GO-SLAM: Global Optimization for Consistent 3D Instant Reconstruction

GO-SLAM: Global Optimization for Consistent 3D Instant Reconstruction

Youmin Zhang Fabio Tosi Stefano Mattoccia Matteo Poggi
Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
{youmin.zhang2, fabio.tosi5, stefano.mattoccia, m.poggi}@unibo.it
https://youmi-zym.github.io/projects/GO-SLAM/

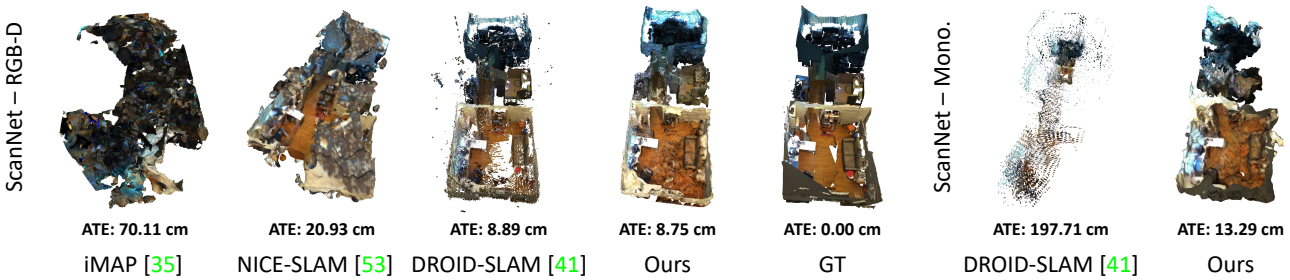

Figure 1: **3D Reconstruction and trajectory error on *scene0054_00* (ScanNet [12]).** From left to right: RGB-D methods (iMAP [35], NICE-SLAM [53], DROID-SLAM [41] and ours), ground truth scan, and monocular methods (DROID-SLAM [41] and ours).

Abstract

Neural implicit representations have recently demonstrated compelling results on dense Simultaneous Localization And Mapping (SLAM) but suffer from the accumulation of errors in camera tracking and distortion in the reconstruction. Purposely, we present GO-SLAM, a deep-learning-based dense visual SLAM framework globally optimizing poses and 3D reconstruction in real-time. Robust pose estimation is at its core, supported by efficient loop closing and online full bundle adjustment, which optimize per frame by utilizing the learned global geometry of the complete history of input frames. Simultaneously, we update the implicit and continuous surface representation on-the-fly to ensure global consistency of 3D reconstruction. Results on various synthetic and real-world datasets demonstrate that GO-SLAM outperforms state-of-the-art approaches at tracking robustness and reconstruction accuracy. Furthermore, GO-SLAM is versatile and can run with monocular, stereo, and RGB-D input.

1. Introduction

The demand for high-fidelity 3D object and scene reconstructions grows continuously in various fields, including robotics and augmented/virtual reality applications. Thus, faithfully representing objects and scenes in three-dimensional space is paramount to modelling them as continuous surfaces rather than discrete points. However, despite the significant advancements in 3D reconstruction techniques, obtaining high-quality representations in real-time without compromising accuracy and spatial resolution remains challenging. This fact is further demanding in online reconstruction scenarios, where handling camera motions and achieving real-time performance are critical.

Dense visual Simultaneous Localization and Mapping (SLAM) systems [28, 45, 32, 13, 46] have been introduced recently, enabling real-time, dense indoor scene reconstructions using RGB-D sensors. In particular, BundleFusion [13] is the first volumetric approach that focuses on globally consistent 3D reconstruction on large-scale scenes at a real-time rate. However, consumer depth sensors have a limited working range [32, 12] and could yield extremely noisy [27] measurements. This issues make the representation mapped by RGB-D SLAM suffer from blurring or over-smoothed geometric details, degrading the accuracy of pose estimation and reconstruction. In parallel, scene reconstruction from monocular imagery is emerging as a more convenient solution compared to RGB-D or LiDAR sensors. Camera sensors are lightweight, inexpensive, and represent the most straightforward configuration. Several deep-learning approaches [38, 4, 11, 51, 39, 32, 41] have advanced monocular 3D reconstruction. However, their surface representations – point cloud, surfel-based and volu-

metric representations – lack flexibility at shape extraction and thus inhibit high-fidelity reconstruction.

More recently, the advent of Neural Radiance Fields (NeRFs) also impacted dense visual SLAM, offering photometrically accurate 3D representations of the world. The implicit representation yielded by continuous radiance fields allows for high-quality rendering of both visible and occluded regions, enabling extraction of the underlying shapes at arbitrary resolution. Recent [35, 19, 53] and concurrent [20, 31, 9, 52] works demonstrate that NeRF-based visual SLAM can yield precise 3D reconstructions and camera pose estimation in small-scale scenes. However, due to the lack of global online optimization, such as loop closure (LC) and global bundle adjustment (BA), camera drift error accumulates as the number of processed frames grows, and the 3D reconstruction quickly collapses, as shown in Fig. 1.

Purposely, this paper introduces GO-SLAM, a deep-learning-based SLAM system featuring on-the-fly, globally consistent 3D reconstruction, facilitated by our robust camera tracking and real-time implicit surface updates. As real-world 3D scenes may be very complex, drifting cannot be completely avoided by only locally tracking camera motion, especially in the monocular camera setting, due to the lack of explicit depth measurements. In addition to the local registration commonly performed by current SLAM systems, we present an efficient loop closing to correct trajectory in real-time, accompanied by an online full BA module to actively optimize the 3D geometry of the complete keyframes history. In contrast to previous works performing BA or LC with sparse visual features [26, 6, 13, 11], our end-to-end global optimization procedure is naturally robust to challenging regions, thanks to richer features and geometry cues (*e.g.*, pixel-wise flow) learned by neural networks. Furthermore, GO-SLAM implements instant mapping based on a neural implicit network with multi-resolution hash encoding [24]. Its compact, multiscale representation enables updating the 3D reconstruction at high-frequency according to newly-optimized camera poses and depths from our global optimization system, thus ensuring global consistency in the dense map and capturing local details. Our contributions can be resumed as follows:

- A novel deep-learning-based, real-time global pose optimization system that considers the complete history of input frames and continuously aligns all poses.
- An efficient alignment strategy that enables instantaneous loop closures and correction of global structure, being both memory and time efficient.
- An instant 3D implicit reconstruction approach, enabling on-the-fly and continuous 3D model update with the latest global pose estimates. This strategy facilitates real-time 3D reconstructions.
- The first deep-learning architecture for joint robust pose estimation and dense 3D reconstruction suited for any setup: monocular, stereo, or RGB-D cameras.

2. Related Work

Here, we review the literature relevant to our work.

