymbol{C}_k$ and a depth map D_k), we compute the local vertex map $\boldsymbol{V}_k^l$ and the local normal map $\boldsymbol{N}_k^l$ using the intrinsic camera parameters. Using the estimated camera pose $\boldsymbol{T}_{g,k}$, $\boldsymbol{V}_k^l$, and $\boldsymbol{N}_k^l$ are transformed into the global vertex map $\boldsymbol{V}_k^g$ and global normal map $\boldsymbol{N}_k^g$ in world space. Following InfiniTAM, we perform standard SDF fusion to update the SDF and color values in a global hash table. Afterwards, ray-casting is performed, yielding the ray-casting vertex map $\boldsymbol{V}_k^{*,g}$ and normal map $\boldsymbol{N}_k^{*,g}$ in world space. Then the SDF-rendered color map $\boldsymbol{C}_{tk}$ is computed by interpolation using $\boldsymbol{V}_k^{*,g}$, and the SDF-rendered depth map D_{tk} is computed by projecting $\boldsymbol{V}_k^{*,g}$ onto the image plane.

Camera tracking. We adopt the standard ICP method for camera tracking [5, 9], which minimizes the point-to-plane distance as

$$E(\boldsymbol{\xi}) = \sum \left\| (\boldsymbol{T}_{g,k}\boldsymbol{V}_k^l(\boldsymbol{u}) - \boldsymbol{V}_{k-1}^{*,g}(\hat{\boldsymbol{u}})) \cdot \boldsymbol{N}_{k-1}^{*,g}(\hat{\boldsymbol{u}}) \right\| \tag{5}$$

Here, $\boldsymbol{\xi}$ is the Lie algebra representation of the estimated transformation $\boldsymbol{T}_{g,k}$. $\boldsymbol{u}$ is the pixel in the current frame, and $\hat{\boldsymbol{u}} = \pi(\boldsymbol{K}\boldsymbol{T}_{g,k-1}\boldsymbol{V}_k^l(\boldsymbol{u}))$ is the projection of $\boldsymbol{V}_k^l(\boldsymbol{u})$ into the previous frame. $\pi(\cdot)$ performs perspective projection. A resolution hierarchy of the depth map is used in our implementation to improve convergence behavior.

3.2.2 Gaussian reconstruction

Based on the SDF reconstruction, we perform Gaussian reconstruction to correct the residual color errors. Since adjacent views often contain a large amount of redundant information, we perform Gaussian reconstruction at certain time intervals δ_k. Gaussian reconstruction includes Gaussian insertion, Gaussian optimization, and Gaussian removal. Note that Gaussian reconstruction does not affect SDF reconstruction, so the two reconstruction processes can be executed in parallel for speed.

Gaussian insertion. Instead of densifying Gaussians based on gradient magnitude [27], we design an efficient Gaussian insertion strategy explicitly targeting regions with appearance details not well represented by SDF. Specifically, for the k-th frame with SDF-rendered image $\boldsymbol{C}_{tk}$ and depth map $\boldsymbol{D}_{tk}$, we apply the second-pass rendering using the existing Gaussians in the scene to obtain the final rendered image $\boldsymbol{C}_k^*$ and the Gaussian weight map W_{Gk}. Then a mask $M_{\widetilde{\boldsymbol{u}}}$ is created to determine for which pixels a Gaussian should be added:

$$M_{\widetilde{\boldsymbol{u}}} = \{\widetilde{\boldsymbol{u}} \,|\, |\boldsymbol{C}_k^*(\boldsymbol{u}) - \boldsymbol{C}_k(\boldsymbol{u})| > \delta_{\boldsymbol{c}} \text{ and } W_{\mathrm{G}k} < \delta_W\} \tag{6}$$

$M_{\widetilde{\boldsymbol{u}}}$ represents those pixels with apparent color errors but insufficient Gaussian counts. $\delta_{\boldsymbol{c}} = 0.05$ is the threshold of color error and $\delta_W = 4$ is the threshold of Gaussian weights.

While $M_{\widetilde{\boldsymbol{u}}}$ indicates regions where Gaussians might be needed, the substantial GPU memory requirement of inserting Gaussians for all $M_{\widetilde{\boldsymbol{u}}}$ significantly impedes real-time performance. Therefore, we uniformly sample 25% pixels on $M_{\widetilde{\boldsymbol{u}}}$ and create a new Gaussian for each sampled pixel $\boldsymbol{u}$. For each Gaussian, the 3D position $\boldsymbol{p}$ and the zero-order component of **SH** are initialized to $\boldsymbol{V}_k^{*,g}(\boldsymbol{u})$ and $\boldsymbol{C}_k(\boldsymbol{u})$. The Gaussian's shape is initialized as a flat surface-aligned disc, as in RTG-SLAM. The shortest axis of the Gaussian is initialized to align with $\boldsymbol{N}_k^{*,g}(\boldsymbol{u})$ and the initial opacity is set to 0.5. Please refer to Appendix A for more details.

