MIPS-Fusion: Multi-Implicit-Submaps for Scalable and Robust Online Neural RGB-D Reconstruction

YIJIE TANG*, National University of Defense Technology, China
JIAZHAO ZHANG*, CFCS, Peking University, China
ZHINAN YU, National University of Defense Technology, China
HE WANG, CFCS, Peking University, China
KAI XU†, National University of Defense Technology, China

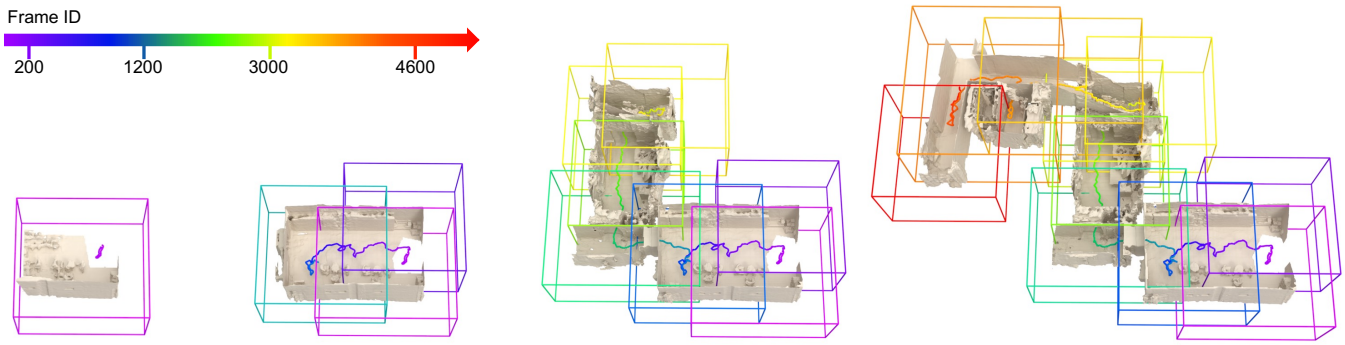

Fig. 1. We introduce MIPS-Fusion, an online RGB-D reconstruction based on a novel neural implicit representation – multi-implicit-submap. Neural submaps are allocated incrementally alongside the scanning trajectory, learned efficiently with local bundle adjustment, refined distributively with back-end optimization, and optimized globally with loop closure. The divide-and-conquer scheme attains both flexibility and scalability. We also propose a hybrid tracking approach where randomized optimization is made possible in the neural setting, enabling efficient and robust tracking even under fast camera motions.

We introduce MIPS-Fusion, a robust and scalable online RGB-D reconstruction method based on a novel neural implicit representation – multi-implicit-submap. Different from existing neural RGB-D reconstruction methods lacking either flexibility with a single neural map or scalability due to extra storage of feature grids, we propose a pure neural representation tackling both difficulties with a divide-and-conquer design. In our method, neural submaps are incrementally allocated alongside the scanning trajectory and efficiently learned with local neural bundle adjustments. The submaps can be refined individually in a back-end optimization and optimized jointly to realize submap-level loop closure. Meanwhile, we propose a hybrid tracking approach combining randomized and gradient-based pose optimizations. For the first time, randomized optimization is made possible in neural tracking with several key designs to the learning process, enabling efficient and robust tracking even under fast camera motions. The extensive evaluation demonstrates that our method attains higher reconstruction quality than the state of the arts for large-scale scenes and under fast camera motions.

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: Online RGB-D reconstruction, neural implicit representation, random optimization

ACM Reference Format:

Yijie Tang*, Jiazhao Zhang*, Zhinan Yu, He Wang, and Kai Xu†. 2023. MIPS-Fusion: Multi-Implicit-Submaps for Scalable and Robust Online Neural RGB-D Reconstruction. *ACM Trans. Graph.* 40, 4, Article 56 (August 2023), 16 pages. https://doi.org/10.1145/3450626.3459676

1 INTRODUCTION

The recent decade has witnessed a proliferation of online dense reconstruction based on RGB-D cameras since the seminal work of KinectFusion [Izadi et al. 2011; Newcombe et al. 2011a]. Its core technique is simultaneous camera localization (tracking) and depth fusion (mapping). Camera tracking has been a long-standing problem in 3D vision and robotics and has gained extensive research. Recently, the tracking robustness under fast camera motions has been drastically improved based on randomized optimization [Zhang et al. 2022, 2021]. Contrary to the rapidly advanced frontiers of tracking, mapping received relatively less attention and the mainstream approach has been largely confined to volumetric [Curless and Levoy 1996] and point-based fusion [Keller et al. 2013; Whelan et al. 2015] until recently when neural implicit mapping came along.

*Joint first authors.
†Corresponding author: Kai Xu (kevin.kai.xu@gmail.com)

Authors' addresses: Yijie Tang*, National University of Defense Technology, China; Jiazhao Zhang*, CFCS, Peking University, China ; Zhinan Yu, National University of Defense Technology, China; He Wang, CFCS, Peking University, China; Kai Xu†, National University of Defense Technology, China.

0730-0301/2023/8-ART56 $15.00
https://doi.org/10.1145/3450626.3459676

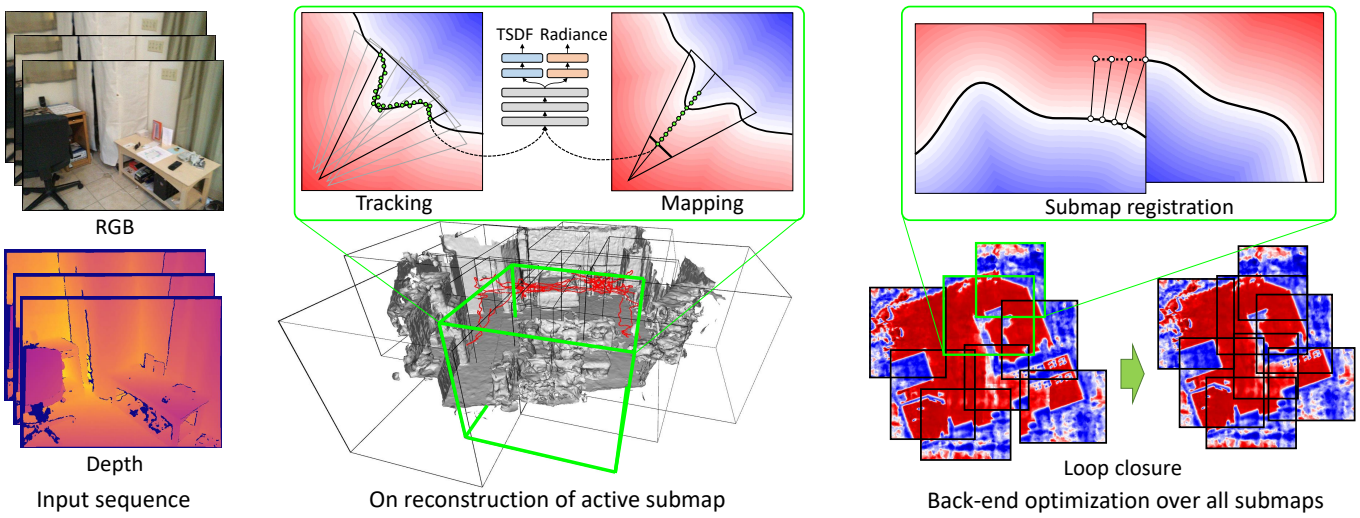

Fig. 2. Method overview. Our method is comprised of online reconstruction of active submap based on neural tracking and mapping and back-end optimization over all inactive submaps based on intra-submap refinement and and inter-submap registration. The latter facilitates submap-level loop closure.

The learning of implicit representation of 3D objects and scenes is drawing increasing attention lately, with many powerful methods proposed and a recent climax reached by neural radiance fields (NeRF) [Mildenhall et al. 2021]. Traditional dense reconstruction approaches adopting explicit volumetric representation suffer from the scalability issue. The storage cost makes it difficult to map a large scene such as a floor of a building of a moderate size. Neural implicit representation seems a promising solution to scalable mapping since it encodes the scene with a compact, end-to-end learnable neural network. In several neural SLAM and RGB-D reconstruction works [Sucar et al. 2021], camera tracking can be jointly optimized with the neural map representation. Despite the encouraging progress that has been made, however, neither the scalability of neural mapping nor the robustness of neural tracking is satisfactory to date.

Neural networks have the issue of limited learning capacity. Some recent works devise dense feature grids to mitigate the issue at the cost of cubic spatial complexity which is again hard to scale. NICE-SLAM [Zhu et al. 2022] adopts hierarchical feature grids to improve scalability to some extent. We advocate the use of *purely neural maps* and aim to fully exploit the potential of implicit representations. To mitigate the limited capacity issue without scarifying scalability, we propose incremental allocation and on-the-fly learning of multiple neural fields alongside the scanning trajectory (see Figure 1). Each neural field, coined *neural submap*, governs a local subvolume and encodes the scene geometry and colors defined in its *local coordinate frame*. The neural submaps are allocated incrementally and learned efficiently with a local bundle adjustment (BA). To achieve a smooth map transition, we ensure that adjacent neural submaps are spatially overlapping and updated with their shared keyframes jointly. This on-demand *multi-implicit-submap* scheme allows a scalable reconstruction with rich local geometric details.

To achieve high tracking robustness, especially under fast camera motions, we propose a hybrid tracking scheme combining both gradient-based (GO) and randomized optimizations (RO). Realizing RO in the implicit setting is conceptually straightforward but computationally prohibitive since it needs to evaluate fitness for a sufficiently large number of hypothetic camera poses and each evaluation involves many times of network inference. To accelerate this process, we devise two key designs. *First*, we propose a *depth-to-TSDF loss* for which network inference is done only for points unprojected from the depth map and transformed by a hypothetic pose; no expensive volumetric depth rendering is needed as in the previous works [Sucar et al. 2021; Zhu et al. 2022]. Meanwhile, this loss is differentiable and admits GO. This allows for a scheduled optimization scheme: RO is used in early iteration steps to obtain a good initialization, which is followed by GO-based refinements. We can optionally optimize a photometric loss based on RGB rendering and GO when the RGB observations are reliable (e.g., texture-rich and blur-free). *Second*, to further speed up the fitness evaluation, we opt for a *light-weight network for classification-based TSDF prediction* trained to output a probabilistic distribution over a discrete set of distances. This makes our neural submaps easier to learn. More importantly, the epistemic uncertainty of TSDF classification can be used to build a weighted fitness for improved tracking accuracy.

The on-the-fly allocation of multiple neural submaps facilitates distributed map refinement. The submap corresponding to the current keyframe, referred to as *active submap*, is usually insufficiently trained for the sake of maintaining realtime framerate. To this end, in parallel to the online updating of the active submap, we fine-tune the *inactive submaps* in a separate thread using denser sampling of keyframes and depth pixels. Furthermore, our method also supports *loop closure of neural submaps*. Once a non-trivial loop is detected, we perform a submap-level BA to jointly optimize the poses of all submaps in the loop. Since our neural submaps are defined in their local coordinate frames, map adjustment can be realized efficiently by transforming the submaps which is much faster than neural updating of learned submaps. See Figure 2 for an overview.

The design philosophy of our approach is a divide-and-conquer mapping scheme for flexibility and scalability, together with a hybrid tracking scheme for efficiency and robustness, both enabled by lightweight neural representations assisted with an easier task of classification. We have evaluated our method on several public benchmarks and a newly introduced dataset of large-scale scenes. On all benchmarks, our method outperforms the state-of-the-art neural SLAM/reconstruction methods. It also successfully reconstructs the challenging sequences with fast camera motions on which all previous neural methods failed. In summary, the contributions of our work include:

- We propose a purely neural mapping approach which achieves scalable dense RGB-D reconstruction through incrementally allocating and on-the-fly learning multiple neural submaps.
- We propose a robust neural tracking method which works well for fast camera motions via combining gradient-based and randomized optimizations in the neural representation.
- Our multi-implicit-submap approach supports parallel fine-tuning of submaps and, for the first time, realizes loop closure in neural mapping with submap-level BA.

2 RELATED WORK

We will focus on works on *dense SLAM and online RGB-D reconstruction* and review them in terms of mapping and tracking separately covering both traditional and neural approaches, followed by a discussion on loop closure in the same context.

