Neural Graph Map: Dense Mapping with Efficient Loop Closure Integration

Leonard Bruns[1] Jun Zhang[2] Patric Jensfelt[1]
[1]KTH Royal Institute of Technology, Stockholm, Sweden
[2]TU Graz, Graz, Austria
{leonardb, patric}@kth.se, jun.zhang@tugraz.at

Abstract

Neural field-based SLAM methods typically employ a single, monolithic field as their scene representation. This prevents efficient incorporation of loop closure constraints and limits scalability. To address these shortcomings, we propose a novel RGB-D neural mapping framework in which the scene is represented by a collection of lightweight neural fields which are dynamically anchored to the pose graph of a sparse visual SLAM system. Our approach shows the ability to integrate large-scale loop closures, while requiring only minimal reintegration. Furthermore, we verify the scalability of our approach by demonstrating successful building-scale mapping taking multiple loop closures into account during the optimization, and show that our method outperforms existing state-of-the-art approaches on large scenes in terms of quality and runtime. Our code is available open-source at https://github.com/KTH-RPL/neural_graph_map.

1. Introduction

Simultaneous localization and mapping (SLAM) using cameras, often referred to as visual SLAM, has been a long standing problem in computer vision [7, 9, 14]. In particular, dense visual SLAM aims to construct a detailed geometric representation of the environment enabling various applications, such as, occlusion handling in augmented reality [27], planning and collision checking in robotics [1], or camera localization [22]. Often volumetric scene representations are employed as they are well-suited for online data integration. Traditional volumetric representations use grid-based structures [26], often accelerated through the use of octrees [10, 19] or voxel hashing [28]. However, incorporating loop closure constraints in volumetric maps is difficult [6, 31, 41, 44] compared to sparse keypoint-based maps, which can easily be deformed.

Recently, neural fields have emerged as a promising volumetric scene representation due to their amenability to differentiable rendering-based optimization [38]. Building on the first neural field-based SLAM method iMAP [36], various adaptations of this method have been proposed. These adaptations primarily focused on enhancing optimization speed by altering network architecture, sampling scheme, and rendering formulation. Nevertheless, the majority of existing neural field-based SLAM methods [12, 36, 42, 46, 47] remain constrained by their monolithic data structure (i.e., a single fixed-size architecture). In that regard, neural fields share the same limitation as other volumetric scene representations: after data has been integrated, the volumetric scene cannot be easily deformed to take loop closure constraints into account.

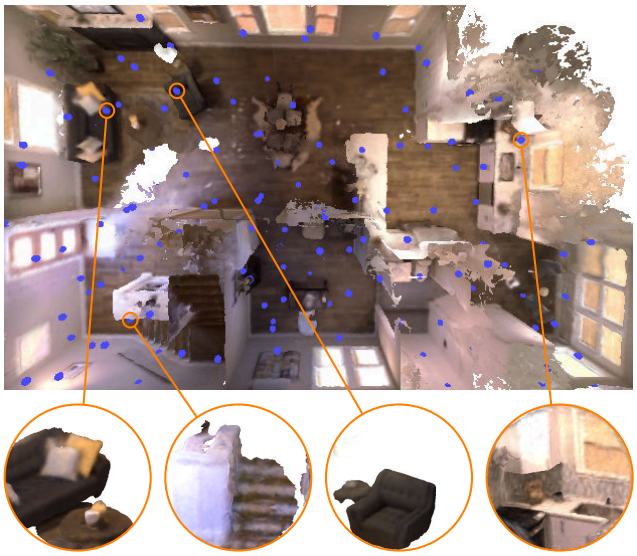


Figure 1. We propose to represent a scene by a set of neural fields (centers indicated by blue spheres) anchored to keyframes in a pose graph with each field capturing the scene within a ball surrounding it. This allows to dynamically extend the scene while also incorporating loop closure deformations into the volumetric scene representation without requiring full reintegration.

To address this limitation, submap-based approaches

have been proposed that maintain a small number of neural field-based submaps [16, 18, 39]. However, submap-based methods come with their own set of difficulties such as inaccurate submap registration, transition artifacts at submap boundaries, and higher computational cost.

Instead of few, larger submaps, we propose to represent the scene by an extendable set of smaller, lightweight neural fields (see Fig. 1). These fields are dynamically anchored to keyframes in a pose graph, collectively forming the volumetric map while maintaining the ability to deform as the keyframe poses change upon loop closure. Our design allows to combine the benefits of sparse pose graph-based SLAM methods while maintaining a volumetric scene representation that remains consistent with the pose graph without requiring costly reintegration of previous sensor data. Furthermore, our method eliminates the necessity for fixed scene boundaries common among existing neural field-based SLAM methods. It achieves this by allocating additional fields dynamically as new parts of the scene come into view.

To summarize, our contributions are

- a novel RGB-D neural mapping framework that combines the robust, accurate tracking and efficient loop closure handling of sparse visual SLAM with the differentiable-rendering-based dense mapping of neural scene representations,
- a multi-field scene representation in which neural fields are anchored relative to a pose graph allowing for large-scale map deformations while limiting necessary reintegration, and
- a thorough comparison to multiple state-of-the-art methods on scenes of varying scales, including a novel set of sequences for the larger Replica scenes that allow to benchmark robustness and scalability.

2. Related Work

Traditional Volumetric Loop Closure The difficulty of incorporating loop closure constraints in volumetric scene representations is not limited to neural representations. Traditional volumetric representations such as sparse grids and octrees also require solutions to efficiently adapt to pose graph changes. Notable examples include: Kintinuous [44] in which the volumetric map is only used for local fusion and globally a mesh is deformed based on a deformation graph; BundleFusion [6] in which frames whose pose has changed are removed and reintegrated; VoxGraph [31] in which submaps are aligned and resulting constraints are included in the pose graph optimization; and recently LivePose [34] which extends the idea of BundleFusion to an RGB setting. Our work aims to enable efficient loop closure integration specifically for neural field-based representations for which de- and reintegration is difficult.

Neural Field-Based Scene Representations Neural fields are parametrized differentiable functions that map coordinates to quantities at that point in space. Initially, neural fields were trained using 3D supervision to represent shapes as occupancy [4, 20] or signed distance fields (SDFs) [29]. Shortly after, neural radiance fields (NeRFs) were introduced optimized using only 2D images through differentiable volume rendering [21]. Replacing NeRF's density with an SDF-based representation higher quality surface reconstructions can be achieved [2, 43].

Few works have investigated the use of multiple neural fields. In Block-NeRF [37] multiple fields are combined to represent whole neighborhoods. However, the field positions are predefined. In Nerflets [45], fields of varying size are composed to create an editable scene representation. NeRFuser [8] proposes methods for registering and blending multiple NeRFs. Finally, vMAP [15] uses multiple fields to represent individual objects highlighting the potential of vectorizing neural field evaluations. Our work follows similar motivations: by splitting the scene into multiple independent fields, it becomes both, efficiently deformable upon loop closures, and dynamically extendable as the camera moves.

Neural Field-Based SLAM iMAP [36] was the first SLAM method using a multilayer perceptron (MLP) as the underlying scene representation. During optimization previous keyframes are continuously reintegrated to avoid forgetting. Subsequently, various modifications of this approach have been suggested aiming to alleviate this forgetting issue and improve run-time.

NICE-SLAM [47] addresses the forgetting issue by employing a hierarchical feature grid combined with a fixed, pretrained decoder MLP. By adopting an SDF-based field combined with efficient learnable positional encodings, ESLAM [12] and Co-SLAM [42] demonstrate significantly improved optimization times. Specifically, ESLAM uses a tri-plane [3] encoding, whereas Co-SLAM uses the aforementioned multi-resolution voxel hash encoding [23]. Co-SLAM further complements the hash encoding with one-blob encodings [24] to improve scene completion. Point-SLAM [33] replaces NICE-SLAM's regular feature grid with with an irregular neural point cloud allowing to allocate more capacity in complex areas. However, none of the aforementioned approaches support integration of loop closure constraints, preventing their use for larger scenes, in which accumulation of drift is unavoidable.

[18] follows a similar idea to ours, also using multiple fields posed relative to a pose graph. However, they employ space warping in which each field in principle covers the entire scene. Synthesized views from multiple fields are merged through alpha compositing. This approach is well-suited for novel view synthesis; however, it generalizes poorly to mesh extraction and other geometric queries. In-

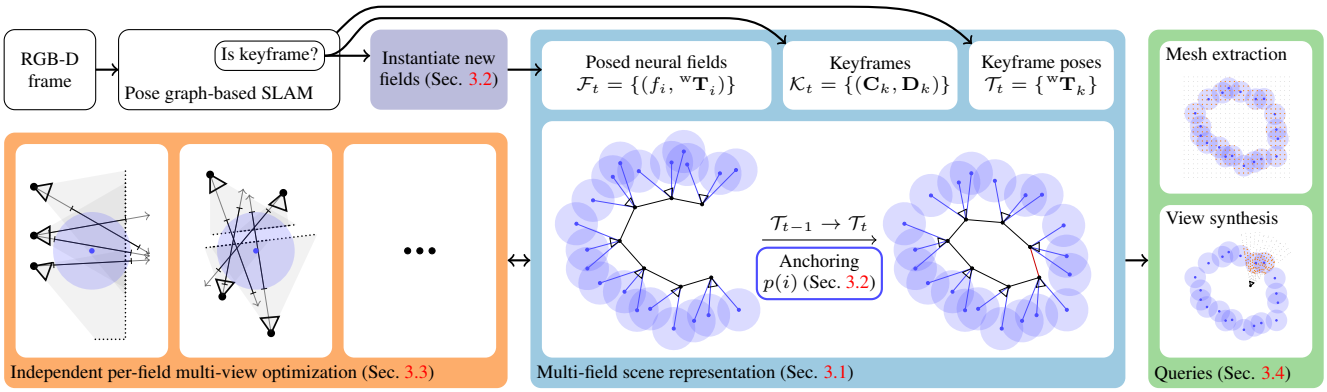


Figure 2. Neural graph map overview. Our framework can be used with any pose graph-based SLAM system and maintains a set of neural fields anchored relative to the keyframes of the pose graph. Each field captures the scene in a sphere surrounding it in local coordinates. This set of fields represents a deformable, volumetric scene representation that can easily adapt to and stay in sync with the pose graph. The fields are optimized independently always using the latest pose graph information for supervision. This design reduces transition artifacts, enables efficient optimization, and allows to average the output of overlapping fields in 3D leading to well-defined queries.

stead, in our work, each field covers a Euclidean ball around it, which allows general geometric queries. In MIPS-Fusion [39] global bundle adjustment at the level of fields is performed. Loopy-SLAM [16] also describes a submap-based system built on top of Point-SLAM [33]. Instead of larger keyframe-centric submaps, our approach represents the scene by many small fields, each covering a sphere surrounding it. By adjusting the field poses relative to the pose graph, while avoiding hard assignments of keyframes to fields during the optimization, our method allows to adapt to local and global pose graph deformations while avoiding transition artifacts common for submap-based methods.

