Incremental Online Scene Reconstruction by 3D Gaussian Triangulation

Yanjin Zhu[*1], Shaofan Liu[*2], and Jianke Zhu[†1,3]

[1] Zhejiang University, Hangzhou, China
[2] Hefei University of Technology, Hefei, China
[3] Shenzhen Loop Area Institute, Shenzhen, China

Abstract. Incremental scene reconstruction is essential for real-world applications. Although 3D Gaussian Splatting shows strong potential, most existing approaches require offline conversion of the optimized Gaussians into an intermediate implicit field for explicit mesh extraction, which hinders seamless integration with downstream tasks. To address this limitation, we propose a novel online framework that incrementally reconstructs and updates high-fidelity explicit meshes by directly triangulating a dense geometric Gaussian representation, which supports both high-quality rendering and incremental surface reconstruction. Moreover, we present a direct meshing algorithm that efficiently extracts and updates the mesh from the Gaussian set. To ensure mesh accuracy, we enforce a plane-based pulling constraint that dynamically aligns 3D Gaussian primitives to the approximated local surface. Furthermore, our framework significantly reduces memory and computational overhead during long-sequence processing by dynamically freezing fully optimized historical regions. Experiments on public datasets demonstrate that our method outperforms conventional Gaussian-based methods on both rendering quality and reconstruction accuracy.

Keywords: Gaussian Triangulation · Incremental Reconstruction · Gaussian Splatting

1 Introduction

3D scene reconstruction is a fundamental task with extensive applications in computer vision and robotics, notably Augmented Reality [21] and scene perception [8]. Typically, 3D scene reconstruction techniques can be categorized into two groups. One is implicit methods [17, 28], and the other is explicit approaches [7, 12, 27]. Unfortunately, most high-fidelity reconstruction methods require processing the entire scene before generating the recovered meshes, which poses a significant bottleneck in real-world applications. For example, the memory constraints of computing devices make it infeasible to represent continuously expanding scenes. For time-critical tasks in autonomous robotics, the delay in

[*] Equal contribution.
[†] Corresponding author.

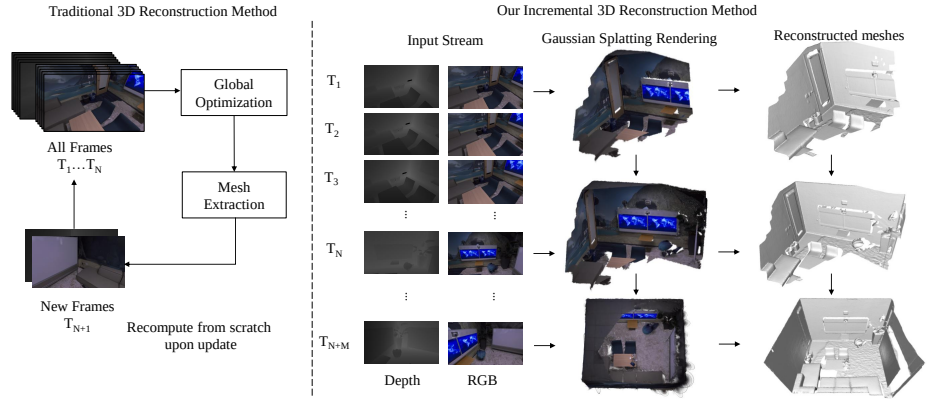


Fig. 1: Comparison between traditional batch reconstruction and our proposed incremental framework. Left: Traditional methods typically rely on global optimization over all available frames ($T_1 ... T_N$). Once a new frame (T_{N+1}) is obtained, the entire mesh extraction pipeline has to be recomputed from scratch. Right: Our approach performs in a fully incremental manner. As the input stream arrives ($T_1 \ldots T_{N+M}$), we progressively update the Gaussian representation and the reconstructed mesh. This allows for continuous scene expansion without global re-optimization.

obtaining a complete model is impractical, as early access to partial results is crucial for making immediate decisions. Consequently, incremental 3D scene reconstruction is urgently needed to overcome these fundamental limitations.

Traditional online scene reconstruction methods, such as KinectFusion [18] and its derivatives, are typically based on volumetric integration of the Truncated Signed Distance Field (TSDF). These approaches incrementally fuse sequential depth maps to update a volumetric grid, from which a triangle mesh is subsequently extracted by Marching Cubes [13] at a much lower frequency than the TSDF fusion. However, they are often constrained by fixed voxel resolution and high memory consumption, which prevent detailed capture and large-scale scene reconstruction.

A remedy is to utilize continuous neural implicit representations, notably Neural Radiance Fields (NeRF) [17], which models scenes as radiance fields with adaptive sampling. Through spatial interpolation [23] and feature priors [29], these implicit methods can effectively reduce reconstruction artifacts and fill in missing geometry due to noisy input. However, these methods introduce new challenges, including heavy load during training and rendering due to computationally intensive network queries, and limited scalability in unbounded scenes. Furthermore, extracting explicit meshes from NeRF still requires dense sampling and offline post-processing, which hinders it from incremental scenarios.

To deal with the challenges of reconstruction and rendering speed, 3D Gaussian Splatting (3DGS) provides a notable advance in real-time novel view synthesis through its explicit and differentiable representation, though its potential in mesh reconstruction is overlooked. Existing methods like SuGaR [7] and 2D

Gaussian Splatting [9] attempt to address this by facilitating surface-aligned regularization and explicit 2D planar Gaussian representations to represent surfaces. In general, these methods decouple mesh extraction from the Gaussian representation and rely on global Poisson reconstruction [11] or Marching Cubes [13], which require offline processing of the fully optimized Gaussian set.