Online 3D Reconstruction and SLAM. Real-time, dense, and globally consistent 3D reconstruction is crucial in the SLAM literature. As a core element for high-quality reconstruction, the underlying representation can be approximately categorized as depth point [25, 26, 6, 51, 15, 4, 38, 11, 41], height map [17], surfel [32, 45] and volumetric representation [28, 29, 13]. Especially some of them [26, 6, 13, 32, 11], make an effort for globally consistent reconstruction by implementing global BA and LC systems. However, due to discrete and limited surface representations (*e.g.*, point-, surfel- or voxel-based), these methods suffer from the accumulation of errors in camera tracking and distortion in the reconstruction. DROID-SLAM [41] achieves impressive trajectory estimations by using neural networks to leverage richer context from images, yet it performs global bundle adjustment only offline, at the end of camera tracking. However, for some challenging cases, it gets hard to eliminate the drift error with offline refinement solely, as shown in Fig. 1, pointing out the importance of online BA for on-the-fly drift correction.

NeRF-based Visual SLAM. Neural implicit fields (NeRF) have recently emerged as one of the promising and widely applicable methods for 3D representation, opening up many new research opportunities including novel view synthesis [23, 44, 43, 2, 3, 49, 36, 16, 24], multi-view 3D reconstruction [8, 7, 37], and large-scale scene reconstruction [42, 54, 21, 50, 30, 1, 48]. A common requirement of these methods is the posed images. [22, 47] tries to relax this constraint starting from imperfect camera poses on small objects. More recently, using neural implicit representation in visual SLAM [35, 53, 52, 31, 9, 20] has achieved better scene completeness, especially for unobserved regions, and it has also allowed for continuous 3D modeling at arbitrary resolution. Two pioneer works, iMAP [35] and NICE-SLAM [53], extend the neural implicit representation to RGB-D SLAM system, which learns camera pose tracking and room-scale mapping from scratch. Concurrent works [20, 52], by removing the reliance on depth sensor input, achieve visual SLAM when only RGB sequences are available. Instead of naïve pose optimization with NeRF, Orbeez-SLAM [9] resorts to visual odometry from ORB-SLAM2 [26] for accurate pose estimation. However, the lack of many of the core capabilities of modern SLAM systems, such as LC and global BA, inhibits the ability of these methods to perform large-scale reconstructions. Concurrent to our work, NeRF-SLAM [31] integrates DROID-SLAM [41] for camera tracking. How-

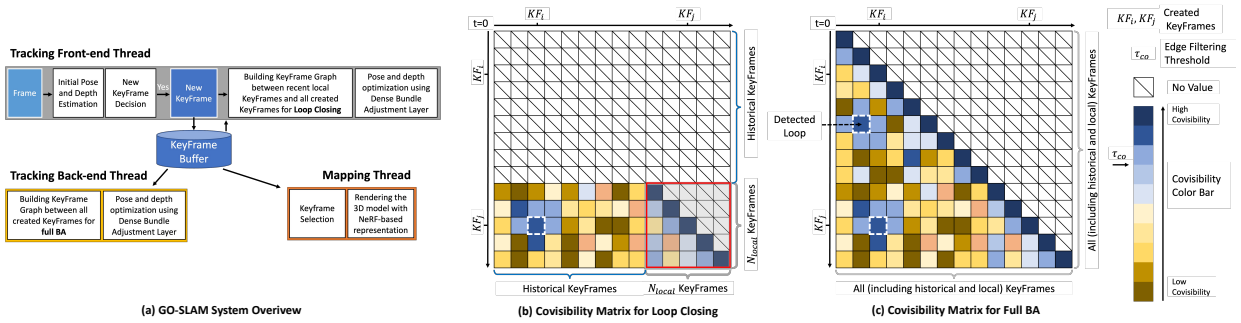

Figure 2: **Architecture Overview.** Our GO-SLAM framework consists of three parallel threads (a): front-end tracking (including keyframe initialization and loop closing), back-end tracking, and instant mapping. The front-end tracking thread uses the video stream as input and iteratively updates the pose and depth of the current frame while determining whether it should be promoted as a new keyframe. Moreover, it also actively performs efficient loop closing (b). The back-end tracking thread focuses on generating globally consistent pose and depth predictions through full bundle adjustment (c). Simultaneously, instant mapping updates the 3D reconstruction on-the-fly according to the latest geometry changes.

ever, it also inherits its limitations, *i.e.*, the absence of online loop closing and full BA, which restricts its ability to perform globally consistent 3D reconstruction.

3. Method

Our GO-SLAM framework, depicted in Fig. 2, uses a keyframe-based SLAM paradigm to achieve real-time, globally-consistent 3D reconstruction. This is made possible by the online drift-corrected pose tracking and instant mapping capabilities of our system. By performing full bundle adjustment and loop closing, pose optimization can be carried out globally. Meanwhile, instant mapping adapts to continuous changes in optimized global poses and depths.

3.1. Tracking with Global Optimization

Loop closing and global bundle adjustment are crucial for robust pose estimation and long-term map consistency. In this work, we extend the tracking component of DROID-SLAM [41] by several key features. These enhancements effectively reduce the drift and pave the way for a globally optimal map. In the front-end tracking, we initialize a keyframe if sufficient motion is observed – there, we also introduce LC – while the back-end is equipped with full BA for online global refinement. Fig. 3 shows the effect of both LC and BA on tracking and 3D reconstruction qualitatively, correcting the large errors occurring in their absence.

Front-End Tracking. Our system takes as input a live video stream, which can be either monocular, stereo, or RGB-D, and applies a recurrent update operator based on RAFT [40] to compute the optical flow of each new frame compared to the last keyframe. If the average flow is larger than a predefined threshold τ_{flow}, a new keyframe is created out of the current frame and added to the maintained keyframe buffer for further refinement.