Gaussian optimization. After adding and initializing Gaussians in world space, their optimization is guided by selected views. If only recent

frames are optimized, this may result in overfitting issues, degrading the quality of the global Gaussian map. On the other hand, since the SDF is continuously being fused and updated, some Gaussians that were optimized in the past may not fit the latest SDF. Therefore, we use keyframes for Gaussian optimization to ensure global consistency in optimization. Our keyframe selection strategy is inspired by Ref. [33]. The keyframe list is constructed based on the camera motion. If the rotation angle relative to the last keyframe exceeds a threshold δ_{angle}, or the relative translation exceeds δ_{move}, we add a new keyframe. We randomly sample n_{global} global keyframes from the keyframe history and evenly select n_{local} recent local frames from the frames during an optimization time interval δ_k. These frames collectively form the target views for this optimization, as shown in Fig. 4.

After determining the views to be optimized, we use the latest SDF to perform ray-casting on these views, recording the SDF-rendered color images and depth maps to avoid repeated ray-casting during the optimization process. We perform Gaussian optimization based on the color loss between the input and rendered RGB images. We use the L_1 loss for optimization

$$L_{\text{color}} = |\boldsymbol{C}_k^* - \boldsymbol{C}_k| \tag{7}$$

Gaussian removal. Although we consider historical views in the optimization, the number of views used in each optimization is still small to ensure speed. This could potentially lead to overfitting, resulting in some excessively large Gaussians that can significantly impact global quality.

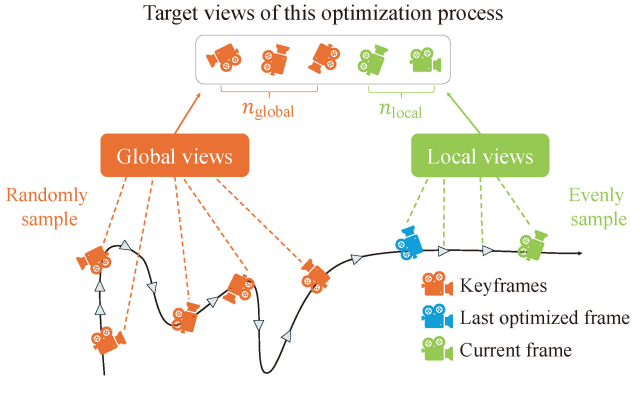

Fig. 4 Optimization view selection. We randomly sample n_{global} views from global keyframes and evenly select n_{local} views from local frames during an optimization time interval δ_k. This approach allows us to optimize recent frames while mitigating both overfitting and catastrophic forgetting.

Meanwhile, the optimization process can also generate some Gaussians with too small scale or opacity. These Gaussians make relatively little contribution to the reconstruction and can be considered redundant. After each Gaussian optimization, we delete Gaussians in the scene that satisfy the criteria in Eq. (8):

$$M_{\mathcal{G}} = \{\mathcal{G}_i | \sigma_i < \delta_\sigma, \text{ or } \max(\boldsymbol{s}_i^{1,2,3}) > \delta_{\boldsymbol{s}_{\max}}, \text{ or } \max(\boldsymbol{s}_i^{1,2,3}) < \delta_{\boldsymbol{s}_{\min}}\} \tag{8}$$

where $\delta_{\boldsymbol{s}_{\max}} = 0.1$ is the threshold for excessively large Gaussians and $\delta_{\boldsymbol{s}_{\min}} = 0.003$ is the threshold for too small Gaussians. $\delta_\sigma = 0.005$ is the threshold for low opacity Gaussians.

4 Evaluation

4.1 Experimental setup

4.1.1 Implementation details

We implemented our SLAM system on a desktop computer with an AMD 9950X3D CPU and a 4 GB NVIDIA RTX 4090 GPU. We developed the entire system in C++ using the Libtorch framework and wrote custom CUDA kernels for rasterization and backpropagation. In all experiments, Gaussian optimization was launched every 10 frames, with 20 iterations performed each time. The voxel size for SDF fusion was set to 0.5 cm. Please refer to Appendix C for further details.

4.1.2 Datasets

We evaluated our method and all baselines on three public datasets: Replica, TUM-RGBD, ScanNet++, and a our own indoor dataset captured by scanning. Replica is the simplest benchmark due to its synthetic, highly accurate, and complete RGB-D images. TUM-RGBD is a widely used dataset in the SLAM field for evaluating tracking accuracy because it provides accurate camera poses from an external motion capture system. We tested on three sequences (fr1_desk, fr2_xyz, fr3_office), which have been commonly used in previous studies. ScanNet++ is a large-scale dataset that combines high-quality and commodity-level geometry and color capture of indoor scenes. Its depth maps are rendered from models reconstructed from laser scanning. Unlike other benchmarks, the camera poses in ScanNet++ are very far apart from one another. Following Refs. [1, 2], two challenging sequences (b20a261fdf,

清華大學出版社 Tsinghua University Press Available on IEEE Xplore®

8b5caf3398) were sampled for evaluation. We also scanned five real-world indoor scenes with an Azure Kinect RGB-D camera to build our own indoor dataset.