In contrast to the conventional offline holistic methods, we propose a novel online incremental scene reconstruction framework that progressively integrates incoming data to generate triangular meshes. Our key idea is to treat optimized 3D Gaussians as surfel primitives for direct triangulation. By directly extracting watertight meshes from these primitives, our method leverages the flexibility of the explicit Gaussian representation to correct noisy depth regions, achieving fast and accurate reconstruction and rendering. To ensure these primitives are mesh-ready, we introduce: (i) a geometric planar constraint that aligns Gaussians to local surfaces, and (ii) a sparsity regularization that eliminates outliers while maintaining triangulation-conforming density. Furthermore, by freezing fully optimized mesh regions, our method prevents unbounded resource growth during long-sequence scanning. As illustrated in Fig. 1, this enables scalable, continuous surface reconstruction.

The main contributions of this paper are as follows: 1) a novel triangulation algorithm on the Gaussian set, which achieves fast mesh extraction and incremental update from Gaussian surfels; 2) a plane-based pulling constraint that guides Gaussian surfels to form a coherent structure in order to rectify input noise and faithfully represent the underlying surfaces; 3) a memory-efficient incremental scene reconstruction method that progressively freezes optimized mesh regions and focuses optimization on newly observed areas.

2 Related Work

2.1 Novel View Synthesis and Gaussian Splatting

We give a brief overview of Novel View Synthesis (NVS) methods for scene reconstruction. Neural Radiance Field (NeRF) [17] and its variants [1] have shown remarkable capabilities in synthesizing impressive novel views from multi-view images. However, the volumetric rendering is computationally intensive, which severely limits its efficiency. In contrast to NeRF, 3D Gaussian Splatting [12] is an explicit scene representation, which offers a more efficient alternative for novel view synthesis. Some subsequent works [14, 26] have improved rendering quality. Most of them depend on the sparse point clouds generated by Structure-from-Motion (SfM) as initial input, which limits their applicability. To produce more compact and accurate 3D Gaussians, GaussianPro [4] presents a novel progressive propagation strategy. GaMeS [24] takes the explicit mesh as input and parameterizes Gaussian ellipsoids with triangle meshes. Using sparse SfM point clouds or meshes as initialization for 3D Gaussian Splatting typically requires additional preprocessing steps and may introduce certain limitations. For large-scale scenes, RTG-SLAM [19] proposes a compact Gaussian representation and on-the-fly Gaussian optimization to facilitate real-time 3D reconstruction

and superior renderings. Despite the fact that RTG-SLAM is able to incrementally reconstruct a Gaussian map, it still requires additional data structures and routines to obtain the reconstructed mesh. Our method not only synthesizes high-quality novel views, but also achieves incremental surface reconstruction.

2.2 Surface Reconstruction from Gaussians

Although 3D Gaussian Splatting has demonstrated remarkable NVS performance, its discrete and unstructured nature poses challenges for surface reconstruction. SuGaR [7] regularizes the Gaussians to be flat and well distributed over the scene surface and derives a volume density from the Gaussians. To achieve a close surface alignment with Gaussians, 2D GS [9] and Gaussian surfels [5] employ surfels as surface representation. This involves conceptually flattening 3D Gaussian points into 2D by setting their z-scale to zero. GS-Pull [28] aligns 3D Gaussians on the zero level-set of the neural SDF using neural pulling and updates the neural SDF through the pulled 3D Gaussians. These methods usually rely on either Poisson reconstruction, derived from an implicit field generated by Gaussians, or TSDF fusion using multi-view depth maps. Moreover, these surface extraction methods are constrained by high memory usage for large scenes. Gaussian Opacity Fields [27] infers minimal opacity across views to keep multi-view consistency, followed by surface extraction using Marching Tetrahedra. The computation of the opacity values using all training views may lead to redundant computations. Without the construction and maintenance of implicit fields, our proposed method directly obtains the detailed surface reconstruction from a Gaussian set and yields photorealistic rendering results.

3 Method

3.1 Overview

We present a unified framework for incremental mesh reconstruction and high-fidelity rendering, as illustrated in Fig. 2. Our method takes a sequence of M RGB-D frames $\{I_i, D_i\}_{i=1}^{M}$ with known camera poses as input. The foundation of our method is the Dense Geometric Gaussian representation, denoted as $\mathcal{G}$, which serves as the fundamental primitive for both volumetric rendering and explicit surface extraction (Sec. 3.2). To achieve incremental meshing from Gaussians, dense geometric Gaussian representation is dynamically maintained and progressively optimized under photometric and structural constraints. The combined supervision enforces the optimized Gaussians in order to adhere to the underlying surface geometry (Sec. 3.3). At each timestamp, local meshes are obtained by triangulating a selected Geometric Gaussian Set. To ensure a memory-efficient incremental scene reconstruction, our method progressively freezes these well-optimized mesh regions and focuses optimization solely on newly observed areas (Sec. 3.4).

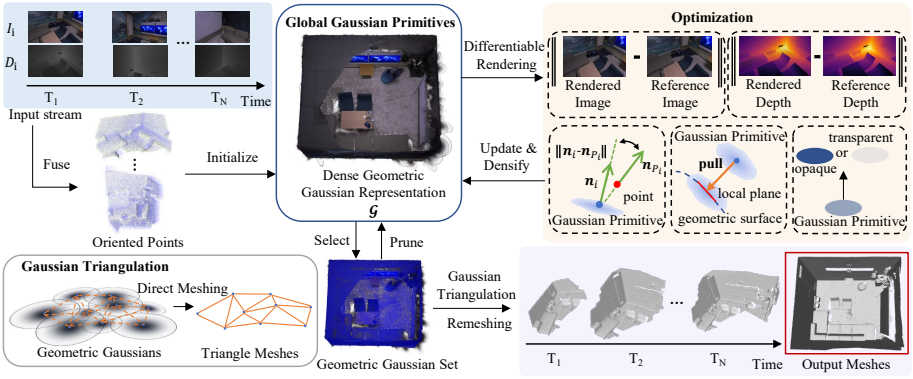


Fig. 2: Overview of our proposed approach. We first initialize Gaussians from a continuous stream of oriented point clouds. The dense geometric Gaussian representation is progressively refined by loss-guided densification and optimization against depth, color, and geometric constraints. Next, a geometric Gaussian set is selected for meshing, during which process redundant primitives are pruned to maintain compactness. A novel triangulation algorithm (illustrated in the bottom-left inset) is subsequently applied to the selected Gaussians to incrementally reconstruct and update meshes.