We use the set of keyframes $\{\mathbf{KF}_k\}_{k=1}^{N_{KF}}$ created so far to build a keyframe-graph $(\mathcal{V}, \mathcal{E})$ for performing LC. This process involves two steps: (1) select high co-visibility connections between the most recent N_{local} keyframes, and (2) detect loop closures between local keyframes and historical keyframes outside the local window. Accordingly, we compute a co-visibility matrix of size $N_{local} \times N_{KF}$ between N_{local} local keyframes and all the N_{KF} created keyframes to find valuable edge connections, as shown in Fig. 2 (b). In practice, the co-visibility is represented by the mean rigid flow between keyframe pairs using efficient back-projection, and those with low co-visibility, *i.e.*, mean flow larger than τ_{co}, are filtered out. Among local keyframes – the red-borders sub-matrix in Fig. 2 (b) – we build edges for keyframe pairs temporally adjacent or with high co-visibility. To avoid redundancy, once an edge connection (*e.g.*, $\mathrm{KF}_i \leftrightarrow \mathrm{KF}_j$) is added to the keyframe-graph, we suppress all possible neighboring edges between $\{\mathbf{KF}_k\}_{k=i-r_{local}}^{i+r_{local}}$ and $\{\mathbf{KF}_k\}_{k=j-r_{local}}^{j+r_{local}}$, where r_{local} is a hyper-parameter denoting a temporal radius. The loop detection step is quite similar. We sample edges from the unexplored part of the co-visibility matrix in descending order of co-visibility and suppress neighboring edges with radius r_{loop}. More strictly, to accept a loop candidate, we detect consecutively three loop candidates and validate them if their mean flow is lower than τ_{co}. The value of r_{loop} is determined empirically based on the observation that keyframes within a local window observe almost the same scene. We set r_{loop} to $\frac{N_{local}}{2}$, which allows for only one loop closure between the recent local region and one revisited region. In addition to the edge connections within local keyframes, several extra loop edges can be added to the keyframe graph depending on how many times the current local region is revisited. In general, the number of edges in the graph is linear to N_{local} with an upper-bound $N_{local} \times N_{local}$ + few loop closures. Through neighbor-

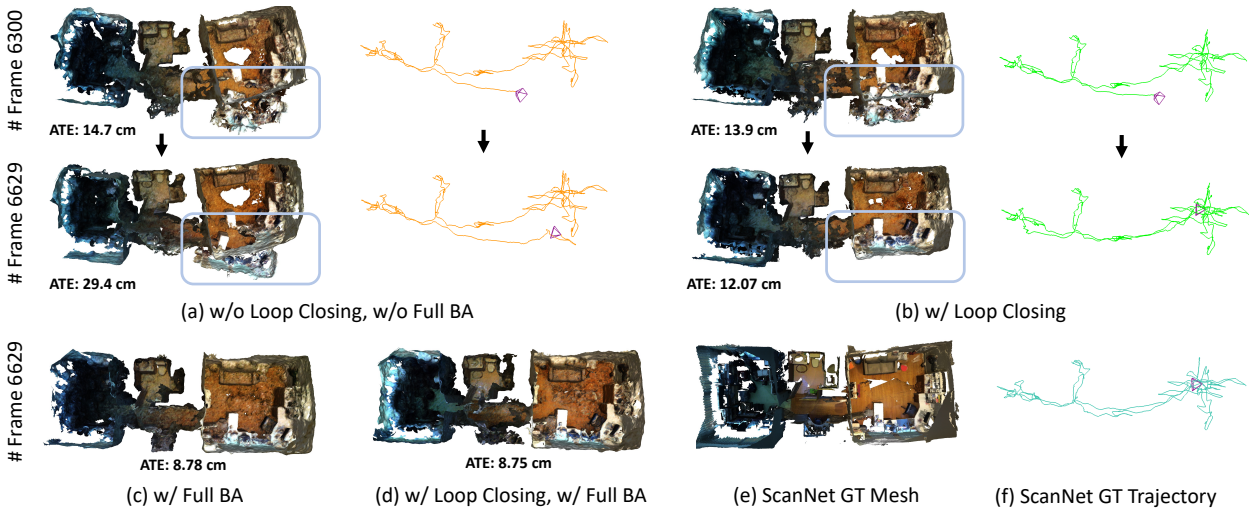

Figure 3: **Qualitatives examples of LC and full BA on *scene0054_00* (ScanNet [12]) with a total of 6629 frames.** In (a), a significant error accumulates when no global optimization is available. With loop closing (b), the system is able to eliminate the trajectory error using global geometry. Additionally, online full BA optimizes (c) the poses of all existing keyframes. The final model (d), which integrates both loop closing and full BA, achieves a more complete and accurate 3D model prediction.

hood suppression and co-visibility filtering, we further limit the number of edges in the keyframe-graph to $s_{edge} \cdot N_{local}$, so that the efficiency of optimization of the entire keyframe graph, *i.e.*, the whole front-end tracking, can be further ensured.

Afterward, we use the differentiable Dense Bundle Adjustment (DBA) layer proposed in [41] to solve a non-linear least squares optimization problem over the cost function in order to correct the camera pose $\mathbf{G} \in SE(3)$ and inverse depth $\mathbf{d} \in \mathbb{R}_{+}^{H \times W}$ of each keyframe in the keyframe-graph:

$$\mathbf{E}(\mathbf{G}, \mathbf{d}) = \sum_{(i,j) \in \mathcal{E}} \left\| \mathbf{p}_{ij}^{*} - \Pi_c(\mathbf{G}_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}_i)) \right\|_{\Sigma_{ij}}^{2}, \tag{1}$$

where $(i, j) \in \mathcal{E}$ denotes any edge in keyframe-graph, Π_c and Π_c^{-1} are the projection and back-projection functions, $\mathbf{p}_i$ is the back-projected pixel position from keyframe $\mathbf{KF}_i$, $\mathbf{G}_{ij}$ is the pose transformation from $\mathbf{KF}_i$ to $\mathbf{KF}_j$, $\mathbf{p}_{ij}^{*}$ and $\mathbf{w}_{ij}$ are estimated flow and associated confidence map, $\Sigma_{ij} = \operatorname{diag} \mathbf{w}_{ij}$ and $\|\cdot\|_{\Sigma}$ is the Mahalanobis distance which weights the error terms based on the confidence weights $\mathbf{w}_{ij}$. The cost function allows the update of camera poses and dense per-pixel depth to maximize their compatibility with flow $\mathbf{p}_{ij}^{*}$ predicted by the recurrent update operator. For the sake of efficiency, we only compute the Jacobians with respect to the depths and poses of the local keyframes. After computing residuals and Jacobians at each iteration, a damped Gauss-Newton algorithm is applied to find the optimal poses and depths of all local keyframes.

Back-End Tracking. Simultaneously optimizing overall historical keyframes can be computationally expensive, as observed in [26, 6, 41]. To address this issue, we follow a similar approach as previous SLAM algorithms [26, 6] by running the full BA online in a separate thread, allowing the system to continue tracking new frames and loop closing. Similarly to the proposed front-end tracking, we start a new keyframe-graph and insert keyframe pairs with high co-visibility, as well as temporal adjacent keyframes, as shown in Fig. 2 (c). When a new edge is built, we suppress the redundant neighboring edges with radius r_{global}. As the trajectory error of the latest keyframes has been corrected with global geometry featured by loop closing, it eases the real-time requirement for the full BA. Our proposed full BA is efficient up to tens of thousands of input frames, as shown in Fig. 4.