Instead of using submaps, GO-SLAM [46] adapts to loop closures by biasing the optimization to keyframes whose pose changed the most. However, it still uses a single neural field, which requires reoptimization upon loop closure. Instead, we achieve instant map deformation upon loop closure by moving fields relative to the pose graph.

3. Method

We consider the problem of online RGB-D mapping without ground-truth poses. That is, given a stream of RGB-D frames $(\mathbf{C}_t, \mathbf{D}_t)$ composed of color images $\mathbf{C}_t \in \mathbb{R}^{H \times W \times 3}$ and depth maps $\mathbf{D}_t \in \mathbb{R}^{H \times W}$, our goal is to build a dense scene representation at each time step t.

Fig. 2 gives an overview of the proposed framework. Similar to other recent neural field-based mapping approaches [15, 18, 46], our approach uses an off-the-shelf keyframe-based SLAM system to provide a set of posed keyframes as well as the pose of the current frame.

3.1. Multi-Field Scene Representation

Let $\mathcal{K}_t = \{(\mathbf{C}_k, \mathbf{D}_k) \mid k = 1, \ldots, K_t\}$ and $\mathcal{T}_t = \{{}^{\mathrm{w}}\mathbf{T}_k \in \mathrm{SE}(3) \mid k = 1, \ldots, K_t\}$ denote the set of keyframes and keyframe poses at time step t, respectively.

We propose to represent the scene by an extendable set of posed neural fields

$$\mathcal{F}_t = \{(f_i, {}^{\mathrm{w}}\mathbf{T}_i) \mid i = 1, ..., F_t\}, \tag{1}$$

where each field

$$\begin{aligned} f_i : \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \times \mathbb{R} \\ \boldsymbol{x} &\mapsto (\boldsymbol{c}, s) \end{aligned} \tag{2}$$

maps a 3D point $\boldsymbol{x}$ in its local reference frame to a color $\boldsymbol{c}$ and truncated signed distance s at that point. Each field's reference frame is defined by its pose ${}^{\mathrm{w}}\mathbf{T}_i \in \mathrm{SE}(3)$, which can be decomposed into a position ${}^{\mathrm{w}}\boldsymbol{t}_i \in \mathbb{R}^3$ and orientation ${}^{\mathrm{w}}\mathbf{R}_i \in \mathrm{SO}(3)$. Each field only captures the scene within a sphere of fixed radius r.

We further define a function $p(i)$ that anchors a field i to a parent keyframe k. At each iteration, given a field i and its parent keyframe $k_i = p(i)$, we compute the field's pose according to ${}^{\mathrm{w}}\mathbf{T}_i = {}^{\mathrm{w}}\mathbf{T}_{k_i}\, {}^{k_i}\widetilde{\mathbf{T}}_{\mathrm{w}}\, {}^{\mathrm{w}}\widetilde{\mathbf{T}}_i$, where $\widetilde{\mathbf{T}}$ denotes transforms from the previous time step. That is, fields move as if they are rigidly connected to their parent keyframe. The parent keyframe can however change over time depending on the anchoring strategy defined by $p(i)$.

In the following sections, the proposed strategies to instantiate and dynamically anchor fields to the pose graph (Sec. 3.2), optimize the fields (Sec. 3.3), and query (e.g., for mesh extraction and view synthesis) the full scene representation (Sec. 3.4) are described. Finally, further architecture details are provided (Sec. 3.5).

3.2. Field Instantiation and Anchoring

Whenever a new keyframe k is added to the pose graph, new fields are instantiated such that all observed 3D points ${}^{\mathrm{w}}\mathcal{X}_k$ in the keyframe are covered by at least one field of radius r. An approximate two-stage algorithm is used (see

Fig. 3). First, the uncovered 3D points $\mathcal{X}_{\text{unc}} = \{\boldsymbol{x} \in {}^{\text{w}}\mathcal{X}_k \mid \|\boldsymbol{x} - {}^{\text{w}}\boldsymbol{t}_i\| > r \,\forall\, i \in \mathcal{N}_f\}$ are found. Second, the space is divided into voxel cells with a side length of $g = 2r/\sqrt{3}$, such that a cell is fully covered by a field of radius r when the center of the field is at the center of the cell. New fields are instantiated in the center of all cells that contain a point from $\mathcal{X}_{\text{unc}}$ and no field center. This scheme ensures that a minimum distance of $g/2$ to existing fields is maintained; however, points in $\mathcal{X}_{\text{unc}}$ might remain uncovered when fields move away from the cell centers through pose graph deformations. To alleviate the chance of uncovered points over time, the voxel grid is randomly shifted for every added keyframe.

New fields added this way are initially anchored to the keyframe that triggered the creation. Over time fields are supervised by all keyframes that observe it (see Sec. 3.3), some of which might be significantly closer than the initial keyframe. Therefore, we define the parenting function $p(i)$ as the closest keyframe (Euclidean distance) with valid depth observation used to supervise the field.

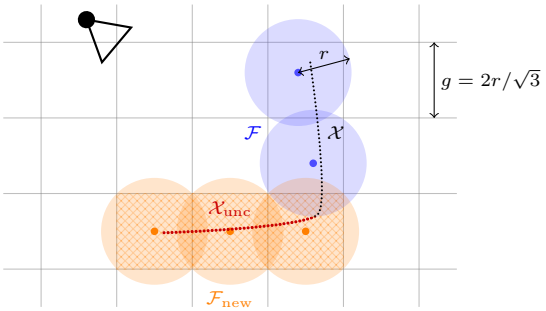


Figure 3. Grid-based instantiation scheme. A new keyframe (top-left) observes world points $\mathcal{X}$ (black & red dots). The points $\mathcal{X}_{\text{unc}}$ (red dots) that are not covered by any of the existing fields $\mathcal{F}$ (blue circles) are determined. Cells that contain uncovered points and no existing field center (hatched orange) are used to instantiate new fields $\mathcal{F}_{\text{new}}$ (orange circles). The new fields are positioned in the center of the uncovered cells and anchored to the keyframe.

3.3. TSDF-Based Optimization

At each time step t, a fixed number of optimization steps N_{it} are performed on up to N_{f} fields. During the optimization the fields f_i are trained independently, allowing for efficient parallelization [15]. Only the fields' parameters are used as optimization variables. The fields' poses are updated solely according to the pose graph as described in Sec. 3.1.

Due to its well-defined isosurface for mesh extraction, we model the geometry as a truncated signed distance field (TSDF) adopting the TSDF-based rendering model from [2] and modify it to be occlusion-aware. Specifically, instead of directly converting signed distances to sample weights as done in [2, 42], we first convert them to occupancy probabilities. That is, given N_{s} samples along a ray $\boldsymbol{x}_i = \boldsymbol{o} + l_i\boldsymbol{d}, i = 1, ..., N_{\text{s}}$ with origin $\boldsymbol{o}$ and direction $\boldsymbol{d}$, the colors and signed distances $(\boldsymbol{c}_i, s_i) = f(\boldsymbol{x}_i)$ along the ray are computed. The signed distances are converted to occupancy probabilities according to

$$o_i = 4\sigma\left(\frac{\eta s_i}{\tau}\right)\sigma\left(-\frac{\eta s_i}{\tau}\right), \tag{3}$$

where σ denotes the sigmoid function, τ is the truncation distance, and η is a parameter that determines how sharply occupancy probability decays around the surface. Note that this definition ensures $o_i = 1.0$ for $s_i = 0.0$. The ray's rendered color and depth are then computed via the weight $w_i = o_i \prod_{j=1}^{i-1}(1 - o_j)$ as

$$\boldsymbol{c} = \sum_{i=1}^{N_{\text{s}}} w_i \boldsymbol{c}_i \quad d = \sum_{i=1}^{N_{\text{s}}} w_i l_i \tag{4}$$

assuming the ray's direction $\boldsymbol{d}$ was scaled appropriately.

Sampling Strategy Each optimization iteration follows a three-stage sampling procedure. Specifically, we sample: (1) N_{f} fields that will be optimized in the next iteration; (2) N_{r} ray segments (with associated observed color $\mathbf{c}$ and distance d) for each sampled field; (3) N_{s} query points along each sampled ray segment. Importantly, no assignments of keyframes to fields are made. Instead, which keyframes observe which field are recomputed in each iteration based on the latest pose graph information. This avoids transition effects stemming from different optimization targets of neighboring fields. The sampling strategy is visualized in Fig. 4 and details are provided in the supplementary material.

Loss Our loss is a weighted sum of four terms: color, depth, TSDF, and free-space. The weighing has been found experimentally to achieve a good trade-off between appearance and geometry quality.

The color loss is computed as the mean L1-norm of the difference between observed and estimated color

$$l_{\text{color}} = \frac{1}{|\mathcal{C}|}\sum_{(\boldsymbol{c},\tilde{\boldsymbol{c}})\in\mathcal{C}} \|\boldsymbol{c} - \tilde{\boldsymbol{c}}\|_1 \tag{5}$$

and the depth loss is computed as the mean Huber loss [11] between observed and estimated depth with $\delta = 5\,\text{cm}$

$$l_{\text{depth}} = \frac{1}{|\mathcal{D}|}\sum_{(d,\tilde{d})\in\mathcal{D}} \text{Huber}_\delta(d, \tilde{d}), \tag{6}$$

where $\mathcal{C}$ and $\mathcal{D}$ are sets containing tuples of the observed and estimated color and depth, respectively.