3.2 Dense Geometric Gaussian Representation

Geometric Gaussian Primitives. Following [12], we represent the scene using a set of 3D Gaussian primitives $\{G_i\}$. Each Gaussian comprises a 3D mean $\boldsymbol{\mu}_i \in \mathbb{R}^3$, opacity value $\alpha_i \in [0,1]$, RGB color values $\mathbf{c}_i \in \mathbb{R}^3$, scale vector $\mathbf{s}_i = [s_1^i, s_2^i, s_3^i]^\top \in \mathbb{R}^3$ and rotation $\mathbf{R}_i = [\mathbf{r}_1^i\ \mathbf{r}_2^i\ \mathbf{r}_3^i] \in \mathbb{R}^{3\times3}$.

Unlike standard 3DGS optimized for volumetric radiance fields, our objective is high-fidelity surface reconstruction. To this end, we explicitly constrain the Gaussian primitives to act as Planar Elliptical Surfels:

- **Planar Constraint:** We enforce the third scale component $s_3^i \to 0$, effectively flattening the Gaussian to align with the local tangent plane of the surface.
- **Opacity Constraint:** To approximate hard physical geometry, we enforce high opacity ($\alpha_i \approx 1$).

These constraints allow our representation to seamlessly unify mesh reconstruction and rendering within a single framework.

Gaussian Initialization. We initialize the geometric Gaussians from geometric attributes of points in the oriented point cloud $\mathcal{P}$, which is incrementally constructed from RGB-D streams. An oriented point in $\mathcal{P}$ has a position $\mathbf{p}_i$ and normal $\mathbf{n}_i$. The mean $\boldsymbol{\mu}_i$ is directly inherited from $\mathbf{p}_i$. Following [10], we construct the rotation matrix $\mathbf{R}_i$ such that its third column $\mathbf{r}_3^i$ aligns with the normal vector $\mathbf{n}_i$. To ensure proper surface coverage, the tangential scales (s_1, s_2) are determined by the local point spacing:

$$s_1^i = s_2^i = \lambda \cdot \max_{j \in \mathcal{N}(i)} \|\mathbf{p}_i - \mathbf{p}_j\|_2, \tag{1}$$

where $\mathcal{N}(i)$ denotes the k-nearest neighbors in 3D space, $k = 4$ and $\lambda = 1.5$. Consistent with planar elliptical surfels, we initialize $s_3^i = 10^{-6}$ and $\alpha_i = 0.99$.

Geometry-Aware Differentiable Rendering. We adopt a dual-branch rendering strategy to optimize both photometric and geometric consistency. We render the optimizable Gaussians $\{G_i\}$ into RGB images $\hat{\mathbf{C}}$ through the standard alpha-blending proposed in [12],

$$\hat{\mathbf{C}}(u,v) = \sum_{i \in N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{2}$$

where $\mathbf{c}_i$ denotes the Gaussian color, and α_i is its opacity.

The standard alpha-blending depth is insufficient for precise mesh reconstruction. To model the local geometric plane as opaque Gaussian primitives, we compute the intersection of the view ray and the front opaque Gaussian primitive to obtain the depth at a pixel $\hat{\mathbf{D}}(u,v)$, as in [19]. The depth map $\hat{\mathbf{D}}$ is computed as below

$$\hat{\mathbf{D}}(u,v) = \begin{cases} -1 & \text{if no intersection with opaque Gaussians,} \\ (\mathbf{T}_w^c(\frac{\boldsymbol{\mu}_j^{\mathbf{r}} \cdot \mathbf{n}_j^{\mathbf{r}}}{\mathbf{r} \cdot \mathbf{n}_j^{\mathbf{r}}} \cdot \mathbf{r}))_z & \text{if } \langle \mathbf{n}_j^{\mathbf{r}}, \mathbf{r} \rangle < 0.5, \\ (\mathbf{T}_w^c \boldsymbol{\mu}_j^{\mathbf{r}})_z & \text{otherwise.} \end{cases} \tag{3}$$

where $\mathbf{r}$ denotes the normalized ray in the world coordinate. $\boldsymbol{\mu}_j^{\mathbf{r}}$ and $\mathbf{n}_j^{\mathbf{r}}$ are the position and normal of the intersected Gaussian, respectively. $\mathbf{T}_w^c$ represents the transformation from the world coordinate to the current camera, and $(\cdot)_z$ extracts the z-component of the 3D vector.

3.3 Online Mapping

Gaussian Adding. Based on the newly added oriented point cloud, we first add corresponding Gaussians. To compensate for the information loss prone to insufficient depth observations, we dynamically supplement missing regions driven by both geometric and photometric rendering discrepancies. For significant depth differences $(\mathbf{D}_i - \hat{\mathbf{D}}_i)$, we add opaque Gaussians to improve the completeness of the reconstructed mesh. Similarly, we leverage the RGB difference $\Delta\mathbf{C}_i = \mathbf{C}_i - \hat{\mathbf{C}}_i$ to add transparent Gaussians (set $\alpha_i = 0.1$) so that the rendering quality can be greatly enhanced.

Gaussian Pruning. Since our representation relies on opaque Gaussians to fit local surfaces, we prune opaque Gaussians that do not participate in depth rendering and transparent Gaussians that contribute minimally to photometric rendering. The pruning operates concurrently with Gaussian optimization process, incurring negligible computational overhead.

Gaussian Optimization. After adding Gaussians for the active window, we iteratively optimize their attributes by randomly sampling one frame per iteration. To ensure accurate reconstruction and rendering, we supervise this process with a combination of photometric and structural constraints.