3.2. Instant Mapping

The proposed instant mapping aims at updating the global 3D reconstruction in real-time by incorporating newly-optimized geometry from tracking. However, this goal presents two challenges for the mapping thread: i) ensuring that the updated reconstruction remains globally consistent, and ii) enabling fast rendering of the reconstructed scene. Updating all existing keyframes at once is the simplest approach to ensure global consistency, but it can quickly become impractical as the number of keyframes increases. Therefore, it is crucial to prune the keyframe candidates for updating selectively. Additionally, high-speed rendering of the scene is necessary to meet real-time requirements. To achieve these goals, we introduce our novel

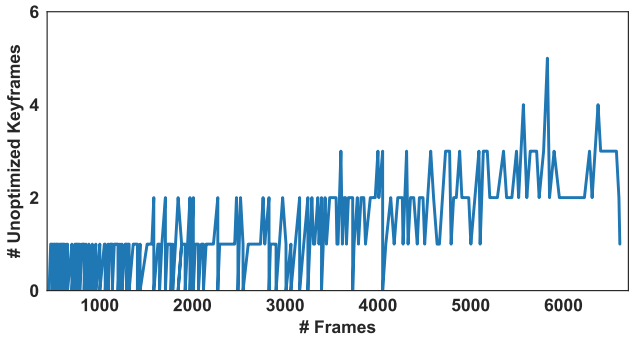

Figure 4: **Number of unoptimized keyframes at each timestamp with full BA.** Experiment on *scene0054_00* (ScanNet [12]) with 6629 frames (357 total keyframes).

keyframe selection strategy and discuss our use of spatial hashing [24] and rendering networks in the remainder.

Keyframe Selection. At the beginning of each update of the 3D reconstruction, the instant mapping thread first takes a snapshot of all existing keyframe poses and depths tracked, to ensure that the geometry remains consistent during the mapping period. For keyframe selection, we prioritize those with the most relevant optimization updates, following the principle established by previous works [13]. Firstly, we ensure that the latest two keyframes and those not optimized by mapping are always included. In addition, following [13], we sort all keyframes in descending order of pose difference between the current and last updated state and select the top 10 keyframes from the sorted list when accessing. Furthermore, to prevent the mapping from forgetting previous 3D geometry, we also select 10 keyframes using a stratified sampling [23] from all the available keyframes.

Rendering. Drawing inspiration from the recent advancements in implicit neural network techniques [23] – Instant-NGP [24] in particular – we can construct 3D models from scratch with remarkable speed and accuracy. Specifically, given depth $\mathbf{D}$ (converted from inverse depth $\mathbf{d}$), pose $\mathbf{G}$, and image $\mathbf{I}$ for each selected keyframe, we randomly select M pixels for training. We then generate $N_{ray} = N_{strat} + N_{imp}$ total sampling points along the emitted ray crossing each pixel, where N_{strat} points are sampled using stratified sampling and N_{imp} points are selected near the depth value following [53]. For each 3D sampling point $\mathbf{x}$, we map it to multi-resolution hash encodings [24] $h_{\Theta_{hash}}(\mathbf{x})$ with trainable encoding parameters Θ_{hash} at each entry of the hash table. As the hash encodings $h_{\Theta_{hash}}(\mathbf{x})$ have explicitly stored the geometric and intensity information for each spatial position, we can predict a signed distance function (SDF) $\Phi(\mathbf{x})$ and color $\Omega(\mathbf{x})$ using shallow networks.

More specifically, our SDF network $f_{\Theta_{sdf}}$, which consists of a single multi-layer perceptron (MLP) with learnable parameters Θ_{sdf}, takes the point position $\mathbf{x}$ and corresponding hash encodings $h_{\Theta_{hash}}(\mathbf{x})$ as input and predicts the SDF as:

$$\Phi(\mathbf{x}),\ \mathbf{g} = f_{\Theta_{sdf}}(\mathbf{x}, h_{\Theta_{hash}}(\mathbf{x})), \tag{2}$$

with $\mathbf{g}$ being the learned geometry feature vector. The color network $f_{\Theta_{color}}$ processes $\mathbf{g}$, $\mathbf{x}$ and the gradient of SDF $\mathbf{n}$ with respect to $\mathbf{x}$ to estimate color $\Omega(\mathbf{x})$ as:

$$\Omega(\mathbf{x}) = f_{\Theta_{color}}(\mathbf{x}, \mathbf{n}, \mathbf{g}). \tag{3}$$

where Θ_{color} is the set of learnable parameters of the color network, i.e., a two-layers MLP.

The depth and color of each pixel/ray are calculated by unbiased volume rendering following NeuS [43]. Specifically, for point $\mathbf{x}_i,\ i \in \{1, \cdots, N_{ray}\}$ along a ray, given the camera center $\mathbf{o}$, view direction $\mathbf{v}$, and sampled depth D_i^{ray}, it can be formulated as $\mathbf{x}_i = \mathbf{o} + D_i^{ray}\mathbf{v}$. At the same time, the unbiased ray termination probability at this point is modeled as $w_i = \alpha_i \Pi_{j=1}^{i-1}(1-\alpha_j)$, where the opacity value α_i is computed as:

$$\alpha_i = \max\left(\frac{\sigma(\Phi(\mathbf{x_i})) - \sigma(\Phi(\mathbf{x_{i+1}}))}{\sigma(\Phi(\mathbf{x_i}))}, 0\right). \tag{4}$$

where σ is the modulated Sigmoid function [43]. With the weight w, the predictions of pixel-wise color $\hat{\mathbf{c}}$ and depth $\hat{\mathbf{D}}$ are accumulated along the ray:

$$\hat{\mathbf{c}} = \sum_{i=1}^{N_{ray}} w_i \Omega(\mathbf{x}_i), \quad \hat{\mathbf{D}} = \sum_{i=1}^{N_{ray}} w_i D_i^{ray}. \tag{5}$$

Training Losses. To optimize the rendering networks, taking keyframe image $\mathbf{I}$ and depth $\mathbf{D}$ as ground truth, the RGB and depth losses $\mathcal{L}_c$ and $\mathcal{L}_{dep}$ imposed on the selected M pixels are $\mathcal{L}_c = \frac{1}{M}\sum_{m=1}^{M} |\mathbf{c}_m - \hat{\mathbf{c}}_m|,\ \mathbf{c}_m \in \mathbf{I}$ and

$$\mathcal{L}_{dep} = \frac{1}{M}\sum_{m=1}^{M} \frac{|\mathbf{D}_m - \hat{\mathbf{D}}_m|}{\sqrt{\hat{\mathbf{D}}_m^{var}}}, \tag{6}$$

respectively, with $\hat{\mathbf{D}}_m^{var} = \sum_{i=1}^{N_{ray}} w_i (\hat{\mathbf{D}}_m - D_{m,i}^{ray})^2$ being the predicted depth variance used for down-weighting uncertain regions in the reconstructed geometry [35, 53]. Furthermore, we also introduce regularization to the predicted SDF, following [30, 42]. Specifically, to encourage the gradient of SDF to unit length, we adopt the Eikonal term [18]:

$$\mathcal{L}_{eik} = \frac{1}{MN_{ray}}\sum_{m,i}(1 - ||\mathbf{n}_{m,i}||)^2 \tag{7}$$