4.1.3 Baselines

We compared our method to existing state-of-the-art Gaussian RGB-D SLAM methods including GSFusion, GS-ICP SLAM, and RTG-SLAM. We also compared our method to the concurrent work GauS-SLAM. For GS-ICP SLAM, the number of map iterations varies depending on tracking speed. The authors provided two cases: one with tracking speed limited to 30 fps, and the other with unlimited tracking speed. We used the unlimited mode to evaluate performance under the fastest conditions. We reproduced the results using the published code and conducted all experiments on the same desktop computer.

4.2 Evaluation of online reconstruction

4.2.1 Time/memory performance

To ensure that our system can offer immediate feedback during real-time scanning, we compared its speed and GPU memory usage to those of all baseline approaches: see Table 1. Across all evaluated datasets, our method consistently delivers ultra-fast and stable performance, significantly outperforming other methods, including GSFusion, which is also based on SDF fusion and implemented in C++. Our system runs at only 79 fps on ScanNet++ because of its much higher resolution (1752×1168, which is 2.2× that of standard RGB-D images). Please note that GS-ICP SLAM and GauS-SLAM exhaust the entire 24 GB of GPU memory when processing the high-resolution ScanNet++ dataset. Furthermore, our method achieves 250 fps on the synthetic Replica dataset. The Replica scenes are larger than our indoor scenes, yet the overall memory overhead is similar. It shows that the Gaussian reconstruction on the Replica dataset consumes less memory than that on our indoor dataset. This is attributed to the high-quality SDF reconstruction, which reduces the number of Gaussians required. Figure 5 later illustrates the differences in SDF and Gaussian rendering results between real-world and synthetic datasets.

Furthermore, we conducted a detailed time cost analysis on Replica *office0*. We report in Table 2

Fig. 5 Comparison including images rendered with only SDF or only 3D Gaussians.

the mapping time per frame and the optimization time per frame. We also record the optimization time per iteration, the total iteration count, the total Gaussian count, the PSNR, and the overall system fps. The slow reconstruction of GauS-SLAM and RTG-SLAM arises from their high iteration counts and long optimization times per iteration. GS-ICP SLAM exhibits the fewest iteration counts; however, this is because its tracker operates so quickly that its mapper does not fully optimize the Gaussians, and the rendering quality is significantly degraded. GSFusion accelerates the optimization time by reducing the number of Gaussians. However, its Gaussian initialization mechanism, which relies on a quadtree scheme for each input RGB image, is excessively time-consuming (53.5 ms per frame). In contrast to GSFusion, our method eliminates the complex Gaussian initialization process and achieves superior reconstruction results with fewer iterations.

Table 2 Time costs on Replica *office0* (2000 frames). * indicates that we manually remove GauS-SLAM's tracking time when calculating the overall FPS for a fair comparison, eventhough mapping and tracking are implemented in a single thread in its published code

Method	Mapping /frame	Optimization /frame	Optimization /iteration	Iteration count	Gaussian count	PSNR↑	FPS↑
GauS-SLAM	341.2 ms	336.9 ms	14.3 ms	47,120	901,685	**43.11**	2.93*
RTG-SLAM	58.3 ms	37.6 ms	4.5 ms	16,700	268,779	38.62	17.15
GSFusion	65.0 ms	6.0 ms	1.2 ms	10,000	**112,282**	37.64	15.37
GS-ICP SLAM	5.7 ms	3.6 ms	8.5 ms	**843**	1,679,211	37.33	174.20
GPS-SLAM (ours)	**2.6 ms**	**2.2 ms**	**1.1 ms**	4000	137,200	41.15	**380.72**

4.2.2 Tracking accuracy

We report the average tracking accuracy on both the synthetic Replica dataset and the real-world TUM-RGBD dataset in Table 3. Here, we also include three classic traditional SLAM systems: Kintinuous [34], BundleFusion [8], and ORB-SLAM [35] for a more comprehensive comparison. GauS-SLAM demonstrates high accuracy on both datasets. However, its tracker is slow because it optimizes the camera pose by minimizing the difference between the Gaussian-rendered color and the input image. Among the remaining methods, our method achieves comparable results on the Replica dataset. However, on the TUM-RGBD dataset, which contains inaccurate depth input, our method exhibits relatively lower accuracy, like all ICP-based methods.

4.2.3 Rendering quality

We compared rendering quality on the Replica, ScanNet++, and Indoor datasets; results are shown in Table 1. Since the ScanNet++ dataset contains many areas with missing depth, and consistent with previous methods [2, 4], we only calculate image metrics within regions where depth is available. Our method achieves results comparable to state-of-the-art approaches. Moreover, it has significantly improved image quality compared to GS-ICP SLAM, which is the only other SLAM system running at over 100 fps. The results of GS-ICP SLAM are worse than those reported in its original paper. This is because its tracker runs faster in our environment, resulting in fewer Gaussian optimization iterations. Qualitative comparisons on our indoor dataset are shown in Fig. 6. All other methods exhibit noticeable artifacts and blurring in some regions. Our Gaussian-plus-SDF representation overcomes this limitation, achieving high-quality reconstruction in these regions. We also evaluated the novel view synthesis quality of our method using the test split of ScanNet++. The results are presented in Table 4 and demonstrate that our method can also achieve comparable results in novel view synthesis.

