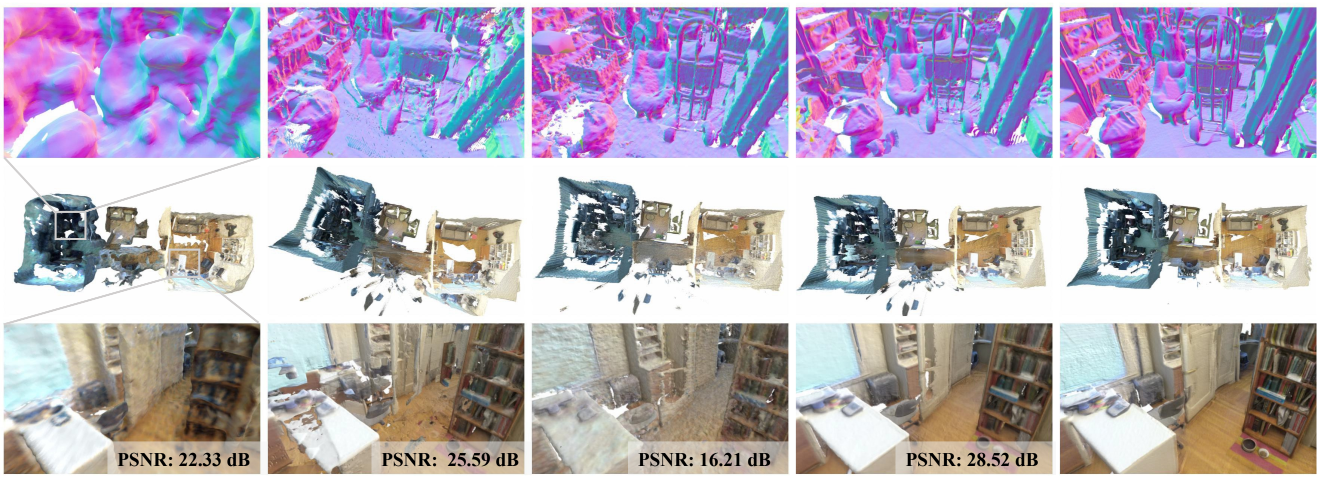
LoopSplat: Loop Closure by Registering 3D Gaussian Splats

Liyuan Zhu[1] Yue Li[2] Erik Sandström[3] Shengyu Huang[3] Konrad Schindler[3] Iro Armeni[1]

[1]Stanford University [2]University of Amsterdam [3]ETH Zurich



GO-SLAM [98] Gaussian-SLAM [92] Loopy-SLAM [39] LoopSplat (Ours) Ground Truth

Figure 1. **Dense Reconstruction on ScanNet [17]** `scene0054`. LoopSplat demonstrates superior performance in geometric accuracy, robust tracking, and high-quality re-rendering. This is enabled by our globally consistent reconstruction approach utilizing 3DGS [37].

Abstract

Simultaneous Localization and Mapping (SLAM) based on 3D Gaussian Splats (3DGS) has recently shown promise towards more accurate, dense 3D scene maps. However, existing 3DGS-based methods fail to address the global consistency of the scene via loop closure and/or global bundle adjustment. To this end, we propose LoopSplat, which takes RGB-D images as input and performs dense mapping with 3DGS submaps and frame-to-model tracking. LoopSplat triggers loop closure online and computes relative loop edge constraints between submaps directly via 3DGS registration, leading to improvements in efficiency and accuracy over traditional global-to-local point cloud registration. It uses a robust pose graph optimization formulation and rigidly aligns the submaps to achieve global consistency. Evaluation on the synthetic Replica and real-world TUM-RGBD, ScanNet, and ScanNet++ datasets demonstrates competitive or superior tracking, mapping, and rendering compared to existing methods for dense RGB-D SLAM. Code is available at loopsplat.github.io.

1. Introduction

Dense Simultaneous Localization and Mapping (SLAM) with RGB-D cameras has seen steady progress throughout the years from traditional approaches [8, 18, 52, 53, 65, 84] to neural implicit methods [32, 39, 45, 61, 62, 72, 75, 80, 87, 98, 100] and recent methods that employ 3D Gaussians [37] as the scene representation [29, 35, 47, 85, 92]. Existing methods can be split into two categories, *decoupled* and *coupled*, where *decoupled* methods [15, 29, 48, 59, 98] do not leverage the dense map for the tracking task, while the *coupled* methods [35, 39, 42, 47, 61, 62, 72, 75, 80, 85, 87, 92, 100] employ frame-to-model tracking using the dense map. Decoupling mapping and tracking generally creates undesirable redundancies in the system, such as inefficient information sharing and increased computational overhead. On the other hand, all *coupled* 3DGS SLAM methods lack strategies for achieving global consistency on the map and the poses, which leads to an accumulation of pose errors and distorted maps. Among the recent methods that enforce global consistency via loop closure and/or global bundle adjustment (BA), GO-SLAM [98] requires costly retraining of the hash grid features to deform the map and Photo-SLAM [29] similarly requires additional optimization of the 3D Gaussian parameters to resolve pose updates from the ORB-SLAM [51] tracker. These re-integration techniques need to save all mapped frames in memory, which limits their scalability. To avoid saving all mapped frames, Loopy-SLAM [39] uses submaps of neural point clouds and rigidly updates them after loop closure. However, to compute the loop edge constraints, Loopy-SLAM uses traditional global-to-local point cloud registration. This is not only slow, but also fails to leverage the property of the scene representation itself. To address limitations of current systems, we seek a *coupled* SLAM system that avoids saving all mapped input frames and is able to extract loop constraints directly from the dense map, without redundant compute. Framed as a research question, we ask: *Can we use the map representation (i.e., 3DGS) itself for loop closure in a SLAM system?* To this end, we propose a dense RGB-D SLAM system that uses submaps of 3D Gaussians for local frame-to-model tracking and dense mapping and is based on existing systems [47, 92]. Different to the latter, we achieve global consistency via online loop closure detection and pose graph optimization. Importantly, we show that traditional point cloud registration techniques are not suitable to derive the loop edge constraints from 3D Gaussians and propose a new registration method that directly operates on the 3DGS representation, hence using 3DGS as a unified scene representation for tracking, mapping, and maintaining global consistency. Our key **contributions** are:

1. We introduce *LoopSplat*, a coupled RGB-D SLAM system based on Gaussian Splatting, featuring a novel loop closure module. This module operates directly on Gaussian splats, integrating both 3D geometry and visual scene content for robust loop detection and closure.
2. We develop an effective way to register two 3DGS representations, so as to efficiently extract edge constraints for pose graph optimization. Leveraging the fast rasterization of 3DGS, it is seamlessly integrated into the system, outperforming traditional techniques in terms of both speed and accuracy.
3. We enhance the tracking and reconstruction performance of 3DGS-based RGB-D SLAM system, demonstrating marked improvements and increased robustness across diverse real-world datasets.

2. Related Work

Dense Visual SLAM. The seminal work of Curless and Levoy [16] paved the way for dense 3D mapping with truncated signed distance functions. Using frame-to-model tracking, KinectFusion [52] showed that real-time SLAM is possible from a commodity depth sensor. To address the cubic memory scaling to the scene size, numerous works utilized voxel hashing [18, 33, 48, 53, 54] and octrees [10, 40, 46, 66, 87] for map compression. Point-based representations have also been popular [8, 12, 33, 36, 39, 61, 65, 84, 95, 96], with surfels and lately using neural points or 3D Gaussians [35, 47, 63, 85, 92, 95]. To tackle the issue of accumulating pose errors, globally consistent dense SLAM methods have been developed, where a subdivision of the global map into submaps is common [5, 8, 13, 18, 22, 26, 33, 34, 39, 43–45, 48, 58, 69, 75, 75], followed by pose graph optimization [8, 13, 19, 20, 25–27, 34, 38, 39, 44, 45, 48, 48, 58, 65, 69, 75, 79, 86] to deform the submaps between them. Additionally, some works employ global BA for refinement [8, 15, 18, 27, 48, 65, 75, 76, 86, 88, 98]. 3D Gaussian SLAM with RGB-D input has also been shown, however, methods fail to consider global consistency [35, 47, 85, 92], leading to error accumulation in the map and pose estimates. Most similar to our work is Loopy-SLAM [39], which uses the explicit neural point cloud representation of Point-SLAM [61] and equips it with global consistency via loop closure on submaps. LoopSplat differentiates itself from Loopy-SLAM and demonstrates improvements in three key areas: *(i)* We improve the accuracy and efficiency of the relative pose constraints by directly registering 3DGS, instead of resorting to classical techniques like FPFH [60] with RANSAC, followed by ICP [4]. *(ii)* We avoid having to mesh the submaps in a separate process for registration and use the 3D Gaussians directly. *(iii)* For loop detection, we rely on a combination of image matching and overlap between submaps, leading to better detections than using only image content as in [39].