We formulate the rendering losses to supervise the Gaussian optimization:

$$\mathcal{L}_{color} = \sum_{i}^{N} |\mathbf{C}_i - \hat{\mathbf{C}}_i|, \quad \mathcal{L}_{depth} = \sum_{i}^{N} |\mathbf{D}_i - \hat{\mathbf{D}}_i|. \tag{4}$$

By leveraging the complementary nature of photometric and geometric constraints, Gaussian surfels can robustly recover scene geometry, mutually compensating for sensor noise and depth missing.

To enable the direct reconstruction of smooth and accurate surfaces from Gaussian sets, we introduce two geometric constraints. The first constraint aims to align the extracted Gaussians with the underlying local geometry. Inspired by GS-Pull [28], we consider our oriented point clouds as a discrete approximation of the surface zero level set $Z(f)$. Accordingly, we introduce a local plane loss to pull Gaussian centers $\boldsymbol{\mu}_i$ toward the underlying surface points represented by the oriented point clouds as follows

$$\mathcal{L}_{plane} = \sum_{(\boldsymbol{\mu}_i, \mathbf{p}_i) \in \mathcal{C}} |\mathbf{n}_i^\top (\boldsymbol{\mu}_i - \mathbf{p}_i)|, \tag{5}$$

where $\mathcal{C}$ is the set of correspondences between geometric Gaussians and oriented points, $\mathbf{p}_i \in \mathcal{P}$ is the point position, and $\mathbf{n}_i$ is the point normal. $\mathcal{P}$ denotes oriented point clouds. The plane-based pulling loss encourages the Gaussian centers to lie on the local plane while preserving maximal freedom during optimization.

Our second geometric constraint enforces the orientational alignment between the Gaussians and the local surfaces. Accordingly, we employ a normal consistency loss defined as

$$\mathcal{L}_n = 1 - |\mathbf{n}_g^\top \cdot \mathbf{n}_i|, \tag{6}$$

where $\mathbf{n}_g$ is the Gaussian normal, which is approximated by the shortest-axis direction of each Gaussian surfel following GaussianShader [10].

To facilitate photometric rendering and geometric optimization, we utilize a sparse loss [10] to encourage Gaussians to become either fully transparent or fully opaque.

$$\mathcal{L}_{\text{sparse}} = \frac{1}{|\alpha|} \sum_{\alpha_i} \left[\log(\alpha_i) + \log(1 - \alpha_i) \right]. \tag{7}$$

Above all, we optimize the attributes of Gaussians via the following objective function,

$$\mathcal{L} = \lambda_c \mathcal{L}_{color} + \lambda_d \mathcal{L}_{depth} + \lambda_p \mathcal{L}_{plane} + \lambda_n \mathcal{L}_n + \lambda_s \mathcal{L}_{sparse}. \tag{8}$$

3.4 Gaussian Triangulation

Geometric Gaussian Set. To enable accurate meshing from dense geometric Gaussian representation, we need to filter out Gaussian primitives that serve merely for visual rendering rather than for true geometry. Therefore, we propose

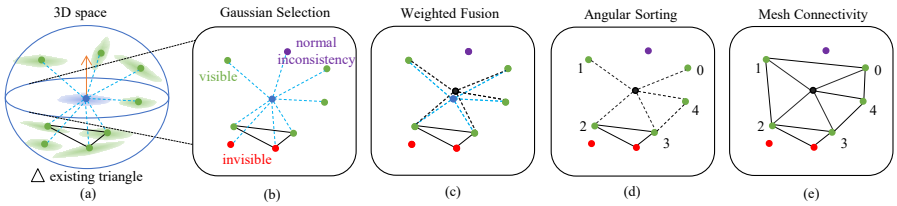


Fig. 3: Illustration of Gaussian triangulation pipeline. (a) Neighbors within an adaptive radius are identified and projected onto the tangent plane. (b) Valid candidates (green) are retained, while invisible (red) and normal-inconsistent (purple) ones are pruned. (c) The central Gaussian (blue) is shifted to a refined position (black) via weighted fusion. (d) Neighbors are re-projected onto the new tangent space and sorted angularly (indexed 0-4). (e) Mesh connectivity is established via an angle-based greedy algorithm.

a simple and effective strategy that relies on the opacity and depth fidelity to extract the Geometric Gaussian Set G_{geo}, a subset of $\{G_i\}$. Specifically, the geometric Gaussian set G_{geo} is defined as

$$G_{geo} = \{G_g \in \{G_i\} \mid \alpha_g > 0.9 \wedge (|(\mathbf{T}_{cw}\mu_g)_z - D| < \tau)\}. \tag{9}$$

where τ denotes a predefined threshold. The opacity criterion removes semi-transparent Gaussians that model volumetric or view-dependent artifacts, while the depth fidelity criterion eliminates geometrically inaccurate primitives. This strategy ensures that only robust, highly opaque primitives corresponding to actual physical structures are retained to reconstruct the explicit meshes.

Fast Gaussian Triangulation. Inspired by fast point set triangulation [6, 15], we propose a triangulation approach for the geometric Gaussian set, as illustrated in Fig. 3. To ensure computational efficiency, we employ a compressed octree. For a given Gaussian G_i, an adaptive search radius is defined based on the mean distance to its k-nearest neighbors ($k = 4$ empirically). Within this radius, we identify a valid neighbor subset $\mathcal{N}_G$ by enforcing two critical geometric constraints on each candidate G_j: mutual visibility on the tangent plane and normal consistency ($\langle \mathbf{n}_i, \mathbf{n}_j \rangle > 0.9$). To yield a more coherent surface, the 3D mean of each Gaussian is refined through a weighted averaging of these candidates. Following the global update of the Gaussian set, we perform a second neighbor search on the refined Gaussians to establish the final connectivity for triangulation.

To establish the topological connectivity, we project all candidate neighbors onto the tangent plane of the central Gaussian. In this 2D local frame, the central Gaussian acts as the pivot vertex. We employ an angle-based greedy strategy to reconstruct the triangular meshes: neighbors are first sorted by polar angle around the center. We then iteratively form triangles by connecting the central Gaussian with pairs of angularly adjacent neighbors. To ensure mesh quality and prevent the formation of degenerate sliver triangles, we enforce a constraint that the subtended angle must exceed 10°.