The query points $\boldsymbol{x}_i$ are filtered into two sets $\mathcal{X}_{\text{tsdf}}$ and $\mathcal{X}_{\text{fs}}$ based on whether a query point's projected depth l_i is within the truncation distance τ of the corresponding observed depth $l_{\text{obs},i}$ or is more than τ in front of it,

respectively. A query point associated estimated signed distance $\tilde{s}_i$ and its signed distance to the observed depth $s_i = l_{\text{obs},i} - l_i$ are then used to compute the TSDF loss as

$$l_{\text{tsdf}} = \frac{1}{|\mathcal{X}_{\text{tsdf}}|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_{\text{tsdf}}} (\tilde{s}_i - s_i)^2 \tag{7}$$

and the free-space loss as

$$l_{\text{fs}} = \frac{1}{|\mathcal{X}_{\text{fs}}|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_{\text{fs}}} (\tilde{s}_i - \tau)^2. \tag{8}$$

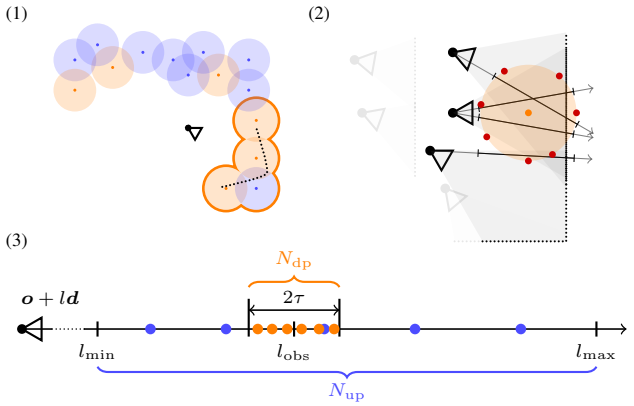


Figure 4. Multi-view sampling procedure. (1) A subset of fields (orange circles) to optimize in the next iteration is sampled biased to the currently observed fields (orange outline). (2) For each field, the observing keyframes (opaque frustums) are approximated based on whether samples on the field boundary (red dots) fall into the observed keyframe region. Ray segments to supervise the field are then sampled from all observing keyframes. (3) Each sampled ray segment $[l_{\text{min}}, l_{\text{max}}]$ is approximated by N_{up} uniformly sampled query points that cover the whole field and N_{dp} depth-guided query points distributed within the truncation distance τ around the observed distance l_{obs}.

3.4. Novel View Synthesis and Mesh Extraction

While the independent optimization ensures that overlapping regions will be supervised with the same data, fields might be at different stages of optimization and unobserved regions might converge to different solutions due to epistemic uncertainty. To reduce the resulting transition artifacts at the boundary of fields, it is possible to query the k nearest fields and average the outputs (see supplementary material for details). For query points outside of all fields' spheres empty space is assumed, that is, $s = 1$. For novel view synthesis, since no depth observation is available at inference time, a fixed far distance l_{far} is used. Query points are uniformly distributed within the ray interval $[0, l_{\text{far}}]$ at the same density as the depth-guided points, that is, $N_{\text{p,inf}} = l_{\text{far}} N_{\text{dp}} / (2\tau)$ samples are used. To extract a mesh the marching cubes algorithm [17] with an unbiased isosurface level of 0 is used.

3.5. Architecture Details

ORB-SLAM2 is used as the keyframe-based SLAM system [25]. However, our proposed method is agnostic to the exact SLAM framework as long as a pose graph as described in Sec. 3.1 is available.

Each field is parameterized by a permutohedral hash encoding [32] followed by a linear layer with ReLU activation and another linear layer mapping to the color and distance. We also experimented with other positional encodings, such as Fourier encoding [38], triplane encoding [3, 12], and voxel hash encoding [23]; however, the best results given the same time budget were achieved with the permutohedral hash encoding closely followed by the voxel hash encoding.

To efficiently evaluate and optimize multiple neural fields in parallel, optimization and evaluation is vectorized following [15]. One limitation imposed by PyTorch's [30] vectorization framework is that the batch size has to be the same for all evaluated networks. The design choices described in Sec. 3.3 are tailored to this limitation sampling the same number of rays per supervised field.

4. Experiments

4.1. Experimental Setup

Datasets We evaluate our method quantitatively on three synthetic datasets of varying difficulty: the *Replica* [35] sequences from iMAP [36], the *NRGBD* dataset by [2], and three novel sequences on the larger Replica scenes (*Replica-Big*). The latter contains challenging trajectories with larger loops spanning multiple rooms. We also include results on two real-world datasets: ScanNet [5] and Kintinuous [44]. The latter contains a large-scale loop and the tested ScanNet sequences typically include one or two smaller loops. Quantitative evaluation of the map is not possible for the real-world datasets, because no accurate ground-truth is available in these datasets: ScanNet provides meshes from BundleFusion [6], Kintinuous provides none.

Metrics To assess the reconstruction quality of the methods we report the accuracy ratio, completion ratio, and F1-score [40] at a 5 cm threshold. In the supplementary material we further report the non-robust metrics, accuracy and completion. We do not evaluate tracking metrics, since we use ORB-SLAM2 as the tracking backend and our focus is on dense mapping, while efficiently supporting map deformations due to loop closures.

Baselines We compare our method to five state-of-the-art neural SLAM methods: NICE-SLAM [47] and Co-SLAM [42], which do not support loop closure; GO-SLAM [46], MIPS-Fusion [39], and Loopy-SLAM [16], which support loop closure. Of these, GO-SLAM uses a monolithic scene representation; MIPS-Fusion few, large submaps; and

Table 1. Comparison of mesh reconstruction quality (**best** ●, second best ●, third best ●).

		Replica	NRGBD	Replica-Big		
		Avg.*	Avg.*	apt0	apt1	apt2
NICE-S. [47]	Acc. R. (%)	**93.17** ●	78.10	30.67	41.48	28.20
	Comp. R. (%)	90.11	70.58	70.58	42.21	26.80
	F1-Score (%)	91.60	74.07	74.07	41.84	27.49
Co-S. [42]	Acc. R. (%)	87.02	87.23	18.85	62.11	34.71
	Comp. R. (%)	86.39	82.86 ●	21.93	66.37	36.75
	F1-Score (%)	86.70	84.94 ●	20.27	64.17	35.70
GO-S. [46]	Acc. R. (%)	89.31	84.10	**57.93** ●	74.38	**80.22** ●
	Comp. R. (%)	74.20	66.19	**59.21** ●	73.90 ●	81.06 ●
	F1-Score (%)	81.03	73.98	**58.56** ●	74.14 ●	**80.64** ●
MIPS-F. [39]	Acc. R. (%)	92.63	88.29 ●	28.00	75.97 ●	33.16
	Comp. R. (%)	**91.38** ●	75.85	28.11	72.14	34.76
	F1-Score (%)	92.00 ●	81.40	28.06	74.01	33.94
Loopy-S.† [16]	Acc. R. (%)	99.99	80.52	✗	31.34	59.87
	Comp. R. (%)	89.28	72.66	✗	21.71	79.04
	F1-Score (%)	94.32	76.29	✗	25.65	68.14
Ours-SF	Acc. R. (%)	93.11 ●	**93.11** ●	54.69 ●	93.14 ●	76.86 ●
	Comp. R. (%)	91.28 ●	**85.45** ●	56.07 ●	**95.42** ●	79.57 ●
	F1-Score (%)	**92.19** ●	**89.02** ●	55.37 ●	94.27 ●	78.19 ●
Ours	Acc. R. (%)	92.76 ●	93.09 ●	56.25 ●	**94.14** ●	79.28 ●
	Comp. R. (%)	90.87 ●	84.70 ●	56.89 ●	95.25 ●	**81.81** ●
	F1-Score (%)	91.80 ●	88.58 ●	56.57 ●	**94.69** ●	80.52 ●

* Full results are available in the supplementary material.
† Excluded from ranking because it uses ground-truth depth during evaluation.

Loopy-SLAM uses neural point-based submaps. In addition we compare to a monolithic, single-field ablation of our method (Ours-SF), which is discussed in Sec. 4.4.

Loopy-SLAM is the only non-volumetric method representing only the volume within centimeters of the surfaces. When querying (for rendering, which in their case is used for subsequent mesh extraction), Loopy-SLAM relies on ground-truth depth maps. Because none of the other methods do so, it leads to an unfair advantage especially on the noise-free synthetic scenes. Therefore, we exclude Loopy-SLAM from the ranking in the tables.

Implementation Details For all experiments we use the same parameters, most notably, $r = 1\,\mathrm{m}$, $N_{\mathrm{it}} = 5$, $N_{\mathrm{f}} = 32$, $N_{\mathrm{r}} = 512$, $N_{\mathrm{up}} = 8$, $N_{\mathrm{dp}} = 16$ and $k = 2$. We adjust the truncation distance τ from $0.1\,\mathrm{m}$ for synthetic data to $0.2\,\mathrm{m}$ for real data to account for the increased depth noise. For ORB-SLAM2 we use the default parameters in the RGB-D setup increasing the number of extracted features in challenging scenes with low texture. To facilitate reproducibility of our results, the full evolution of the pose graph was precomputed and played back as if ORB-SLAM is running in parallel. We will publish these recordings in conjunction with our code. All remaining parameters are provided in the supplementary material and as part of the released code.

4.2. Reconstruction Quality Evaluation

Tab. 1 reports quantitative results on all datasets. Our method achieves state-of-the-art results on the smaller scenes (Replica, NRGBD) and outperforms existing methods on larger scenes (Replica-Big). The qualitative results on Replica-Big shown in Fig. 5 further illustrate the differences. NICE-SLAM and Co-SLAM drift off and cannot correct for the drift upon returning to previous frames leading to poor results. GO-SLAM also benefiting from loop closures successfully maps large parts; however, fails in the last section of `apt0` and generally exhibits worse reconstruction quality. While GO-SLAM achieves quantitatively slightly better results as our method on `apt0`, we note that qualitatively our method exhibits better scene completion and higher fidelity overall[1]. The submap-based methods MIPS-Fusion and Loopy-SLAM fail to achieve globally consistent result on most scenes. Particularly Loopy-SLAM failed on all scenes of Replica-Big[2]. Our method and GO-SLAM benefit from their underlying SLAM systems ORB-SLAM2 and DROID-SLAM, respectively.

Fig. 5 shows results on three scenes of the ScanNet dataset (the "ground-truth" was obtained using BundleFusion in this case [6]). The benefit of neural scene representations over classic representations in terms of scene completion can easily be observed. While less drift accumulates on these scenes, our method still achieves on par results to the best performing method Co-SLAM. The point-based Loopy-SLAM appears less robust to noisy measurements and creates more noisy, less complete reconstructions.

Fig. 6 shows additional results on the Kintinuous sequences before and after loop closure highlighting the efficient loop closure integration of our method. Because none of the other methods successfully closed the loop, we compare with the monolithic ablation of our approach (see Sec. 4.4 for further discussion).

4.3. Run-Time and Model Size Analysis

Tab. 2 reports run-time[3] and model size on three representative scenes of different sizes. Our approach compares favorably in terms of run-time despite the overhead of parallelized training of multiple fields. GO-SLAM and Loopy-SLAM slow down significantly for longer sequences, whereas our method slows down the least of all evaluated methods (5% difference between the smallest and largest scene). Model size is competitive with the submap-

[1] The upper floor has a slight consistent orientation error even after loop closure, which makes many walls fall outside the $5\,\mathrm{cm}$ threshold explaining the mismatch between perceived quality and evaluation result.