Geometric Registration. Geometric registration is an important component of building edge constraints for pose graphs. Specifically, point cloud registration aims to find a rigid transformation that aligns two point cloud fragments into the same coordinate framework. Traditional methods leverage hand-crafted local descriptors [60, 77] for feature matching, followed by RANSAC for pose estimation. Recent learning-based methods either use patch-based local descriptors [23, 93] or efficient fully-convolutional ones [3, 14]. BUFFER [1] balances the efficiency and generalization of local descriptors by combining fully-convolutional backbones for key-point detection with a patch-based network for feature description. To address fragment registration with low overlap, Predator [31] uses attention mechanisms [78] to guide key-point sampling, significantly improving the robustness of algorithms. This has been further enhanced through coarse-to-fine matching [57]. Point clouds lack the continuous, view-dependent, and multi-scale representation capabilities of NeRFs, limiting their ability to fully capture complex 3D scenes in SLAM.

Neural Radiance Fields (NeRFs) [50] have been widely adopted for various applications beyond scene reconstruction, including scene understanding [21], autonomous driving [82], and SLAM [55, 73]. When modeling large-scale

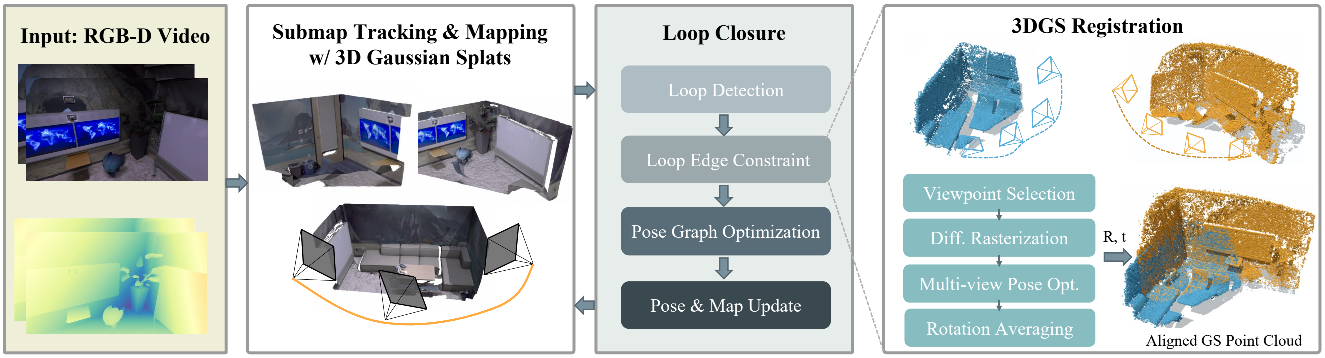


Figure 2. **LoopSplat Overview.** LoopSplat is a *coupled* RGB-D SLAM system that uses Gaussian splats as a **unified** scene representation for tracking, mapping, and maintaining global consistency. In the front-end, it continuously estimates the camera position while constructing the scene using Gaussian splats. When the camera traverses beyond a predefined threshold, the current submap is finalized, and a new one is initiated. Concurrently, the back-end loop closure module monitors for location revisits. Upon detecting a loop, the system generates a pose graph, incorporating loop edge constraints derived from our proposed **3DGS registration**. Subsequently, pose graph optimization (PGO) is executed to refine both camera poses and submaps, ensuring overall spatial coherence.

scenes with NeRFs, it is necessary to partition the scene into blocks to manage memory constraints and to ensure sufficient representation power. Consequently, registering NeRFs to merge different partitions emerged as a research problem. iNeRF [89] aligns a query image to the NeRF map through analysis-by-synthesis: it optimizes the camera pose so that the rendered image matches the query. However, this method is only suitable for local refinement due to its non-convex nature, which can cause the model to get stuck in local minima. NeRF2NeRF [24] aims to align two NeRFs by extracting surface points from the density field and aligning manually selected keypoints to estimate the pose. DReg-NeRF [11] addresses NeRF registration similarly to point cloud registration, by first extracting surface points and then applying a fully convolutional feature extraction backbone. Recently, Gaussian Splatting [37] has started to replace NeRFs due to its efficient rasterization and flexible editing capabilities, afforded by the explicit representation. GaussReg [9] pioneered learning-based 3D Gaussian Splatting (3DGS) registration, drawing on the fast rendering of 3DGS. However, all previous NeRF and 3DGS registration methods [9, 11, 24, 89] assume ground truth camera poses for training views, which is not applicable to real-world SLAM scenarios. Moreover, these methods have only explored pairwise registration in small-scale scenes. Our method, without any training or preprocessing, directly operates on estimated camera poses from the SLAM front-end and can be integrated into loop closure on the fly.

3. LoopSplat

LoopSplat is an RGB-D SLAM system that simultaneously estimates the camera poses and builds a 3D Gaussian map from input frames in a globally consistent manner. This section begins with a recap of the Gaussian-SLAM system described in [92] (Sec. 3.1) – which is the base of LoopSplat, followed by the introduction of the proposed 3DGS registration module (Sec. 3.2). Finally, the integration of loop closure into the Gaussian-SLAM system, enabled by the registration module, is presented in Sec. 3.3. Please see Fig. 2 for an overview of the proposed system.

3.1. Gaussian Splatting SLAM

We follow [39, 92] and represent the scene using a collection of submaps, each modeling several keyframes with a 3D Gaussian point cloud $\mathbf{P}^s$, where

$$\mathbf{P}^s = \{G_i(\mu, \Sigma, o, C) | i = 1, \dots, N_G\}, \tag{1}$$

with the submap index s, individual Gaussian mean $\mu \in \mathbb{R}^3$, covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, opacity value $o \in \mathbb{R}$, number of Gaussians N_G in each submap, and RGB color $C \in \mathbb{R}^3$.

Submap Initialization. Starting from the first keyframe $\mathbf{I}^s_f$, each submap models a sequence of keyframes observing a specific region. As the explored scene space expands, a new submap is initialized to avoid processing the entire global map simultaneously. Unlike previous approaches that use a fixed number of keyframes [13, 18, 43], we dynamically trigger new submap initialization when the current frame's relative displacement or rotation to the first keyframe $\mathbf{I}^s_f$ exceeds the predefined thresholds, d_{thre} or θ_{thre}.