Besides, to supervise the SDF for accurate surface reconstructions, we approximate the ground truth SDF of sampling point $\mathbf{x}_i$ by computing its distance to the keyframe's

	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	avg
ORB-SLAM2 [26]	✗	0.071	✗	0.023	✗	✗	✗	✗	0.010	-
ORB-SLAM3 [6]	✗	0.017	0.210	✗	0.034	✗	✗	✗	**0.009**	-
DeepV2D [39]	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
DeepFactors [11]	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
DROID-SLAM [41]	0.111	0.018	0.042	**0.021**	**0.016**	0.049	0.026	**0.048**	0.012	0.038
Ours	**0.089**	**0.016**	**0.028**	0.025	0.026	0.052	**0.019**	**0.048**	0.010	**0.035**

	fr1/desk	fr2/xyz	fr3/office
Kintinuous[28]	0.037	0.029	0.030
BAD-SLAM[32]	0.017	0.011	0.017
ORB-SLAM2[26]	0.016	**0.004**	**0.010**
iMAP [35]	0.049	0.020	0.058
NICE-SLAM [53]	0.027	0.018	0.030
GO-SLAM (ours)	**0.015**	0.006	0.013

Table 1: **ATE[m] on the TUM RGB-D [34] benchmark.** The left table shows the results of **monocular** SLAM on sequences from the freiburg1 set. The right one reports the accuracy for **RGB-D** SLAM on sequences from freiburg1, freiburg2 and freiburg3 respectively. '✗' denotes tracking failure, '-' no available data. Results of monocular SLAM [26, 6, 39, 11, 41] and RGB-D SLAM [28, 32, 26, 35, 53] are taken from [41] and [53] respectively.

depth $\mathbf{D}_m$, *i.e.*, $\mathbf{b}(\mathbf{x}_i) = \mathbf{D}_m - D^{ray}_{m,i}$. For SDF learning, we have $|\Phi(\mathbf{x}_i)| \leq |\mathbf{b}(\mathbf{x}_i)|, \forall \mathbf{x}_i$. To satisfy this bound, for near-surface points ($|\mathbf{b}(\mathbf{x}_i)| \leq \tau_{trunc}$, where τ_{trunc} is a hyper-parameter denoted the truncation threshold, set to 16cm), the SDF loss is defined as $\mathcal{L}_{near} = |\Phi(\mathbf{x}_i) - \mathbf{b}(\mathbf{x}_i)|$, while for elsewhere, *i.e.*, free space, we apply a relaxed loss:

$$\mathcal{L}_{free} = \max\left(e^{-\beta\Phi(\mathbf{x}_i)} - 1, \Phi(\mathbf{x}_i) - \mathbf{b}(\mathbf{x}_i), 0\right) \quad (8)$$

where β is a hyper-parameter to apply a penalty when the predicted SDF $\Phi(\mathbf{x}_i)$ is negative in free space. Therefore, our full SDF loss is defined as:

$$\mathcal{L}_{sdf} = \frac{1}{MN_{ray}} \sum_{m,i} \begin{cases} \mathcal{L}_{near} & \textbf{if } |\mathbf{b}(\mathbf{x}_i)| \leq \tau_{trunc} \\ \mathcal{L}_{free} & \textbf{otherwise.} \end{cases} \quad (9)$$

Our instant mapping thread continuously optimizes the scene reconstruction over all sampled pixels of all selected keyframes. Specifically, for each set of selected keyframes, we run the mapping process for a fixed number N_{iter} of iterations. Our total loss is defined as:

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_{dep} \mathcal{L}_{dep} + \lambda_{eik} \mathcal{L}_{eik} + \lambda_{sdf} \mathcal{L}_{sdf}, \quad (10)$$

with $\mathcal{L}_c$, $\mathcal{L}_{dep}$, $\mathcal{L}_{eik}$, and $\mathcal{L}_{sdf}$ being loss balance weights. Given the pose and depth of each selected keyframe, our mapping thread directly uses them without refinement since our full BA and LC have exploited the global geometry to optimize both, being naturally robust in handling occluded regions and back faces.

4. Experimental Results

This section investigates our experimental evaluation, including implementation details, datasets, and key findings.

4.1. Implementation Details

Our system runs on a PC with a 3.5GHz Intel Core i9-10920X CPU and an NVIDIA RTX 3090 GPU. For tracking, we utilize pre-trained weights from DROID-SLAM [41], whereas the rendering networks are trained from scratch. The experiments are performed with the following default settings unless otherwise specified: local window size $N_{local} = 25$ for RGB-D and stereo input, $N_{local} = 50$ for monocular mode, neighboring radius $r_{local} = 1$, $r_{global} = 5$, co-visibility threshold $\tau_{co} = 25.0$, the factor used to constrain the maximum allowed edge $s_{edge} = 8$, sampling points along a ray $N_{strat} = 24$, $N_{imp} = 48$, pixel samples $M = 200$, penalty parameter $\beta = 5.0$, iterations $N_{iter} = 2$, and the loss weights $\lambda_c = 1.0$, $\lambda_{dep} = 1.0$, $\lambda_{eik} = 0.1$, $\lambda_{sdf} = 1.0$. The mesh reconstruction of a scene is achieved by running marching cubes on the SDF values of the queried points. The supplementary material provides more details about SLAM configuration and also qualitative results on various datasets.

4.2. Datasets

We evaluate our GO-SLAM system on several datasets with different input modalities, including the TUM RGB-D, EuRoC, ETH3D-SLAM, ScanNet, and Replica. The TUM RGB-D benchmark [34] is a small-scale indoor dataset with accurate ground truth obtained from an external camera motion capture system. The EuRoC dataset [5] contains 11 indoor stereo sequences recorded from a micro aerial vehicle (MAV). The ETH3D-SLAM dataset [32] provides real-world RGB-D image sequences captured with synchronized global shutter cameras. The ScanNet dataset [12] features richly annotated RGB-D scans of real-world environments, including challenging short and long trajectories. Finally, the Replica dataset [33] provides high-fidelity 3D models of photo-realistic indoor scenes, enabling us to assess the reconstruction performance of our approach. For TUM RGB-D, EuRoC, and ETH3D-SLAM, images are resized to 384×512 resolution, while 240×320 and 320×640 for ScanNet and Replica datasets, respectively.