Table 3 Comparison of tracking accuracy on the TUM-RGBD and Replica datasets. Numbers represents absolute trajectory error (ATE) root mean square error (RMSE) in cm

Category	Method	Replica	TUM
Image loss	GauS-SLAM	**0.06**	1.54
ICP loss	GS-ICP SLAM	0.15	2.89
	Kintinuous	0.29	3.20
	GPS-SLAM (ours)	0.17	3.08
ICP loss & backend	BundleFusion	0.16	2.07
	ORB-SLAM2	0.68	**1.00**
	RTG-SLAM	0.18	1.06

4.2.4 Geometry quality

Following NICE-SLAM, we use the following metrics to evaluate the scene geometry on ScanNet++: *Accuracy*, *Completion*, *Accuracy Ratio* (< 3 cm), and *Completion Ratio* (< 3 cm). Following Ref. [20], we remove unseen regions that are not inside any camera's frustum. We extract meshes from the reconstructed SDF volume using marching cubes for GSFusion and our method. For RTG-SLAM, we uniformly sample an equal number of points from the reconstructed Gaussians for evaluation. To eliminate the impact of tracking accuracy, we use the ground truth camera pose for reconstruction. Results are reported in Table 5. Our method outperforms GSFusion because its default voxel size is 1 cm, while our voxel size is 0.5 cm. This demonstrates that good geometry can be reconstructed using high-precision

Table 4 Comparison of novel view synthesis on the ScanNet++ dataset

Method	PSNR↑	SSIM↑	LPIPS↓
GSFusion	24.73	**0.884**	**0.264**
RTG-SLAM	25.07	0.852	0.286
GPS-SLAM (ours)	**25.81**	0.872	0.278

清華大學出版社 Tsinghua University Press Available on IEEE Xplore®

Table 5 Comparison of geometric accuracy on the ScanNet++ dataset

Metric	GSFusion	RTG-SLAM	GPS-SLAM
Accuracy ↓	1.38	0.94	**0.60**
Accuracy ratio↑	92.51	96.87	**99.93**
Completion ↓	1.26	1.09	**0.61**
Completion ratio↑	95.74	97.31	**99.50**

SDF fusion alone, without the need for a Gaussian representation.

4.3 Ablation and related studies

4.3.1 Rendering using only SDF or Gaussians

As Fig. 5 shows, using the synthetic Replica dataset, SDF fusion achieves good quality. In this case, our method requires only a small number of Gaussians to fit some high-frequency details. However, on our real-world indoor dataset, SDF fusion introduces significant artifacts and incompleteness due to more view-dependent appearances and missing depths. Consequently, our method uses more Gaussians to represent this information, resulting in slower performance on real-world datasets.

4.3.2 SDF voxel size

In our framework, the SDF voxel size plays a crucial role, influencing not only the quality of the reconstructed geometry, but also the Gaussian optimization and model size. We performed experiments with different voxel sizes using Replica *room0*. The results are summarized in Table 6 and illustrated in Fig. 7. The experiments demonstrate that