Frame-to-model Tracking. To localize a incoming frame j within the current submap $\mathbf{P}^s$, we first initialize the camera pose $\mathbf{T}_j$ based on the constant motion assumption as: $\mathbf{T}_j = \mathbf{T}_{j-1} \cdot \mathbf{T}^{-1}_{j-2} \cdot \mathbf{T}_{j-1}$. Next, we optimize $\mathbf{T}_j$ by minimizing the tracking loss $\mathcal{L}_{\text{tracking}}(\hat{\mathbf{I}}^s_j, \hat{\mathbf{D}}^s_j, \mathbf{I}^s_j, \mathbf{D}^s_j, \mathbf{T}_j)$, which measures the discrepancy between the rendered color $\hat{\mathbf{I}}^s_j$ and depth $\hat{\mathbf{D}}^s_j$ images at viewpoint $\mathbf{T}_j$, and the input color $\mathbf{I}^s_j$ and depth $\mathbf{D}^s_j$. To stabilize tracking, we use an alpha mask

M_{a} and an inlier mask M_{in} to address gross errors caused by poorly reconstructed or previously unobserved areas. The final tracking loss is a sum over the valid pixels as

$$\mathcal{L}_{\text{tracking}} = \sum M_{\text{in}} \cdot M_{\text{a}} \cdot (\lambda_c |\hat{\mathbf{I}}^s_j - \mathbf{I}^s_j|_1 + (1-\lambda_c)|\hat{\mathbf{D}}^s_j - \mathbf{D}^s_j|_1), \tag{2}$$

where λ_c is a weight that balances the color and depth losses, and $|\cdot|$ denotes the $\mathcal{L}_1$ loss between two images. Please refer to the supplementary material for more details.

Submap Expansion. Keyframes are selected by fixed interval for the submap. Once the current keyframe $\mathbf{I}^s_j$ is localized, we expand the 3D Gaussian map primarily in sparsely covered regions for efficient mapping. We first compute a posed dense point cloud from the RGB-D input and then uniformly sample M_k points from areas where the accumulated alpha values are below a threshold α_{thre} or where significant depth discrepancies occur. These points are initialized as anisotropic 3D Gaussians, with scales defined based on the nearest neighbor distance within the current submap. New 3D Gaussian splats are added to the current submap only if there is no existing 3D Gaussian mean within a radius ρ.

Submap Update. After new Gaussians are added, all Gaussians in the current submap are optimized for a fixed number of iterations by minimizing the rendering loss $\mathcal{L}_{\text{render}}$, computed over all keyframes of the submap, with at least 40% of the compute allocated to the most recent keyframe. The rendering loss has three components: color loss $\mathcal{L}_{\text{color}}$, depth loss $\mathcal{L}_{\text{depth}}$, and a regularization term $\mathcal{L}_{\text{reg}}$:

$$\mathcal{L}_{\text{render}} = \lambda_{\text{color}} \cdot \mathcal{L}_{\text{color}} + \lambda_{\text{depth}} \cdot \mathcal{L}_{\text{depth}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}, \tag{3}$$

where λ_* are hyperparamters. Similar to the tracking loss, the depth loss is the $\mathcal{L}_1$ loss between rendered and ground truth depth maps. For color supervision, we use a weighted combination of the $\mathcal{L}_1$ and SSIM [81] losses:

$$\mathcal{L}_{\text{col}} = (1-\lambda_{SSIM}) \cdot |\hat{\mathbf{I}} - \mathbf{I}|_1 + \lambda_{SSIM}\big(1 - \text{SSIM}(\hat{\mathbf{I}}, \mathbf{I})\big), \tag{4}$$

where $\lambda_{SSIM} \in [0, 1]$. To regularize overly elongated 3D Gaussians in sparsely covered or barely observed regions, we add an isotropic regularization term [47]

$$\mathcal{L}_{\text{reg}} = \frac{1}{K} \sum_{k \in K} |s_k - \overline{s}_k|_1, \tag{5}$$

where $s_k \in \mathbb{R}^3$ is the scale of a 3D Gaussian, $\overline{s}_k$ is its mean, and K is the number of Gaussians in the submap. During optimization, to preserve geometry directly measured from the depth sensor and reduce computation time, we do not clone or prune the Gaussians [37].

3.2. Registration of Gaussian Splats

LoopSplat's first contribution relates to the registration of Gaussian splats which is formulated as following. Consider two overlapping 3D Gaussian submaps $\mathbf{P}$ and $\mathbf{Q}$, each reconstructed using different keyframes and not aligned. The goal is to estimate a rigid transformation $\mathbf{T}_{\mathbf{P}\to\mathbf{Q}} \in SE(3)$ that aligns $\mathbf{P}$ with $\mathbf{Q}$. Each submap is also associated with a set of viewpoints. The set of viewpoints $\mathbf{V}^{\mathbf{P}}$ for a submap $\mathbf{P}$ is:

$$\mathbf{V}^{\mathbf{P}} = \{\mathbf{v}^{\mathbf{P}}_i = (\mathbf{I}, \mathbf{D}, \mathbf{T})_i | i = 0, \ldots, N_P\}, \tag{6}$$

where $\mathbf{I}$ and $\mathbf{D}$ are the individual RGB and depth measurements, respectively, and $\mathbf{T}$ is the estimated camera pose in Sec. 3.1. N_P the number of viewpoints in the submap.

Overlap Estimation. Knowing the approximate overlap between the source and target submaps $\mathbf{P}$ and $\mathbf{Q}$ is crucial for robust and accurate registration, and this co-contextual information can be extracted by comparing feature similarities [31]. While the means of the Gaussian splats do form a point cloud, we found that estimating the overlap region directly from them, by matching local features, does not work well (*cf*. Sec. 4.5). Instead, we identify viewpoints from each submap that share similar visual content. Specifically, we first pass all keyframes through NetVLAD [2] to extract their global descriptors. We then compute the cosine similarity between the two sets of keyframes and retain the top-k pairs for registration.

Registration as Keyframe Localization. Given that the 3DGS submap and its viewpoints can be treated as one rigid body, we propose to approach 3DGS registration as a keyframe localization problem. For a selected viewpoint $\mathbf{v}^{\mathbf{P}}_i$, determining its camera pose $\mathbf{T}^q_i$ within $\mathbf{Q}$ allows one to render the same RGB-D image from $\mathbf{Q}$. Hence, the rigid transformation $\mathbf{T}_{\mathbf{P}\to\mathbf{Q}}$ can be computed as $\mathbf{T}^q_i \cdot \mathbf{T}^{-1}_i$.

During keyframe localization, we keep the parameters of $\mathbf{Q}$ fixed and optimize the rigid transformation $\mathbf{T}_{\mathbf{P}\to\mathbf{Q}}$ by minimizing the rendering loss $\mathcal{L} = \mathcal{L}_{col} + \mathcal{L}_{depth}$ [49], where both $\mathcal{L}_{col}$ and $\mathcal{L}_{depth}$ are $\mathcal{L}_1$ losses.

We estimate the rigid transformations for the selected viewpoints, from $\mathbf{P}$ to $\mathbf{Q}$ for viewpoints in $\mathbf{V}^{\mathbf{P}}$ and vice versa for $\mathbf{V}^{\mathbf{Q}}$, in parallel. The rendering residuals ϵ are also saved upon completion of the optimization. By using the sampled top-k viewpoints from the estimated overlap region as the selected viewpoints, the registration efficiency is greatly improved without redundancy in non-overlapping viewpoints. Viewpoint transformations are estimated first, then used to compute the submap's global transformation, as described next.

Multi-view Pose Refinement. Given a set of viewpoint transformations $\{(\mathbf{T}_{\mathbf{P}\rightleftharpoons\mathbf{Q}}, \varepsilon)_i\}_{i=1}^{2k}$, where the first k estimates are from $\mathbf{P} \rightarrow \mathbf{Q}$ and the last k estimates from $\mathbf{Q} \rightarrow \mathbf{P}$, one must find a global consensus for the transformation $\bar{\mathbf{T}}_{\mathbf{P}\rightarrow\mathbf{Q}}$. As the rendering residual indicates how well the transformed viewpoint fits the original observation, we take the reciprocal of the residuals as a weight for each estimate and apply weighted rotation averaging [6, 56] to compute the global rotation:

$$\bar{\mathbf{R}} = \arg\min_{\mathbf{R}\in SO3} \sum_{i=1}^{k} \frac{1}{\varepsilon_i} \|\mathbf{R}-\mathbf{R}_i\|_F^2 + \sum_{i=k+1}^{2k} \frac{1}{\varepsilon_i} \|\mathbf{R}-\mathbf{R}_i^{-1}\|_F^2, \tag{7}$$

where $\|\cdot\|_F^2$ denotes the Frobenius norm. The global translation is found as the weighted mean over individual estimates.