4.3. Evaluation Metrics

Following the common protocol in the SLAM literature [53, 26, 41], we evaluate the estimated trajectory by aligning it to the ground truth and then calculating the camera pose accuracy based on the Absolute Trajectory Error

	MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg
ORB-SLAM2 [26]	0.035	0.018	0.028	0.119	0.060	**0.035**	0.020	0.048	0.037	0.035	-	-
SVO [15]	0.040	0.070	0.270	0.170	0.120	0.040	0.040	0.070	0.050	0.090	0.790	0.159
ORB-SLAM3 [6]	0.029	0.019	0.024	0.085	0.052	**0.035**	0.025	0.061	0.041	0.028	0.521	0.084
DROID-SLAM [41]	**0.015**	**0.013**	0.035	0.048	**0.040**	0.037	**0.011**	**0.020**	0.018	0.015	**0.017**	**0.024**
Ours	0.016	0.014	**0.023**	**0.045**	0.045	0.037	**0.011**	0.023	**0.016**	**0.010**	0.022	**0.024**

	V103	V202	V203
ORB-SLAM3 [6]	0.037	0.022	✗
DSO [14]	0.903	0.132	1.152
DROID-SLAM [41]	0.020	0.013	**0.014**
Li *et al.* [20]	✗	0.178	✗
GO-SLAM (ours)	**0.018**	**0.011**	0.017

Table 2: **ATE [m] on the EuRoC dataset [5].** In the left table, the results of all methods were obtained by running them on **stereo** video. In the right table, we report the trajectory error of **monocular** SLAM. '✗' denotes tracking failure. Results of [26, 6, 15, 41, 14] and [20] are adopted from [41] and [20] respectively.

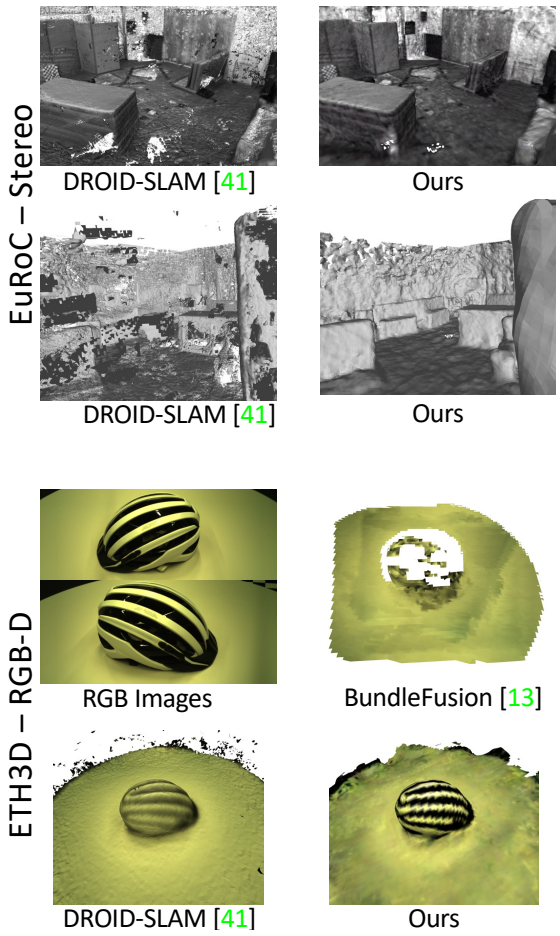

Figure 5: **Qualitative results.** Top: V1_02_Medium (EuRoC), bottom: Helmet (ETH3D). DROID-SLAM [41] reconstructions obtained with TSDF-Fusion [10].

(ATE) RMSE. The reconstruction metrics include *Accuracy* [cm], *Completion* [cm], *Completion Ratio* [$<$ 5cm %], and F-score [$<$ 5cm %]. For the evaluation, we remove regions not observed by any camera. Furthermore, using ground truth trajectory for depth rendering, we evaluate the Depth L1 metric [53] by computing the absolute error between rendered depths from estimated and ground truth meshes.

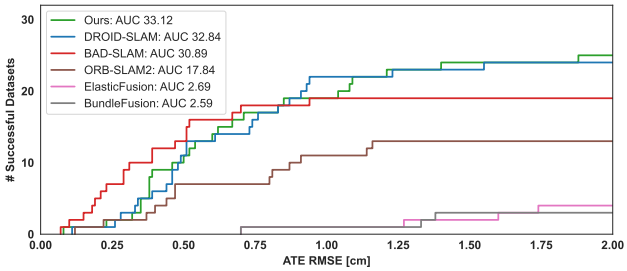

Figure 6: **RGB-D ETH3D-SLAM test benchmark.** Number of successful trajectories (y-axis) as a function of the ATE RMSE (x-axis). Maximum ATE RMSE: 2cm. Results by published SLAM systems [41, 32, 26, 45, 13] and ours.

4.4. Comparison with state-of-the-art SLAM

Here, we evaluate GO-SLAM in synthetic and real-world scenarios and compare it to state-of-the-art SLAM systems in monocular, stereo, and RGB-D settings.

TUM RGB-D (Monocular & RGB-D). In this dataset, we compare with monocular and RGBD SLAM systems. In Tab. 1 (left), we focus on the former methods: traditional SLAM [26, 6] with point-based representation fails on camera tracking on most of the challenging sequences. Among all deep-learning-based methods [39, 11, 41], GO-SLAM with online LC and full BA achieves the lowest average trajectory error. Furthermore, following [53], we also evaluate our method on RGB-D input. Tab. 1 (right) illustrates the clear superiority of our pose estimation compared to recent NeRF-base RGB-D SLAM [35, 53], effectively shrinking the gap with traditional SLAM systems.

EuRoC (Stereo & Monocular). Tab. 2 (left) shows the camera trajectory error of GO-SLAM for all stereo sequences compared to stereo SLAM methods. While existing systems based on neural implicit representation are designed for monocular [52, 20, 31, 9] or RGB-D [35, 53] input only, our method predicts trajectories comparable to state-of-the-art stereo SLAM [26, 6, 15, 41], while also producing globally dense and consistent 3D reconstruction, as shown in Fig. 5. Compared to the noisy result of DROID-SLAM with several holes and floating points, GO-SLAM produces a more complete, smoother surface and a