Mapping. Since the seminal work of DTAM [Newcombe et al. 2011b], dense SLAM has been extensively studied over the years (see the survey by [Taketomi et al. 2017]). Taking advantage of RGB-D cameras, KinectFusion [Izadi et al. 2011; Newcombe et al. 2011a] achieves the first online RGB-D reconstruction via realizing real-time volumetric depth fusion [Curless and Levoy 1996]. In order to handle larger environments, spatial hierarchies [Chen et al. 2013] and hashing schemes [Kahler et al. 2015; Nießner et al. 2013] have been proposed. Some recent works propose to learn depth map fusion to account for fusion errors [Cao et al. 2018; Weder et al. 2020], handle outliers [Weder et al. 2021], or preserve details [Li et al. 2022a; Lionar et al. 2021]. Another line of works adopt point-, surfel- [Henry et al. 2014; Keller et al. 2013; Liu et al. 2016; Pradeep et al. 2013; Wang et al. 2019; Whelan et al. 2012, 2016] which leads to better mapping scalability but produces lower map density. Recently, Xu et al. [2022] propose a unique mapping scheme based on on-the-fly implicits of Hermite Radial Basis Functions (HRBFs) demonstrating good accuracy and robustness of RGB-D reconstruction.

Another approach to scalable mapping is to represent the global map as a combination of submaps, which dates back at least to the Atlas framework [Bosse et al. 2003]. The existing works that utilize explicit TSDF subvolumes to maintain map consistency can be largely classified into two categories, i.e., those which attempt to partition space and minimize overlap between subvolumes [Henry et al. 2013; Kähler et al. 2016] and those which do not partition space [Fioraio et al. 2015; Millane et al. 2018].

Neural implicit representation offers new opportunities for scalable mapping, taking advantage of the expressiveness and compactness of learned geometric priors. CodeSLAM [Bloesch et al. 2018] trains an encoder-decoder network to embed depth maps as low-dimensional codes which can be used to optimize key-frame poses. DI-Fusion [Huang et al. 2021] proposes to learn geometric priors to embed 3D points in a low-dimensional latent space which can then be decoded into SDF values. Such learned geometric prior is, however, inaccurate in handling complex geometric details. Recently, iMAP trains an implicit network [Sucar et al. 2021] online to represent a scene. Several careful designs are made to attain a good trade-off between compactness and accuracy of mapping. Inspired by that, iSDF [Ortiz et al. 2022] learns to map with a neural SDF with novel self-supervision and sampling strategies. Azinović et al. [2022] propose to represent scene surface using an implicit TSDF and incorporate this representation in the NeRF framework for rendering-based learning. Block-NeRF [Tancik et al. 2022] scales NeRF to render city-scale scenes spanning multiple blocks but not support online reconstruction.

Observing that the prior works such as iMAP use a single MLP to represent the entire scene, which can only be updated globally and hence suffers from the forgetting issue when scanning a large scene, the following works NICE-SLAM [Zhu et al. 2022] and Vox-Fusion [Yang et al. 2022] propose a hybrid representation which combines multi-level grid-based features and a neural decoder, inspired by several recent works [Li et al. 2022b; Liu et al. 2020; Peng et al. 2020; Sun et al. 2022]. The learnable grid-based features can be seen as a "spatially distributed network" with immense representation capacity. The decoder can be either pretrained *a priori* or learned on-the-fly. Rosinol et al. [2022] propose a geometric and photometric 3D mapping pipeline from monocular images based on hierarchical volumetric neural radiance fields. Recently, Wang et al. [2023] adopt parametric encoding to accelerate learning convergence based on the multiresolution hash encoding [Müller et al. 2022] on NeRF. More recently, Johari et al. [2023] propose a new scene representation consisting of multi-scale axis-aligned perpendicular feature planes (tri-plane features).

Our method differs from the existing works in that it utilizes multiple MLPs to jointly represent the scene. The neural submaps can be learned and refined independently, achieving a balance of expressiveness, compactness, and flexibility.

Tracking. Regarding camera tracking, KinectFusion and DTAM estimate poses for the input depth maps using frame-to-model alignment based on point-to-plane ICP. To improve robustness, many works further adopt photometric and/or feature-based tracking [Dai et al. 2017b; Whelan et al. 2015]. Bylow et al. [2013] realize a feature-free tracking through optimizing an objective defined with depth-to-TSDF conformation. While most state-of-the-art tracking approaches rely on gradient-based optimization, Zhang et al. [2021] argue that gradient-based methods are brittle when handling fast camera motions due to the high nonlinearity of large pose optimization. They propose ROSEFusion which minimizes a depth-to-TSDF objective similar to [Bylow et al. 2013] using randomized optimization and achieves highly robust camera tracking under fast motions.

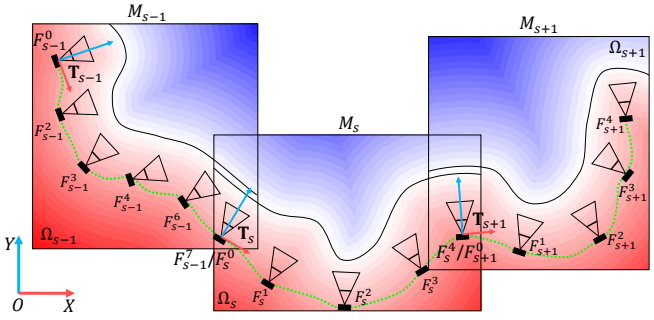

Fig. 3. The multi-implicit-submap representation for online RGB-D reconstruction. Three submaps M_{s-1}, M_s and M_{s+1} with their subvolumes and keyframes are shown. For each submap, the first keyframe is the anchor at which the local coordinate frame of the submap is defined. Adjacent submaps share at least one keyframe, e.g., F^7_{s-1} of M_{s-1} is F^0_s of M_s.

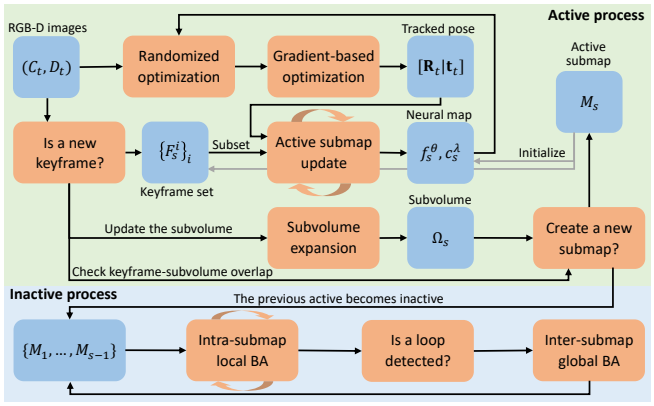

Fig. 4. System pipeline. Our method runs two processes in parallel. The active process works on active submap and performs tracking of the current RGB-D frame, selection of keyframes, and mapping based on both the current frame and a subset of the keyframes. The subvolume of the active submap expands as new keyframes comes in. When a new active submap is created, the previous active submap becomes inactive. Back-end optimization is performed and loop closure conducted whenever available.

Later, the method is extended to realize depth-inertial odometry with even higher tracking robustness [Zhang et al. 2022].

Some methods include back-end optimizations such as bundle adjustment [Dai et al. 2017b; Schops et al. 2019] and pose-graph optimization [Kähler et al. 2016; Kerl et al. 2013] to improve tracking accuracy. These back-end optimizations are typically time-consuming and therefore conducted only for keyframes and invoked sparsely in time. Back-end optimization is also used for loop closure; see below.

In the context of neural SLAM and neural online reconstruction, camera tracking is solved either in a *coupled* way based on inverse neural representation learning via differentiable volumetric rendering, or in a *decoupled* manner where camera poses are optimized independently without relying on the neural representation. Coupled approaches, such as iMAP, NICE-SLAM, and Vox-Fusion, adopt a render-and-compare paradigm where both RGB and depth maps are rendered and compared to the corresponding observations. Since volumetric rendering is expensive, it is done only for subsampled keyframes and pixels. Generally speaking, while coupled solutions seem neat and more integrated, decoupled ones usually lead to more robust tracking results. Note, however, that decoupled approaches can employ not only traditional tracking methods (e.g., [Chung et al. 2022; Koestler et al. 2022]), but also neural tracking models. For example, iDF-SLAM [Ming et al. 2022] learns a neural feature detector and DROID-SLAM [Teed and Deng 2021] learns a neural optical flow estimator for frame-to-frame registration.

Our tracking method belongs to *coupled* approach since it runs completely on the neural representation. To attain high robustness, we, for the first time, integrate gradient-based and randomized optimizations in the neural setting, although not fully differentiable due to the randomized part. We use the same objective function for both optimization processes, facilitating a natural switching between the two for a scheduled optimization. This objective function also saves depth rendering. Combining an efficient GPU implementation, we realize the first neural tracker working under fast camera motions.

Loop closure. Loop closure is a classic technique in the back-end of SLAM systems. Deep learning has been mainly used in loop closure detection (e.g., [Merrill and Huang 2018]). Once detected, traditional methods are used for global adjustment of the camera trajectory and the map. Regarding bundle adjustment, BA-Net [Tang and Tan 2018] proposes a learnable bundle adjustment layer which optimizes over a number of coefficients used to linearly combine a depth basis as well as the damping factor of the Levenberg-Marquardt algorithm. DROID-SLAM proposes a differentiable dense bundle adjustment layer which computes a Gauss-Newton update to camera poses and dense per-pixel depth to match the estimated optical flow. However, they were not shown to work for loop closure. In fact, adjusting maps, especially neural maps, is much harder than camera trajectories. Yuan and Nüchter [2022] propose an algorithm for the $SE(3)$-transformation of neural implicit maps for remapping in loop closure. We did not follow this method since our submaps can be updated locally and transformed globally. In fact, multi-submap adjustment admits more DoFs than $SE(3)$ transformations.

3 METHOD

The input to online reconstruction is an RGB-D sequence $\{C_t, D_t\}_{t=0:T}$ (C and D are color and depth images, respectively) captured by an RGB-D camera and the output is a surface reconstruction of the scene being captured as well as a trajectory of 6DoF camera poses, $\{[\mathbf{R}_t|\mathbf{t}_t]\}_{t=0:T}$ ($[\mathbf{R}|\mathbf{t}] \in SE(3)$ represents a 6D camera pose in the world coordinate frame). Our method is built upon the neural mapping framework of iMAP [Sucar et al. 2021]. The key problem of online neural RGB-D reconstruction is the joint optimization of the neural map and the 6D camera pose of each frame.

Figure 2 gives an overview of our method. In this section, we first introduce our multi-implicit-submap representation (Section 3.1). Based on the representation, we describe the optimization losses used for mapping and tracking (Section 3.2). We then elaborate on the optimization processes for camera poses (Section 3.3) and neural maps (Section 3.4). Finally, we discuss the back-end optimization

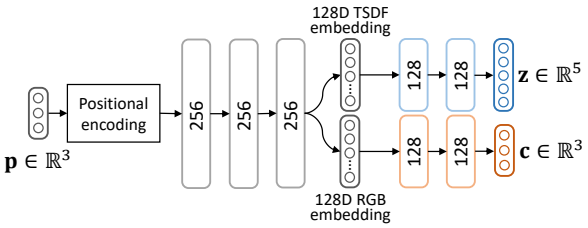

Fig. 5. The network architecture of SDF and color prediction.

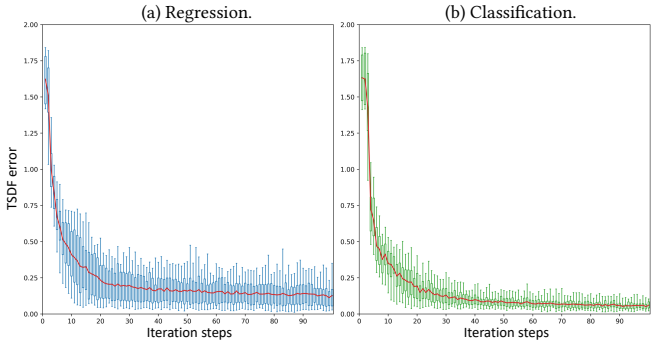

Fig. 6. Convergence rate (learning speed) comparison between regression- and classification-based TSDF prediction.

together with loop closure (Section 3.5). Figure 4 shows our system pipeline which will be elaborated below.

