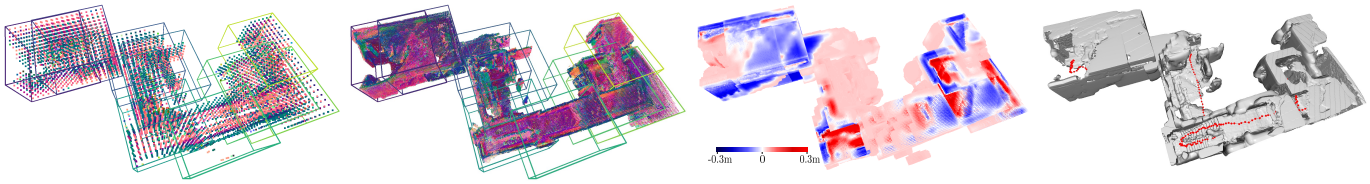
MISO: Multiresolution Submap Optimization for Efficient Globally Consistent Neural Implicit Reconstruction

Yulun Tian, Hanwen Cao, Sunghwan Kim and Nikolay Atanasov
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093, USA
Email: {yut034,h1cao,suk063,natanasov}@ucsd.edu



(a) Coarse level features (b) Fine level features (c) SDF reconstruction (d) Mesh and trajectory estimates

Fig. 1: Demonstration of MISO on the FastCaMo-Large dataset [1]. MISO leverages *multiresolution submaps* that organize neural implicit features at different spatial resolutions (visualized in (a) and (b) using principal component analysis). By performing hierarchical optimization within (local) and across (global) submaps, MISO can efficiently and accurately reconstruct SDF (c) and estimate mesh and robot trajectory (d). For clarity, we only visualize the features and SDF values within 30 cm of the surface. The scene size is 26.0 m $\times$ 16.7 m $\times$ 7.5 m.

***Abstract*—Neural implicit representations have had a significant impact on simultaneous localization and mapping (SLAM) by enabling robots to build continuous, differentiable, and high-fidelity 3D maps from sensor data. However, as the scale and complexity of the environment increase, neural SLAM approaches face renewed challenges in the back-end optimization process to keep up with runtime requirements and maintain global consistency. We introduce MISO, a hierarchical optimization approach that leverages *multiresolution submaps* to achieve efficient and scalable neural implicit reconstruction. For local SLAM within each submap, we develop a hierarchical optimization scheme with learned initialization that substantially reduces the time needed to optimize the implicit submap features. To correct estimation drift globally, we develop a hierarchical method to align and fuse the multiresolution submaps, leading to substantial acceleration by avoiding the need to decode the full scene geometry. MISO significantly improves computational efficiency and estimation accuracy of neural signed distance function (SDF) SLAM on large-scale real-world benchmarks.**

I. Introduction

In recent years, *neural fields* [2] have emerged as a new frontier for scene representation in simultaneous localization and mapping (SLAM). Compared to conventional approaches based on hand-crafted features or volumetric representations, neural SLAM [3] offers advantages including continuous and differentiable scene modeling, improved memory efficiency, and better handling of measurement noise. However, a crucial limitation remains in the back-end optimization of neural SLAM: as the environment size and mission time grow, most existing approaches consider increasingly larger optimization problems that ultimately limit their real-time performance.

A powerful idea to achieve more efficient SLAM is to depend on a *hierarchical* representation that explicitly disentangles coarse and fine information of the environment. Equipped with such a hierarchical model, a robot can perform inference over varying spatial resolutions, *e.g.*, by first capturing the core structure in the environment and then optimizing the fine details later. In SLAM, this idea dates back to several seminal works such as [4]–[6]. Recently, hierarchical or multiresolution representations have also achieved success in neural fields [7]–[9], demonstrating state-of-the-art performance and cost-quality trade-offs in many computer vision tasks. Nevertheless, neural SLAM systems have yet to benefit from these recent advances, as the majority of back-end solvers do not fully utilize the hierarchical form of the representations.

In this work, we develop a *hierarchical optimization* approach that directly uses multiresolution implicit features for neural SLAM. This approach enables us to solve a significant portion of the back-end optimization in the *implicit feature space*, and thus obtain substantial gains in efficiency and robustness compared to existing methods that depend on geometric reconstruction [1], [10]. To scale to larger scenes, we adopt a submap-based design that models the environment as a collection of local neural implicit maps. In this context, we show that the proposed hierarchical optimization significantly enhances both *local submap optimization* and *global submap fusion* stages in SLAM. We apply our approach to neural signed distance function (SDF) SLAM [11] and demonstrate its effectiveness on real-world, large-scale datasets.