	Scene ID	0000	0054	0233	0465	0059	0106	0169	0181	Avg.
	# Frames	5578	6629	7643	6306	1807	2324	2034	2349	
RGB-D	iMAP* [35]	55.95	70.11	86.42	85.03	32.06	17.50	70.51	32.10	56.21
	NICE-SLAM [53]	8.64	20.93	9.00	22.31	12.25	8.09	10.28	12.93	13.05
	DROID-SLAM [41] (VO)	8.00	29.28	6.75	11.37	11.30	9.97	8.64	7.38	11.59
	DROID-SLAM [41]	5.36	8.89	4.90	8.32	7.72	7.06	8.01	6.97	7.15
	Ours	**5.35**	**8.75**	**4.78**	**8.15**	**7.52**	**7.03**	**7.74**	**6.84**	**7.02**
Mono.	ORB-SLAM3 [6]	73.93	243.26	25.01	181.86	90.67	178.13	60.15	104.93	119.74
	DROID-SLAM [41] (VO)	11.05	204.31	71.08	117.84	67.26	11.20	16.21	9.94	63.61
	DROID-SLAM [41]	**5.48**	197.71	72.23	114.36	9.00	**6.76**	**7.86**	**7.41**	52.60
	Ours	5.94	**13.29**	**5.31**	**79.51**	**8.27**	8.07	8.42	8.29	**17.59**

Table 3: **ATE[cm] on ScanNet dataset [12].** For DROID-SLAM [41], we also report results for the visual odometry (VO) variant, which does not include the final global bundle adjustment. Results of iMAP* and NICE-SLAM are from [53].

	RGB-D			Mono.					
	iMAP* [35]	NICE-SLAM [53]	**Ours**	Orbeez-SLAM [9]	NeRF-SLAM [31]‡	DROID-SLAM [41]	Li *et al.* [20]	NICER-SLAM [52] ‡	**Ours**
ATE RMSE[cm] ↓	-	1.95	**0.34**	-	-	0.42	0.46	1.88	**0.39**
Depth L1[cm] ↓	7.64	3.53	**3.38**	11.88	4.49	-	-	-	**4.39**
Acc.[cm] ↓	6.95	2.85	**2.50**	-	-	5.03	4.03	**3.65**	3.81
Comp.[cm] ↓	5.33	**3.00**	3.74	-	-	8.49	4.20	**4.16**	4.79
Comp. Ratio[<5cm %] ↓	66.60	89.33	88.09	-	-	64.72	**79.60**	79.37	78.00
Avg. FPS ↑	**10**	$\ll 1$	8	≈ 20	10	**21**	3	$\ll 1$	8

Table 4: **Reconstruction results and ATE[cm] on the Replica dataset (average over 8 scenes).** The results of iMAP* [35] are adopted from NICE-SLAM [53], while results for other methods are taken from the respective original papers. ‡ denotes concurrent works yet unpublished.

Sampling Strategy			Avg.
Latest	Stratified	Top-Ranked	F-score ↑
✓	✗	✗	49.55
✓	✓	✗	83.13
✓	✓	✓	**85.56**

Table 5: **Impact of keyframe selection.** We report average F-score achieved by different sampling strategies.

cleaner reconstruction. In the Tab. 2 (right), we also report the results of monocular SLAM. Specifically, NeRF-based SLAM [20] without online BA fails in most cases, while our approach gives accurate predictions in all sequences.

ETH3D-SLAM (RGB-D). The ETH3D-SLAM dataset provides a public online leaderboard for RGB-D SLAM evaluation. The results in Fig. 6 demonstrate the significant advantage of our deep-learning-based method over published RGB-D SLAM systems with point- [26], surfel- [32, 45] or voxel-based [13] representations. Similarly to our approach, BundleFusion [13] targets globally consistent reconstruction. However, its pose estimation is highly susceptible to errors, resulting in inferior reconstruction as shown in Fig. 5, whereas our method yields smoother reconstruction with photometrically convincing rendering.

ScanNet (Monocular & RGB-D). For an exhaustive evaluation, we test SLAM methods on both short (sequences with less than 5000 frames) and long, complex sequences. Tab. 3 shows that, although DROID-SLAM [41] performs well on short sequences and RGB-D inputs, its accuracy drops dramatically when processing longer sequences in the monocular setting, even performing a final global bundle adjustment. In contrast, GO-SLAM consistently yields the best results, thanks to online loop closing and full BA. Furthermore, as anticipated in Fig. 1, due to the absence of global optimization to eliminate the accumulated trajectory error, NeRF-based SLAM, *e.g.*, iMAP, and NICE-SLAM [53] provide significantly poorer results for pose estimation and 3D reconstruction.

Replica (Monocular & RGB-D). Finally, Tab. 4 shows pose and reconstruction performance achieved by GO-SLAM on the Replica dataset [33]. Our system achieves comparable accuracy with respect to existing RGB-D methods [53] and concurrent monocular [52, 20] ones. However, none of the latter runs in real-time, whereas GO-SLAM achieves a frame rate of 8 FPS with maximum GPU memory consuming 18 GB. Although iMAP [35] alone can achieve a similar speed as GO-SLAM (10 FPS), it yields much worse results. In particular, close to ours, NeRF-SLAM [31] is a concurrent, monocular SLAM system based on DROID-SLAM [41], running at nearly 10 FPS. However, their system lacks BA, which GO-SLAM provides. Moreover, NeRF-SLAM and Orbeez-SLAM [9] lack dense 3D reconstruction results, limiting the comparison. We can only compare on depth rendering quality, for which GO-SLAM achieves better results.

$\mathcal{L}_c$	$\mathcal{L}_{dep}$	$\mathcal{L}_{sdf}$	$\mathcal{L}_{eik}$	Avg. F-score ↑
✓	✗	✗	✗	34.64
✓	✓	✗	✗	83.50
✓	✗	✓	✗	84.00
✓	✓	✓	✗	84.83
✓	✓	✓	✓	**85.56**

Table 6: **Impact of single losses.** We report average F-score achieved with different losses configurations.

Skipping Frames	Speedup	Avg. F-score ↑	Avg. ATE RMSE [cm] ↓
None	1×	**85.56**	**7.02**
1/2	2×	84.67	7.08
3/4	4×	84.80	7.16
7/8	8×	84.41	7.28

Table 7: **Impact of frames skipping.** We report average F-score and ATE when running in real-time.

4.5. Ablation Study

We conclude by studying the impact of the novel designs in the proposed GO-SLAM with RGB-D input.

Keyframe Selection. In Tab. 5, we investigate the impact of different keyframe selection strategies by computing the mean F-score in eight sequences of the Replica dataset [33]. The experimental results prove that updating the 3D model with the latest keyframes or stratified sampling is not sufficient to obtain the best results. Monitoring the pose and depth of each frame continuously and updating the 3D model according to the largest change is crucial for globally consistent reconstruction.

Losses. Tab. 6 evaluates on Replica the effectiveness of each loss term. Results show that the RGB loss alone cannot lead to satisfying results. Geometric supervision, such as depth or SDF loss, provides similar accuracy while integrating all terms produces the best results.