3.1 Multi-Implicit-Submap Representation

Our scene representation is a sequence of S neural submaps $\{M_s\}_{s=1:S}$ allocated alongside the scanning trajectory. Each submap is a tuple $M_s = \langle f_s^\theta, c_s^\lambda, \mathbf{T}_s, \mathcal{F}_s, \Omega_s \rangle$, where f_s^θ is the truncated signed distance function (TSDF) and c_s^λ the radiance field, both implemented as a multi-layer perceptron (MLP) and parameterized by θ and λ, respectively. More advanced scene representation techniques [Müller et al. 2022; Wang et al. 2023] could be used for enhanced reconstruction quality. While c_s^λ is trained as usual, f_s^θ is learned as a classifier to facilitate fast mapping and tracking. Both the two functions are defined in the local coordinate frame of the submap M_s with $\mathbf{T}_s$ being the global pose of the submap in the world coordinate frame. $\mathcal{F}_s$ is the set of keyframes associated to M_s. The global pose of the first keyframe $F_s^0 \in \mathcal{F}_s$ is set to the submap pose $\mathbf{T}_s$. Therefore, F_s^0 is also referred to as the anchor keyframe of M_s. Ω_s is the axis-aligned cuboid subvolume that M_s governs. See Figure 3 for illustration.

Learning local neural map functions gains flexibility such that each submap can be transformed as a whole for efficient global alignment of submaps in loop closure. Meanwhile, local functions are generally easier to learn due to the low data bias caused by localized data distributions in local coordinate frames.

Classification-based neural TSDF. Given a 3D point defined in the world coordinate frame and located in the subvolume of M^s, i.e., $\mathbf{x}^\mathrm{W} \in \Omega^s$, its TSDF value by M^s can be computed as $\psi_s(\mathbf{x}^s) = f_s^\theta(\mathbf{x}^s)$, where $\mathbf{x}^s = \mathbf{T}_s^{-1}\mathbf{x}^\mathrm{W}$ is the point transformed into the local coordinate frame of M^s; see Figure 5. We use a sinusoidal positional encoding to encode the 3D position before feeding it into the neural network [Mildenhall et al. 2021]. For a point located in the overlapping area of two subvolumes, e.g., $\mathbf{x}^W \in \Omega^s \cap \Omega^t$, its global TSDF value can be evaluated as a weighted combination of the local TSDF values given by the corresponding two submaps:

$$\psi(\mathbf{x}^\mathrm{W}) = \frac{w_s(\mathbf{x}^s)\psi_s(\mathbf{x}^s) + w_t(\mathbf{x}^t)\psi_t(\mathbf{x}^t)}{w_s(\mathbf{x}^s) + w_t(\mathbf{x}^t)}, \tag{1}$$

where $\mathbf{x}^* = \mathbf{T}_*^{-1}\mathbf{x}^\mathrm{W}$ and the weight is $w_* = \frac{1}{h_*(\mathbf{x}^*)^2}$ with h_* being the uncertainty of TSDF prediction by the submap (see below).

In particular, we choose 5 signed distance values uniformly from the interval $[-\tau, \tau]$, i.e., $\{\ell_1 = -\tau, \ell_2 = -\frac{\tau}{2}, \ell_3 = 0, \ell_4 = \frac{\tau}{2}, \ell_1 = \tau\}$, where $\tau = 0.1$m is the truncation distance. Given a point $\mathbf{x}^s$, f_s^θ outputs a normalized 5D vector $\mathbf{z} = (z_i)_{i=1,\ldots,5}$ corresponding to the five distance values with z_i indicating how probable $\mathbf{x}^s$'s SDF value is close to ℓ_i. We can then approximate $\mathbf{x}^s$'s SDF value with a soft argmax:

$$\psi_s(\mathbf{x}^s) = \sum_{i=0}^{5} \frac{e^{\beta z_i}}{\sum_{i=1}^{5} e^{\beta z_i}} \ell_i, \tag{2}$$

where we use a coldness parameter $\beta = 10$. With soft argmax, our method obtains gradients from the training losses, facilitating effective learning of submaps. Moreover, the coldness parameter controls the smoothness of the probabilities over multiple classes, leading to a better approximation of SDF distributions. Figure 6 shows that our classification-based TSDF prediction converges faster and hence learns faster than regression-based one.

Defining the probability distribution over the five classes as $p_i = \frac{e^{\beta z_i}}{\sum_{i=1}^{5} e^{\beta z_i}}, i = 1, \ldots, 5$, we can measure the uncertainty of the TSDF classification as the Shannon entropy: $h_s(\mathbf{x}^s) = -\sum_{i=1}^{5} p_i \log p_i$. The plots in Figure 7 demonstrate that the uncertainty measurement is useful in filtering points with inaccurate TSDF predictions, and is insensitive to class count.

Neural radiance field. In addition to the neural geometry representation, we also learn for each submap a neural appearance representation [Mildenhall et al. 2021], c_s^λ, for optimizing mapping and tracking with photometric losses. Similar to [Sucar et al. 2021], we omit the encoding of view directions since we are not interested in modeling view-dependent effects such as specularities. Implemented also with MLPs, it takes as input a 3D position (after sinusoidal encoding) $\mathbf{x}^s$ and regresses a radiance value $c_s^\lambda(\mathbf{x}^s)$ as output. This simplification also makes c_s^λ light-weight and faster to learn.

Color and depth map rendering. We render a color image as a weighted sum of radiance values of points $\mathbf{q} = \mathbf{o} + d_p(\mathbf{q})\mathbf{v}_p$ sampled along the ray $\mathbf{v}_p$ shooting from the camera center $\mathbf{o}$ to an image pixel p, with $d_p(\mathbf{q})$ being $\mathbf{q}$'s depth. The weights are computed directly from signed distance values as the product of two sigmoid functions [Azinović et al. 2022]:

$$\omega_p(\mathbf{q}) = \sigma\left(\frac{\psi(\mathbf{q})}{\tau}\right)\sigma\left(-\frac{\psi(\mathbf{q})}{\tau}\right). \tag{3}$$

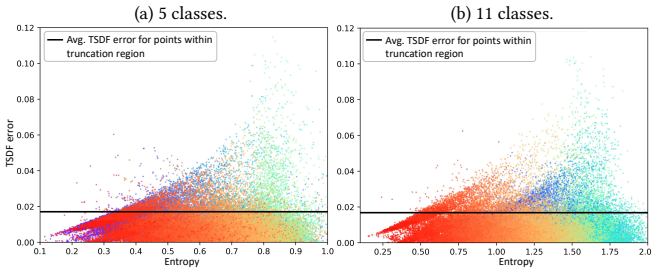

Fig. 7. Scattered plots of uncertainty (Shannon entropy) of samples with different TSDF errors. The two plots are for TSDF classification against 5 and 11 classes, respectively. The colors of the dots indicate the TSDF values (red is large (free space), blue is small ($-\tau$) and green corresponds largely to the zero-level set). Points with larger TSDF errors generally have higher uncertainties and vice versa, making the uncertainty measurement useful in filtering points with inaccurate TSDF predictions. Note also that the majority of dots reside in the lower right instead of the upper left meaning that the entropy-based measure prefers false positive (FP) over false negative (FN). FN tends to trust predictions with higher TSDF error, which is more harmful to reconstruction accuracy.

The color along of pixel p is approximated as a weighted sum of radiance sampled on ray $\mathbf{v}_p$ *within only the truncation region*:

$$\tilde{C}(p) = \frac{1}{\sum_{\mathbf{q}\in\mathcal{S}_p^{\text{tr}}} \omega_p(\mathbf{q})} \sum_{\mathbf{q}\in\mathcal{S}_p^{\text{tr}}} \omega_p(\mathbf{q}) c_s^{\lambda}(\mathbf{q}, \mathbf{v}_p), \tag{4}$$

where $\mathcal{S}_p^{\text{tr}}$ is the set of sampled points in truncation region. Sampling only within truncation region leads to a much faster rendering with limited quality degrading since the weights drop quickly to zero outside the truncation according to Eq. (3). The simplifications on rendering is fine to our task. Depth can be rendered similarity:

$$\tilde{D}(p) = \frac{1}{\sum_{\mathbf{q}\in\mathcal{S}_p^{\text{tr}}} \omega_p(\mathbf{q})} \sum_{\mathbf{q}\in\mathcal{S}_p^{\text{tr}}} \omega_p(\mathbf{q}) d(\mathbf{q}). \tag{5}$$

Note that, however, the depth rendering is used only for visualizing the learned geometry; neither our mapping or tracking involves depth rendering loss due to its high computational cost.

3.2 Optimization Losses for Mapping and Tracking

To realize mapping and tracking, we optimize the neural scene representations together with the keyframe poses through minimizing different combinations of four different losses. The four losses include (1) a depth-to-TSDF loss $\mathcal{L}_{\text{d2t}}$ for imposing the confirmation between the posed depth map and the learned TSDF, (2) a TSDF truncation-region loss $\mathcal{L}_{\text{tr}}$ for learning the SDF values within the truncation region, (3) a TSDF free-space loss $\mathcal{L}_{\text{fs}}$ for learning the truncation of TSDF on the visible side of the surface within the viewing frustum, and (4) an RGB rendering loss $\mathcal{L}_{\text{rgb}}$ for enforcing photometric consistency.

While $\mathcal{L}_{\text{d2t}}$ is used for tracking (RO and GO), $\mathcal{L}_{\text{tr}}$, $\mathcal{L}_{\text{fs}}$ and $\mathcal{L}_{\text{rgb}}$ are used for both mapping and GO-based pose optimization. While the latter three losses are commonly seen in recent works, the depth-to-TSDF loss is new to neural SLAM and we show through evaluations that it is highly effective and efficient for tracking optimization. We do not use the depth rendering loss of [Yang et al. 2022] since it has been encompassed by $\mathcal{L}_{\text{tr}}$ and $\mathcal{L}_{\text{fs}}$.

Depth-to-TSDF loss. Given the depth image of the current frame D_t and the current neural submap f_s^{θ}, our task is to compute the 6-DoF camera pose of the current frame in the world coordinate frame $[\mathbf{R}_t|\mathbf{t}_t] \in SE(3)$ while optimizing the f_s^{θ} with the 3D information of the posed depth map. To this end, we define a frame-to-model error metric to measure the fitness of how well D_t "fits into" the TSDF under pose $[\mathbf{R}_t|\mathbf{t}_t]$ [Bylow et al. 2013; Zhang et al. 2021]. For each pixel p of D_t, we can compute based on its depth $D_t(p)$ the corresponding 3D point $\mathbf{x}_p$ in the camera coordinate frame of the current frame. We can then transform this point into the world coordinate frame:

$$\mathbf{x}_p^{\text{W}} = \mathbf{R}_t \mathbf{x}_p + \mathbf{t}_t. \tag{6}$$

We use the unprojected 3D points to query the TSDF map defined in the world coordinate system and obtain point-to-surface distances directly. If the camera pose is correct, it is expected that the point-to-surface distances of all unprojected 3D points should be zero. Assuming that the depth measurements contain Gaussian noise and that all pixels are independent and identically distributed, the likelihood of observing depth image D_t from camera pose $[\mathbf{R}_t|\mathbf{t}_t]$ is

$$p(D_t|\mathbf{R}_t, \mathbf{t}_t) \propto \prod_{p\in\mathcal{P}} \exp\left(-\psi_s(\mathbf{T}_s^{-1}(\mathbf{R}_t\mathbf{x}_p + \mathbf{t}_t))^2\right), \tag{7}$$

where $\mathcal{P}$ is the set of sampled pixels. Our depth-to-TSDF loss is defined as the negative log-likelihood:

$$\mathcal{L}_{\text{d2t}}(\mathbf{R}_t, \mathbf{t}_t) = -\log p(D_t|\mathbf{R}_t, \mathbf{t}_t) = \sum_{p\in\mathcal{P}} \psi_s(\mathbf{T}_s^{-1}(\mathbf{R}_t\mathbf{x}_p + \mathbf{t}_t))^2. \tag{8}$$

The loss is used only for pose optimization of the current frame.