Local Remeshing and Freezing. In incremental scene reconstruction, the system generates a local triangular mesh for the current viewpoint at timestamp t and attempts to fuse it with the existing surface mesh. Due to observational variations across different viewpoints, direct integration inevitably leads to vertex misalignments and topological conflicts in adjacent regions. To address this issue, we introduce a local remeshing strategy to eliminate topological conflicts and achieve seamless stitching. Following the isotropic mesh optimization approach proposed by Botsch et al. [3], we begin by splitting edges longer than $1.5\bar{L}$ (where $\bar{L}$ is the local average edge length). Subsequently, edges shorter than $0.5\bar{L}$ are collapsed to simplify the topology. Then, edges are evaluated and flipped if it optimizes the degree of neighboring vertices towards the target value of 6. Finally, vertices are relocated to the centroid of their neighbors via Laplacian smoothing. We repeat this process for three iterations to enforce strict geometric consistency between the newly observed region and the prior mesh.

Furthermore, to prevent the unbounded growth of computational and memory overhead associated with long-sequence reconstruction, we freeze the triangular meshes and Gaussian surfels in regions that possess extensive historical observations and are fully optimized. Specifically, the system quantifies the convergence state of a region by evaluating the multi-view observation counts and the loss of its Gaussians. When the cumulative effective observation count of Gaussians within a local region exceeds a predefined threshold N_{obs}, and their corresponding optimization loss $\mathcal{L}$ has stably converged to a minimum ϵ_{gs}, the region is determined to be fully optimized. Once this condition is met, the system removes the Gaussian surfels and mesh of this region from the active optimization computation graph and freezes them. Such combined strategy of dynamically freezing stable regions alongside local remeshing ensures that the system maintains excellent memory efficiency when processing long-sequence scenes.

4 Experiments

In this section, we present the details of our experiments and conduct experiments on public datasets to evaluate the reconstruction accuracy and rendering quality of our proposed approach.

4.1 Experimental Setup

Implementation Details. Our framework is implemented in Python, with the Gaussian set triangulation and oriented point cloud generation encapsulated in C++ modules. We leverage CUDA parallel computing to accelerate the generation of oriented point clouds. For Gaussian optimization, we set $\lambda_c = 0.8, \lambda_d = 1.0, \lambda_p = 0.05, \lambda_n = 0.2, \lambda_s = 0.001$. For Gaussian filtering, we use $\tau = 0.001$ to select the geometric Gaussian set. Additionally, the observation count threshold N_{obs} is set to 10, and the geometric selection loss threshold ϵ_{gs} is set to 0.1. The active window size and optimization iterations are set to 6 and 50 for Replica [22], and 3 and 75 for ScanNet++ [25], respectively. All experiments

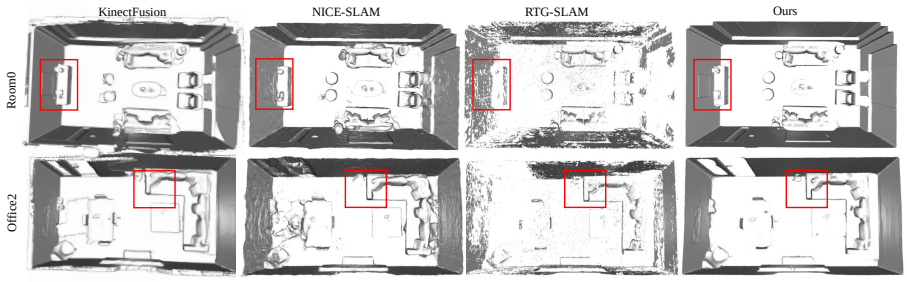


Fig. 4: Comparison of Mesh Reconstruction results across different Replica sequences. For RTG-SLAM [19], we employ the ball pivoting algorithm [2] to generate meshes from Gaussian balls.

Table 1: Comparison of geometry accuracy on Replica.

Method	Metric	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
KinectFusion [18]	Acc.[cm]↓	10.39	19.27	17.25	11.10	9.99	13.43	9.88	10.77	12.76
	Acc. Ratio↑	44.02	22.36	24.68	37.13	36.64	34.31	39.58	35.98	34.34
	Comp. Ratio↑	79.20	47.23	50.98	72.25	70.57	60.93	62.26	59.30	62.84
NICE-SLAM [29]	Acc.[cm]↓	3.22	42.36	2.61	2.32	3.22	3.85	4.70	2.99	8.16
	Acc. Ratio↑	89.33	39.01	91.77	93.39	87.02	83.26	68.12	91.44	80.42
	Comp. Ratio↑	**88.39**	73.80	**90.73**	**94.80**	**94.38**	**84.12**	67.16	**86.71**	85.01
MonoGS [16]	Acc.[cm]↓	3.26	3.23	2.89	2.32	1.93	2.76	2.72	3.02	2.77
	Acc. Ratio↑	82.83	81.95	85.97	90.01	91.45	87.10	89.05	85.46	86.73
	Comp. Ratio↑	80.45	82.59	82.45	91.39	86.91	76.04	75.44	79.33	81.83
RTG-SLAM [19]	Acc.[cm]↓	1.67	1.23	1.32	1.16	0.92	1.48	1.80	1.67	1.41
	Acc. Ratio↑	99.16	**99.90**	**99.74**	99.91	99.86	99.40	98.85	99.54	99.54
	Comp. Ratio↑	78.80	83.97	84.87	88.40	87.66	76.79	74.51	78.68	81.71
Ours	Acc.[cm]↓	**1.52**	**1.20**	**1.29**	**1.09**	**0.88**	**1.44**	**1.78**	**1.54**	**1.34**
	Acc. Ratio↑	**99.85**	99.85	99.53	**99.93**	**99.99**	**99.67**	**99.05**	**99.73**	**99.70**
	Comp. Ratio↑	85.24	**86.35**	85.08	91.53	88.32	83.71	**82.84**	84.51	**85.95**

were conducted on a PC equipped with an NVIDIA RTX 3090 GPU. To facilitate a fair evaluation of reconstruction quality and rendering performance, we utilize ground truth poses for all baselines in our experiments.