3.3. Loop Closure with 3DGS

Loop closure aims to identify pose corrections (*i.e.* relative transformations *w.r.t.* the current estimates) for past submaps and, thus, keyframes to ensure global consistency. This process is initiated when a new submap is created, and upon detecting a new loop, the pose graph, which includes all historical submaps, is constructed. The loop edge constraints for the pose graph are then computed using 3DGS registration (Sec. 3.2). Subsequently, Pose Graph Optimization (PGO) [13] is performed to achieve globally consistent multi-way registration of 3DGS.

Loop Closure Detection. To effectively detect system revisits to the same place, we first extract a global descriptor $\mathbf{d} \in \mathbb{R}^{1024}$ using a pretrained NetVLAD [2]. We compute the cosine similarities of all keyframes within the i-th submap and determine the self-similarity score s^i_{self} corresponding to their p-th percentile. We then apply the same method to compute the cross-similarity $s^{i,j}_{\text{cross}}$ between the i-th and j-th submaps. A new loop is added if $s^{i,j}_{\text{cross}} > \min(s^i_{\text{self}}, s^j_{\text{self}})$. However, relying solely on visual similarity for loop closure [39] can generate false loop edges, potentially degrading PGO performance. To mitigate that risk, we additionally evaluate the initial geometric overlap ratio r [31] between the Gaussians of two submaps, and retain only loops with $r > 0.2$. See Supp. for more details.

Pose Graph Optimization. We create a new pose graph every time a new loop is detected and ensure that its connections match the previous one, besides the new edges introduced by the new submap. The relative pose corrections $\{\mathbf{T}_{c^i} \in SE(3)\}$ to each submap are defined as nodes in the pose graph, which are connected with odometry edges and loop edges. Here $\mathbf{T}_{c^i}$ denotes the correction applied to i-th submap. The nodes and edges connecting adjacent nodes (*i.e.*, odometry edges) are initialized with identity matrices. Loop edge constraints are added at detected loops and initialized according to the Gaussian splatting registration (Sec. 3.2). The information matrices for edges are computed directly from the Gaussian centers and incorporated into the pose graph. PGO is triggered after loop detection and we use a robust formulation based on line processes [13].

Method	LC	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
Neural Implicit Fields										
NICE-SLAM [100]	✗	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.06
Vox-Fusion [87]	✗	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
ESLAM [32]	✗	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63	0.63
Point-SLAM [61]	✗	0.61	0.41	0.37	0.38	0.48	0.54	0.69	0.72	0.52
MIPS-Fusion [74]	✓	1.10	1.20	1.10	0.70	0.80	1.30	2.20	1.10	1.19
GO-SLAM [98]	✓	0.34	0.29	0.29	0.32	0.30	0.39	0.39	0.46	0.35
Loopy-SLAM [39]	✓	**0.24**	0.24	0.28	0.26	0.40	0.29	0.22	0.35	0.29
3D Gaussian Splatting										
SplaTAM [35]	✗	0.31	0.40	0.29	0.47	0.27	0.29	0.32	0.72	0.38
MonoGS [47]	✗	0.33	**0.22**	0.29	0.36	0.19	**0.25**	**0.12**	0.81	0.32
Gaussian-SLAM [92]	✗	0.29	0.29	0.22	0.37	0.23	0.41	0.30	0.35	0.31
*Photo-SLAM [29]	✓	0.54	0.39	0.31	0.52	0.44	1.28	0.78	0.58	0.60
LoopSplat (Ours)	✓	0.28	**0.22**	**0.17**	**0.22**	**0.16**	0.49	0.20	**0.30**	**0.26**

Table 1. **Tracking Performance on Replica [67]** (ATE RMSE ↓ [cm]). LC indicates loop closure. The best results are highlighted as **first**, second, and third. LoopSplat performs the best. *Photo-SLAM [29] is a *decoupled* method using ORB-SLAM3 [7] for tracking and loop closure.

Globally Consistent Map Adjustment. From the PGO output, we obtain a set of pose corrections $\{\mathbf{T}_{c^i} = [\mathbf{R}_{c^i}|\mathbf{t}_{c^i}]\}_{i=1}^{N_s}$ for N_s submaps, with c_i denoting correction for submap i. For each submap, we update camera poses, the Gaussian means, and covariances

$$\mathbf{T}_j \leftarrow \mathbf{T}_{c^i}\mathbf{T}_j \ , \tag{8}$$

$$\boldsymbol{\mu}_i \leftarrow \mathbf{R}_{c^i}\boldsymbol{\mu}_{\mathbf{S}^i} + \mathbf{t}_{c^i}, \ \ \boldsymbol{\Sigma}_i \leftarrow \mathbf{R}_{c^i}\boldsymbol{\Sigma}_{\mathbf{S}^i}\mathbf{R}_{c^i}^T \ . \tag{9}$$

Here, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ represent the sets of centers and covariance matrices, respectively, of the Gaussians in the i-th submap $\mathbf{S}^i$, index j is iterated over the keyframe span of the submap. We omit spherical harmonics (SH) to reduce the Gaussian map size and improve pose estimation accuracy [47].

4. Experiments

Here we describe our experimental setup and compare our method to state-of-the-art baselines. We evaluate tracking, reconstruction, and rendering performance on synthetic and real-world datasets, with a dedicated ablation study for loop closure. For implementation details, please refer to Supp.

Datasets. We evaluate on four datasets: *Replica* [68] is a synthetic dataset with high-quality 3D indoor reconstructions. We use the same RGB-D sequences as [72]. *ScanNet* [17] is a real-world dataset with its poses estimated by BundleFusion [18]. We evaluate on eight scenes with loops

Method	a	b	c	d	e	Avg.
Neural Implicit Fields						
Point-SLAM [61]	246.16	632.99	830.79	271.42	574.86	511.24
ESLAM [32]	25.15	2.15	27.02	20.89	35.47	22.14
GO-SLAM [98]	176.28	145.45	38.74	85.48	106.47	110.49
Loopy-SLAM [39]	N/A	N/A	25.16	234.25	81.48	113.63
3D Gaussian Splatting						
SplaTAM [35]	1.50	**0.57**	**0.31**	443.10	1.58	89.41
MonoGS [92]	7.00	3.66	6.37	3.28	44.09	12.88
Gaussian SLAM [92]	1.37	2.82	6.80	3.51	**0.88**	3.08
LoopSplat (Ours)	**1.14**	3.16	3.16	**1.68**	0.91	**2.05**

Table 2. **Tracking Performance on ScanNet++ [90]** (ATE RMSE ↓ [cm]). LoopSplat achieves the highest accuracy and can robustly deal with the large camera motions in the sequence.

Method	00	59	106	169	181	207	54	233	Avg.
Neural Implicit Fields									
Vox-Fusion [87]	16.6	24.2	8.4	27.3	23.3	9.4	-	-	-
Co-SLAM [80]	7.1	11.1	9.4	**5.9**	11.8	7.1	-	-	-
MIPS-Fusion [74]	7.9	10.7	9.7	9.7	14.2	7.8	-	-	-
NICE-SLAM [100]	12.0	14.0	7.9	10.9	13.4	6.2	20.9	9.0	13.0
ESLAM [32]	7.3	8.5	7.5	6.5	9.0	**5.7**	36.3	**4.3**	10.6
Point-SLAM [61]	10.2	7.8	8.7	22.2	14.8	9.5	28.0	6.1	14.3
GO-SLAM [98]	5.4	7.5	**7.0**	7.7	**6.8**	6.9	8.8	4.8	**6.9**
Loopy-SLAM [39]	**4.2**	7.5	8.3	7.5	10.6	7.9	**7.5**	5.2	7.7
3D Gaussian Splatting									
MonoGS [47]	9.8	32.1	8.9	10.7	21.8	7.9	17.5	12.4	15.2
SplaTAM [35]	12.8	10.1	17.7	12.1	11.1	7.5	56.8	4.8	16.6
Gaussian-SLAM [92]	21.2	12.8	13.5	16.3	21.0	14.3	37.1	11.1	18.4
LoopSplat (Ours)	6.2	**7.1**	7.4	10.6	8.5	6.6	16.0	4.7	8.4

Table 3. **Tracking Performance on ScanNet [17]**. LoopSplat outperforms 3DGS-based systems by a large margin and is on par with the state-of-the-art baselines.

following [39, 98]. *ScanNet++* [90] is a real, high-quality dataset. We use five DSLR-captured sequences where poses are estimated with COLMAP [64] and refined with the help of laser scans. *TUM-RGBD* [70] is a real-world dataset with accurate poses obtained from a motion capture system.