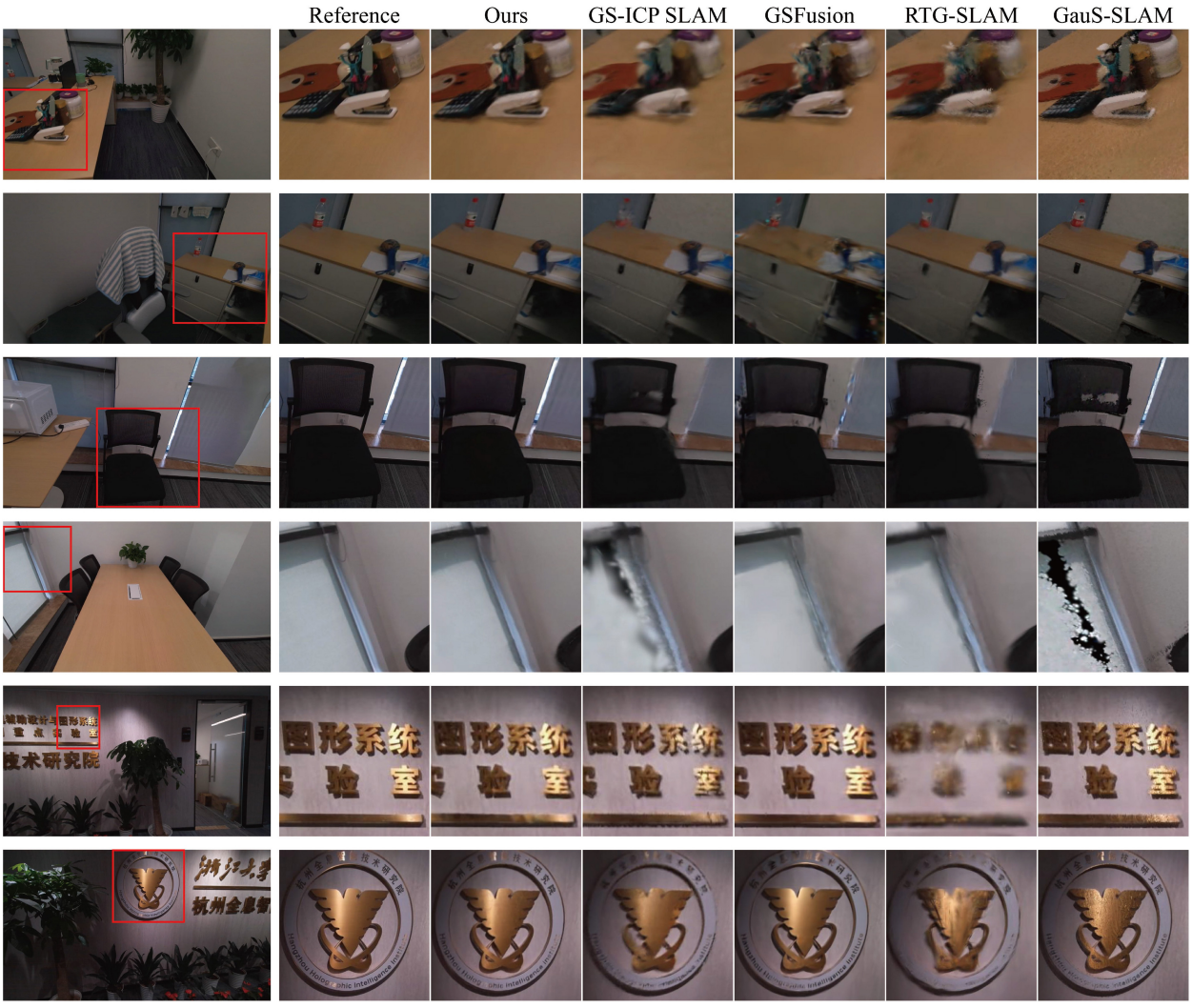

Fig. 6 Qualitative comparison of rendering quality on our indoor dataset. Our method achieves high-fidelity reconstruction, while other baselines produce blurred results or significant artifacts. Please note that the poor reconstruction quality of GS-ICP SLAM results from its mapping operation being slower than its tracking.

清華大學出版社 Tsinghua University Press Available on IEEE Xplore®

Table 6 Quantitative comparisons of different SDF voxel sizes using Replica *room0*. Geometric quality is evaluated using the chamfer distance (CD) metric

Metric	0.5 cm	1 cm	2 cm
PSNR↑	**34.73**	34.09	33.01
FPS↑	**133.88**	107.48	96.02
ATE↓	**0.16 cm**	0.21 cm	0.26 cm
CD↓	**1.72 cm**	1.77 cm	1.97 cm
Gaussian size	**155 MB**	193 MB	222 MB
Mesh size	2284 MB	560 MB	**138 MB**

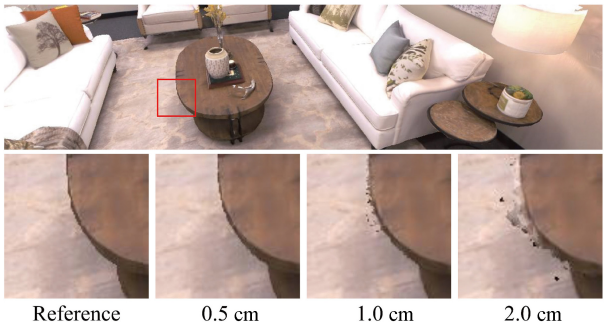

Fig. 7 Effect of SDF voxel size on Replica *room0*.

smaller voxel sizes result in higher tracking accuracy and improved image quality, especially in border areas. Moreover, reconstruction speed is not sacrificed. This is because the primary bottleneck of our system is Gaussian optimization, and more accurate SDF reconstruction reduces the number of Gaussians needed, thus accelerating the optimization process.

4.3.3 Sorting-free Gaussian rendering

Our sorting-free Gaussian rendering also contributes to the ultra-fast performance. We test the runtime performance of two rendering methods on the *activity room* scene of our indoor dataset, reporting the forward and backward times in Table 7. All results were averaged over three trials to ensure statistical reliability. Sorting-free Gaussian rendering accelerates the forward pass by 10% and the backward pass by 19%, resulting in an overall system speedup of 17%.