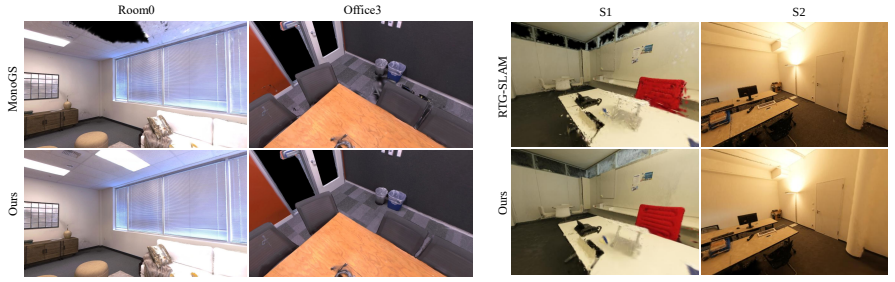
Datasets. We mainly evaluate the proposed method on two datasets, namely Replica [22] and ScanNet++ [25]. For Replica, we use eight sequences featuring living room and office scenarios. For ScanNet++, we use the DSLR captures from two scenes (8b5caf3398 (S1) and b20a261fdf (S2)).

4.2 Reconstruction Evaluation

We evaluate the reconstruction quality of our method on the Replica dataset, comparing it against the volumetric TSDF method KinectFusion [18], the NeRF-based RGB-D SLAM method NICE-SLAM [29], and recent Gaussian SLAM approaches, including RTG-SLAM [19] and MonoGS [16]. Meshes for RTG-SLAM

Table 2: Comparison of rendering performance on Replica [22].

Method	Metric	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
NICE-SLAM [29]	PSNR↑	24.72	26.79	27.06	30.21	32.78	26.59	26.22	24.74	27.39
	SSIM↑	0.787	0.799	0.807	0.881	0.906	0.816	0.801	0.834	0.829
	LPIPS↓	0.431	0.372	0.329	0.322	0.275	0.321	0.288	0.333	0.334
Point-SLAM [20]	PSNR↑	32.40	34.08	35.50	38.26	39.16	33.99	33.48	33.49	35.17
	SSIM↑	0.97	0.98	0.98	0.98	**0.99**	0.96	0.96	0.98	0.98
	LPIPS↓	0.11	0.12	0.11	0.10	0.12	0.16	0.13	0.14	0.12
MonoGS [16]	PSNR↑	34.83	36.43	37.49	39.95	42.09	**36.24**	**36.70**	36.07	37.50
	SSIM↑	0.95	0.96	0.97	0.97	0.98	0.96	0.96	0.96	0.96
	LPIPS↓	0.07	0.08	0.08	0.07	0.06	0.08	0.07	0.10	0.07
RTG-SLAM [19]	PSNR↑	31.56	34.21	35.57	39.11	40.27	33.54	32.76	36.48	35.43
	SSIM↑	0.97	0.98	0.98	**0.99**	**0.99**	0.98	0.98	0.98	0.98
	LPIPS↓	0.13	0.11	0.12	0.07	0.08	0.13	0.13	0.12	0.11
Ours	PSNR↑	**35.12**	**36.58**	**37.61**	**42.65**	**42.43**	35.30	34.86	**38.27**	**37.85**
	SSIM↑	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
	LPIPS↓	**0.04**	**0.04**	**0.03**	**0.02**	**0.02**	**0.03**	**0.05**	**0.03**	**0.03**



(a) Comparison on Replica (vs. MonoGS)

(b) Comparison on ScanNet++ (vs. RTG-SLAM)

Fig. 5: Qualitative comparisons of novel view synthesis across different datasets. (a) Results on Replica compared with MonoGS. Our method reconstructs sharper geometry details with fewer artifacts. **(b)** Results on ScanNet++ compared with RTG-SLAM. Note that our approach produces cleaner textures in complex indoor environments.

are generated using the ball pivoting algorithm [2], which effectively exploits the geometric attributes of its Gaussians. As illustrated in Fig. 4, our method produces finer geometric details and better preserves the smoothness of planar surfaces, such as walls. To evaluate the quality of the reconstructed geometry, we adopt the standard metrics: Accuracy, Accuracy Ratio (< 5cm), and Completion Ratio (< 5cm). For MonoGS, we uniformly sample an equal number of points from the reconstructed Gaussians, while other baselines are directly evaluated on the generated meshes. As shown in Tab. 1, our proposed method not only

Table 3: Novel & Train View Rendering Performance on ScanNet++ [25].

Methods	Metrics	Novel View			Training View		
		Avg.	S1	S2	**Avg.**	S1	S2
Point-SLAM [20]	PSNR ↑	22.53	22.03	23.02	25.94	25.34	26.54
	SSIM ↑	0.77	0.75	0.78	0.86	0.83	0.89
	LPIPS ↓	0.47	0.47	0.46	0.32	0.36	0.27
RTG-SLAM [19]	PSNR ↑	21.65	19.01	24.29	22.29	19.73	24.85
	SSIM ↑	0.83	0.81	0.84	0.88	0.87	**0.90**
	LPIPS ↓	0.33	0.34	**0.33**	0.24	0.26	**0.23**
Ours	PSNR ↑	**24.40**	**23.78**	**25.03**	**26.89**	**26.65**	**27.14**
	SSIM ↑	**0.85**	**0.85**	**0.85**	**0.91**	**0.91**	**0.90**
	LPIPS ↓	**0.32**	**0.30**	0.34	**0.23**	**0.21**	0.24

Table 4: Comparison of runtime and peak memory usage on Replica (Off0).

Method	FPS ↑	Mem. (MB) ↓
MonoGS	1.48	5434
RTG-SLAM	3.65	2751
Ours	**10.34**	**2325**

Table 5: Efficiency Comparison of Gaussian-to-Mesh Conversion.