TSDF truncation-region loss. The truncation-region loss is devised to supervise the MLP to output correct SDF values for points within the truncation region:

$$\mathcal{L}_{\text{tr}}(\Theta) = \frac{1}{|\mathcal{P}|} \sum_{p\in\mathcal{P}} \frac{1}{|\mathcal{S}_p^{\text{tr}}|} \sum_{\mathbf{q}\in\mathcal{S}_p^{\text{tr}}} \left(\psi_s(\mathbf{T}_s^{-1}\mathbf{q}) - (D_t(p) - d_p(\mathbf{q}))\right)^2, \tag{9}$$

where $\mathcal{S}_p^{\text{tr}}$ is the set of points sampled on ray $\mathbf{v}_p$ and within the truncation region. $D_t(p) - d_p(\mathbf{q})$ is the signed distance value of sample point $\mathbf{q}$ with $d_p(\mathbf{q})$ being the sampled depth along ray $\mathbf{v}_p$ of pixel p. Θ may encompass the parameters of the TSDF θ and the camera pose, depending on whether the task is mapping or tracking. The predicted signed distance $\psi_s(\mathbf{T}_s^{-1}\mathbf{q})$ is computed based on the output of f_s^{θ} according to Eq. (2). This loss is used for optimizing the neural map parameters θ. To ensure a meaningful uncertainty measurement of f_s^{θ}'s output $(z_i)_{i=1,\ldots,5}$, we additionally minimize the following EMD-based distribution loss:

$$\mathcal{L}_{\text{tr-emd}}(\Theta) = \frac{1}{|\mathcal{P}|} \sum_{p\in\mathcal{P}} \frac{1}{|\mathcal{S}_p^{\text{tr}}|} \sum_{\mathbf{q}\in\mathcal{S}_p^{\text{tr}}} \sum_{i=1}^{5} z_i |i - y(\mathbf{q})|, \tag{10}$$

where $y(\mathbf{q})$ is the ground-truth label of TSDF classification at $\mathbf{q}$.

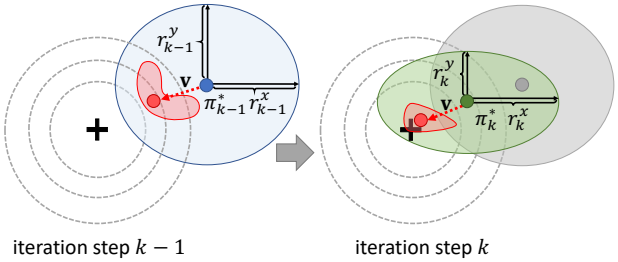

Fig. 8. Moving and rescaling PST from iteration step $k-1$ to k. At each step, we first identify the Advantage Particle Set (APS, shaded in red) which is a subset of the current PST Ω (blue ellipse), and then compute the current best solution π_k^* (red dot) as the weighted average of the particles in APS. The PST is then moved to π_k^* with the new axis length proportional to the vector $\mathbf{v} = \pi_k^* - \pi_{k-1}^*$, thus evolving into (green ellipse).

TSDF free-space loss. The free-space loss directs the neural map to output a value equal to the truncation value τ for the empty region in the visible side of the viewing frustum:

$$\mathcal{L}_{\text{fs}}(\Theta) = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}\frac{1}{|\mathcal{S}_p^{\text{fs}}|}\sum_{\mathbf{q}\in\mathcal{S}_p^{\text{fs}}}\left(\psi_s(\mathbf{T}_s^{-1}\mathbf{q}) - \tau\right)^2, \tag{11}$$

where $\mathcal{S}_p^{\text{fs}}$ is the set of sample points in the free space of the visible side of ray $\mathbf{v}_p$. For free-space TSDF, a similar distribution loss $\mathcal{L}_{\text{fs-emd}}$ is defined as in Eq. (10).

Remarks on the TSDF losses. The depth-to-TSDF loss is estimated by direct point query and accounts only for 3D points unprojected from the depth map. This makes it much more efficient than volumetric rendering. Therefore, it is suited for depth-based tracking. The TSDF truncation-region and free-space losses concern about the full occupancy (geometry) information in the viewing frustum of a frame, which is thus well-targeted for the mapping task.

RGB rendering loss. The RGB loss measures the squared differences between the rendered and the input (ground-truth) color images:

$$\mathcal{L}_{\text{rgb}}(\Lambda) = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}\|\tilde{C}(p) - C_t(p)\|, \tag{12}$$

where $C_t(p)$ is the color at pixel p of the input RGB image of the current frame and $\tilde{C}(p)$ is defined in Eq. (4). Λ may encompass the parameters of the radiance field λ and the camera pose, depending on whether the task is mapping or tracking.

3.3 Tracking with Hybrid Optimization

Given the current RGB-D frame, we compute its camera pose by minimizing the depth-to-TSDF loss and the RGB rendering loss while keeping the current active submap fixed. To do so, we employ a hybrid optimization combining both randomized optimization (RO) and gradient-based optimization (GO). The optimization is scheduled as follows. The RO is first performed for the depth-to-TSDF loss only. After a fixed number of RO iterations, the GO is invoked for both two losses for another fixed number of iterations.

Randomized pose optimization. The RO adopts the particle filter optimization (PFO) framework which samples and propagates a population of particles (candidate solutions) iteratively to make them cover the optimal solution as quickly as possible. In our case, a solution is a 6DoF camera pose $\pi = (\mathbf{R}, \mathbf{t}) \doteq (q_x, q_y, q_z, x, y, z)$ with q_x, q_y and q_z being the imaginary part of the rotation quaternion and $\mathbf{t} = (x, y, z)^T$. The key to making PFO realtime capable is to *presample* a set of particles uniformly within the unit sphere in the 6D solution space, referred to as Particle Swarm Template (PST), instead of sampling the particles on the fly throughout the optimization. The PST is then moved and rescaled into a 6D ellipsoid over the optimization iterations to gradually cover the optimal solution. Let us denote the PST at iteration k as Π_k, which is parameterized by a center position $\mathbf{c}$ and a vector of axis lengths (each for one of the six dimensions) $\mathbf{r} = (r_d)_{d=1:6}$.

In each iteration step k, we first evaluate the depth-to-TSDF loss η for each particle $\pi_k^i = (\mathbf{R}_k^i, \mathbf{t}_k^i) \in \Pi_k$ based on the *uncertainty-weighted* depth-to-TSDF loss:

$$\eta(\pi_k^i) = \sum_{p\in\mathcal{P}}\frac{\psi_s(\mathbf{T}_s^{-1}(\mathbf{R}_k^i\mathbf{x}_p + \mathbf{t}_k^i))^2}{h_s(\mathbf{x}_p)^2}, \tag{13}$$

where $\mathbf{x}_p$ is the unprojected 3D point corresponding to pixel p. The uncertainty $h_s(\cdot)$ of TSDF prediction $\psi_s(\cdot)$ has been given in Section 3.1. Among all the particles in Π_k, we collect those whose depth-to-TSDF loss is smaller than π_{k-1}^* (the best solution in step $k-1$) into an Advantage Particle Set (APS). The best state at the current step k takes the centroid of the APS. The PST is moved to be centered at the best state of the current step π_k^*.

To rescale the PST, we compute the axis lengths $\mathbf{r}_k$ of the current step as follows:

$$\mathbf{v} = \pi_k^* - \pi_{k-1}^*, \tag{14}$$

$$\hat{\mathbf{r}}_k = \eta(\pi_k^*)\frac{\mathbf{v}}{\|\mathbf{v}\|} + \epsilon, \tag{15}$$

where $\mathbf{v}$ is an anisotropic attractor which drives the particles towards the best solution $\mathbf{s}_k^*$ (the global best of the particle set). $\hat{\mathbf{r}}_k$ is the (interim) vector of axis lengths of Π_k, which is scaled by depth-to-TSDF loss $\eta(\mathbf{s}_k^*)$ to gradually decrease the search range for stable convergence. ϵ is a 6D vector of small numbers (10^{-3}) used to avoid degenerating PST. Figure 8 gives an illustration of randomized optimization. The final shape of the PST is a blend between the current step axis lengths $\hat{\mathbf{r}}_k$ and previous step axis lengths $\mathbf{r}_{k-1}$:

$$\mathbf{r}_k = \alpha\mathbf{r}_{k-1} + (1-\alpha)\hat{\mathbf{r}}_k, \tag{16}$$

where $\alpha = 0.1$. The scaling factor is computed with $\mathbf{r}_k$ and $\mathbf{r}_{k-1}$.

Gradient-based pose optimization. The GO phase minimizes the following loss over the input pixel batch set $\mathcal{B}$:

$$\mathcal{L}(\mathbf{R}_t, \mathbf{t}_t) = \sum_{b\in\mathcal{B}}\mathcal{L}_{\text{d2t}}^b(\mathbf{R}_t, \mathbf{t}_t) + \omega\mathcal{L}_{\text{rgb}}^b(\mathbf{R}_t, \mathbf{t}_t), \tag{17}$$

where $\mathcal{L}_*^b$ is the average loss over batch b and $\omega = 1$. We adopt the ADAM solver [Kingma and Ba 2014] with a learning rate of 10^{-2}.

Algorithm 1: Mechanism of multi-submap maintenance

Input : RGB-D sequences $\{I_t^c, I_t^d\}$ and corresponding pose $\mathbf{x}_t$
Output: Submaps M_j and corresponding keyframes $\{(I_i^c, I_i^d, \mathbf{x}_i) \in \Omega(M_j)\}$

$M_0 \leftarrow$ `CreateSubmap`$(I_0^c, I_0^d, \mathbf{x}_0)$; // see *submap allocation*
$\Omega(M_0) \leftarrow$ `InsertKeyFrame`$(I_0^c, I_0^d, \mathbf{x}_0, \Omega(M_0))$;
$t \leftarrow 1$;
$j \leftarrow 0$;
repeat
  if `CheckOutBound`$(\mathbf{x}_t, M_{0:j})$ **then**
    $j \leftarrow j + 1$;
    $M_j \leftarrow$ `CreateSubmap`$(\mathbf{x}_t)$;
    $\Omega(M_j) \leftarrow$ `InsertKeyFrame`$(I_t^c, I_t^d, \mathbf{x}_t, \Omega(M_j))$;
  foreach $M_j \in \{M\}$ **do** // see *keyframe selection*
    if `CheckKeyFrame`$(\mathbf{x}_t, M_j)$ **then**
      $\Omega(M_j) \leftarrow$ `InsertKeyFrame`$(I_t^c, I_t^d, \mathbf{x}_t, \Omega(M_j))$;
  $t \leftarrow t + 1$;
until *All frames are processed*;

3.4 Mapping of the Active Submap

Given the sequentially acquired RGB-D frames, the mapping process optimizes the network parameters of f_s^θ and c_s^λ via minimizing the TSDF truncation-region and free-space losses, along with the RGB rendering loss. In this subsection, we focus on the mapping of the active submap and leave the refinement of inactive ones for the next subsection. Algorithm 1 describes the mechanism of multi-submap maintenance.

Submap allocation. The subvolume governed by a neural submap is an axis-aligned bounding box enveloping the viewing frustums (the far clipping plane is set to 5m) of all keyframes. The subvolume of the active submap grows dynamically as new keyframes are being added. Whenever a new keyframe is selected, the subvolume is enlarged by a minimum expansion to enclose its viewing frustums. When any side of the subvolume reaches a predefined maximum length (set to 7m), the subvolume stops expanding along that dimension. When the overlap between the viewing frustum of a keyframe and the subvolume is less than 75% of the frustum (see `CheckOutBound()` in Algorithm 1), a new submap is allocated and set as active and that keyframe is selected as its first/anchor keyframe. The previous active submap then becomes inactive.

Submap initialization: When a new submap is created, we perform initialization using its first keyframe shared with the previous active submap for 500 epochs (found through experiments), to make sure that the MLP of the new submap is optimized sufficiently for a smooth transition of tracking across submaps (see `CreateSubmap()` in Algorithm 1).

Keyframe selection. The selection of keyframes is via measuring the information gain of a frame (see `CheckKeyFrame()` in Algorithm 1). Based on the depth-to-TSDF loss, we compute for each frame an information gain used for filtering those frames which does not induce much novel information. Given a frame, we compute the depth-to-TSDF loss for each pixel: $\psi_s(\mathbf{T}_s^{-1}(\mathbf{R}_t\mathbf{x}_p + \mathbf{t}_t))$. If the proportion of pixels having a small error (< 0.05) is lower than 65%, the frame is selected as a keyframe. These thresholds were found through experiments and are then kept fixed. To avoid selecting keyframes too frequently, we stipulate that the minimum spacing between two keyframes is 30 frames (see `InserKeyFrame()` in Algorithm 1).