Contributions. We present MISO (MultIresolution Submap Optimization), a hierarchical optimization approach for neural implicit SLAM. MISO performs local pose and submap optimization and global submap fusion, which can be used to achieve SDF SLAM from depth images or LiDAR scans. For *local* submap optimization, MISO introduces a learning-based hierarchical initialization method to generate multiresolution

submap features, which are subsequently refined through joint optimization with robot poses. As a theoretical motivation, we derive a closed-form solution to the initialization problem under the special case of linear least squares optimization. Leveraging this theoretical insight, we design hierarchical encoders to learn effective initializations in the general case. For *global* submap fusion, MISO presents a hierarchical optimization method to align and fuse submaps in the global frame. Compared to previous works, our approach achieves faster and more robust performance by directly using information stored in the hierarchical implicit features rather than relying on geometric reconstruction. Evaluation on benchmark datasets shows that MISO achieves competitive estimation quality and significantly outperforms existing methods in computational efficiency. Fig. 1 demonstrates MISO on the real-world FastCaMo-Large dataset [1].

Notation. Unless stated otherwise, lowercase and uppercase letters denote vectors and matrices, respectively. We define $[n] \triangleq \{1, 2, \ldots, n\}$ as the set of positive integers from 1 to n. The special Euclidean group in 3D is denoted by $\mathrm{SE}(3)$, and $\mathrm{SE}(3)^n$ denotes its product manifold. A local perturbation on the tangent space of $\mathrm{SE}(3)$ is represented by a vector $\varepsilon \in \mathbb{R}^6$. The exponential map $\mathrm{Exp} : \mathbb{R}^6 \to \mathrm{SE}(3)$ is given by $\mathrm{Exp}(\varepsilon) = \exp([\varepsilon]_\times)$, where $[\varepsilon]_\times \in \mathfrak{se}(3)$ is the Lie algebra element corresponding to ε and $\exp$ is the standard matrix exponential. The inverse of the exponential map is denoted as $\mathrm{Log} : \mathrm{SE}(3) \to \mathbb{R}^6$. Given $T = (R, t) \in \mathrm{SE}(3)$ and a point $x \in \mathbb{R}^3$, $Tx = Rx + t \in \mathbb{R}^3$ denotes the transformed point.

II. Related Work

We review related work on neural implicit representations for SLAM, with particular focus on neural SDF reconstruction and submap decompositions. The reader is referred to recent surveys [2], [3] for further discussions and reviews of alternative representations including 3D Gaussian splatting [12].

A. Neural Implicit Representations

Neural implicit representations offer continuous and differentiable modeling of 3D scenes with high fidelity, memory efficiency, and robustness to noise [13]–[16]. Early methods such as DeepSDF [13] and NeRF [15] rely solely on 3D coordinates and a single multi-layer perceptron (MLP) to reconstruct the scene. However, this approach is insufficient for capturing larger scenes or complex details, prompting subsequent works to introduce hybrid methods that combine MLP decoders with additional implicit features. The implicit features are commonly organized in a 3D grid [7]–[9], [17]. To enable continuous scene modeling, trilinear interpolation is used to infer a feature at an arbitrary query location that is subsequently passed through the MLP decoder to predict the environment model (*e.g.*, occupancy, distance, radiance). Recent works propose several alternative approaches to improve the memory efficiency over 3D feature grids. K-Planes [18] factorizes the scene representation into multiple 2D feature planes rather than using a full 3D voxel grid. Similarly, TensoRF [19] employs tensor decomposition to compactly represent radiance fields. PointNeRF [20] constructs the scene representation directly from point clouds by efficiently aggregating local features at surface points.

Hierarchical strategies for organizing the implicit features have been particularly effective at capturing different levels of detail while maintaining efficiency [7]–[9], [21], [22]. DVGO [8] performs progressive scaling that gradually increases the feature grid resolution during training. InstantNGP [9] significantly accelerates feature grid training and inference by introducing a multiresolution hash encoding scheme. Neuralangelo [21] extends this concept with a coarse-to-fine optimization scheme that preserves fine-grained details. In parallel, octree-based frameworks provide an adaptive representations for large scenes by varying spatial resolution where needed [7], [22]. H_2-Mapping [23] achieves incremental neural SDF mapping by combining octree-based coarse SDF and multiresolution feature grids, where the latter is optimized to learn residual geometry. Hierarchical representations have also been explored for fast RGB-D surface reconstruction [16], [24]. In this work, we leverage these hierarchical neural representations to achieve efficient and accurate back-end optimization for neural SLAM.