Table 7 Benefits of sorting-free Gaussian rendering on our indoor *activity room*. Our sorting-free Gaussian rendering simultaneously speeds up both forward and backward process

Rendering method	Forward	Backward	System FPS
With sort	0.97 ms	1.87 ms	152.86
Without sort	**0.87 ms**	**1.51 ms**	**179.40**

5 Conclusions

We have presented a real-time 3D reconstruction system which enables ultra-fast, high-fidelity reconstruction. The core of our system is a novel representation called Gaussian-plus-SDF, which can minimize the computational overhead of Gaussian optimization. We have designed highly efficient Gaussian insertion, Gaussian optimization, and Gaussian removal based on Gaussian-plus-SDF in the SLAM system. Our system achieves an order-of-magnitude faster reconstruction speed than state-of-the-art methods, with comparable reconstruction quality.

Our system currently only includes front-end camera tracking, which can lead to drift in large-scale scenes. Incorporating global pose optimization like Loopy-SLAM might address this issue. Additionally, our Gaussian rendering relies on depth culling provided by SDF. As Fig. 8 shows, significant geometric errors in the SDF reconstruction can degrade rendering quality. In future, we plan to explore the ultra-fast reconstruction of large-scale outdoor scenes with LiDAR sensors. We believe Gaussian-plus-SDF holds the promise of being compatible with LiDAR data.

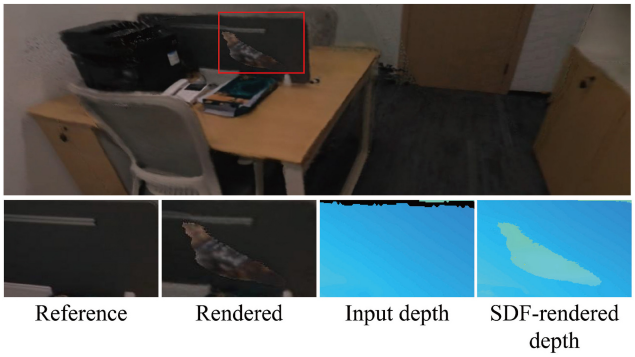

Fig. 8 A failure example. The thin panels exhibit holes during SDF reconstruction, which causes colors to be incorrectly rendered onto the opposite side.

Appendix

A Gaussian initialization

Here we explain how we compute the scales $\boldsymbol{s_u}$ of each newly added Gaussian for pixel $\boldsymbol{u}$ in detail. Each new Gaussian is initialized as a thin circular disc as in RTG-SLAM. However, RTG-SLAM calculates the scales using all Gaussians in the scene, resulting in an excessively large k-NN scope and impacting speed. Therefore, we only use the selected pixels

清華大學出版社 Tsinghua University Press Available on IEEE Xplore®

in $\boldsymbol{V}_k^{*,g}$. Specifically, we select the three vertices $\boldsymbol{V}_k^{*,g}(\boldsymbol{u}_1), \boldsymbol{V}_k^{*,g}(\boldsymbol{u}_2), \boldsymbol{V}_k^{*,g}(\boldsymbol{u}_3)$ closest to the pixel $\boldsymbol{V}_k^{*,g}(\boldsymbol{u})$, where $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3 \in M\widetilde{\boldsymbol{u}}$. The scales of $\mathcal{G}_u$ are initialized based on Formula (A.1):

$$\boldsymbol{s}_{\boldsymbol{u},1} = \begin{cases} 0.1, & \text{if } \sqrt{\frac{1}{3}\sum_{i=1}^{3}\left(||\boldsymbol{V}_k^{*,g}(\boldsymbol{u}) - \boldsymbol{V}_k^{*,g}(\boldsymbol{u}_i)||\right)} \geqslant 0.1 \\ \sqrt{\frac{1}{3}\sum_{i=1}^{3}\left(||\boldsymbol{V}_k^{*,g}(\boldsymbol{u}) - \boldsymbol{V}_k^{*,g}(\boldsymbol{u}_i)||\right)}, & \text{otherwise} \end{cases}$$

$$\boldsymbol{s}_{\boldsymbol{u},2} = \boldsymbol{s}_{\boldsymbol{u},1}, \quad \boldsymbol{s}_{\boldsymbol{u},3} = 0.1\boldsymbol{s}_{\boldsymbol{u},1} \tag{A.1}$$

Here, we truncate the maximum scale to below 0.1 to avoid large-scale outliers.

B Gaussian optimization

Although the sort-free rasterization method allows us to launch threads per Gaussian, the significant variation in Gaussian scale means that directly launching a thread for each Gaussian would lead to substantial computational imbalance between threads. To address this, we calculate the number of pixels covered by each Gaussian based on its radius in pixel space. We then divide the Gaussians into groups of a fixed size. During backpropagation, threads are launched per group, ensuring a consistent number of iterations per thread and maximizing parallel efficiency.

C Implementation details

Now we list all the parameters in detail. For Gaussian optimization learning rates, we set $\mathrm{lr}_{\mathrm{position}} = 0.00016$, $\mathrm{lr}_{\mathrm{SH0}} = 0.0025$, $\mathrm{lr}_{\alpha} = 0.05$, $\mathrm{lr}_{\mathrm{scale}} = 0.005$, and $\mathrm{lr}_{\mathrm{rotation}} = 0.001$ for all datasets. The learning rate for other SH coefficients is 0.0005. The motion thresholds for keyframe creation, δ_{angle} and δ_{move}, are set to 30° and 0.3 m, respectively.