Baselines. We compare LoopSplat with state-of-the-art *coupled* RGB-D SLAM methods, categorized into two groups based on the underlying scene representation: *(i)* Neural implicit fields: MIPS-Fusion [74], GO-SLAM [98], and Loopy-SLAM [39], all of which incorporate loop closure; and *(ii)* 3DGS: MonoGS [47], SplaTAM [35], Gaussian-SLAM [92], and Photo-SLAM [29]. For completeness, we include Photo-SLAM [29] in our evaluation, noting that it utilizes ORB-SLAM3 [7] for tracking and loop closure, setting it apart from all other tested methods.

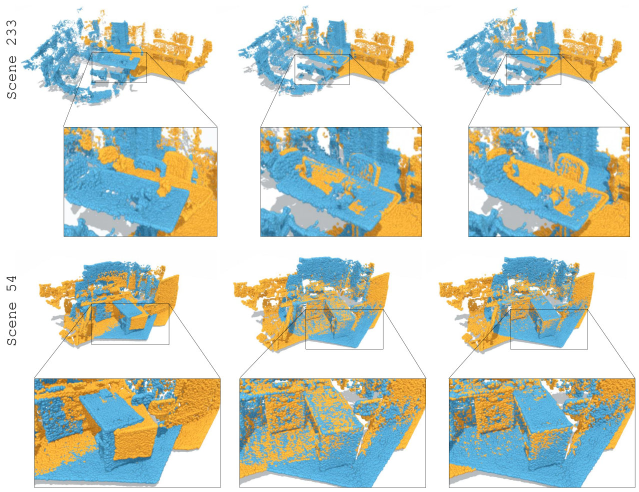
Evaluation Metrics. *Tracking* accuracy is measured by the root mean square absolute trajectory error (ATE RMSE) [70]. For *reconstruction*, we follow [61] and evaluate via meshes extracted with marching cubes [41], using a voxel size of 1 cm. We measure rendered mesh depth error at sampled novel views as in [100] and the F1-score, *i.e.*, the harmonic mean of precision and recall *w.r.t.* ground truth mesh vertices. *Rendering* quality is evaluated by comparing full-resolution rendered images to input training views in terms of PSNR, SSIM [81], and LPIPS [97]. We note that comparing to training views may yield too optimistic results, but it enables a consistent comparison with existing methods. To assess *map size*, we measure the total memory needed for the map and the peak GPU memory usage. *Runtime* is reported as average per-frame tracking and map optimization time, as well as loop edge registration runtime.

Method	LC	fr1/desk	fr1/desk2	fr1/room	fr2/xyz	fr3/off.	Avg.
Neural Implicit Fields							
DI-Fusion [30]	✗	4.4	N/A	N/A	2.0	5.8	N/A
NICE-SLAM [100]	✗	4.26	4.99	34.49	6.19	3.87	10.76
Vox-Fusion [87]	✗	3.52	6.00	19.53	1.49	26.01	11.31
MIPS-Fusion [74]	✓	3.0	N/A	N/A	1.4	4.6	N/A
Point-SLAM [61]	✗	4.34	4.54	30.92	1.31	3.48	8.92
ESLAM [32]	✗	2.47	3.69	29.73	1.11	2.42	7.89
Co-SLAM [80]	✗	2.40	N/A	N/A	1.70	2.40	N/A
GO-SLAM [98]	✓	**1.50**	N/A	**4.64**	0.60	1.30	N/A
Loopy-SLAM [39]	✓	3.79	3.38	7.03	1.62	3.41	3.85
3D Gaussian Splatting							
SplaTAM [35]	✗	3.35	6.54	11.13	1.24	5.16	5.48
MonoGS [47]	✗	1.59	7.03	8.55	1.44	1.49	4.02
Gaussian-SLAM [92]	✗	2.73	6.03	14.92	1.39	5.31	6.08
*Photo-SLAM [29]	✓	2.60	N/A	N/A	**0.35**	1.00	N/A
LoopSplat (Ours)	✓	2.08	3.54	6.24	1.58	3.22	3.33
Classical							
BAD-SLAM [65]	✓	1.7	N/A	N/A	1.1	1.7	N/A
Kintinuous [83]	✓	3.7	7.1	7.5	2.9	3.0	4.84
ORB-SLAM2 [51]	✓	1.6	**2.2**	4.7	0.4	1.0	**1.98**
ElasticFusion [84]	✓	2.53	6.83	21.49	1.17	2.52	6.91
BundleFusion [18]	✓	1.6	N/A	N/A	1.1	2.2	N/A
Cao *et al.* [8]	✓	**1.5**	N/A	N/A	0.6	**0.9**	N/A
Yan *et al.* [86]	✓	1.6	N/A	5.1	N/A	3.1	N/A

Table 4. **Tracking Performance on TUM-RGBD [71]** (ATE RMSE ↓ [cm]). * indicates using ORB-SLAM3 [7] for tracking and loop closure. LoopSplat performs the best among *coupled* SLAM, further closing the gap to sparse solver-based SLAM.

4.1. Tracking

We report the camera tracking performance in Tabs. 1 to 4. On Replica, we outperform all the baselines, achieving a 10% higher accuracy compared to the second best one. On real-world datasets, we achieve the highest pose accuracy on TUM-RGBD and ScanNet++ among all neural implicit field-based and 3DGS-based baselines, improving tracking accuracy by 14% and 33%, respectively. It is worth noting that, for all 3DGS-based baselines [35, 47, 92], trajectory errors accumulate as trajectories grow longer in larger scenes with loops and motion blur, *e.g.*, ScanNet `00`, `59`, `181` and TUM-RGBD `fr1/desk2` and `fr1/room`. We attribute our superior tracking performance to the robust 3DGS registration that underpins our loop closure. On ScanNet, we obtain the third-best performance. We note that the ground truth poses in ScanNet, derived from BundleFusion [18], appear to have limited accuracy: visual inspection suggests that our method achieves better alignment and reconstruction than the ground truth; see Fig. 1, Fig. 3, and `scene 233` in Fig. 4. Additional qualitative



Gaussian-SLAM [92] Ours Ground Truth

Figure 3. **Comparison of Submap Alignment on ScanNet [17].** We visualize the centers of 3D Gaussians as point clouds. Two submaps are colorized differently. LoopSplat consistently aligns the submaps better than Gaussian-SLAM [92].

examples are in Supp. Besides superior tracking accuracy, our coupled method avoids redundant computations for separate tracking and map reconstruction, in contrast to *decoupled* ones like GO-SLAM [98] and Photo-SLAM [29].

4.2. Reconstruction

We evaluate the mesh reconstruction quality on Replica, the only dataset with high accuracy ground truth mesh, in Tab. 5[1]. LoopSplat outperforms all 3DGS-based baselines attributed to more accurate pose estimates. LoopSplat falls behind Loopy-SLAM [39] and Point-SLAM [61] but note that the latter two require ground truth depth to determine where to sample points during ray-marching, thus assuming perfect input depth. Fig. 4 compares ScanNet meshes reconstructed with LoopSplat to those of the best-performing baselines, GO-SLAM and Loopy-SLAM (both also including loop closure), as well as to a 3DGS baseline, Gaussian-SLAM (which does not perform loop closure). Our method recovers more geometric details (*e.g.*, on the chairs). On ScanNet `233`, the visual quality and completeness of our reconstruction appears even better than the ground truth, especially on the floor, desk, and bed.