[2] We could not achieve a successful run on `apt0` without exceeding 24 GB of GPU memory due to too many created submaps.

[3] We report time per frame by benchmarking the execution time for the whole sequence and dividing it by the number of frames. Mesh extraction has been deactivated for all methods for fair comparison. All evaluations were run with an Intel Core i9-13900KF and NVIDIA GeForce RTX 4090.

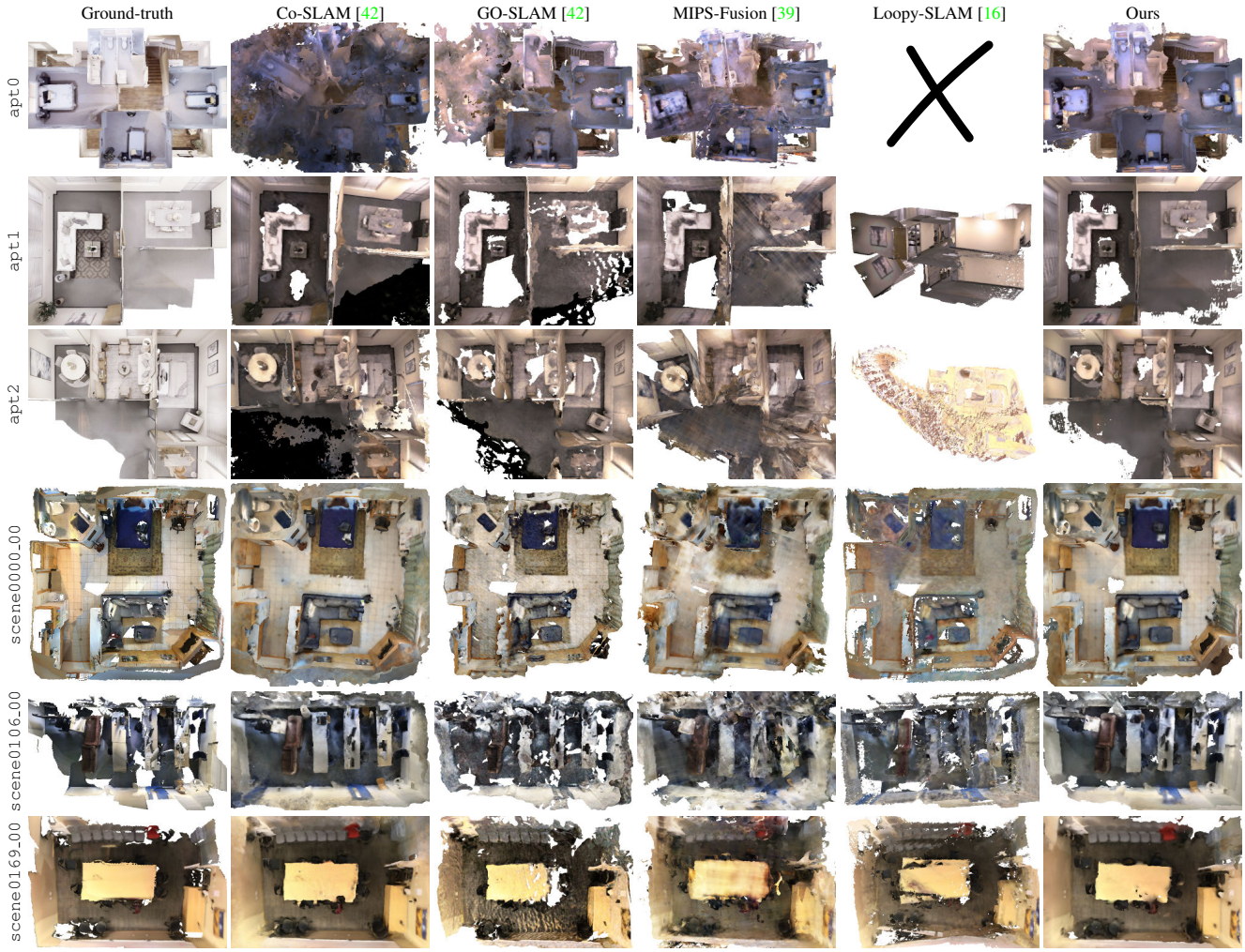
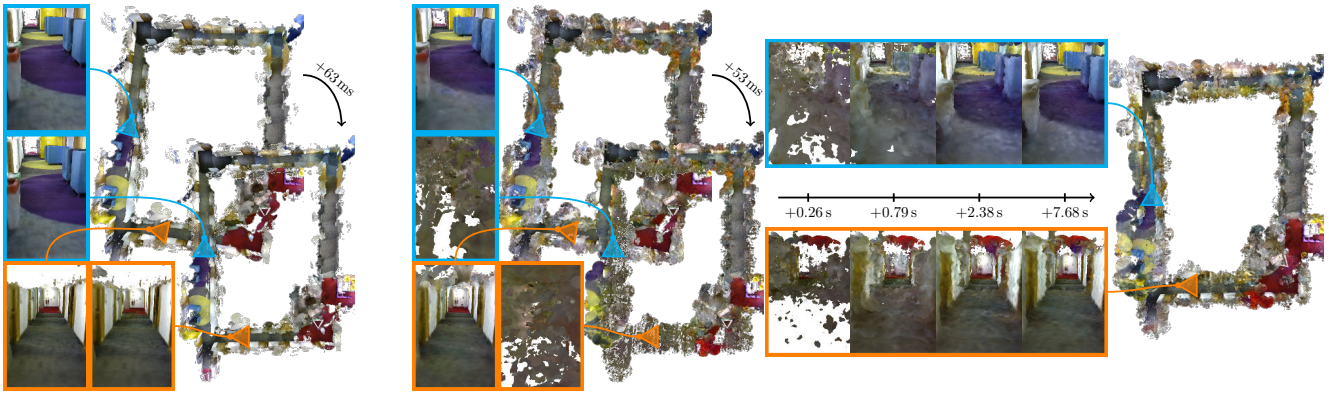


Figure 5. Qualitative comparison of final reconstruction on Replica-Big and ScanNet. Alternative view is shown for significant failure cases. The submap-based methods MIPS-Fusion and Loopy-SLAM exhibit artifacts at submap borders and erroneous submap registrations. Our method allows submap-like map deformation at much finer granularity, while significantly reducing transition artifacts.



(a) Multi-field representation

(b) Single-field representation

Figure 6. Qualitative results on the Kintinuous sequence for our multi-field representation and the ablated, monolithic variant. We show a top-down and two additional views of the mesh right before and after specific durations. Our multi-field representation adapts within just one frame with minimal loss in quality, whereas the monolithic variant does not reach the same quality even after 7 s of optimization.

Table 2. Run-time and model size comparison of all methods. Our method compares favorably in terms of processing time compared to all baselines. The model size is larger than for the monolithic baselines, but competitive with the submap-based methods. Loopy-SLAM failed after 16 hours (0.1 Hz) on `apt0`.

	FPS (Hz)			Model Size (MB)		
	`br`	`ck`	`apt0`	`br`	`ck`	`apt0`
NICE-SLAM [47]	2.74	2.70	2.25	5.3	21.6	57.1
Co-SLAM [42]	12.70	12.84	10.75	7.0	7.0	7.0
GO-SLAM [46]	17.35	12.06	6.36	66.5	66.5	66.5
MIPS-Fusion [39]	4.02	3.64	3.31	36.2	72.4	398.3
Loopy-SLAM [16]	0.38	0.36	✗	21.9	48.1	✗
Ours-SF	19.78	18.92	18.84	8.4	8.4	8.4
Ours	16.67	15.93	15.68	28.1	57.7	229.2

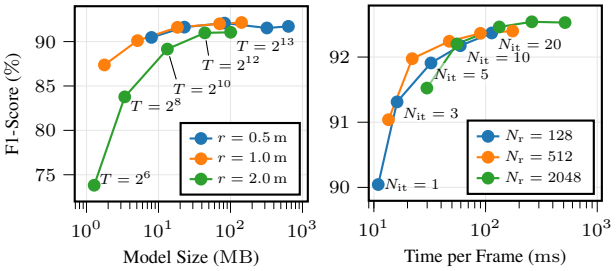


Figure 7. Left: reconstruction quality and model size for varying field radii r and hash table sizes T. Right: reconstruction quality and run-time for varying number of iterations per frame N_{it} and number of rays per field N_{r}.

based baselines, but worse in comparison to monolithic methods. This might be due to the more constrained hash tables, which can be overparameterized for simple areas.

Run-time results such as these are difficult to interpret due to varying implementation quality and large number of parameters. Especially NICE-SLAM and Loopy-SLAM are an order-of-magnitude slower than the other methods. Whether this is due to inefficient implementation or inherent complexity is unclear. Also, the baselines perform tracking and mapping, whereas our method uses precomputed ORB-SLAM results. However, ORB-SLAM is a CPU-only method and could therefore run in parallel without significantly affecting the run-time of our approach.

Field Radius and Hash Map Size In Fig. 7 (left) the memory requirement and F1-score for different field radii r and hash table sizes T is reported. In general, larger fields require larger hash maps to maintain the same quality. Our approach works across a wide range of r, but the ability to adapt to loop closures will be limited for larger r.

Batch Size and Iterations In Fig. 7 (right) the processing time per frame and F1-score for varying batch sizes and number of iterations is reported. Increasing the number of iterations per frame or the batch size leads to improved reconstruction quality. Notably, our approach maintains high reconstruction quality even at frame rates of more than 50 Hz.

Figure 8. Single-view supervision leads to artifacts when the opposite side of a wall is observed. Multi-view supervision significantly reduces this local forgetting effect.

4.4. Ablation Study

Multi-Field Representation In Tab. 1 and Tab. 2 a comparison to a single-field variant of our method to isolate the effect of the multi-field representation. This variant uses the same rendering, poses, and optimization scheme as the multi-field variant, but represents the scene by a global monolithic field with $T = 2^{16}$. On the scenes with significant loop closures (`apt0` and `apt2`) the multi-field representation outperforms the monolithic variant.

These results are at the end of the sequence and hence the monolithic variant has time to reintegrate with updated poses. Fig. 6 shows the map quality right after a large-scale loop closures. Our proposed multi-field representation adapts immediately to the pose adjustments keeping the map intact, while the single-field variant degrades significantly and only slowly reoptimizes without reaching the previous quality after more than 7 s.