D Dataset details

We used a laptop with an Intel i7 10870-H CPU and NVIDIA 3070 Laptop GPU connected to an Azure Kinect RGB-D camera for data acquisition. The RGB-D images captured by the camera were transmitted to a desktop computer through a wireless network, and the desktop computer completed the SLAM computations. Results were sent back to a viewer on the laptop for visualization. A statistical summary of our indoor dataset is given in Table 8.

Table 8 Statistical summary of our indoor dataset

Statistic	*Signboard*	*Activity room*	*Office0*	*Office1*	*Office2*
Trajectory (m)	13.8	13.3	21.7	15.2	18.2
Area (m^2)	37.3	30.3	88.5	32.5	39.8
Frames	2200	2680	2950	2150	2500

Availability of data and materials

We used the publicly available Replica, TUM-RGBD, and ScanNet++ datasets in our experiments. Our own indoor dataset can be downloaded from `https://gapszju.github.io/GPS-SLAM`.

Funding

This research was supported by the National Natural Science Foundation of China (U23A20311, 62421003).

Author contributions

Kun Zhou introduced the idea, and designed the algorithm with Tianjia Shao. Zhexi Peng implemented the algorithm, designed and conducted the experiments, and prepared the draft paper. Kun Zhou and Tianjia Shao had regular discussions with Zhexi Peng, provided guidance and suggestions, and revised the paper.

Acknowledgements

The authors appreciate the support from Adobe Research and the XPLORER Prize.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article. The author Kun Zhou is an Associate Editor of this journal.

Electronic Supplementary Material

Supplementary material is available in the online version of this article at `https://doi.org/10.26599/CVM.2025.9450513`.

References

[1] Keetha, N.; Karhade, J.; Jatavallabhula, K. M.; Yang, G.; Scherer, S.; Ramanan, D.; Luiten, J. SplaTAM:

Splat, track & map 3D Gaussians for dense RGB-D SLAM. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 21357–21366, 2024.

[2] Su, Y.; Chen, L.; Zhang, K.; Zhao, Z.; Hou, C.; Yu, Z. GauS-SLAM: Dense RGB-D SLAM with Gaussian Surfels. *arXiv preprint* arXiv:2505.01934, 2025.

[3] Peng, Z.; Shao, T.; Liu, Y.; Zhou, J.; Yang, Y.; Wang, J.; Zhou, K. RTG-SLAM: Real-time 3D reconstruction at scale using Gaussian splatting. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Papers, 2024.

[4] Ha, S.; Yeon, J.; Yu, H. RGBD GS-ICP SLAM. In: *Computer Vision – ECCV 2024. Lecture Notes in Computer Science, Vol. 15094*. Sun, Q.; Zhou, H.; Zhou, W.; Li, L.; Li, H. Eds. Springer Cham, 180–197, 2024.

[5] Newcombe, R. A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A. J.; Kohi, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality, 127–136, 2012.

[6] Steinbrucker, F.; Kerl, C.; Cremers, D.; Sturm, J. Large-scale multi-resolution surface reconstruction from RGB-D sequences. In: Proceedings of the IEEE International Conference on Computer Vision, 3264–3271, 2014.

[7] Nießner, M.; Zollhöfer, M.; Izadi, S.; Stamminger, M. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics* Vol. 32, No. 6, Article No. 169, 2013.

[8] Dai, A.; Nießner, M.; Zollhöfer, M.; Izadi, S.; Theobalt, C. BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 1, 2017.

[9] Kähler, O.; Adrian Prisacariu, V.; Yuheng Ren, C.; Sun, X.; Torr, P.; Murray, D. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics* Vol. 21, No. 11, 1241–1250, 2015.

[10] Ye, K.; Shao, T.; Zhou, K. When Gaussian meets surfel: Ultra-fast high-fidelity radiance field rendering. *ACM Transactions on Graphics* Vol. 44, No. 4, Article No. 113, 2025.

[11] Kähler, O.; Prisacariu, V. A.; Murray, D. W. Real-time large-scale dense 3D reconstruction with loop closure. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9912*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 500–516, 2016.

[12] Straub, J.; Whelan, T.; Ma, L.; Chen, Y.; Wijmans, E.; Green, S.; Engel, J. J.; Mur-Artal, R.; Ren, C.; Verma, S.; et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint* arXiv:1906.05797, 2019.

[13] Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 573–580, 2012.

[14] Yeshwanth, C.; Liu, Y. C.; Nießner, M.; Dai, A. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 12–22, 2024.

[15] Zeng, M.; Zhao, F.; Zheng, J.; Liu, X. Octree-based fusion for realtime 3D reconstruction. *Graphical Models* Vol. 75, No. 3, 126–136, 2013.

[16] Fuhrmann, S.; Goesele, M. Fusion of depth maps with multiple scales. *ACM Transactions on Graphics* Vol. 30, No. 6, 1–8, 2011.