Active submap optimization. The optimization of the active submap *at each frame* involves five different frames. First of all, the current frame, after being tracked, always participates in map optimization. The first/anchor keyframe of the active submap is also selected with its pose being fixed during optimization. Fixing the anchor pose avoids free drifting of the entire submap. Besides the above two frames, we randomly selected another three keyframes in between. If there are fewer than three keyframes in between, already selected frames are duplicated up to five. We found through experiments that using such five frames for optimization leads to a good balance between accuracy and efficiency. In summary, each frame participates in the optimization at least once, and more times if it is a keyframe. The neural submap and the poses of the five frames (except for the anchor pose) are jointly optimized, which is essentially a local bundle adjustment for the active submap. This local BA optimizes the following loss for 15 iterations with a learning rate of 10^{-2} for submap update and 10^{-3} for pose optimization:

$$\mathcal{L}(\theta, \lambda, \mathbf{R}_{i_1,\ldots,i_5}, \mathbf{t}_{i_1,\ldots,i_5}) = \sum_{b\in\mathcal{B}} \omega_{\text{rgb}}\mathcal{L}_{\text{rgb}}^b + \omega_{\text{fs}}\mathcal{L}_{\text{fs}}^b + \omega_{\text{tr}}\mathcal{L}_{\text{tr}}^b, \quad (18)$$

which sums up losses over the pixel batch set $\mathcal{B}$ for all the involved five frames. The weights are: $\omega_{\text{rgb}} = 1$, $\omega_{\text{fs}} = 10$, and $\omega_{\text{tr}} = 1000$.

3.5 Back-end Optimization and Loop Closure

We create two threads running in parallel, one for the tracking and mapping of the active submap and one for the refinement of the inactive ones. This can improve the global map quality while ensuring realtime frame rate of online reconstruction.

Optimization of inactive submaps. The inactive thread optimizes the inactive submaps sequentially and repeatedly. For the optimization of each inactive submap, we randomly select four keyframes belonging to the submap. The four keyframes, together with the first/anchor keyframe, are used to update the neural submap jointly. The poses of the four keyframes are also optimized with a small learning rate (10^{-3}). Such intra-submap local BA optimization is conducted for 10 iterations. The number was determined through experiments for a trade-off between accuracy and efficiency.

Handling pose jump at submap revisiting. When the camera moves into the subvolume of an inactive submap built previously, the inactive submap is re-activated. At this time, the overlapping keyframe, whose pose was just optimized in the last active submap, is now optimized again with the new active submap against its map built previously. Since the two maps may be misaligned due to drift, the

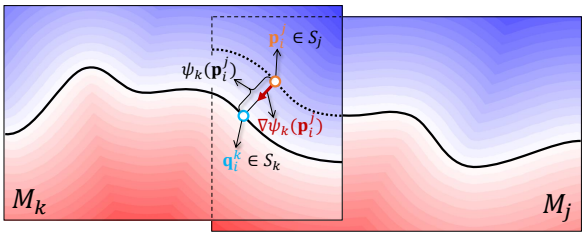

Fig. 9. Finding correspondence for a pair of adjacent submaps. Given a surface point $\mathbf{p}_i^j$ of one submap M_j, its correspondence is found by moving it along the gradient of *the other* submap M_k, denoted by $\nabla\psi_k(\mathbf{p}_i^j)$, for a distance of $\psi_k(\mathbf{p}_i^j)$. Submap transformations are omitted for brevity.

pose of the keyframe may jump across the two optimizations. To avoid this jump, we first perform a local BA of the new active submap using this overlapping keyframe (together with other keyframes of it) for 10 epochs, before starting tracking. This alignment of adjacent submaps makes the tracking transit smoothly.

Loop detection and closure. Since we concern about submap-level loop closure, loop detection is detected simply by checking whether the camera moves into the subvolume of an inactive submap. We detect only non-trivial loops involving at least four submaps. A more sophisticated scheme of loop detection can also be used.

To construct the optimization problem for loop closure, we first find for each pair of adjacent submaps (with subvolume overlap), M_j and M_k, a set of point correspondences denoted as C_{jk}. To do so, we first identify the overlapping region between M_j and M_k. In the overlapping region, we extract a set of surface points of M_j at the zero level set of ψ_j, denoted by $\mathcal{S}_j$. For each point $\mathbf{p}_i^j \in \mathcal{S}_j$, its correspondence on the surface of $\mathbf{q}_i^k \in \mathcal{S}_k$ can be found by first transforming it into the local coordinate frame of M_k and then moving it along the gradient of ψ_k for a distance of its TSDF value, similar to [Fioraio et al. 2015]:

$$\mathbf{q}_i^k = \mathbf{T}_k^{-1}\mathbf{T}_j\mathbf{p}_i^j - \psi_k(\mathbf{T}_k^{-1}\mathbf{T}_j\mathbf{p}_i^j)\nabla\psi_k(\mathbf{T}_k^{-1}\mathbf{T}_j\mathbf{p}_i^j), \tag{19}$$

where $\nabla\psi$ denotes the gradient of TSDF field which is defined only within the truncation region. Therefore, our method can only find correspondences lying in the truncation regions. This suffices for our method since the drift between two consecutive submaps is usually quite small. Between the loop-closing submaps (the first and the last), however, the drift can be very large, for which we employ the existing feature detection and matching techniques [Choi and Christensen 2012]. Given the correspondence $(\mathbf{p}_i^j, \mathbf{q}_i^k) \in C_{jk}$, we can formulate the following point-to-plane inter-submap pose constraint:

$$e_i^{jk} = \left(\mathbf{p}_i^j - \mathbf{T}_j^{-1}\mathbf{T}_k\mathbf{q}_i^k\right) \cdot \mathbf{n}_i^j, \tag{20}$$

where $\mathbf{n}_i^j = \nabla\psi_j(\mathbf{p}_i^j)$ is the normal of $\mathbf{p}_i^j$ in submap M_j. In case two consecutive submaps have too low overlap such that their correspondence set is too small to pin down their relative transformation, we simply use a pose-to-pose constraint based on the tracked motion between the two submaps, i.e., $\mathbf{M}_{s-1,s}$:

$$e^{s-1,s} = \log\left(\mathbf{M}_{s-1,s}\mathbf{T}_s^{-1}\mathbf{T}_{s-1}\right), \tag{21}$$

where $\log : SE(3) \to \mathfrak{se}(3)$ is the logarithmic map. Putting the two constraints together, we solve for all submap poses by optimizing:

$$\underset{\{\mathbf{T}_1,\ldots,\mathbf{T}_S\}}{\arg\min} \sum_j \sum_k \sum_i \|e_i^{jk}\|^2 + \sum_s \|e^{s-1,s}\| \tag{22}$$

The optimization is solved by Ceres [Agarwal et al. 2010] with the Levenberg-Marquardt method.

4 IMPLEMENTATION DETAILS

Parameter settings. For efficiency, all losses are computed with downsampled 384 pixels for both depth and RGB images following the sampling method of [Zhang et al. 2021] in which images are stripe downsampled into 16 rows by 24 columns, approximately $\frac{1}{30}$ of the original resolution which is 640×480. We provide a detailed illustration to explain the downsampling algorithm in the supplemental material. For each pixel, the sampling of 3D points on its ray is performed in two phases. First, we uniformly sample 20 points along the ray across the free space and the truncation region. We then sample additional 20 points uniformly within the truncation region. In the second phase, we sample 10 more points around the point having the smallest TSDF value. Therefore, we sample 50 points per ray in total. For the tracking of active submap, RO is performed for 10 iterations and GO for 10 epochs. The optimization of mapping also runs for 10 epochs in each update. For RO, 2048 particles are pre-sampled and evolved; other parameter settings follow [Zhang et al. 2020]. The batch size is 19, 200 (384 pixels $\times$ 50 points) for GO in tracking and 96, 000 (384 pixels $\times$ 50 points $\times$ 5 keyframes) for mapping. All these numbers were found through experiments.

GPU implementation. The most time-consuming operation in a neural SLAM/reconstruction is the MLP training and inference based on all sample points in a batch. Furthermore, the fitness evaluation and filtering of particles in RO is also costly. To accelerate the computation, we make use of the Graph Execution mode of Tensorflow and compile all the core computations above into computational graphs. The computations are "traced" only once and can be called repeatedly and run efficiently in the GPU.

The optimization of the active and inactive submaps runs in separate processes in the GPU concurrently. Most of the time, the two processes work independently and they communicate with each other only when a new submap is created and the switching of active submaps happens. When the inactive submap adjacent to the active is being optimized, the two processes may update their overlapping keyframes jointly. To avoid "dirty write" of the overlapping keyframe residing in the shared memory, we set a write lock to ensure that it is optimized alternately by the active and inactive processes.

5 RESULTS AND EVALUATIONS

We provide both quantitative and qualitative results in this section. *Live demos can be found in the accompanying video.*

5.1 Datasets and Metrics

Datasets. We evaluated our method on three diverse public distastes, including `Replica` [Straub et al. 2019], `ScanNet` [Dai et al. 2017a], and `FastCaMo` [Zhang et al. 2021]. `Replica` is a synthetic dataset containing rendered (with noise added) RGB-D sequences. `ScanNet` is a real dataset of captured RGB-D sequences. `FastCaMo` is a challenging dataset of sequences with fast camera motions. The dataset is composed of a synthetic part (`FastCaMo-Synth`) and a real captured (`FastCaMo-Real`) part. `FastCaMo-Synth` is built with 10 `Replica` scenes. The RGB-D sequences have the linear speed of camera motion varying in $[1, 4]$ m/s and the angular speed in $[0.9, 2.2]$ rad/s, with synthesized motion blur effect for RGB images and depth noise for depth maps. `FastCaMo-Real` contains 24 real RGB-D sequences captured for 12 scenes with fast camera motions (linear speed up to 5.47m/s). For each sequence, a full and dense reconstruction scanned with a laser scanner is provided as ground truth for evaluating reconstruction accuracy and completeness.

To better evaluate the scalability of RGB-D reconstruction methods, we contribute a new real-world dataset of RGB-D sequences capturing six large-scale indoor scenes (with area up to 200m^2), named FastCaMo-Large. The sequences were captured using an Azure Kinect DK under fast camera motions, and the individual size of each scene can be found in the supplemental material.

Evaluation metrics. When the ground-truth trajectory is available, we measure the camera tracking quality based on the Absolute Trajectory Error (ATE) [Sturm et al. 2012]. To estimate ATE, the trajectory to be evaluated is first rigidly aligned to the ground truth. ATE is then estimated as the mean of pose differences of all frames. We also measure the per-frame pose accuracy based on Translation Error (TE). In addition, we use Relative Pose Error (RPE) to evaluate the relative pose differences over a fixed time interval between the estimated and the ground-truth trajectories. RPE is suited for evaluating local trajectory accuracy. TE and RPE do not require a pre-alignment of the estimated and ground-truth trajectories. As long as the trajectories start from the same initial pose of the very first frame, they can always be estimated for the following frames in the reference system of the first frame. For a fair comparison, we conducted multiple runs of our method and other open-sourced neural-based methods using different random seeds. Specifically, we executed each method five times and recorded the average result as the final outcome. This approach helps mitigate the impact of random variations and provides a more reliable and robust evaluation of performance.

To evaluate the reconstruction quality, we measure the reconstruction completeness and accuracy based on ground-truth surface reconstruction. Following [Zhang et al. 2021], we measure completeness as the percentage of the inlier portion of the ground-truth surface and accuracy by RMS error of all reconstructed points against the ground-truth surface.

5.2 Ablation Studies

We conduct a series of ablation studies to verify the necessity of the various key design choices of our method:

Table 1. Ablation study of seven design choices on tracking accuracy (ATE in cm) over 6 sequences of `ScanNet` (top rows) and 4 of `FastCaMo` (bottom rows). The best results for each sequence are highlighted in **blue** color. '–' indicates that the tracking failed for the corresponding method.

Method	No C	No RO	No GO	No U	No SI	No SR	No LC	Full
`Scene0000`	11.5	12.8	19.9	12.1	–	17.5	27.5	**7.9**
`Scene0106`	10.8	13.9	15.3	11.3	17.3	20.9	35.5	**9.7**
`scene0169`	12.3	15.1	36.5	13.5	–	–	–	**9.7**
`scene0181`	14.9	17.5	29.6	14.7	–	18.4	15.1	**14.2**
`scene0207`	10.2	8.9	19.5	8.2	20.8	19.5	–	**7.8**
`scene0011`	9.1	14.2	18.6	7.9	–	10.1	15.4	**7.5**
`Apartment_1`	10.5	27.6	10.2	11.0	–	–	13.9	**7.0**
`Hotel_0`	7.3	14.3	6.2	6.9	–	9.5	10.3	**4.8**
`Office_0`	6.9	19.1	7.6	6.8	–	6.8	6.6	**3.6**
`Room_0`	8.1	40.6	8.9	7.2	–	20.1	28.0	**4.8**

- **No Classification (No C)**: TSDF prediction is implemented with regression as in existing methods.
- **No RO**: The tracking optimization is performed by GO only as in existing methods.
- **No GO**: There is no GO-based pose refinement after RO.
- **No Uncertainty (No U)**: The fitness evaluation (Eq. (13)) in RO is not weighted by uncertainty.
- **No Submap Initialization (No SI)**: No initialization is performed for newly allocated submaps.
- **No Smooth Revisit (No SR)**: No handling of pose jump for smooth transition is done at submap revisiting.
- **No Loop Closure (No LC)**: No loop closure optimization of submaps is conducted.