Method	Mesh Size (MB) ↓	Time (s) ↓
GOF (MC)	964.95	444.26
GOF (PSR)	14.05	415.27
Ours	**3.32**	**5.34**

achieves the best average performance but also attains the highest accuracy on each Replica sequence.

4.3 Rendering Evaluation

We quantitatively evaluate the rendering performance of our method on the Replica dataset and adopt PSNR, SSIM, and LPIPS as the evaluation metrics. To demonstrate the efficacy of our proposed approach, we compare our method with NeRF RGB-D SLAM methods like NICE-SLAM [29], Point-SLAM [20], and two Gaussian SLAM methods, including MonoGS [16] and RTG-SLAM [19]. The results are directly from the publications [16, 19]. As shown in Tab. 2, our approach achieves superior rendering performance over all baseline methods in almost all sequences. The novel view synthesis results in Fig. 5a demonstrate that our approach effectively reduces artifacts and preserves fine-grained details while generating more complete scenes.

We further assess the rendering capability of our method through quantitative and qualitative evaluations on the ScanNet++ dataset. The experiments for RTG-SLAM and Point-SLAM [20] were conducted based on their official setup. Tab. 3 shows that our method achieves superior rendering quality over Point-SLAM and RTG-SLAM in both training views and novel views. Furthermore, the novel view rendering results in Fig. 5b qualitatively demonstrate that our method achieves finer details with fewer artifacts. Both quantitative and qualitative results on Replica and ScanNet++ demonstrate that our approach

Table 6: Ablation studies on key modules and loss terms for mesh reconstruction.

Key Modules	Acc.[cm]↓	Acc. Ratio↑	Comp. Ratio↑	Loss Terms	Acc.[cm]↓	Acc. Ratio↑	Comp. Ratio↑
w/o Planar Constraint	1.19	99.04	88.58	Baseline	1.13	98.58	86.34
w/o Neighborhood Selection	1.26	98.59	85.14	$+L_{sparse}$	1.09	98.92	86.50
w/o Direct Triangulation	0.89	99.98	84.49	$+L_n + L_{sparse}$	0.95	98.72	86.13
w/o Local Remeshing	1.32	97.99	**91.10**	$+L_{plane} + L_{sparse}$	0.93	99.78	86.50
w/o Freezing	0.92	99.59	89.27	$+L_{plane} + L_n$	0.90	99.77	86.04
Full Method	**0.88**	**99.99**	88.32	Full losses	**0.88**	**99.99**	**88.32**

surpasses baseline methods in detail preservation, scene completeness, and artifact reduction.

4.4 Efficiency

To demonstrate the efficiency of our approach, we evaluate the whole mapping frame rate (FPS) and peak memory usage during mapping on the 'Office0' sequence of Replica. We compare with MonoGS and RTG-SLAM, conducting experiments on the same workstation for fairness. As shown in Tab. 4, our method achieves the highest mapping speed at 10.34 FPS and the lowest peak memory of 2325 MB. This efficiency stems from our incremental mesh freezing strategy. By freezing the triangular meshes and Gaussian surfels within fully optimized historical regions, our system restricts active optimization to a limited local window, substantially reducing memory overhead and accelerating computation.

To investigate mesh extraction efficiency, we compare with Gaussian Opacity Field (GOF) [27], which reconstructs meshes via a volumetric opacity field followed by Marching Cubes (MC) or Poisson Surface Reconstruction (PSR). Since GOF requires excessive memory for a global implicit field, we restrict the comparison to a small subset of the 'Office0' scene from Replica. Table 5 shows our method takes only 5.34 seconds, much faster than GOF-MC (444.26 s) and GOF-PSR (415.27 s). This speedup comes from directly triangulating Gaussian surfels locally without constructing dense 3D voxel grids or evaluating global implicit fields. Regarding mesh size, GOF demands excessively massive meshes (e.g., 964.95 MB for GOF-MC) to capture fine details. In contrast, our planar constraints align surfels tightly to the true surface, allowing higher-fidelity geometry with a compact mesh.

4.5 Ablation Study

Core Components. Tab. 6 reports ablation results on Replica (Office1). The left part evaluates key modules, while the right part studies the contribution of optimization loss terms. In 'w/o Direct Triangulation', we adopt Poisson surface reconstruction in place of the proposed direct triangulation. Each module improves reconstruction quality, and the full model achieves the best Acc. and Acc. ratio. All loss terms contribute positively, and the complete objective yields the best overall performance, reducing the accuracy error from 1.13 cm to 0.88 cm and improving the completion ratio from 86.34% to 88.32%.

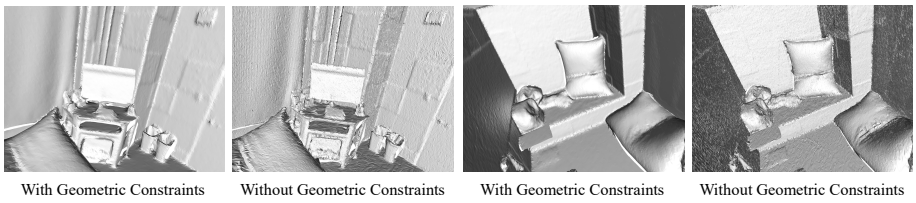


Fig. 6: Qualitative effect of geometric constraints on mesh reconstruction. We display the reconstruction details. When only depth constraints are applied without geometric constraints, the reconstruction captures the general outline of the scene but suffers from low surface fidelity and exhibits noticeably coarse textures.

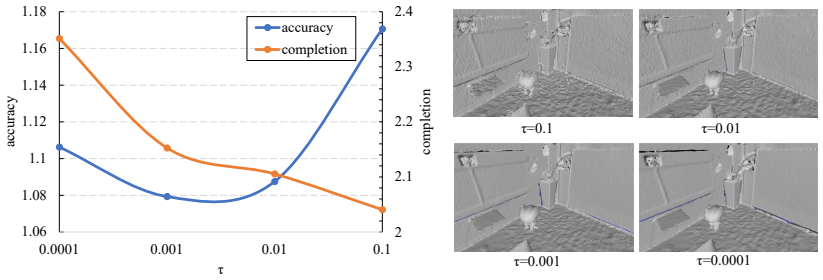


Fig. 7: Ablation Study on parameter τ to explore its impact on reconstruction quality.