B. Neural SDF SLAM

Recent SLAM systems have achieved remarkable progress by modeling the environment using neural implicit SDF. Building on DeepSDF [13], iSDF [11] uses a single MLP for online SDF reconstruction from streaming RGB-D data. iSDF selects keyframes based on information gain [25] and samples free-space and near-surface points along camera rays to train the MLP. VoxFusion [26] leverages a sparse octree to organize implicit features and Morton coding for efficient allocation and retrieval, enabling real-time SLAM in dynamically expanding environments. Vox-Fusion++ [10] extends the method to large-scale scenes through submap support. NICER-SLAM [27] transforms estimated SDF to density for volume rendering during monocular SLAM. NeRF-LOAM [28] similarly uses SDF to represent the geometry for neural lidar odometry and mapping, and develops a dynamic voxel embeddings generation method to speed up octree queries. PIN-SLAM [29] reconstructs SDF via sparse neural point features, and employs voxel hashing to speed up spatial querying for online SLAM. PINGS [30] is a concurrent work that extends PIN-SLAM to enable photorealistic rendering via Gaussian Splatting. The neural point features are decoded to spawn Gaussian primitives locally, and trained with both SDF- and GS-based losses to enhance geometric consistency. ESLAM [31] uses multi-scale axis-aligned tri-plane feature grids with a TSDF representation to achieve memory-efficient reconstruction and localization. Co-SLAM [32] combines smooth one-blob coordinate encoding with local-detail hash-grid embeddings to improve camera tracking. GO-SLAM [33] supports loop closing and online bundle adjustment with a multi-resolution hash-grid design for both SDF and color. Despite these advancements, achieving globally consistent SDF reconstruction of large-scale scenes remains challenging. In this work, we address this limitation

by developing a hierarchical approach for both local and global multiresolution submap optimization.

C. Submap-based Neural SLAM

An effective strategy for large-scale 3D reconstruction is to partition the scene into multiple submaps. Kähler *et al.* [34] create submaps storing truncated SDF values based on visibility criteria and align them by optimizing the relative poses of overlapping keyframes. MIPS-Fusion [1] extends this idea by incrementally generating MLP-based submaps based on the camera's field of view and aligning them via point-to-plane refinement. Vox-Fusion++ [10] adopts a dynamic octree structure for each submap and performs joint camera tracking and submap alignment by optimizing a differentiable rendering loss. Loopy-SLAM [35] uses a neural-point-based approach, creating submaps upon large camera rotations and later constructing a pose graph with iterative closest point (ICP) to detect loop closures. More recently, PLGSLAM [36] combines axis-aligned tri-planes for high-frequency features with an MLP for low-frequency components, enabling multiple local representations to be merged efficiently. NEWTON [37] employs a spherical coordinate system to create local maps that accommodate flexible boundary adjustments. Multiple-SLAM [38] and CP-SLAM [39] consider collaborative scenarios and fuse local neural implicit maps from multiple agents. Although effective, existing methods require reconstructing the scene's geometry to align submaps, which can be costly and inaccurate in real-world settings. In contrast, our method aligns submaps directly in the feature space via hierarchical optimization, providing both fast and robust performance without explicit geometric reconstruction.

III. Overview

In MISO, we represent the scene as a collection of posed submaps. Correspondingly, the back-end optimization involves two types of problems: (i) local SLAM within each submap and (ii) global alignment and fusion across all submaps. See Fig. 2 for an illustration.

Given odometry and point-cloud observations from depth images or LiDAR scans, a robot aims to estimate its trajectory and build a local map represented as a multiresolution feature grid (Fig. 2a). Organizing implicit features into a *hierarchy of grids* effectively disentangles information at different spatial resolutions. At inference time, interpolated features from different hierarchy levels are aggregated and processed by a *decoder network* to predict the scene geometry. To speed up local optimization, we introduce *hierarchical encoder networks* to initialize the grid features at each hierarchy level directly from input observations. To achieve further acceleration and enable generalization to new environments, both the encoder and decoder networks are pre-trained offline over multiple scenes and fixed during online SLAM. Sec. IV presents in detail our local SLAM method.