The evaluation is conducted on 6 sequences from `ScanNet` and 4 from `FastCaMo-Synth`. Table 1 compares the tracking accuracy (ATE) of our method and the various baselines. It can be seen that "No SI" and "No SR" cause the most accuracy drop (and even failures) among all baselines, suggesting their importance to stable tracking. This also suggests that the handling of smooth transition between two submaps especially when revisiting an inactive submap is critical to the overall tracking quality. The combination of RO and GO produces better accuracy than either one of them. In particular, RO's effect is more prominent for fast-camera-motion sequences. For sequences with ordinary camera motions, GO's contribution seems more significant. Classification is also influential in tracking accuracy as implied by the results of "No C". Although relatively less significant, "No U" does affect the final tracking accuracy, hinting that the uncertainty estimated by the classification network output is indeed meaningful. The effect of "No LC" manifests the necessity of our submap-level loop closure for fast-camera-motion sequences. Note, however, that the local BAs of inactive submaps are also turned off in "No LC". This is because the local BAs rely on globally consistent frame poses provided by the global optimization of loop closure.

In Figure 10, we show plots of tracking accuracy over iteration steps for RO only, GO only and our RO+GO. The tracking accuracy is measured by per-frame translation error averaged over all frames of `scene0207` of `ScanNet`. The full ranges of TE variation are depicted

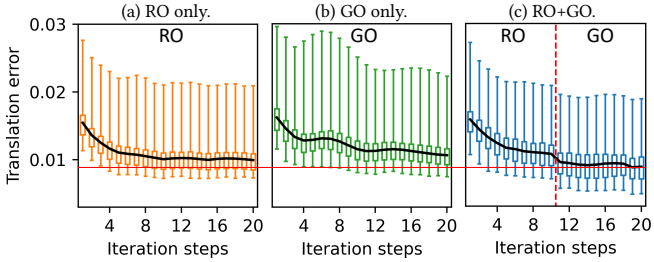

Fig. 10. Plots of per-frame tracking accuracy (TE) over iteration steps for (a) RO only, (b) GO only, and (c) RO and then GO (our method). Each plot shows TE averaged over all frames of a sequence (black curve) and ranges of variation (colored bars). In (c), the dashed red line indicates the switching point from RO to GO. The horizontal solid line across the three plots is drawn for a clear comparison of the converging error by the three methods.

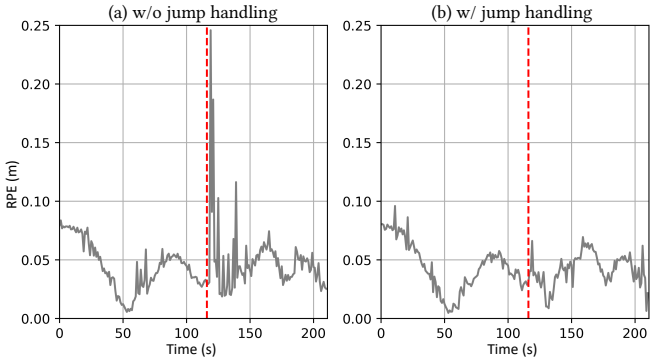

Fig. 11. Plots of Relative Pose Error (RPE) over time with and without jump handling at submap revisiting. The red dashed line indicates the time at which an inactive submap is revisited and a spike is observed when no jump handling is conducted.

by bars and the ranges of medium half by boxes. Although RO finds a good solution efficiently and converges faster than GO, GO can be used to refine the solution found by RO, leading to a better solution. Indeed, RO excels at finding a good initial solution by getting rid of local minima due to its randomized nature. GO is good at finding a better optimum in the vicinity of an initial guess. Combining the two makes our method enjoy the advantages of both worlds.

To demonstrate the effect of pose jump handling when revisiting an inactive submap, Figure 11 plots of the tracking accuracy (RPE) over time with and without jump handling. When no jump handling is conducted, a spike of the RPE curve is observed, which also affect adversely the pose tracking of the following frames (see the higher RPEs of the following time stamps).

5.3 Quantitative Comparisons

We quantitatively evaluate our method against several state-of-the-art methods for both ordinary and fast-motion sequences.

Comparison on `Replica`. Table 2 compares ATE RMSE on 8 sequences of `Replica` between our method and three state-of-the-art neural online RGB-D reconstruction methods (iMAP [Sucar et al. 2021], NICE-SLAM [Zhu et al. 2022] and Vox-Fusion [Yang et al. 2022]). On these relatively easy sequences, our method achieves comparable accuracy to Vox-Fusion with much less running time and memory footprint (see Section 5.5). We also evaluate the reconstruction quality of our method in comparison to VoxFusion and NICE-SLAM. The results demonstrate that our method achieves comparable reconstruction quality to VoxFusion, while outperforming NICE-SLAM. *The results can be found in the supplemental material.*

Table 2. Comparing tracking accuracy (ATE RMSE in cm) on 8 RGB-D sequences of **`Replica`**. The best and the second best results for each sequence are highlighted in **blue** and **green** colors, respectively.

Sequence	iMAP	NICE-SLAM	Vox-Fusion	MIPS-Fusion
`Room-0`	70.1	1.7	**0.3**	**1.1**
`Room-1`	4.5	2.0	**1.3**	**1.2**
`Room-2`	2.2	1.6	**0.5**	**1.1**
`Office-0`	2.3	**1.0**	**0.7**	**0.7**
`Office-1`	1.7	**0.9**	1.1	**0.8**
`Office-2`	4.9	1.4	**0.5**	**1.3**
`Office-3`	58.4	4.0	**0.3**	**2.2**
`Office-4`	2.6	3.1	**0.6**	**1.1**

Table 3. Comparing tracking accuracy (ATE RMSE in cm) on 8 RGB-D sequences of **`ScanNet`**. The best and the second best results for each sequence are highlighted in **blue** and **green** colors, respectively. '–' indicates that the tracking failed for the corresponding method.

Sequence	iMAP	NICE-SLAM	Vox-Fusion	MIPS-Fusion
`scene0000`	––	**12.1**	17.6	**7.9**
`scene0106`	17.5	**9.2**	**8.8**	9.7
`scene0169`	––	**11.2**	20.0	**9.7**
`scene0181`	32.1	**13.9**	19.0	**14.2**
`scene0207`	11.9	**6.2**	**7.5**	7.8
`scene0011`	18.4	8.2	**7.4**	**7.5**
`scene0024`	––	––	**31.0**	**7.8**
`scene0059`	––	**12.8**	35.5	**10.7**

Comparison on `ScanNet`. Table 3 reports the comparison on 8 sequences of `ScanNet` (index "0" for each scene). These sequences include not only those tested by the alternatives [Yang et al. 2022; Zhu et al. 2022] in their papers but also new ones which we believe are more challenging. Our method achieves comparable accuracy to the best-performing method. Sequences such as `scene0000`, `scene0181`, `scene0011` and `scene0059` contain complex camera trajectories (with multiple loops), on which our method demonstrates good results due to the back-end optimization. The `scene0024` sequence is the most challenging one which contains large open areas and lacks geometric details. Our method works the best on this sequence due to the robust tracking method employed.

Comparison on `FastCaMo-Synth`. Table 4 reports a comparison on 10 sequences of `FastCaMo-Synth`. All these sequences were recorded with fast camera motions. Among the methods, ours is the only one that can reconstruct all the sequences with decent accuracy. `Office_3` is the most challenging one due to the large camera rotations involved, on which our method achieves an ATE of 17.4cm. Our method is the first, to our knowledge, that realizes

Table 4. Comparing tracking accuracy (ATE RMSE in cm) on 10 fast-camera-motion RGB-D sequences of **FastCaMo-Synth** (noise-free). The best and the second best results for each sequence are highlighted in **blue** and **green** colors, respectively. '–' indicates that the tracking was failed for the corresponding method.

Sequence	iMAP	NICE-SLAM	Vox-Fusion	MIPS-Fusion
Apartment_1	–	–	**9.1**	**7.0**
Apartment_2	–	–	**4.1**	**1.5**
Frl_apartment_2	–	–	**7.2**	**1.9**
Hotel_0	20.3	**4.2**	5.0	**4.8**
Office_0	39.2	8.4	**4.8**	**3.6**
Office_1	–	13.7	**4.6**	**5.6**
Office_2	–	–	**10.2**	**7.4**
Office_3	–	**14.3**	–	**17.4**
Room_0	–	–	**8.2**	**4.4**
Room_0	–	29.7	**5.8**	**5.1**

Table 5. Comparing reconstruction quality (completeness and accuracy) on 10 fast-camera-motion RGB-D sequences of **FastCaMo-Synth** (noise-free). The best results for each sequence are highlighted in **blue**. '–' indicates that the tracking failed for the corresponding method.

	NICE-SLAM		Vox-Fusion		Ours	
	Compl.(↑)	Acc.(↓)	Compl.(↑)	Acc.(↓)	Compl.(↑)	Acc.(↓)
Apartment_1	-	-	63.4	4.8	**73.9**	**4.2**
Apartment_2	-	-	**93.1**	**2.4**	64.8	4.9
Frl_aparment_2	-	-	62.3	5.1	**78.0**	**4.3**
Hotel_0	84.4	3.9	66.0	4.6	**88.8**	**3.4**
Office_0	92.9	**2.9**	44.8	6.5	**94.2**	**2.9**
Office_1	53.8	5.7	**72.3**	5.9	67.9	**4.3**
Office_2	-	-	51.6	6.3	**62.7**	**4.8**
Office_3	**63.4**	**4.9**	-	-	44.4	6.4
Room_0	-	-	37.6	7.0	**65.6**	**4.8**
Room_1	65.3	4.8	37.7	7.0	**83.4**	**3.4**

online neural RGB-D reconstruction under fast camera motions. Besides tracking accuracy, Table 5 compares the reconstruction quality. Our method exhibits the best completion and accuracy for most of the sequences. We also compare our method with two traditional RGB-D reconstruction methods, i.e., BunldeFusion [Dai et al. 2017b] and ElasticFusion [Whelan et al. 2015]. The results are reported in Table 6. Generally speaking, the performance of the current neural SLAM approaches is still not comparable to traditional ones. On the fast-camera-motion sequences, however, our method performs better than the two traditional methods, thanks to the integration of gradient-based and randomized optimizations in the neural setting and the efficient learning of submaps.

Comparison on TUM RGB-D. Table 7 reports a comparison of tracking accuracy on three TUM RGB-D sequences with slow camera motions and small scene scales. Traditional methods (rows 5-8) are generally more accurate than neural-based ones (rows 1-4) with dedicated designs such as feature tracking and depth noise modeling. Our method achieves comparable performance to NICE-SLAM. The advantage of our method is more prominent for fast-motion and large-scale sequences.

Table 6. Comparing tracking accuracy (ATE RMSE in cm) on 10 fast-camera-motion RGB-D sequences of **FastCaMo-Synth** (with noise). The best results for each sequence are highlighted in **blue** color. '–' indicates that the tracking was failed for the corresponding method.

Sequence	ElasticFusion	BundleFusion	MIPS-Fusion
Apartment_1	40.9	**4.6**	6.6
Apartment_2	40.7	**2.2**	3.1
Frl_apartment_2	43.8	83.6	**2.6**
Hotel_0	22.3	**2.7**	5.2
Office_0	**2.3**	17.3	7.6
Office_1	-	-	**17.4**
Office_2	-	-	**24.9**
Office_3	43.8	-	**6.0**
Room_0	-	8.2	**4.4**
Room_0	31.0	5.8	**3.6**

Table 7. Comparing tracking accuracy (ATE in cm) on three RGB-D sequences of **TUM RGB-D**. The best results for each sequence are highlighted in **blue**.