4.3. Rendering

Tab. 6 reports our rendering performance on training views. To conduct a fair comparison, we merge all the submaps into a global one and optimize the global map with estimated cameras pose, to avoid local overfitting on submaps[2].

Method	Metric	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
Neural Implicit Fields										
NICE-SLAM [100]	Depth L1 [cm] ↓	1.81	1.44	2.04	1.39	1.76	8.33	4.99	2.01	2.97
	F1 [%] ↑	45.0	44.8	43.6	50.0	51.9	39.2	39.9	36.5	43.9
Vox-Fusion [87]	Depth L1 [cm] ↓	1.09	1.90	2.21	2.32	3.40	4.19	2.96	1.61	2.46
	F1 [%] ↑	69.9	34.4	59.7	46.5	40.8	51.0	64.6	50.7	52.2
ESLAM [32]	Depth L1 [cm] ↓	0.97	1.07	1.28	0.86	1.26	1.71	1.43	1.06	1.18
	F1 [%] ↑	81.0	82.2	83.9	78.4	75.5	77.1	75.5	79.1	79.1
Co-SLAM [80]	Depth L1 [cm] ↓	1.05	0.85	2.37	1.24	1.48	1.86	1.66	1.54	1.51
GO-SLAM [98]	Depth L1 [cm] ↓	-	-	-	-	-	-	-	-	3.38
	*Depth L1 [cm] ↓	4.56	1.97	3.43	2.47	3.03	10.3	7.31	4.34	4.68
	F1 [%] ↑	17.3	33.4	24.0	43.0	31.8	21.8	17.3	22.0	26.3
Point-SLAM [61]	Depth L1 [cm] ↓	0.53	0.22	0.46	0.30	0.57	**0.49**	0.51	0.46	0.44
	F1 [%] ↑	86.9	92.3	90.8	93.8	**91.6**	**89.0**	88.2	85.6	89.8
Loopy-SLAM [39]	Depth L1 [cm] ↓	**0.30**	**0.20**	**0.42**	**0.23**	**0.46**	0.60	**0.37**	**0.24**	**0.35**
	F1 [%] ↑	**91.6**	**92.4**	90.6	**93.9**	**91.6**	88.5	**89.0**	**88.7**	**90.8**
3D Gaussian Splatting										
SplaTAM [35]	Depth L1 [cm]↓	0.43	0.38	0.54	0.44	0.66	1.05	1.60	0.68	0.72
	F1 [%]↑	89.3	88.2	88.0	91.7	90.0	85.1	77.1	80.1	86.1
Gaussian SLAM [92]	Depth L1 [cm]↓	0.61	0.25	0.54	0.50	0.52	0.98	1.63	0.42	0.68
	F1 [%]↑	88.8	91.4	90.5	91.7	90.1	87.3	84.2	87.4	88.9
LoopSplat (Ours)	Depth L1 [cm] ↓	0.39	0.23	0.52	0.32	0.51	0.63	1.09	0.40	0.51
	F1 [%] ↑	90.6	91.9	**91.1**	93.3	90.4	88.9	88.7	88.3	90.4

Table 5. **Reconstruction Performance on Replica [67].** LoopSplat obtains the second-best F1-score, falling behind only to Loopy-SLAM. It is noteworthy that both the NeRF-based Loopy-SLAM and Point-SLAM methods require ground truth depth input to guide the depth rendering, whereas our method, leveraging 3DGS, only requires estimated camera poses at rendering time.

Dataset	***Replica* [67]**			***TUM* [71]**			***ScanNet* [17]**		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NICE-SLAM [100]	24.42	0.892	0.233	14.86	0.614	0.441	17.54	0.621	0.548
Vox-Fusion [87]	24.41	0.801	0.236	16.46	0.677	0.471	18.17	0.673	0.504
ESLAM [32]	28.06	0.923	0.245	15.26	0.478	0.569	15.29	0.658	0.488
Point-SLAM [61]	35.17	0.975	0.124	16.62	0.696	0.526	19.82	0.751	0.514
Loopy-SLAM [39]	35.47	0.981	0.109	12.94	0.489	0.645	15.23	0.629	0.671
SplaTAM [35]	34.11	0.970	**0.100**	**22.80**	**0.893**	**0.178**	19.14	0.716	**0.358**
Gaussian-SLAM [92]	42.08	0.996	0.018	25.05	0.929	0.168	27.67	0.923	0.248
LoopSplat (Ours)	**36.63**	**0.985**	0.112	22.72	0.873	0.259	**24.92**	**0.845**	0.425

Table 6. **Rendering Performance on 3 Datasets**. LoopSplat achieves competitive results on synthetic and real-world datasets. Gray indicates evaluation on submaps instead of a global map.

LoopSplat surpasses all competing methods in terms of PSNR and LPIPS on Replica and ScanNet, and is competitive with SplaTAM on TUM-RGBD. Note the significant margin over baselines that employ implicit neural representations. We report the per-scene rendering results in Supp.

4.4. Memory and Runtime Analysis

Tab. 7 profiles the runtime and memory usage of LoopSplat. While our per-frame tracking and map optimization time falls behind the fastest baselines, our Gaussian Splatting-based registration significantly shortens the loop edge registration time compared to Loopy-SLAM. Through careful control of submap growth, our Gaussian splats embedding size is 8× smaller than that of the 3DGS baseline SplaTAM. Additionally, we require the least GPU memory to process a room-sized scene. In contrast, baselines like ESLAM, GO-SLAM or SplaTAM require >15 GB of GPU memory.

[1] *Depth L1 for GO-SLAM is based on results reproduced by [39] using random poses, as GO-SLAM originally evaluates on ground truth poses.

[2] Gaussian-SLAM evaluates rendering on local submaps.