Multi-View Supervision Fig. 8 compares optimization with targets sampled from all keyframes as described in Sec. 3.3 to targets sampled from a single view. In the latter case, optimization alternates between a random previous keyframe and the latest keyframe. Single-view supervision shows local forgetting effects, particularly notable when a previously seen wall is observed from the other side. Multi-view optimization avoids this by combining previous and current observations in each optimization step.

5. Conclusion

We have presented a neural mapping framework that anchors lightweight neural fields to the pose graph of a sparse visual SLAM system. This allows to incorporate loop closures into the volumetric map at near-zero cost achieving global map consistency over time. Our approach deforms the map reducing the need for reintegration while allowing geometric queries without requiring any space warping or image-space fusion. Compared to submap-based approaches, it deforms the map at a more granular level and reduces commonly observed artifacts at submap boundaries by avoiding hard assignments of frames to submaps.

Acknowledgments This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world. *RAL*, 7(2):4606–4613, 2022. 1

[2] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D surface reconstruction. In *CVPR*, pages 6290–6301, 2022. 2, 4, 5, 18, 19

[3] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, pages 16123–16133, 2022. 2, 5

[4] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, pages 5939–5948, 2019. 2

[5] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 5, 20

[6] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM TOG*, 36(4):1, 2017. 1, 2, 5, 6

[7] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE TPAMI*, 29(6):1052–1067, 2007. 1

[8] Jiading Fang, Shengjie Lin, Igor Vasiljevic, Vitor Guizilini, Rares Ambrus, Adrien Gaidon, Gregory Shakhnarovich, and Matthew R Walter. NeRFuser: Large-scale scene representation by NeRF fusion. *arXiv preprint arXiv:2305.13307*, 2023. 2

[9] Christopher G Harris and JM Pike. 3D positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988. 1

[10] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34:189–206, 2013. 1

[11] Peter J Huber. Robust estimation of a location parameter. *Ann. Math. Statist.*, 35:73–101, 1964. 4

[12] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. ESLAM: Efficient dense SLAM system based on hybrid representation of signed distance fields. In *CVPR*, pages 17408–17419, 2023. 1, 2, 5

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 12

[14] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, pages 225–234, 2007. 1

[15] Xin Kong, Shikun Liu, Marwan Taher, and Andrew J Davison. vMAP: Vectorised object mapping for neural field SLAM. In *CVPR*, pages 952–961, 2023. 2, 3, 4, 5

[16] Lorenzo Liso, Erik Sandström, Vladimir Yugay, Luc Van Gool, and Martin R Oswald. Loopy-SLAM: Dense neural SLAM with loop closures. In *CVPR*, pages 20363–20373, 2024. 2, 3, 5, 6, 7, 8, 10, 14, 15, 16, 17, 18, 19, 20

[17] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH*, pages 163–169, 1987. 5

[18] Hidenobu Matsuki, Keisuke Tateno, Michael Niemeyer, and Federico Tombari. NEWTON: Neural view-centric mapping for on-the-fly large-scale SLAM. *RAL*, 2024. 2, 3

[19] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982. 1

[20] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, pages 4460–4470, 2019. 2

[21] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, pages 405–421. Springer, 2020. 2

[22] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. LENS: Localization enhanced by NeRF synthesis. In *CoRL*, pages 1347–1356, 2022. 1

[23] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4), 2022. 2, 5

[24] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM TOG*, 38(5):1–19, 2019. 2, 12

[25] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *TRO*, 33(5):1255–1262, 2017. 5

[26] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011. 1

[27] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, pages 2320–2327, 2011. 1

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

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019. 2

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, volume 32, pages 8026–8037, 2019. 5

[31] Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. Voxgraph:

Globally consistent, volumetric mapping using signed distance function submaps. *RAL*, 5(1):227–234, 2019. 1, 2

[32] Radu Alexandru Rosu and Sven Behnke. PermutoSDF: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In *CVPR*, pages 8466–8475, 2023. 5

[33] Erik Sandström, Yue Li, Luc Van Gool, and Martin R Oswald. Point-SLAM: Dense neural point cloud-based SLAM. In *ICCV*, pages 18433–18444, 2023. 2, 3

[34] Noah Stier, Baptiste Angles, Liang Yang, Yajie Yan, Alex Colburn, and Ming Chuang. LivePose: Online 3D reconstruction from monocular video with dynamic camera poses. In *ICCV*, pages 7921–7930, 2023. 2

[35] Julian Straub et al. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 5, 16, 17

[36] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. iMAP: Implicit mapping and positioning in real-time. In *ICCV*, pages 6229–6238, 2021. 1, 2, 5, 16, 17

[37] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. In *CVPR*, pages 8248–8258, 2022. 2

[38] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 33:7537–7547, 2020. 1, 5

[39] Yijie Tang, Jiazhao Zhang, Zhinan Yu, He Wang, and Kai Xu. MIPS-FUSION: Multi-implicit-submaps for scalable and robust online neural RGB-D reconstruction. *ACM TOG*, 42(6):1–16, 2023. 2, 3, 5, 6, 7, 8, 14, 15, 16, 17, 18, 19, 20

[40] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *CVPR*, pages 3405–3414, 2019. 5

[41] Diego Thomas and Akihiro Sugimoto. Modeling large-scale indoor scenes with rigid fragments using RGB-D cameras. *Computer Vision and Image Understanding*, 157:103–116, 2017. 1

[42] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-SLAM: Joint coordinate and sparse parametric encodings for neural real-time SLAM. In *CVPR*, pages 13293–13302, 2023. 1, 2, 4, 5, 6, 7, 8, 12, 13, 14, 15, 16, 17, 18, 19, 20

[43] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 34, 2021. 2

[44] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *IJRR*, 34(4-5):598–626, 2014. 1, 2, 5

[45] Xiaoshuai Zhang, Abhijit Kundu, Thomas Funkhouser, Leonidas Guibas, Hao Su, and Kyle Genova. Nerflets: Local radiance fields for efficient structure-aware 3D scene representation from 2D supervision. In *CVPR*, pages 8274–8284, 2023. 2

[46] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. GO-SLAM: Global optimization for consistent 3D instant reconstruction. In *ICCV*, pages 3727–3737, 2023. 1, 3, 5, 6, 8, 10, 14, 15, 16, 17, 18, 19, 20

[47] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural implicit scalable encoding for SLAM. In *CVPR*, 2022. 1, 2, 5, 6, 8, 14, 15, 16, 17, 18, 19, 20

6. Supplementary Video

In the supplementary video (https://roym899.github.io/videos/ngm_supplementary.mp4), we first showcase the result on the Kintinuous scene visualizing the field centers and their movement during loop closure. The second part shows a comparison with an ablated, single-field version of our method (supplementing Fig. 6 in the main paper). In the last part, we show the mapping on the large-scale `apt0` scene which includes two smaller loop closures. More videos can be found on our project page https://kth-rpl.github.io/neural_graph_mapping/.

7. Limitations and Discussion

While our approach enables efficient integration of loop closure into the volumetric map, it is not without drawbacks. In particular, the multi-field representation is less memory-efficient compared to monolithic neural field representations, since those can use the available network capacity relatively unconstrained, whereas this adaptiveness is constrained to the local spheres in our approach. Sharing hash tables among multiple fields might be one direction to reduce the memory overhead.

Another downside of our method is that the neural scene representation is currently not used to improve the SLAM result and a tighter integration of dense mapping and sparse tracking could lead to improved robustness. A similar limitation applies to GO-SLAM [46], which also does not use the neural map for pose estimation. In [16] this property is referred to as "decoupled". While "coupled" approaches that use the map for frame-to-map alignment appear elegant they are not without drawbacks. Current methods for frame-to-map alignment using neural representation have a small basin of convergence and hence only work for slow sequences with small baselines. They also typically rely on depth measurements to be available. In contrast, feature-based methods can potentially estimtate poses even for large baselines and can directly benefit from progress in image matching. In the future we want to investigate how to combine sparse features and neural maps in a principled way.

8. Method Details

8.1. Sampling Strategy

As described in Sec. 3.3 and illustrated in Fig. 4 in the main paper, a three-stage sampling procedure is used. First, a subset of fields is sampled, then rays are sampled for each field, and finally points are sampled along each ray. In the following we describe the details of each stage.

Fields Especially with larger scenes, it is important that recently added fields and those currently observed are optimized with a higher rate than out-of-view fields that have already been optimized before. To achieve this, the currently observed fields $\mathcal{F}_t^{\text{obs}} \subseteq \mathcal{F}_t$ are determined and sampled with a higher probability. Specifically, a total of N_f fields are sampled; half from the currently observed fields and the remaining ones from all fields $\mathcal{F}_t$ discarding duplicates.

Rays To sample supervision targets, each sampled field i is approximated by a set of points $\boldsymbol{q}_j^i, j = 1, ..., N_{\text{approx}}$ sampled uniformly on the field's sphere of radius r. These points are projected into all keyframes. A field is considered visible in a keyframe, if at least one of the field's points $\boldsymbol{q}_j^i$ is inside the keyframe's frustum and the projected depth of $\boldsymbol{q}_j^i$ is smaller than the observed depth at the projected 2D point. This yields a set of keyframes $\mathcal{K}_t^i \subseteq \mathcal{K}_t$. N_r rays per field (i.e., target rays) are then sampled via the 2D bounding boxes of the projected points $\boldsymbol{q}_j^i$ in the keyframes $\mathcal{K}_t^i$. For each target ray $(\boldsymbol{o}, \boldsymbol{d})$ the closest point to the field center is computed as $\boldsymbol{o} + l_\text{c}\boldsymbol{d}$ and only a ray segment $[l_\text{c} - r, l_\text{c} + r]$ covering the sphere will be considered for optimization.

Points Given a ray segment $[l_{\text{min}}, l_{\text{max}}]$, N_{up} points are uniformly sampled across the segment, and N_{dp} points are uniformly sampled in the truncation interval τ around the observed depth, that is, in the interval $[l_{\text{obs}} - \tau, l_{\text{obs}} + \tau]$; the full ray interval is used, if there is no depth measurement or $l_{\text{obs}} \notin [l_{\text{min}}, l_{\text{max}}]$. This yields a total of $N_\text{p} = N_{\text{up}} + N_{\text{dp}}$ query points per ray segment during optimization.

In total, each optimization iteration will contain a maximum of $N_\text{f} N_\text{r} N_\text{p}$ query points.