Method	fr1/desk	fr2/xyz	fr3/office
iMap	4.9cm	2.0cm	5.8cm
DI-Fusion	4.4cm	2.4cm	15.6cm
NICE-SLAM	**2.7cm**	1.8cm	**3.0cm**
MIPSFusion	3.0cm	**1.4cm**	4.6cm
BAD-SLAM	2.3cm	2.2cm	2.3cm
Kintinuous	2.0cm	1.1cm	1.7cm
ORB-SLAM2	1.6cm	**0.4cm**	1.0cm
[Cao et al. 2018]	1.5cm	0.6cm	0.9cm
HRBF-Fusion	**1.4cm**	0.5cm	**0.7cm**

5.4 Qualitative Results

Visual comparison of neural rendering. We provide the results of the neural rendering in Figure 12 on sequences from FastCaMo-Large, ScanNet, and FastCaMo-Synth. Our method achieves higher rendering quality under challenging lighting conditions in real-world environments (rows 1-2). For the fast-motion sequences (row 4), NICE-SLAM finds difficulty in learning geometry and appearance within a short time interval, leading to suboptimal results. In contrast, our method consistently produces high-quality rendering outputs throughout the sequences (row 4). On the quantitative side, our method outperforms NICE-SLAM by 51.3% in PSNR on FastCaMo-Large. *Please refer to the supplemental material.*

Visual comparison of reconstruction. We compare the reconstruction quality of our method with several mainstream methods including both neural-based [Yang et al. 2022; Zhu et al. 2022] and traditional-based ones [Dai et al. 2017b; Whelan et al. 2015]. The evaluation was performed on the FastCaMo-Real and FastCaMo-Synth datasets and the results are shown in Figure 13. Note that our method achieves better reconstruction quality with fewer artifacts and more complete geometry. This is also reflected by the better trajectory conformance of our method against the ground truths.

In Figure 16, we show a gallery of reconstruction results on several large-scale indoor scenes of the FastCaMo-Large dataset. Here, we

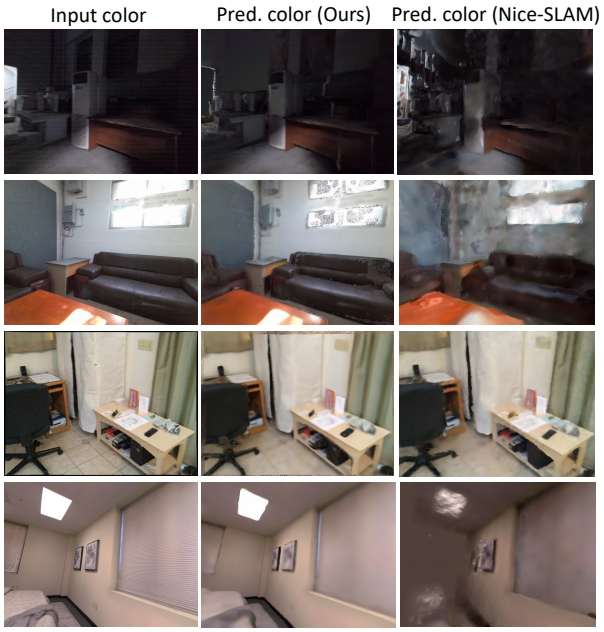

Fig. 12. Rendering results of ours and NICE-SLAM on the sequences of FastCaMo-Large (rows 1-2), ScanNet (row 3), and FastCaMo-Synth (row 4).

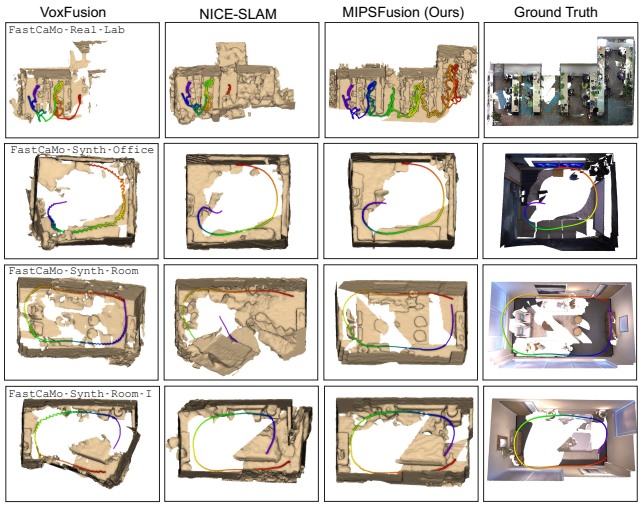

Fig. 13. Comparision of 3D reconstruction on four room-scale indoor scene sequences of FastCaMo-Real and FastCaMo-Synth.

observe that our method exhibits significant advantages in terms of both completion and quality, especially in scenarios with large loops (columns 1 and 4). Our method attains higher quality thanks to 1) the RO-based pose optimization leading to robustness and 2) the classification-based design making the network lightweight and fast to learn. In most sequences, our method preserves geometric details better (see the zoom-in views). We attribute this to our distributed

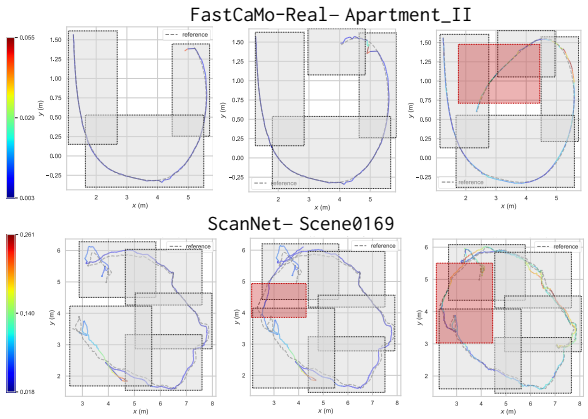

Fig. 14. Visualization of loop closure on two sequences. Camera trajectories are visualized with solid curves and ground-truth reference with dashed curves. The tracking error is color-coded. Submaps are shown as grey boxes. The closing-loop submaps are shaded in red.

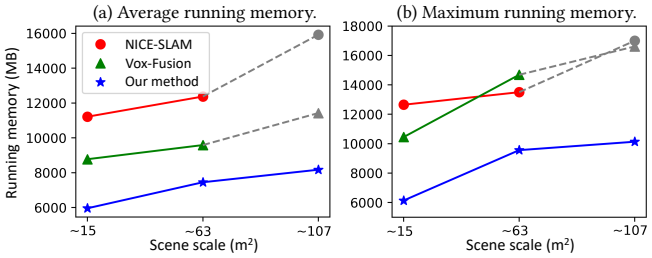

Fig. 15. Comparing the average and maximum running memory cost of NICE-SLAM, Vox-Fusion and our method for increasing scene scales. Our method consumes the least running memory and the cost increases slowly with scene scale. The grey dots mean that the two alternative methods failed on those sequences.

neural representation where each submap takes charge of only a local region and hence more scene details can be memorized.

Visualization of loop closure. In Figure 14, we present a visualization of loop closure with two examples. For the Apartment_II sequence of FastCaMo-Real (top row), the trajectory starts to drift in the middle image. The trajectory is then corrected when a loop is formed and closed by the final submap shown in the right image. The same goes for the sequence shown in the bottom row. After loop closure, the overall tracking error is minimized. Since each submap has been well optimized with local BAs in both active and inactive processes, our method only needs to perform submap-level registrations to close a loop. Thanks to the mechanism of the local-frame definition of a neural submap and pair-wise alignment of adjacent submaps, we can achieve submap-level loop closure straightforwardly and efficiently. This flexibility is a clear advantage over the single-implicit-map [Sucar et al. 2021] and the feature-grid-based approaches [Yang et al. 2022; Zhu et al. 2022] based on which it is hard to realize global map update caused by loop closure. Examples for with and without non-trivial loops can be found in the supplemental material. We observe that our method consistently exhibits robustness across various types of loops.

Table 8. Running time of various algorithmic components of our method profiled on a per-iteration basis.

Sequence	Time
Tracking (RO)	4 ms
Tracking (GO)	8 ms
Mapping (1 frame)	9 ms
Mapping (5 frames)	48 ms
Create new submap	4 ms

5.5 Runtime and Memory Analysis

Table 8 reports the average runtime for one iteration of the various algorithmic components of our method tested on the `scene0000` sequence of `ScanNet`. The time was measured on a workstation with an Intel® Core™ i9-1290K CPU @ 3.9GHz × 16 with 32GB RAM and an Nvidia GeForce RTX 3090Ti GPU with 24GB memory. In terms of total runtime for full reconstruction of the tested sequences, our method is 4× faster than NICE-SLAM and 3× faster than Vox-Fusion.

In Figure 15, we compare the average and maximum running memory cost of NICE-SLAM, Vox-Fusion, and our method for increasing scene scales. Our method leads to the smallest memory footprint and the cost for the three scenes. In fact, the main storage cost of our method is the sample batches for the optimization of the active submap and one of the inactive submaps being refined. Such memory cost does not increase drastically with growing scene scales. Our method does not require extra memory for storing feature grids as in the alternative methods.

6 DISCUSSION AND CONCLUSIONS

With our work, we wish to bring it to the community's attention the potential of grid-free, purely neural representation for scalable and robust online RGB-D reconstruction. Our main design philosophy is two-fold. First, we adopt a flexible divide-and-conquer mapping scheme. Each submap, representing a subscene compactly, can be learned efficiently and refined distributively. The high-quality submaps together constitute a decent full reconstruction of the whole scene with submap-level global pose optimization. We believe that this mapping scheme has accomplished a good trade-off between flexibility and scalability. Second, we propose a hybrid tracking scheme in which randomized optimization is made possible based on two new designs on tracking loss. This enables efficient and robust tracking even under fast camera motions.

Limitations. Our method has several limitations. *Firstly*, our tracking and mapping depends heavily on depth. When the depth input is of low quality, the reconstruction quality is unsatisfactory. *Secondly*, our loop detection is still simplistic. A loop may happen when the camera looks at a previously visited point without actually entering any inactive submap, which will be missed by our detection. *Thirdly*, when aligning two submaps having significant misalignment, robust feature detection and matching is still needed. *Finally*, our method does not handle view-dependent appearance such as specular since it does not model view directions in the neural radiance field as most existing works [Sucar et al. 2021; Zhu et al. 2022].

Future works. We expect that our work will inspire a rich set of future directions:

- How to achieve a smarter submap allocation to ensure a better match between learning capacity and scene complexity? This may need a method for probing the representation forgetting [Davari et al. 2022] of a neural submap against the acquired data.
- How to realize end-to-end trainable loop detection and closure in one framework based on our MIPS representation? For example, it might be interesting to investigate an efficient neural remapping of submaps during loop closure based on a fast $SE(3)$-transformation of neural implicit maps [Yuan and Nüchter 2022].
- How to integrate the geometric and the photometric losses in a more principled way? Specifically, how to bridge and switch smoothly between the two is worth of investigating.
- How to fuse multi-modal input, e.g., inertial measurement, into online neural reconstruction using a similar technique to [Zhang et al. 2022]?
- It seems a natural application to use neural submap representation for distributive and collaborative reconstruction of large scenes with a collection of robots [Dong et al. 2019].
- Another promising and interesting direction is to enhance neural submap representation for semantic scene reconstruction [Vora et al. 2021; Zhang et al. 2020].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments. We are grateful to Qi Wu for the fruitful discussions. This work was supported in part by the National Key Research and Development Program of China (2018AAA0102200), NSFC (62325211, 62132021).

REFERENCES

S. Agarwal, K. Mierle, and Others. 2010. Ceres Solver. http://ceressolver.org.

Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. 2022. Neural RGB-D surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6290–6301.

Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. 2018. CodeSLAM: learning a compact, optimisable representation for dense visual SLAM. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2560–2568.

Michael Bosse, Paul Newman, John Leonard, Martin Soika, Wendelin Feiten, and Seth Teller. 2003. An atlas framework for scalable mapping. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, Vol. 2. IEEE, 1899–1906.

Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. 2013. Real-time camera tracking and 3D reconstruction using signed distance functions. In *Robotics: Science and Systems*, Vol. 2. 2.

Yan-Pei Cao, Leif P. Kobbelt, and Shimin Hu. 2018. Real-time High-accuracy Three-Dimensional Reconstruction with Consumer RGB-D Cameras. *ACM Transactions on Graphics (TOG)* 37 (2018), 1 – 16. https://api.semanticscholar.org/CorpusID:52306210

Jiawen Chen, Dennis Bautembach, and Shahram Izadi. 2013. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 4 (2013), 1–16.