[17] Steinbrücker, F.; Sturm, J.; Cremers, D. Volumetric 3D mapping in real-time on a CPU. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2021–2028, 2014.

[18] Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM* Vol. 65, No. 1, 99–106, 2022.

[19] Sucar, E.; Liu, S.; Ortiz, J.; Davison, A. J. iMAP: Implicit mapping and positioning in real-time. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 6209–6218, 2022.

[20] Zhu, Z.; Peng, S.; Larsson, V.; Xu, W.; Bao, H.; Cui, Z.; Oswald, M. R.; Pollefeys, M. NICE-SLAM: Neural implicit scalable encoding for SLAM. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 12776–12786, 2022.

[21] Yang, X.; Li, H.; Zhai, H.; Ming, Y.; Liu, Y.; Zhang, G. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, 499–507, 2022.

[22] Johari, M. M.; Carta, C.; Fleuret, F. 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, 17408–17419, 2023.

[23] Wang, H.; Wang, J.; Agapito, L. 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, 13293–13302, 2023.

[24] Gong, Z.; Tosi, F.; Zhang, Y.; Mattoccia, S.; Poggi,

M. HS-SLAM: Hybrid representation with structural supervision for improved dense SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation, 8464–8470, 2025.

[25] Sandström, E.; Li, Y.; Van Gool, L.; Oswald, M. R. Point-SLAM: Dense neural point cloud-based SLAM. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 18387–18398, 2024.

[26] Liso, L.; Sandström, E.; Yugay, V.; Van Gool, L.; Oswald, M. R. Loopy-SLAM: Dense neural SLAM with loop closures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 20363–20373, 2024.

[27] Kerbl, B.; Kopanas, G.; Leimkuehler, T.; Drettakis, G. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics* Vol. 42, No. 4, Article No. 139, 2023.

[28] Yugay, V.; Li, Y.; Gevers, T.; Oswald, M. R. Gaussian-SLAM: Photo-realistic dense SLAM with Gaussian splatting. *arXiv preprint* arXiv:2312.10070, 2023.

[29] Yan, C.; Qu, D.; Xu, D.; Zhao, B.; Wang, Z.; Wang, D.; Li, X. GS-SLAM: Dense visual SLAM with 3D Gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 19595–19604, 2024.

[30] Hu, J.; Chen, X.; Feng, B.; Li, G.; Yang, L.; Bao, H.; Zhang, G.; Cui, Z. CG-SLAM: Efficient dense RGB-D SLAM in a consistent uncertainty-aware 3D Gaussian field. *arXiv preprint* arXiv:2403.16095, 2024.

[31] Wei, J.; Leutenegger, S. GSFusion: Online RGB-D mapping where Gaussian splatting meets TSDF fusion. *IEEE Robotics and Automation Letters* Vol. 9, No. 12, 11865–11872, 2024.

[32] Yu, M.; Lu, T.; Xu, L.; Jiang, L.; Xiangli, Y.; Dai, B. GSDF: 3DGS meets SDF for improved neural rendering and reconstruction. *arXiv preprint* arXiv:2403.16964, 2024.

[33] Cao, Y. P.; Kobbelt, L.; Hu, S. M. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Transactions on Graphics* Vol. 37, No. 5, Article No. 171, 2018.

[34] Whelan, T.; Kaess, M.; Johannsson, H.; Fallon, M.; Leonard, J. J.; McDonald, J. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *The International Journal of Robotics Research* Vol. 34, Nos. 4–5, 598–626, 2015.

[35] Mur-Artal, R.; Tardós, J. D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics* Vol. 33, No. 5, 1255–1262, 2017.

Zhexi Peng received his B.S. degree in information and computing sciences from Beijing University of Posts and Telecommunications. He is currently a Ph.D. student at Zhejiang University. His research interests cover computer vision, SLAM, and machine learning.

Kun Zhou is a Cheung Kong Professor of Computer Science at Zhejiang University and the Director of the State Key Lab of CAD&CG. He received his Ph.D. degree from Zhejiang University, and then spent six years with Microsoft Research Asia, serving as a lead researcher of the graphics group before returning to Zhejiang University. He was named one of the world's top 35 young innovators by MIT Technology Review (2011), and received an Asiagraphics Outstanding Technical Contributions Award (2022) and an ACM SIGGRAPH Test-of-Time Award (2024). He is a Fellow of IEEE and ACM.

Tianjia Shao received his B.S. degree from the Department of Automation, and his Ph.D. degree in computer science from the Institute for Advanced Study, both in Tsinghua University. He is currently a ZJU100 Young Professor in the State Key Laboratory of CAD&CG, Zhejiang University. Previously he was a lecturer in the School of Computing, University of Leeds, UK. His current research focuses on 3D scene reconstruction, digital human creation, and 3D AIGC.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made.

The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this licence, visit `http://creativecommons.org/licenses/by/4.0/`.

To submit a manuscript, please go to `https://jcvm.org`.