8.2. K-Nearest Neighbors Queries

We compute the color and signed distance at a query point as the weighted average of the k nearest fields. Specifically, let $\boldsymbol{x} \in \mathbb{R}^3$ denote the query point. Let $\boldsymbol{c}_i, s_i, d_i, i = 1, ..., k$ denote the returned color, signed distance, and distance to the field center for the k nearest fields for the query point $\boldsymbol{x}$. We compute weights based on the softmax of the negative distances, that is,

$$u_i = \frac{e^{-\xi d_i}}{\sum_{j=1}^{k} e^{-\xi d_j}}, \tag{9}$$

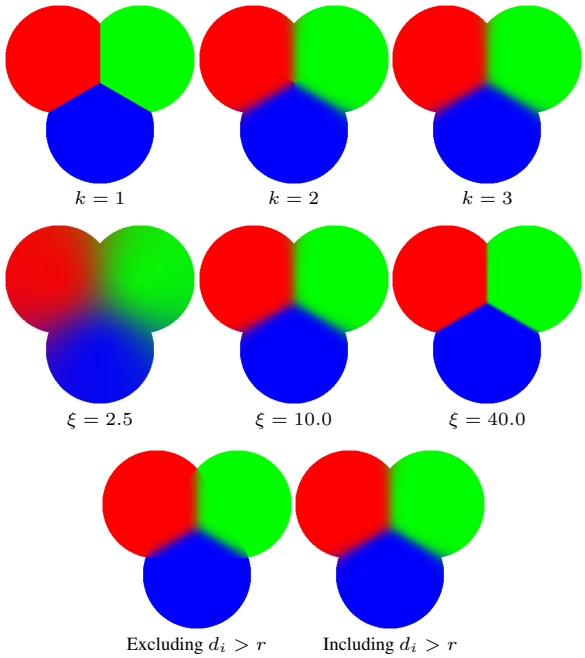


Figure 9. Visualization of k-nearest neighbor distance-based averaging. The top row shows the effect of varying k. The second row shows the effect of varying ξ. The last row shows the effect of excluding fields with distances d_i greater than the field radius r from the averaging.

where ξ determines the transition speed. The combined color and signed distance are then computed as a weighted sum, that is, $\boldsymbol{c} = \sum_{i=1}^{k} u_i \boldsymbol{c}_i$ and $s = \sum_{i=1}^{k} u_i s_i$.

We always use the k nearest fields even when only the closest field is within radius r. However, we set ξ sufficiently high such that the transition region becomes small and fields with $d_i \gg r$ will have no significant contribution to the final value. This is a feasible strategy, since fields are optimized for all ray segments intersecting them even when the segment is terminating outside the sphere. Hence, each field will in practice capture a region larger than a sphere with radius r.

Figure 9 illustrates the effect of this weighted averaging for different values of ξ, k, and with and without $u_i = 0$ for $d_i > r$ on a 2D toy example with three fields of fixed color. Note that the distance-weighted averaging leads to smooth transitions in the overlapping regions. When forcing $u_i = 0$ for $d_i > r$ transitions on the boundaries are unavoidable, hence we opt for the strategy described in the previous paragraph. For the experiments we use $k = 2$ and $\xi = 10$.

8.3. Parameters

In Tab. 3 we provide a full list of parameters, the used value to achieve the experimental results, and a brief description. Parameters were tuned manually and the same setting is used for all experiments (with the exception of τ,

Table 3. Overview of parameters.

Parameter	Value	Description
r	$1\,\mathrm{m}$	Field radius
N_f	2	Number of fields optimized in parallel in each iterations
N_r	512	Number of ray segments sampled per field during optimization
N_up	8	Number of uniformly sampled points distributed along each ray segment during optimization
N_dp	16	Number of depth-guided points distributed along each ray segment during optimization
τ	$0.1\,\mathrm{m}$ or $0.2\,\mathrm{m}^*$	Truncation distance; used for scaling depth-guided sampling and for capping the supervision range (i.e., dividing samples into free-space samples and TSDF samples)
η	20.0	Determines how fast occupancy probability decays around surfaces
k	2	Number of nearest neighbors used during evaluation
ξ	10.0	Distance weighing determining transition speed between two fields
L	1	Number of MLP layers following the permutohedral encoding; excluding the final linear layer
T	2^{12}	Hash table size for permutohedral encoding
N_levels	16	Number of resolution levels for permutohedral encoding
N_fpl	16	Number of features per level for permutohedral encoding
r_coarse	0.1	Coarsest resolution for permutohedral encoding
r_fine	0.0001	Finest resolution for permutohedral encoding
λ_color	1.0	Weight of color loss
λ_depth	1.0	Weight of depth loss
λ_fs	40.0	Weight of free-space loss
λ_tsdf	50.0	Weight of TSDF loss
δ	$5\,\mathrm{cm}$	Huber loss threshold
γ	1×10^{-3}	Learning rate used for Adam optimizer [13]
λ	1×10^{-5}	Weight decay used for Adam optimizer [13]

* The truncation distance is increased for real-world datasets to account for the increase in depth noise.

which is increased for the real-world datasets).

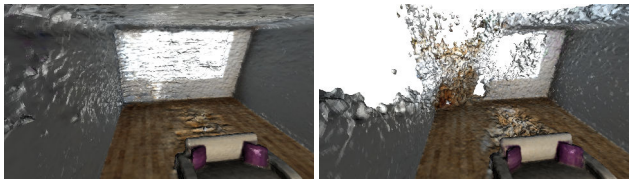
9. Experiment Details

Baseline Setup All baselines are evaluated using the parameters published as part of the published code. For additional datasets for which no parameters were provided, the most similar dataset's parameters were adopted (i.e., for Replica-Big the provided setup for Replica is used; for Kintinuous the setup for ScanNet). Scene boundaries were manually adjusted to cover the observed area with extra margin to account for errors in positioning.

In our experiments, we noticed that Co-SLAM [42] uses the ground-truth pose of the first frame to initialize the SLAM system, which leads to axis-aligned planes. We found that planes (such as walls, floors, and ceilings) which are axis-aligned are significantly better completed using the one-blob encoding [24] than generally-oriented planes. Therefore, for a fair comparison, we modified Co-SLAM's implementation to start from a random orientation instead. We note that this mainly reduces qualitative scene completion, however, on one of the Replica scenes it leads to tracking issues and hence poor reconstruction results. Figure 10 shows an example of the scene completion capability of Co-SLAM with and without ground-truth initialization (i.e., with and without axis-aligned planes).

9.1. Evaluation Protocol

Let ${}^{\mathrm{w}}\mathcal{M}_\mathrm{gt}$ and ${}^{\tilde{\mathrm{w}}}\mathcal{M}_\mathrm{est}$ denote the ground-truth mesh and estimated mesh, respectively. We assume that ${}^{\tilde{\mathrm{w}}}\mathcal{M}_\mathrm{est}$ has already been globally aligned with ${}^{\mathrm{w}}\mathcal{M}_\mathrm{gt}$, which is typically achieved by either aligning the first frame in the sequence or by aligning the trajectories using the Umeyama algorithm. Starting from ${}^{\mathrm{w}}\mathcal{M}_\mathrm{gt}$ and ${}^{\tilde{\mathrm{w}}}\mathcal{M}_\mathrm{est}$ further preprocessing steps are performed before the evaluation metrics are computed.



(a) With axis-alignment (b) Without axis-alignment

Figure 10. Co-SLAM result with and without ground-truth initialization. Ground-truth initialization leads to axis-aligned walls and floors, which in turn leads to significantly better scene completion.

1. Unobserved parts of the ground-truth mesh ${}^{\mathrm{w}}\mathcal{M}_\mathrm{gt}$ are removed. In particular, we apply two removal steps. First, vertices falling more than $2\,\mathrm{cm}$ outside the scene bounding box are removed. The scene bounding box is computed as the intersection of a manually-set bounding box (used to exclude outliers in the depth map present in some scenes) and an automatically computed bounding box (based on the ground-truth trajectory and depth maps). Second, vertices that are not in front or up to $3\,\mathrm{cm}$ behind any rendered depth map are removed. These depth maps are rendered from the ground-truth trajectory and from additional virtual views manually placed to improve the evaluation of

scene completion (same as in Co-SLAM [42]). This yields a culled ground-truth mesh used for evaluation ${}^{\mathrm{w}}\mathcal{M}^{*}_{\mathrm{gt}}$.

2. To further equalize slight differences in alignment between different methods, we perform another alignment step using point-to-plane-based iterative closest point from ${}^{\tilde{\mathrm{w}}}\mathcal{M}_{\mathrm{est}}$'s vertices to ${}^{\mathrm{w}}\mathcal{M}_{\mathrm{gt}}$'s vertices yielding an aligned estimated mesh ${}^{\mathrm{w}}\mathcal{M}_{\mathrm{est}}$.

3. The aligned estimated mesh ${}^{\mathrm{w}}\mathcal{M}_{\mathrm{est}}$ follows the same removal process as the ground-truth mesh (see step 1 above) yielding the culled estimated mesh used for evaluation ${}^{\mathrm{w}}\mathcal{M}^{*}_{\mathrm{est}}$.

For the evaluation, $N_{\mathrm{samples}} = 200\,000$ points are uniformly sampled on both meshes yielding the point sets $\mathcal{G} = \{\boldsymbol{x}_i \sim \mathcal{U}({}^{\mathrm{w}}\mathcal{M}^{*}_{\mathrm{gt}}) \mid i = 1, ..., N_{\mathrm{samples}}\}$ and $\mathcal{E} = \{\boldsymbol{y}_i \sim \mathcal{U}({}^{\mathrm{w}}\mathcal{M}^{*}_{\mathrm{est}}) \mid i = 1, ..., N_{\mathrm{samples}}\}$, where $\mathcal{U}(\cdot)$ denotes the uniform distribution. The point sets are used to compute accuracy, completion, accuracy ratio, and completion ratio as

$$\mathrm{Acc}(\mathcal{G}, \mathcal{E}) = \frac{1}{|\mathcal{E}|} \sum_{\boldsymbol{y} \in \mathcal{E}} \min_{\boldsymbol{x} \in \mathcal{G}} \|\boldsymbol{y} - \boldsymbol{x}\| \tag{10}$$

$$\mathrm{Comp}(\mathcal{G}, \mathcal{E}) = \frac{1}{|\mathcal{G}|} \sum_{\boldsymbol{x} \in \mathcal{G}} \min_{\boldsymbol{y} \in \mathcal{E}} \|\boldsymbol{x} - \boldsymbol{y}\| \tag{11}$$

$$\mathrm{AR}(\mathcal{G}, \mathcal{E}) = \frac{1}{|\mathcal{E}|} \sum_{\boldsymbol{y} \in \mathcal{E}} \left[\min_{\boldsymbol{x} \in \mathcal{G}} \|\boldsymbol{y} - \boldsymbol{x}\| < \Delta \right] \tag{12}$$

$$\mathrm{CR}(\mathcal{G}, \mathcal{E}) = \frac{1}{|\mathcal{G}|} \sum_{\boldsymbol{x} \in \mathcal{G}} \left[\min_{\boldsymbol{y} \in \mathcal{E}} \|\boldsymbol{x} - \boldsymbol{y}\| < \Delta \right], \tag{13}$$

where $[\cdot]$ denotes the Iverson bracket and $\Delta = 5\,\mathrm{cm}$ in our experiments. Since accuracy ratio and completion ratio can be interpreted as precision and recall of the reconstruction, we further use the F1-score

$$F_1(\mathcal{G}, \mathcal{E}) = \frac{2}{\mathrm{AR}(\mathcal{G}, \mathcal{E})^{-1} + \mathrm{CR}(\mathcal{G}, \mathcal{E})^{-1}} \tag{14}$$

to summarize reconstruction performance in one metric.