Changhyun Choi and Henrik I Christensen. 2012. Robust 3D visual tracking using particle filtering on the special Euclidean group: A combined approach of keypoint and edge features. *The International Journal of Robotics Research* 31, 4 (2012), 498–519.

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. 2022. Orbeez-SLAM: A Real-time Monocular Visual SLAM with ORB Features and NeRF-realized Mapping. *arXiv preprint arXiv:2209.13274* (2022).

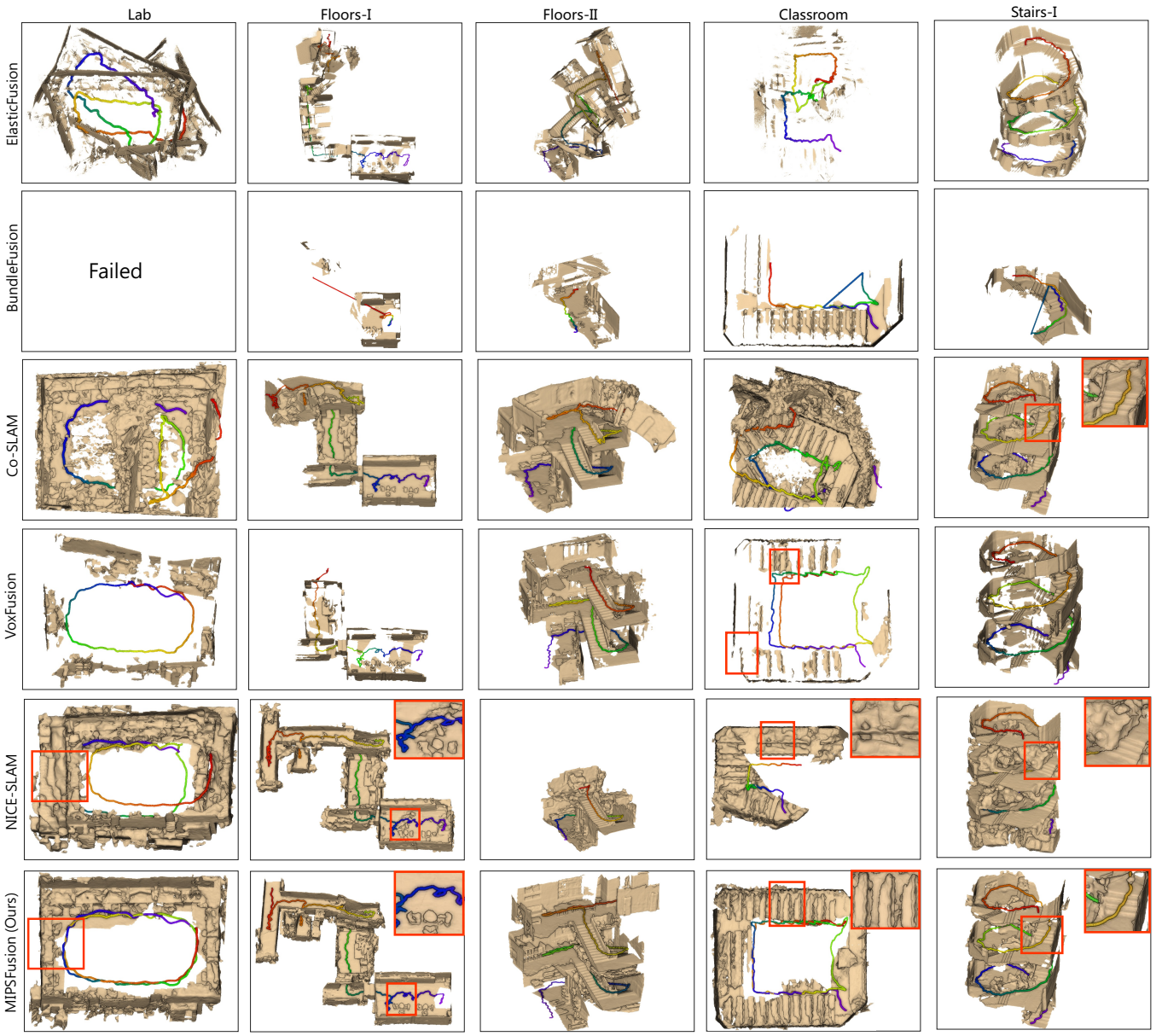

Fig. 16. Comparison of 3D reconstruction results by NICE-SLAM, Vox-Fusion, Co-SLAM, BundleFusion, ElasticFusion, and our MIPS-Fusion over five scenes from the FastCaMo-Large datasets. Our method achieves better accuracy and completeness compared to the alternatives.

Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *Proc. of SIGGRAPH*. 303–312.

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

Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. 2017b. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Reintegration. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 24.

MohammadReza Davari, Nader Asadi, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. 2022. Probing representation forgetting in supervised and unsupervised continual learning. In *Proc. CVPR*. 16712–16721.

Siyan Dong, Kai Xu, Qiang Zhou, Andrea Tagliasacchi, Shiqing Xin, Matthias Nießner, and Baoquan Chen. 2019. Multi-robot collaborative dense scene reconstruction. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–16.

Nicola Fioraio, Jonathan Taylor, Andrew Fitzgibbon, Luigi Di Stefano, and Shahram Izadi. 2015. Large-scale and drift-free surface reconstruction using online subvolume registration. In *Proc. CVPR*. 4475–4483.

Peter Henry, Dieter Fox, Achintya Bhowmik, and Rajiv Mongia. 2013. Patch volumes: Segmentation-based consistent mapping with rgb-d cameras. In *2013 International Conference on 3D Vision-3DV 2013*. IEEE, 398–405.

Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. 2014. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Experimental robotics*. Springer, 477–491.

Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. 2021. Di-fusion: Online implicit 3d reconstruction with deep priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8932–8941.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time 3D Reconstruction and

Interaction Using a Moving Depth Camera. In *UIST*. 559–568.

Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. 2023. ESLAM: Efficient Dense SLAM System Based on Hybrid Representation of Signed Distance Fields. In *Proc. CVPR*.

Olaf Kähler, Victor A Prisacariu, and David W Murray. 2016. Real-time large-scale dense 3D reconstruction with loop closure. In *European Conference on Computer Vision*. Springer, 500–516.

O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S Torr, and D. W. Murray. 2015. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device. *IEEE Trans. Vis. & Computer Graphics (ISMAR)* 22, 11 (2015).

Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. 2013. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision-3DV 2013*. IEEE, 1–8.

Christian Kerl, Jürgen Sturm, and Daniel Cremers. 2013. Dense visual SLAM for RGB-D cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2100–2106.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Lukas Koestler, Nan Yang, Niclas Zeller, and Daniel Cremers. 2022. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In *Conference on Robot Learning*. PMLR, 34–45.

Hai Li, Xingrui Yang, Hongjia Zhai, Yuqian Liu, Hujun Bao, and Guofeng Zhang. 2022b. Vox-Surf: Voxel-Based Implicit Surface Representation. *IEEE Transactions on Visualization and Computer Graphics* (2022).

Kejie Li, Yansong Tang, Victor Adrian Prisacariu, and Philip HS Torr. 2022a. BNV-Fusion: Dense 3D Reconstruction using Bi-level Neural Volume Fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6166–6175.

Stefan Lionar, Lukas Schmid, Cesar Cadena, Roland Siegwart, and Andrei Cramariuc. 2021. Neuralblox: Real-time neural representation fusion for robust volumetric mapping. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 1279–1289.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural sparse voxel fields. *Advances in Neural Information Processing Systems* 33 (2020), 15651–15663.

Shengjun Liu, Charlie CL Wang, Guido Brunnett, and Jun Wang. 2016. A closed-form formulation of HRBF-based surface reconstruction by approximate solution. *Computer-Aided Design* 78 (2016), 147–157.

Nate Merrill and Guoquan Huang. 2018. Lightweight unsupervised deep loop closure. *arXiv preprint arXiv:1805.07703* (2018).

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Alexander Millane, Zachary Taylor, Helen Oleynikova, Juan Nieto, Roland Siegwart, and César Cadena. 2018. C-blox: A scalable and consistent tsdf-based dense mapping approach. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 995–1002.

Yuhang Ming, Weicai Ye, and Andrew Calway. 2022. iDF-SLAM: End-to-End RGB-D SLAM with Neural Implicit Mapping and Deep Feature Tracking. *arXiv preprint arXiv:2209.07919* (2022).

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 1–15.

Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. 2011a. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*. 127–136.

Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. 2011b. DTAM: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*. IEEE, 2320–2327.

M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. 2013. Real-time 3D Reconstruction at Scale using Voxel Hashing. *ACM Trans. on Graph. (SIGGRAPH Asia)* 32, 6 (2013), 169.

Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. 2022. iSDF: Real-Time Neural Signed Distance Fields for Robot Perception. *arXiv preprint arXiv:2204.02296* (2022).

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional occupancy networks. In *European Conference on Computer Vision*. Springer, 523–540.

Vivek Pradeep, Christoph Rhemann, Shahram Izadi, Christopher Zach, Michael Bleyer, and Steven Bathiche. 2013. MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 83–88.

Antoni Rosinol, John J Leonard, and Luca Carlone. 2022. NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields. *arXiv preprint arXiv:2210.13641* (2022).

Thomas Schops, Torsten Sattler, and Marc Pollefeys. 2019. BAD SLAM: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 134–144.

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

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 573–580.

Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. 2021. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6229–6238.

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

Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. 2017. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications* 9, 1 (2017), 1–11.

Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. 2022. Block-nerf: Scalable large scene neural view synthesis. In *Proc. CVPR*. 8248–8258.

Chengzhou Tang and Ping Tan. 2018. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807* (2018).

Zachary Teed and Jia Deng. 2021. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in Neural Information Processing Systems* 34 (2021), 16558–16569.

Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. 2021. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv preprint arXiv:2111.13260* (2021).

Hengyi Wang, Jingwen Wang, and Lourdes Agapito. 2023. Co-SLAM: Joint Coordinate and Sparse Parametric Encodings for Neural Real-Time SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13293–13302.

Kaixuan Wang, Fei Gao, and Shaojie Shen. 2019. Real-time scalable dense surfel mapping. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 6919–6925.

Silvan Weder, Johannes Schonberger, Marc Pollefeys, and Martin R Oswald. 2020. Routedfusion: Learning real-time depth map fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4887–4897.

Silvan Weder, Johannes L Schonberger, Marc Pollefeys, and Martin R Oswald. 2021. NeuralFusion: Online depth fusion in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3162–3172.

Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. 2012. Kintinuous: Spatially Extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*.

Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. 2015. ElasticFusion: Dense SLAM without a pose graph. In *Proc. Robotics: Science and Systems*.

Thomas Whelan, Renato F. Salas-Moreno, Ben Glocker, Andrew J. Davison, and Stefan Leutenegger. 2016. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research* 35 (2016), 1697 – 1716. https://api.semanticscholar.org/CorpusID:21124365

Yabin Xu, Liangliang Nan, Laishui Zhou, Jun Wang, and Charlie C.L. Wang. 2022. HRBF-Fusion: Accurate 3D Reconstruction from RGB-D Data Using On-the-fly Implicits. *ACM Transactions on Graphics (TOG)* 41 (2022), 1 – 19. https://api.semanticscholar.org/CorpusID:246608194

Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. 2022. Vox-Fusion: Dense Tracking and Mapping with Voxel-based Neural Implicit Representation. *arXiv preprint arXiv:2210.15858* (2022).

Yijun Yuan and Andreas Nüchter. 2022. An algorithm for the SE (3)-transformation on neural implicit maps for remapping functions. *IEEE Robotics and Automation Letters* 7, 3 (2022), 7763–7770.

Jiazhao Zhang, Yijie Tang, He Wang, and Kai Xu. 2022. ASRO-DIO: Active Subspace Random Optimization Based Depth Inertial Odometry. *IEEE Transactions on Robotics* (2022).

Jiazhao Zhang, Chenyang Zhu, Lintao Zheng, and Kai Xu. 2020. Fusion-aware point convolution for online semantic 3d scene segmentation. In *Proc. CVPR*. 4534–4543.

Jiazhao Zhang, Chenyang Zhu, Lintao Zheng, and Kai Xu. 2021. ROSEFusion: random optimization for online dense reconstruction under fast camera motion. *ACM Trans. on Graph. (SIGGRAPH)* 40, 4 (2021), 1–17.

Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. 2022. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12786–12796.