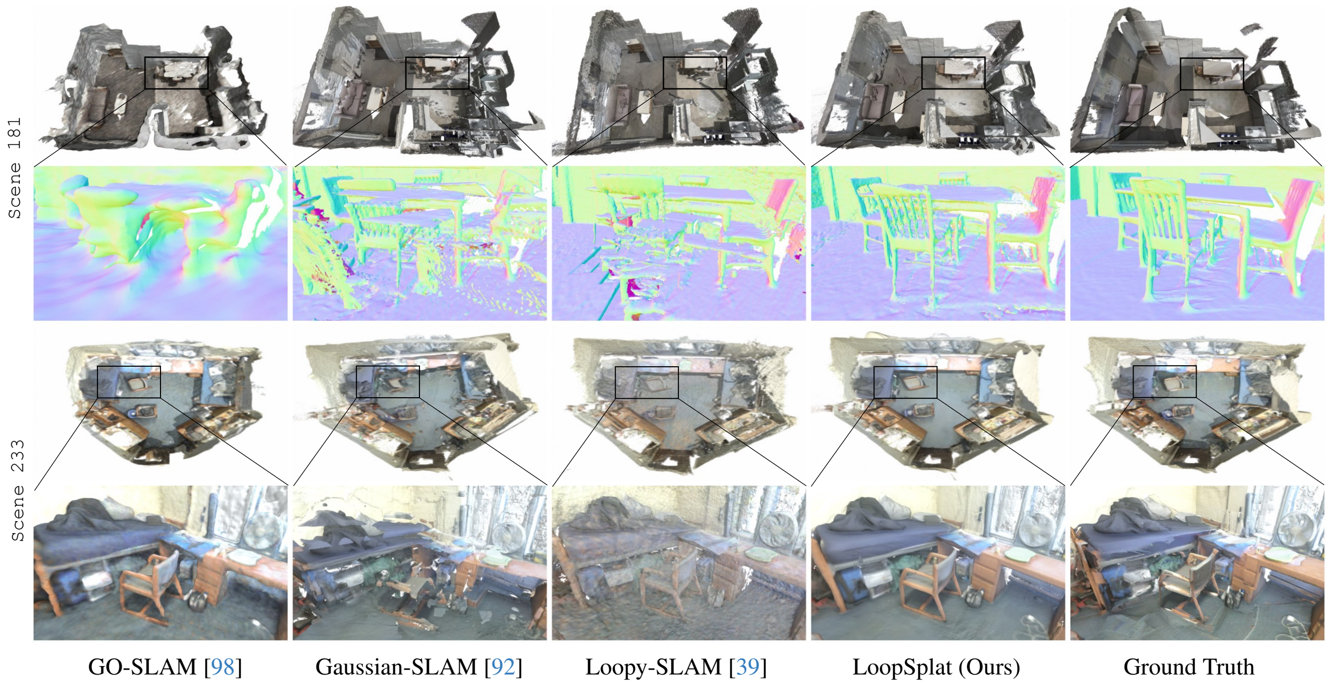


Figure 4. **Comparison of Mesh Reconstruction on two ScanNet [17] scenes**. For the first scene, we highlight shape details with normal shading, showing that LoopSplat yields the best geometry (*e.g.* the chairs). For the second one, we display renderings of the colored mesh. Note the distortions at the desk in ground truth that are not present in ours, indicating accuracy limitations of ScanNet ground truth poses.

Method	Tracking /Frame(s) ↓	Mapping /Frame(s) ↓	Registration /Edge(s) ↓	Embedding Size(MiB)↓	Peak GPU Use(GiB)↓
NICE-SLAM [100]	1.06	1.15	-	95.9	12.0
Vox-Fusion [87]	1.92	1.47	-	**0.15**	17.6
Point-SLAM [61]	1.11	3.52	-	27.2	7.7
ESLAM [42]	0.15	0.62	-	45.5	17.3
GO-SLAM [98]	**0.125**		-	48.1	18.4
SplaTAM [35]	2.70	4.89	-	404.5	18.5
Loopy-SLAM[39]	1.11	3.52	12.0	60.9	9.3
LoopSplat (Ours)	0.83	0.93	**1.36**	49.7	**7.0**

Table 7. **Runtime and Memory Usage on Replica** `office 0`. Per-frame runtime is calculated as the total optimization time divided by the sequence length, profiled on a RTX A6000 GPU. The embedding size is the total memory of the map representation. Note that implicit field-based methods require additional space for their decoders. We take runtime values from [92] and embedding values from [39] for the baselines.

4.5. Ablations

We first demonstrate that straightforward point cloud registration is not suitable to derive loop edge constraints from 3DGS. To illustrate this, we replace the proposed 3DGS registration in our SLAM system with FPFH+ICP [99] and evaluate the trajectory error (ATE) on Replica. As shown in the last row of Tab. 8, FPFH+ICP applied directly to the center points of 3D Gaussians leads to less accurate loop edges compared to our method and deteriorates loop closure. We hypothesize that this is because the center points do not accurately represent the scene surfaces, as previously discussed in [28, 91, 94]. Furthermore, the pre-processing of [99] involves re-rendering and back-projecting 3DGS to

Mul. Opt.	*Ove. Est.*	*Rot. Ave.*	ATE (cm)	Runtime (s)
✗	✗	✗	0.31	-
✗	✓	✗	0.31	**1.25**
✓	✓	✗	0.27	1.36
✓	✗	✓	0.37	11.02
✓	✓	✓	**0.26**	1.36
FPFH+ICP [99]			0.40	12.0

Table 8. **Ablation Study on 3DGS Registration**. The numbers are computed based on average performance of 8 scenes on Replica [68]. *Mul. Opt.* denotes multi-view optimization, *Ove. Est.* and *Rot. Ave.* denote view selection and rotation averaging.

obtain 3D points, downsampling the point clouds and voxelizing them. This heavy pre-processing makes [39] more than 8× slower than our method. In contrast, LoopSplat efficiently reuses the native map representation without any pre-processing, answering the research question we asked in Sec. 1. We also explore the impact of different modules in our registration method. The ablation study confirms that every component contributes to the final performance: Multi-view optimization and rotation averaging greatly improve registration accuracy by fusing information from different viewpoints. View selection via overlap estimation (Sec. 3.2) is crucial to identifying informative viewpoints and ensuring the efficiency of the SLAM system.

5. Conclusion

We presented LoopSplat, a dense RGB-D SLAM system that exclusively uses 3D Gaussian splats as scene representation, achieving global consistency through loop closure.

References

[1] Sheng Ao, Qingyong Hu, Hanyun Wang, Kai Xu, and Yulan Guo. Buffer: Balancing accuracy, efficiency, and generalizability in point cloud registration. In *CVPR*, 2023. 2

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016. 4, 5

[3] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *CVPR*, 2020. 2

[4] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, 1992. 2

[5] Michael Bosse, Paul Newman, John Leonard, Martin Soika, Wendelin Feiten, and Seth Teller. An atlas framework for scalable mapping. In *ICRA*, 2003. 2

[6] Romain Brégier. Deep regression on manifolds: a 3D rotation case study. 2021. 5

[7] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 2021. 5, 6

[8] Yan-Pei Cao, Leif Kobbelt, and Shi-Min Hu. Real-time high-accuracy three-dimensional reconstruction with consumer rgb-d cameras. *ACM TOG*, 2018. 1, 2, 6

[9] Jiahao Chang, Yinglin Xu, Yihao Li, Yuantao Chen, and Xiaoguang Han. Gaussreg: Fast 3d registration with gaussian splatting. In *ECCV*, 2024. 3

[10] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. *ACM TOG*, 2013. 2

[11] Yu Chen and Gim Hee Lee. Dreg-nerf: Deep registration for neural radiance fields. In *ICCV*, 2023. 3

[12] Hae Min Cho, HyungGi Jo, and Euntai Kim. Sp-slam: Surfel-point simultaneous localization and mapping. *IEEE/ASME Transactions on Mechatronics*, 2021. 2

[13] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 2, 3, 5

[14] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *ICCV*, 2019. 2

[15] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H Hsu. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. *arXiv preprint arXiv:2209.13274*, 2022. 1, 2

[16] Brian Curless and Marc Levoy. Volumetric method for building complex models from range images. In *SIGGRAPH Conference on Computer Graphics*, 1996. 2

[17] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 1, 5, 6, 7, 8

[18] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM TOG*, 2017. 1, 2, 3, 5, 6

[19] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *ICRA*, 2012. 2

[20] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014. 2

[21] Francis Engelmann, Fabian Manhardt, Michael Niemeyer, Keisuke Tateno, Marc Pollefeys, and Federico Tombari. Opennerf: Open set 3d neural scene segmentation with pixel-wise features and rendered novel views. *arXiv preprint arXiv:2404.03650*, 2024. 2

[22] Nicola Fioraio, Jonathan Taylor, Andrew Fitzgibbon, Luigi Di Stefano, and Shahram Izadi. Large-scale and drift-free surface reconstruction using online subvolume registration. In *CVPR*, 2015. 2

[23] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *CVPR*, 2019. 2

[24] Lily Goli, Daniel Rebain, Sara Sabour, Animesh Garg, and Andrea Tagliasacchi. nerf2nerf: Pairwise registration of neural radiance fields. In *ICRA*, 2023. 3

[25] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 2012. 2

[26] Peter Henry, Dieter Fox, Achintya Bhowmik, and Rajiv Mongia. Patch volumes: Segmentation-based consistent mapping with rgb-d cameras. In *3DV*, 2013. 2

[27] Jiarui Hu, Mao Mao, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. CP-SLAM: Collaborative neural point-based SLAM system. In *NeurIPS*, 2023. 2