In large environments or over long time durations, the robot trajectory estimates will inevitably drift and cause the submaps to be misaligned. To address this challenge, MISO introduces an approach to align and fuse all submaps in the global reference frame (Fig. 2b). Each submap is associated with a *base pose* that determines the transformation from the local (submap) frame to the global frame. Compared to existing approaches, which rely on decoding the scene geometry into an explicit representation like occupancy, mesh, or distance field, MISO performs alignment and fusion directly using the implicit features in the multiresolution submaps. We show that this results in significantly faster optimization and outperforms other methods under large initial alignment errors. Sec. V presents the details of the global alignment and fusion method.

IV. Local SLAM

This section introduces our submap representation utilizing multiresolution feature grids and our hierarchical submap optimization method.

A. Local SLAM with Multiresolution Feature Grid

We represent each local submap as a multiresolution feature grid [7]–[9], defined formally below.

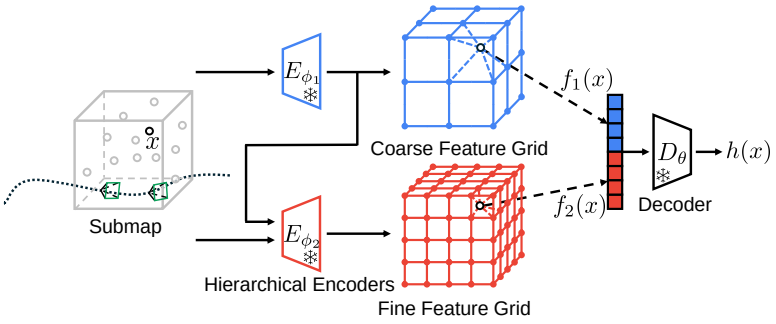
Definition 1 (Multiresolution Feature Grid)**.** A *multiresolution feature grid* contains $L > 1$ levels of regular grids with increasing spatial resolution ordered from coarse ($l = 1$) to fine ($l = L$). At each level l, each vertex located at $z_{l,i} \in \mathbb{R}^3$ stores a learnable feature vector $f_{l,i} \in \mathbb{R}^d$. Together with a kernel function $k_l : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$, the feature grid defines a continuous feature field $f_l(x) = \sum_{i \in I_l} k_l(x, z_{l,i}) f_{l,i}$, where $x \in \mathbb{R}^3$ is any query position, and I_l indexes over all vertices at level l. To obtain a scalar output (*e.g.*, signed distance or occupancy) at query position x, the features at different levels are concatenated (denoted by $\bigoplus$) and processed by a decoder network D_θ,

$$h(x; F, \theta) = D_\theta\big(\bigoplus_{l \in [L]} f_l(x)\big). \tag{1}$$

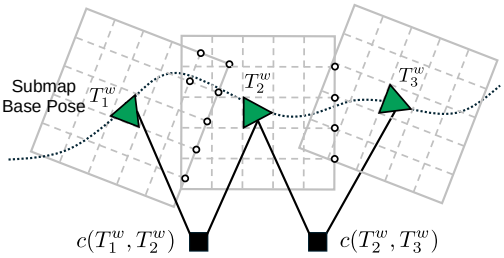
The model has the set of features from all levels F and the decoder parameters θ as learnable parameters.

In this work, we implement the kernel functions k_l using trilinear interpolation. While the multiresolution feature grid offers a powerful representation, directly using it as a map representation in SLAM presents a computational challenge due to the need to train the decoder network D_θ. Even if computation is not a concern, training the decoder with a small dataset or during a single SLAM session may lead to unreliable generalization or catastrophic forgetting [3].

To address these challenges, we pre-train the decoder D_θ offline over multiple scenes, similar to prior works (*e.g.*, [29], [40]). The details of the offline decoder training are presented in Appendix A. During online SLAM, the decoder weights are fixed (as shown in Fig. 2a), and the robot only needs to optimize the grid features F and its own trajectory. Specifically, within each submap s, we are given noisy pose estimates $\{\hat{T}_k^s\}_k$ (*e.g.*, from odometry) and associated observations $\{X^k\}_k$, where each $X^k = \{x_1^k, \dots, x_{m_k}^k\} \subset \mathbb{R}^3$ is a point cloud observed at pose k. Using this information, we



(a) Local mapping with multiresolution feature grid



(b) Global submap alignment and fusion

Fig. 2: **Overview of MISO.** (a) Given point cloud observations, MISO performs local hierarchical SLAM within a submap represented as a multiresolution feature grid (Sec. IV). (b) Given locally optimized submaps, MISO performs global alignment and fusion across submaps to eliminate estimation drift and achieve globally consistent scene reconstruction (Sec. V).

seek to jointly refine the robot's pose estimates and the submap features F via the following optimization problem.