10. Additional Results

Detailed Results on Replica and NRGBD Table 4 and Table 5 shows per-scene results on the Replica and NRGBD dataset, respectively. Overall, Co-SLAM achieves the best result, albeit with small margins; it fails on `room1` leading to worse average results. Loopy-SLAM achieves near perfect accuracy at the cost of worse completion. As noted in the paper it uses the ground-truth depth for rendering and mesh extraction and is hence not fairly comparable to the other methods. Overall, this experiment highlights that our contributions aiming for improved loop closure integration do not significantly worsen performance on small scenes.s

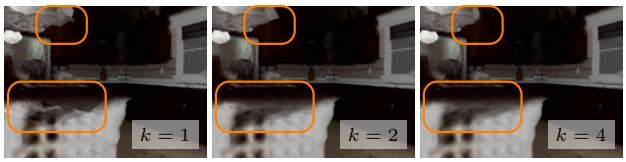


Figure 11. Comparison of rendered views with different number of neighbors k. By increasing the number of nearest neighbors taken into account, the number of visible field transitions decreases. These transitions are most visible in unobserved regions, such as the underside of the countertop shown here (see highlighted regions).

Additional Qualitative Results In Figs. 12 to 16 additional qualitative results on the Replica, NRGBD, and ScanNet datasets are shown. In most scenes our approach performs close to the best performing method Co-SLAM. Co-SLAM achieves slightly more detailed and smoother results, which might be due to their more effective hash table use and the additional smoothness loss. Loopy-SLAM sometimes produces meshes with wrong colors; MIPS-Fusion shows transition and streaking artifacts; and GO-SLAM produces more noisy results particularly in terms of appearance.

Our method performs most consistent compared to other methods that support loop closure.

K-Nearest Neighbor Averaging Figure 11 shows renderings of our model with varying values for k (see Sec. 8.2). For higher k visible transitions between fields are reduced and smoothed out. Note that due to our independent training scheme, all fields are trained in overlapping regions making averaging at the query point a viable strategy. While higher values for k lead to improved results, it also multiplies the number of queries required for rendering and mesh extraction (note that optimization time is unaffected by k).

Table 4. Comparison of mesh quality on Replica (**best** ●, second best ●, third best ●).

		offi0	offi1	offi2	offi3	offi4	room0	room1	room2	Avg.
NICE-SLAM [47]	Acc (cm)	1.90	1.61	3.13	2.92	2.60	2.47	2.21	2.17	2.38
	Acc.-Ratio (%)	94.87	95.30	89.78	90.17	**93.21**	93.38	94.92	**93.75**	**93.17**
	Comp. (cm)	2.37	2.15	2.89	3.42	3.91	2.93	2.31	2.77	2.84
	Comp. Ratio (%)	92.58	92.15	88.78	86.20	85.72	90.90	**93.57**	90.97	90.11
	F1-Score (%)	93.71	93.70	89.28	88.14	89.31	92.12	**94.24**	92.34	91.60
Co-SLAM [42]	Acc (cm)	**1.55**	**1.33**	2.76	**2.61**	**2.22**	**1.99**	19.90	**1.92**	4.28
	Acc.-Ratio (%)	96.15	96.75	90.92	**92.04**	92.70	**95.37**	38.68	93.51	87.02
	Comp. (cm)	**1.54**	**1.68**	**2.39**	**2.73**	**2.47**	**2.37**	17.47	**2.08**	4.09
	Comp. Ratio (%)	**96.04**	**94.57**	**91.99**	**90.92**	**90.96**	**93.43**	40.03	**93.16**	86.39
	F1-Score (%)	**96.09**	**95.65**	91.46	**91.48**	**91.82**	**94.39**	39.34	**93.34**	86.70
GO-SLAM [46]	Acc (cm)	1.88	1.72	3.51	3.97	3.56	3.45	**2.15**	2.90	2.89
	Acc.-Ratio (%)	**96.31**	**97.55**	84.49	79.64	86.34	84.59	**96.04**	89.50	89.31
	Comp. (cm)	3.12	4.17	6.74	7.43	8.39	6.83	4.26	8.46	6.17
	Comp. Ratio (%)	85.21	82.89	69.40	63.81	67.33	69.44	82.89	72.65	74.20
	F1-Score (%)	90.42	89.63	76.20	70.85	75.66	76.27	88.98	80.20	81.03
MIPS-Fusion [39]	Acc (cm)	1.65	1.61	3.16	3.43	2.62	2.30	2.64	2.52	2.49
	Acc.-Ratio (%)	96.16	95.41	91.21	88.96	89.99	94.95	93.76	90.60	92.63
	Comp. (cm)	1.78	1.86	2.99	3.22	2.67	2.78	**2.13**	2.69	2.52
	Comp. Ratio (%)	94.96	93.09	89.16	88.40	89.64	92.37	93.48	89.92	**91.38**
	F1-Score (%)	95.56	94.24	90.17	88.68	89.82	93.64	93.62	90.26	92.00
Loopy-SLAM [16]	Acc (cm)	1.05	0.84	1.33	1.53	1.49	1.45	1.13	1.20	1.25
	Acc.-Ratio (%)	100.00	100.00	100.00	99.98	99.99	99.99	100.00	99.99	99.99
	Comp. (cm)	1.65	2.16	3.64	3.03	3.80	3.45	2.95	2.65	2.92
	Comp. Ratio (%)	93.43	90.80	86.33	88.67	86.32	88.58	89.78	90.32	89.28
	F1-Score (%)	96.61	95.18	92.66	93.98	92.65	93.94	94.62	94.91	94.32
Ours-SF	Acc (cm)	1.55	1.79	2.49	3.09	2.38	2.12	2.52	2.53	**2.31**
	Acc.-Ratio (%)	96.25	94.20	92.51	89.94	92.83	94.51	94.59	90.08	93.11
	Comp. (cm)	1.59	2.01	2.66	2.94	2.89	2.46	2.23	3.16	**2.49**
	Comp. Ratio (%)	95.88	92.32	90.65	89.31	88.24	91.92	93.20	88.75	91.28
	F1-Score (%)	96.07	93.25	**91.57**	89.63	90.47	93.20	93.89	89.41	**92.19**
Ours	Acc (cm)	1.60	1.86	**2.46**	2.81	2.36	2.14	2.73	2.85	2.35
	Acc.-Ratio (%)	95.74	94.20	**92.52**	90.53	92.81	94.46	92.55	89.26	92.76
	Comp. (cm)	1.67	2.00	2.76	2.78	3.01	2.71	2.16	4.00	2.64
	Comp. Ratio (%)	95.27	91.74	89.65	89.01	88.02	91.88	93.41	87.96	90.87
	F1-Score (%)	95.50	92.95	91.06	89.76	90.35	93.15	92.98	88.61	91.80

Table 5. Comparison of mesh quality on NRGBD (**best** ●, second best ●, third best ●).

		br	ck	gr	gwr	ki	ma	sc	tg	wa	Avg.
NICE-SLAM [47]	Acc (cm)	2.46	10.76	2.33	2.71	9.18	1.70	4.55	8.37	7.37	5.49
	Acc.-Ratio (%)	92.41	65.34	93.87	93.63	57.29	95.06	71.69	56.22	77.38	78.10
	Comp. (cm)	4.82	14.21	3.91	3.19	12.82	3.36	10.69	8.02	5.39	7.38
	Comp. Ratio (%)	86.18	53.58	86.66	87.64	50.04	84.06	58.79	59.24	69.04	70.58
	F1-Score (%)	89.19	58.88	90.12	90.54	53.42	89.22	64.60	57.69	72.97	74.07
Co-SLAM [42]	Acc (cm)	2.21	4.73	1.89	2.02	7.40	1.74	3.30	2.07	6.24	3.51
	Acc.-Ratio (%)	93.24	75.16	95.17	94.84	77.72	93.98	78.01	92.05	84.92	87.23
	Comp. (cm)	**2.06**	8.76	2.93	**2.41**	5.14	**2.75**	4.29	**2.83**	3.85	**3.89**
	Comp. Ratio (%)	93.49	63.17	**91.32**	**93.96**	**78.19**	**86.41**	70.90	**86.62**	81.70	82.86
	F1-Score (%)	93.37	68.65	**93.21**	**94.40**	77.96	**90.03**	74.29	**89.26**	83.28	84.94
GO-SLAM [46]	Acc (cm)	3.89	4.08	2.50	2.87	3.28	**1.54**	6.46	**1.48**	5.46	3.51
	Acc.-Ratio (%)	77.64	81.94	91.46	86.87	84.74	**97.44**	66.62	**96.32**	73.90	84.10
	Comp. (cm)	9.25	29.60	9.50	4.50	**5.11**	4.60	12.35	7.26	12.57	10.53
	Comp. Ratio (%)	64.29	54.56	71.31	75.12	72.58	75.53	54.24	70.78	57.33	66.19
	F1-Score (%)	70.34	65.51	80.14	80.57	78.19	85.10	59.80	81.59	64.57	73.98
MIPS-Fusion [39]	Acc (cm)	2.44	3.48	1.94	2.10	9.24	1.72	3.49	1.61	3.30	3.26
	Acc.-Ratio (%)	91.20	84.69	94.53	95.76	75.62	93.68	75.92	93.59	89.61	88.29
	Comp. (cm)	5.05	35.03	8.33	3.07	8.91	3.15	8.63	3.61	11.52	9.70
	Comp. Ratio (%)	83.49	55.36	83.09	89.38	70.86	82.81	64.98	79.77	72.89	75.85
	F1-Score (%)	87.18	66.95	88.44	92.46	73.16	87.91	70.02	86.13	80.39	81.40
Loopy-SLAM [16]	Acc (cm)	1.44	3.41	1.93	2.00	20.41	1.14	2.59	✗	3.02	4.49
	Acc.-Ratio (%)	99.85	85.01	98.99	98.52	61.61	99.93	94.86	✗	85.87	90.58
	Comp. (cm)	3.24	13.57	3.41	3.14	8.31	3.17	3.39	✗	4.40	5.33
	Comp. Ratio (%)	90.29	71.09	88.58	89.86	68.23	83.98	85.03	✗	76.88	81.74
	F1-Score (%)	94.83	77.43	93.50	93.99	64.75	91.26	89.67	✗	81.13	90.58
Ours-SF	Acc (cm)	**1.68**	3.57	1.82	1.95	2.28	1.79	**2.33**	2.55	3.10	2.34
	Acc.-Ratio (%)	**94.57**	86.96	95.12	**96.34**	95.24	93.90	**95.66**	89.75	90.47	**93.11**
	Comp. (cm)	2.16	**7.49**	**2.86**	2.64	25.86	3.10	**3.84**	3.79	**3.73**	6.16
	Comp. Ratio (%)	**94.87**	**78.60**	90.95	91.76	74.19	84.16	**88.77**	82.27	**83.45**	**85.45**
	F1-Score (%)	**94.72**	**82.57**	92.99	93.99	**83.41**	88.76	**92.09**	85.85	**86.82**	**89.02**
Ours	Acc (cm)	1.77	**3.19**	**1.78**	**1.94**	**2.18**	1.78	2.42	2.65	**2.92**	**2.29**
	Acc.-Ratio (%)	93.83	**88.67**	**95.50**	96.30	**96.00**	94.24	94.95	87.40	**90.90**	93.09
	Comp. (cm)	2.46	8.98	2.88	2.67	29.26	3.25	4.33	3.33	4.18	6.82
	Comp. Ratio (%)	93.84	76.89	90.83	91.31	73.69	82.80	87.90	83.53	81.51	84.70
	F1-Score (%)	93.83	82.36	93.11	93.74	83.38	88.15	91.29	85.42	85.95	88.58