Geometric Constraints. To further validate the two geometric constraints $L_{plane}+L_n$, we provide a visual comparison of the reconstruction details in Fig. 6. The reconstructed mesh without geometric constraints captures the coarse geometry and exhibits uneven and rough surfaces. In contrast, our method with geometric constraints recovers smoother and more accurate meshes. This indicates that incorporating geometric constraints guides Gaussians to better conform to the object surface, which ultimately improves the quality of reconstruction.

Geometric Gaussian Set. Since the quality of the geometric Gaussian set directly affects reconstruction, the parameter τ in the selection strategy significantly impacts mesh quality. To explore its influence on reconstruction, we conduct ablation on the Replica dataset with τ settings. We set τ ranging from 0.0001 to 0.1 across orders of magnitude. Accuracy and completion are used as reconstruction quality metrics, where lower values signify better performance. Fig. 7 illustrates both quantitative and qualitative impacts of τ. Increasing τ yields better completion but penalizes accuracy. Furthermore, the qualitative results demonstrate that improperly small values ($\tau = 0.0001$) cause obvious missing geometry. Therefore, in our implementation, we adopt $\tau = 0.001$ to achieve an optimal balance between precision and completeness.

5 Conclusion

This paper proposed an incremental online scene reconstruction method using Gaussian triangulation. A dense geometric Gaussian representation was presented as an explicit and efficient foundation for both high-fidelity rendering and surface reconstruction. Building on this Gaussian representation, a novel direct meshing algorithm was proposed to incrementally reconstruct and update triangular meshes. To improve reconstruction accuracy, we introduced a plane-based pulling constraint that can dynamically align 3D Gaussians to local surface approximations. Furthermore, we achieved memory-efficient incremental reconstruction by progressively freezing fully optimized mesh regions. Experiments on public datasets demonstrate that our method incrementally recovers high-quality meshes online and simultaneously achieves high-fidelity rendering. One limitation of our method is its reliance on depth information and its inability to reconstruct unobserved regions. We hope to explore 3D reconstruction using only RGB information in future work.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No.62376244 and the National Key Research and Development Program of China (No. 2023YFF0905104). It is also supported by the Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 5855–5864 (2021)
2. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE transactions on visualization and computer graphics **5**(4), 349–359 (2002)
3. Botsch, M., Kobbelt, L.: A remeshing approach to multiresolution modeling. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. pp. 185–192 (2004)
4. Cheng, K., Long, X., Yang, K., Yao, Y., Yin, W., Ma, Y., Wang, W., Chen, X.: Gaussianpro: 3d gaussian splatting with progressive propagation. In: Forty-first International Conference on Machine Learning (2024)
5. Dai, P., Xu, J., Xie, W., Liu, X., Wang, H., Xu, W.: High-quality surface reconstruction using gaussian surfels. In: ACM SIGGRAPH 2024 conference papers. pp. 1–11 (2024)
6. Gopi, M.: A fast and efficient projection-based approach for surface reconstruction. Int. J. of High Performance Computer Graphics, Multimedia and Visualization **1**(1), 1–12 (2000)

7. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5354–5363 (2024)
8. Häne, C., Heng, L., Lee, G.H., Fraundorfer, F., Furgale, P., Sattler, T., Pollefeys, M.: 3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. Image and Vision Computing **68**, 14–27 (2017)
9. Huang, B., Yu, Z., Chen, A., Geiger, A., Gao, S.: 2d gaussian splatting for geometrically accurate radiance fields. In: ACM SIGGRAPH 2024 conference papers. pp. 1–11 (2024)
10. Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., Ma, Y.: Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5322–5332 (2024)
11. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. pp. 61–70 (2006)
12. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139–1 (2023)
13. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
14. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20654–20664 (2024)
15. Marton, Z.C., Rusu, R.B., Beetz, M.: On fast surface reconstruction methods for large and noisy point clouds. In: 2009 IEEE international conference on robotics and automation. pp. 3218–3223. IEEE (2009)
16. Matsuki, H., Murai, R., Kelly, P.H., Davison, A.J.: Gaussian splatting slam. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18039–18048 (2024)
17. 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 **65**(1), 99–106 (2021)
18. 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: 2011 10th IEEE international symposium on mixed and augmented reality. pp. 127–136. IEEE (2011)
19. 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: ACM SIGGRAPH 2024 Conference Papers. pp. 1–11 (2024)
20. 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. pp. 18433–18444 (2023)
21. Schöps, T., Oswald, M.R., Speciale, P., Yang, S., Pollefeys, M.: Real-time view correction for mobile devices. IEEE transactions on visualization and computer graphics **23**(11), 2455–2462 (2017)
22. 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)

23. 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. pp. 6229–6238 (2021)
24. Waczyńska, J., Borycki, P., Tadeja, S., Tabor, J., Spurek, P.: Games: Mesh-based adapting and modification of gaussian splatting (2024), https://arxiv.org/abs/2402.01459
25. Yeshwanth, C., Liu, Y.C., Nießner, M., Dai, A.: Scannet++: A high-fidelity dataset of 3d indoor scenes. In: Proceedings of the International Conference on Computer Vision (ICCV) (2023)
26. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 19447–19456 (2024)
27. Yu, Z., Sattler, T., Geiger, A.: Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. ACM Transactions on Graphics (ToG) **43**(6), 1–13 (2024)
28. Zhang, W., Liu, Y.S., Han, Z.: Neural signed distance function inference through splatting 3d gaussians pulled on zero-level set. Advances in Neural Information Processing Systems **37**, 101856–101879 (2024)
29. 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. pp. 12786–12796 (2022)