[28] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024. 8

[29] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular stereo and rgb-d cameras. In *CVPR*, 2024. 1, 2, 5, 6, 7

[30] Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *CVPR*, 2021. 6

[31] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *CVPR*, 2021. 2, 4, 5

[32] M. M. Johari, C. Carta, and F. Fleuret. ESLAM: Efficient dense slam system based on hybrid representation of signed distance fields. In *CVPR*, 2023. 1, 5, 6, 7

[33] Olaf Kähler, Victor Adrian Prisacariu, Carl Yuheng Ren, Xin Sun, Philip H. S. Torr, and David William Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Trans. Vis. Comput. Graph.*, 2015. 2

[34] Olaf Kähler, Victor A Prisacariu, and David W Murray. Real-time large-scale dense 3d reconstruction with loop closure. In *ECCV*, 2016. 2

[35] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. *CVPR*, 2024. 1, 2, 5, 6, 7, 8

[36] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *International Conference on 3D Vision (3DV)*, 2013. 2

[37] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023. 1, 3, 4

[38] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *IROS*, 2013. 2

[39] Lorenzo Liso, Erik Sandström, Vladimir Yugay, Luc Van Gool, and Martin R. Oswald. Loopy-slam: Dense neural slam with loop closures. In *CVPR*, 2024. 1, 2, 3, 5, 6, 7, 8

[40] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 2

[41] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 1987. 6

[42] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. pages arXiv–2211, 2022. 1, 8

[43] Robert Maier, Jürgen Sturm, and Daniel Cremers. Submap-based bundle adjustment for 3d reconstruction from rgb-d data. In *Pattern Recognition: 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings 36*, 2014. 2, 3

[44] R Maier, R Schaller, and D Cremers. Efficient online surface correction for real-time large-scale 3d reconstruction. arxiv 2017. *arXiv preprint arXiv:1709.03763*, 2017. 2

[45] Yunxuan Mao, Xuan Yu, Kai Wang, Yue Wang, Rong Xiong, and Yiyi Liao. Ngel-slam: Neural implicit representation-based global consistent low-latency slam system. *arXiv preprint arXiv:2311.09525*, 2023. 1, 2

[46] Nico Marniok, Ole Johannsen, and Bastian Goldluecke. An efficient octree design for local variational range image fusion. In *German Conference on Pattern Recognition (GCPR)*, 2017. 2

[47] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *arXiv preprint arXiv:2312.06741*, 2023. 1, 2, 4, 5, 6

[48] Hidenobu Matsuki, Keisuke Tateno, Michael Niemeyer, and Federic Tombari. Newton: Neural view-centric mapping for on-the-fly large-scale slam. *arXiv preprint arXiv:2303.13654*, 2023. 1, 2

[49] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. *CVPR*, 2024. 4

[50] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2021. 2

[51] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 2017. 2, 6

[52] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 1, 2

[53] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM TOG*, 2013. 1, 2

[54] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan I. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board MAV planning. In *IROS*, 2017. 2

[55] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss. PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency. *IEEE Transactions on Robotics (TRO)*, 2024. 2

[56] Valentin Peretroukhin, Matthew Giamou, David M Rosen, W Nicholas Greene, Nicholas Roy, and Jonathan Kelly. A smooth representation of belief over so (3) for deep rotation learning with uncertainty. *arXiv preprint arXiv:2006.01031*, 2020. 5

[57] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, Slobodan Ilic, Dewen Hu, and Kai Xu. Geotransformer: Fast and robust point cloud registration with geometric transformer. *IEEE TPAMI*, 2023. 2

[58] Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. *IEEE Robotics and Automation Letters*, 2019. 2

[59] Antoni Rosinol, John J. Leonard, and Luca Carlone. NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields. *arXiv*, 2022. 1

[60] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3d registration. In *ICRA*, 2009. 2

[61] Erik Sandström, Yue Li, Luc Van Gool, and Martin R Oswald. Point-slam: Dense neural point cloud-based slam. In *ICCV*, 2023. 1, 2, 5, 6, 7, 8

[62] Erik Sandström, Kevin Ta, Luc Van Gool, and Martin R Oswald. Uncle-slam: Uncertainty learning for dense neural slam. In *Int. Conf. Comput. Vis. Worksh.*, 2023. 1

[63] Erik Sandström, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Luc Van Gool, Martin R Oswald, and Federico Tombari. Splat-slam: Globally optimized rgb-only slam with 3d gaussians. *arXiv preprint arXiv:2405.16544*, 2024. 2

[64] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 6

[65] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *CVPR*, 2019. 1, 2, 6

[66] Frank Steinbrucker, Christian Kerl, and Daniel Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *ICCV*, 2013. 2

[67] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 5, 7

[68] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 5, 8

[69] Jörg Stückler and Sven Behnke. Multi-resolution surfel maps for efficient dense 3d modeling and tracking. *Journal of Visual Communication and Image Representation*, 2014. 2

[70] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012. 6

[71] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 2012. 6, 7

[72] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *ICCV*, 2021. 1, 5

[73] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *ICCV*, 2021. 2

[74] Yijie Tang, Jiazhao Zhang, Zhinan Yu, He Wang, and Kai Xu. Mips-fusion: Multi-implicit-submaps for scalable and robust online neural rgb-d reconstruction. *ACM TOG*, 2023. 5, 6

[75] Yijie Tang, Jiazhao Zhang, Zhinan Yu, He Wang, and Kai Xu. Mips-fusion: Multi-implicit-submaps for scalable and robust online neural rgb-d reconstruction. *arXiv preprint arXiv:2308.08741*, 2023. 1, 2

[76] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *NeurIPS*, 2021. 2

[77] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *ECCV*, 2010. 2

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[79] Hao Wang, Jun Wang, and Wang Liang. Online reconstruction of indoor scenes from rgb-d streams. In *CVPR*, 2016. 2

[80] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. In *CVPR*, 2023. 1, 6, 7

[81] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 2004. 4, 6

[82] Zian Wang, Tianchang Shen, Jun Gao, Shengyu Huang, Jacob Munkberg, Jon Hasselgren, Zan Gojcic, Wenzheng Chen, and Sanja Fidler. Neural fields meet explicit geometric representations for inverse rendering of urban scenes. In *CVPR*, 2023. 2

[83] Thomas Whelan, John McDonald, Michael Kaess, Maurice Fallon, Hordur Johannsson, and John J. Leonard. Kintinuous: Spatially extended kinectfusion. In *Proceedings of RSS '12 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012. 6

[84] Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems (RSS)*, 2015. 1, 2, 6

[85] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*, 2024. 1, 2

[86] Zhixin Yan, Mao Ye, and Liu Ren. Dense visual slam with probabilistic surfel map. *IEEE TVCG*, 2017. 2, 6

[87] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2022. 1, 2, 5, 6, 7, 8

[88] Xingrui Yang, Yuhang Ming, Zhaopeng Cui, and Andrew Calway. Fd-slam: 3-d reconstruction using features and dense matching. In *2022 International Conference on Robotics and Automation (ICRA)*, 2022. 2

[89] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *(IROS)*, 2021. 3

[90] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *ICCV*, 2023. 6

[91] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient high-quality compact surface reconstruction in unbounded scenes. *arXiv:2404.10772*, 2024. 8

[92] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023. 1, 2, 3, 5, 6, 7, 8

[93] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017. 2

[94] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024. 8

[95] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R Oswald. Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam. *arXiv preprint arXiv:2403.19549*, 2024. 2

[96] Heng Zhang, Guodong Chen, Zheng Wang, Zhenhua Wang, and Lining Sun. Dense 3d mapping for indoor environment based on feature-point slam method. In *2020 the 4th International Conference on Innovation in Artificial Intelligence*, 2020. 2

[97] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

[98] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *ICCV*, 2023. 1, 2, 5, 6, 7, 8

[99] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *ECCV*, 2016. 8

[100] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *CVPR*, 2022. 1, 5, 6, 7, 8