Skipping frames to run in real-time. Tab. 7 shows our GO-SLAM accuracy when skipping RGB-D frames to run at $2\times, 4\times, 8\times$ speed. Remarkably, both mapping accuracy (avg. F-score on Replica [33]) and tracking performance (avg. ATE RMSE on ScanNet [12]) only experience minimal degradation, proving that GO-SLAM is 1) robust to large view changes and 2) ready for real-time usage.

Loop Closing and Full BA. Finally, we study the impact of our efficient loop closing and online full BA on eight scenes of ScanNet [12]. Our baseline DROID-SLAM (VO) [41] (without LC and Full BA) achieves high speed but suffers from a large trajectory error, as shown in Tab. 8. Our efficient LC significantly reduces drift with negligible speed reduction. Introducing Full BA slows down GO-SLAM, but improves global pose estimation significantly.

	Avg. ATE RMSE [cm] ↓	avg. FPS ↑
w/o LC & w/o Full BA	11.59	**30**
w/ LC	8.83	20
w/ Full BA	7.11	12
w/ LC & w/ Full BA	**7.02**	10

Table 8: **Impact of loop closing and full BA.** We report average ATE and FPS with/without our main contributions.

Method	CPU Processing Frequency [GHz]	GPU Consuming Memory [G]↓	Avg. FPS↑	Comp. Ratio[<5cm %] ↑
iMAP* [35]	3.80	**10.13**	10	66.60
NICE-SLAM [53]	3.80	11.72	$\ll 1$	**89.33**
DROID-SLAM [41]	3.50	14.34	**21**	70.52
Ours	3.50	15.63	8	88.09

Table 9: **Hardware requirements and performance.** All results are obtained by running on NVIDIA RTX 3090 GPU and Replica dataset [33] with RGB-D input.

Finally, our full system integrating LC and full BA achieves the best results in pose estimation and 3D reconstruction (see Fig. 3), while enabling real-time performance.

4.6. CPU/GPU Requirements.

We conclude by measuring hardware requirements on the Replica dataset [33]. In Tab. 9 we present several metrics including CPU requirements, maximum GPU memory consumption, average frames per second, and final accuracy, showcasing the performance of various SLAM algorithms during the reconstruction of an entire scene. Specifically, our SLAM system stands out by achieving an optimal balance between CPU/GPU requirements and SLAM performance, combining high accuracy with speed.

5. Conclusions

We introduced a novel, real-time deep-learning-based SLAM algorithm that achieves globally consistent reconstruction with monocular, stereo, or RGB-D input. Our approach explicitly detects loop closures and performs online full BA to minimize trajectory error. Based on NeRF, our 3D reconstruction provides an efficient, compact, and multi-resolution representation. Moreover, our approach continuously updates the dense 3D reconstruction to adapt to the newly-optimized global geometry at high frequency. Our experiments demonstrate the algorithm's robustness in reliably tracking and densely mapping even in large-scale scenes, especially on long monocular trajectories with no depth information, achieving state-of-the-art performance on various datasets.

Acknowledgment. We sincerely thank the scholarship supported by China Scholarship Council (CSC).

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *CVPR*, pages 6290–6301, 2022.

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5855–5864, 2021.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pages 5470–5479, 2022.

[4] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *CVPR*, pages 2560–2568, 2018.

[5] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *IJRR*, 35(10):1157–1163, 2016.

[6] 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. *arXiv preprint arXiv:2007.11898*, 2020.

[7] Di Chang, Aljaž Božič, Tong Zhang, Qingsong Yan, Yingcong Chen, Sabine Süsstrunk, and Matthias Nießner. Rc-mvsnet: unsupervised multi-view stereo with neural rendering. In *ECCV*, pages 665–680. Springer, 2022.

[8] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, pages 14124–14133, 2021.

[9] 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.

[10] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, pages 303–312, 1996.

[11] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE RAL*, 5(2):721–728, 2020.

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017.

[13] 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*, 36(4):1, 2017.

[14] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE TPAMI*, 40(3):611–625, 2017.

[15] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2016.

[16] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, 2022.

[17] David Gallup, Marc Pollefeys, and Jan-Michael Frahm. 3d reconstruction using an n-layer heightmap. In *PR*, pages 1–10. Springer, 2010.

[18] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[19] Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *CVPR*, pages 8932–8941, 2021.

[20] Heng Li, Xiaodong Gu, Weihao Yuan, Luwei Yang, Zilong Dong, and Ping Tan. Dense rgb slam with neural implicit maps. *ICLR*, 2023.

[21] Kejie Li, Yansong Tang, Victor Adrian Prisacariu, and Philip HS Torr. Bnv-fusion: dense 3d reconstruction using bi-level neural volume fusion. In *CVPR*, pages 6166–6175, 2022.

[22] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, pages 5741–5751, 2021.

[23] 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*, pages 405–421, 2020.

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):1–15, 2022.

[25] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[26] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[27] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[28] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*, pages 127–136. IEEE, 2011.

[29] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM TOG*, 32(6):1–11, 2013.

[30] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. *arXiv preprint arXiv:2204.02296*, 2022.

[31] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022.

[32] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *CVPR*, pages 134–144, 2019.

[33] 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.

[34] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012.

[35] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *ICCV*, pages 6229–6238, 2021.

[36] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, pages 5459–5469, 2022.

[37] Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural 3d reconstruction in the wild. In *SIGGRAPH*, pages 1–9, 2022.

[38] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *CVPR*, pages 6243–6252, 2017.

[39] Zachary Teed and Jia Deng. Deepv2d: Video to depth with differentiable structure from motion. *arXiv preprint arXiv:1812.04605*, 2018.

[40] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419. Springer, 2020.

[41] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *NeurIPS*, 34:16558–16569, 2021.

[42] Jingwen Wang, Tymoteusz Bleja, and Lourdes Agapito. Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. *arXiv preprint arXiv:2206.14735*, 2022.

[43] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[44] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, pages 4690–4699, 2021.

[45] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[46] Yabin Xu, Liangliang Nan, Laishui Zhou, Jun Wang, and Charlie CL Wang. Hrbf-fusion: Accurate 3d reconstruction from rgb-d data using on-the-fly implicits. *ACM TOG*, 41(3):1–19, 2022.

[47] 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*, pages 1323–1330. IEEE, 2021.

[48] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *NeurIPS*, 2022.

[49] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[50] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In *CVPR*, pages 5449–5458, 2022.

[51] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *ECCV*, pages 822–838, 2018.

[52] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. *arXiv preprint arXiv:2302.03594*, 2023.

[53] 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*, pages 12786–12796, 2022.

[54] Zi-Xin Zou, Shi-Sheng Huang, Yan-Pei Cao, Tai-Jiang Mu, Ying Shan, and Hongbo Fu. Mononeuralfusion: Online monocular neural 3d reconstruction with geometric priors. *arXiv preprint arXiv:2209.15153*, 2022.