Figure 12. Qualitative comparison of final meshes extracted by all methods on the Replica dataset [35, 36] (part 1).

Figure 13. Qualitative comparison of final meshes extracted by all methods on the Replica dataset [35, 36] (part 2).

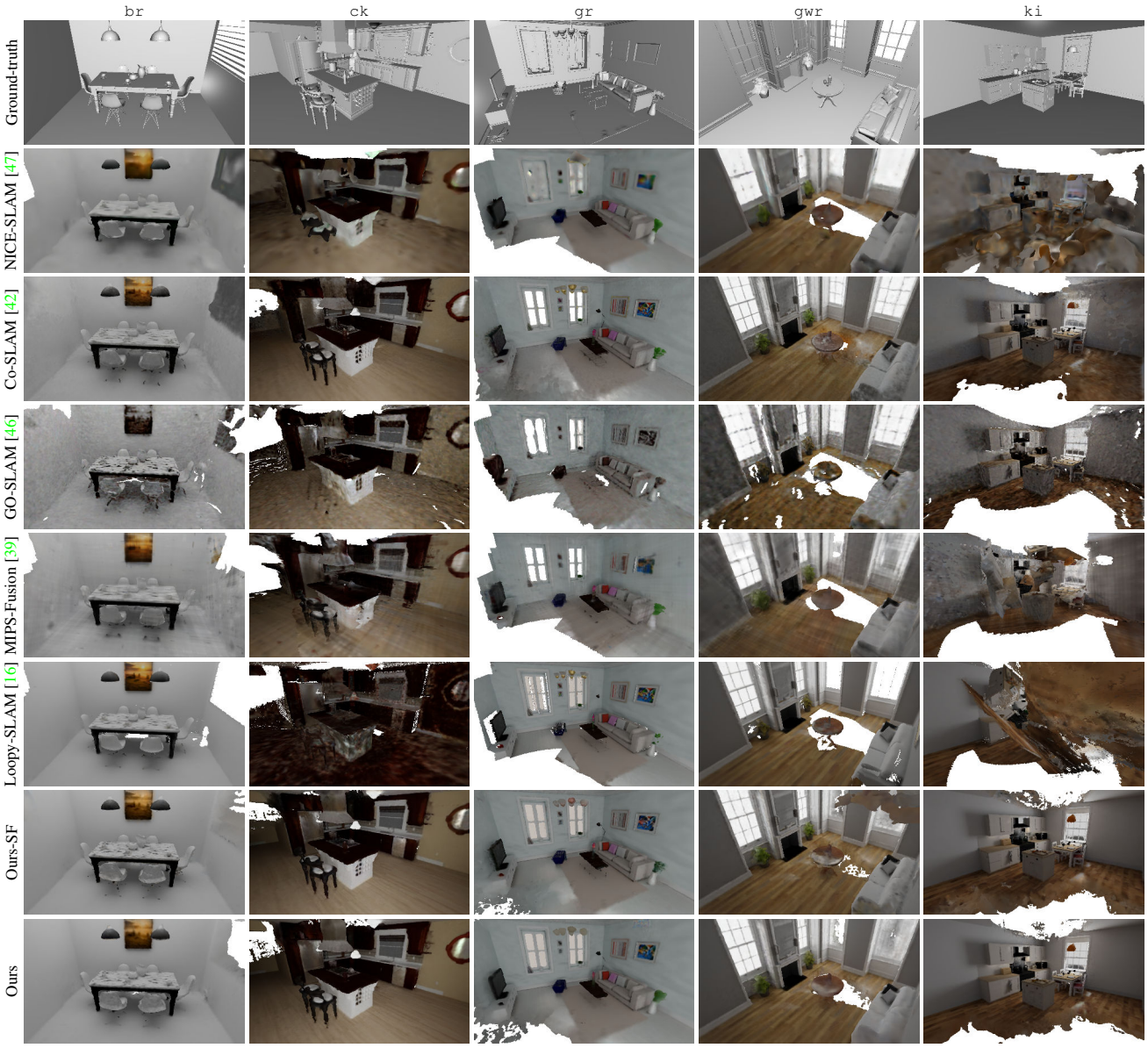


Figure 14. Qualitative comparison of final meshes extracted by all evaluated methods on the NRGBD dataset [2] (part 1).

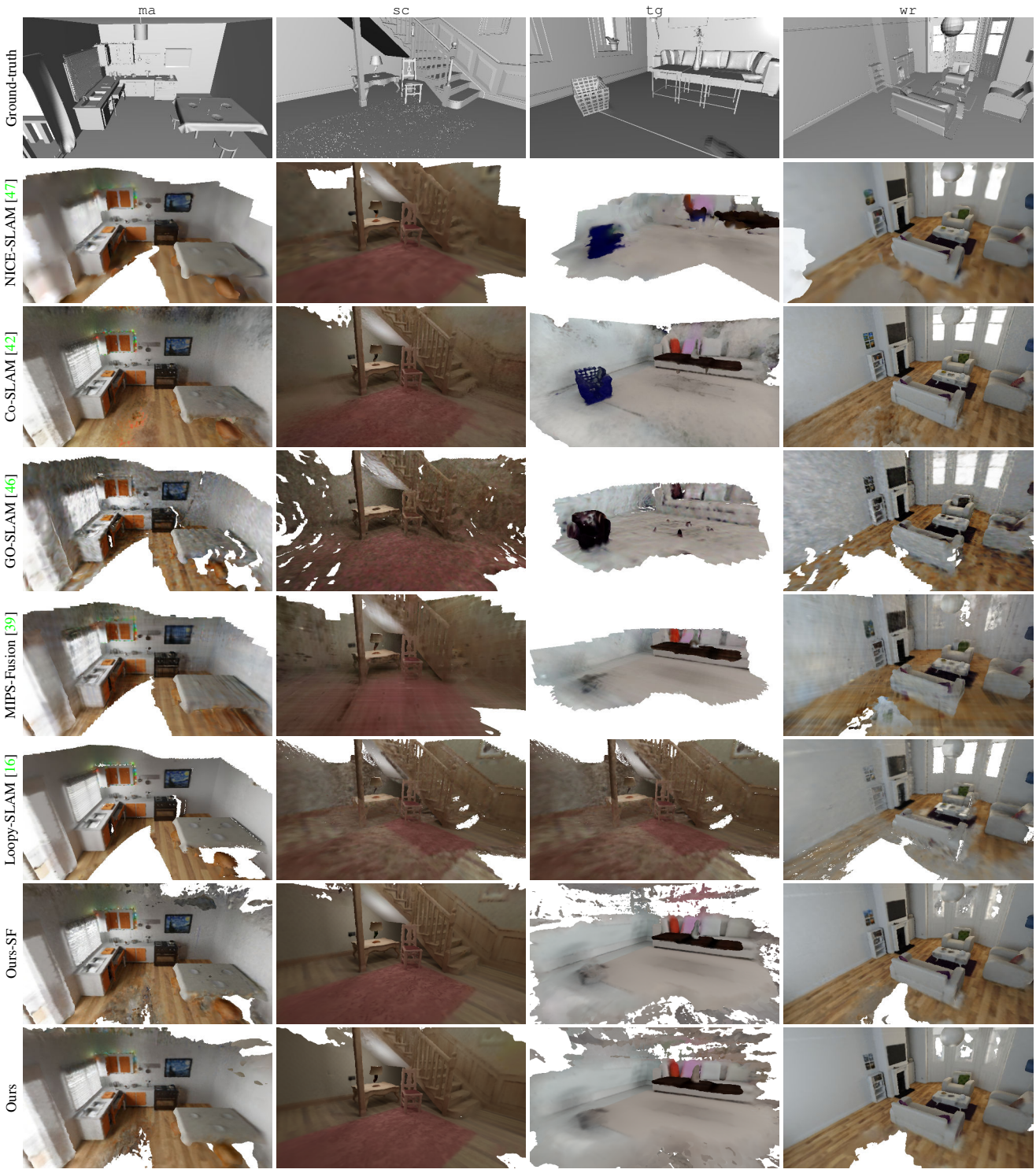


Figure 15. Qualitative comparison of final meshes extracted by all evaluated methods on the NRGBD dataset [2] (part 2).

Figure 16. Qualitative comparison of final meshes extracted by all methods on the ScanNet dataset [5]. Our method fails on `scene0181_00` due to tracking issues in the underlying SLAM system in a feature-less region.