Problem 1 (Local SLAM within a submap)**.** Given n initial pose estimates $\{\hat{T}_k^s\}_k$ in the reference frame of submap s and associated point-cloud observations $\{X^k\}_k$ in the sensor frame, the local SLAM problem is defined as,

$$\min_{F,\{T_k^s\}_k \subset \mathrm{SE}(3)} \sum_{k=1}^{n} \sum_{j=1}^{m_k} c_j\big(h(T_k^s x_j^k; F)\big) + \sum_{k=1}^{n} \rho(\widehat{T}_k^s, T_k^s), \quad (2)$$

where $c_j : \mathbb{R} \to \mathbb{R}$ is a cost function associated with the j-th observation and $\rho : \mathrm{SE}(3) \times \mathrm{SE}(3) \to \mathbb{R}$ is a pose regularization term. We drop the dependence of the model h on the decoder parameters θ to reflect that the decoder is trained offline.

The first group of terms in (2) evaluates the environment reconstruction at observed points x_j^k by transforming them to the submap frame (*i.e.*, $x_j^s = T_k^s x_j^k$) and querying the feature grid model h. Empirical results show that introducing the second group of pose regularization terms helps the optimization remain robust against noisy or insufficient observations. We use regularization inspired by trust-region methods [41, Ch. 4],

$$\rho(\widehat{T}, T) = w^\rho \max\big(\big\|\mathrm{Log}(\widehat{T}^{-1}T)\big\|_2 - \tau, 0\big), \quad (3)$$

which penalizes pose updates larger than the trust-region radius τ, and w^ρ is a weight parameter (default to 10^3).

In our implementation, we solve Problem 1 approximately by parametrizing each pose variable locally as $T_k^s = \widehat{T}_k^s \,\mathrm{Exp}(\varepsilon_k^s)$ where $\varepsilon_k^s \in \mathbb{R}^6$ is the local pose correction. Both the grid features F and the correction terms $\{\varepsilon_k^s\}_k$ are optimized using Adam [42] in PyTorch [43].

We introduce definitions of the cost c_j specific to neural SDF reconstruction next. Whenever clear from context, to ease the notation we use $x_j \equiv x_j^s = T_k^s x_j^k$ to represent a point in the submap frame.

Cost functions for neural SDF reconstruction. We follow iSDF [11] to design measurement costs for SDF reconstruction. Specifically, we classify all observed points as either (i) on or near surface (default to 30 cm as in iSDF), or (ii) in free space. For on or near surface observations, the cost function $c_j = c_j^{\mathrm{sdf}}$ is based on direct SDF supervision,

$$c_j^{\mathrm{sdf}}\left(h(x_j)\right) = w_j^{\mathrm{sdf}} \left|h(x_j) - y_j\right|, \quad (4)$$

where $w_j^{\mathrm{sdf}} > 0$ is measurement weight (default to 5.4 as in iSDF) and $y_j \in \mathbb{R}$ is a measured SDF value on or near surface obtained using the approach from iSDF [11].

For free-space observations, we use the cost to enforce bounds on the SDF values. Specifically, we follow iSDF to obtain lower and upper bounds $\underline{b}_j, \bar{b}_j$ on the SDF from sensor measurements, and define $c_j = c_j^{\mathrm{bnd}}$ as,

$$c_j^{\mathrm{lo}}\left(h(x_j)\right) = \max(e^{\beta(\underline{b}_j - h(x_j))} - 1,\ 0), \quad (5)$$

$$c_j^{\mathrm{up}}\left(h(x_j)\right) = \max(h(x_j) - \bar{b}_j,\ 0), \quad (6)$$

$$c_j^{\mathrm{bnd}}\left(h(x_j)\right) = \max(c_j^{\mathrm{lo}}\left(h(x_j)\right), c_j^{\mathrm{up}}\left(h(x_j)\right)). \quad (7)$$

This cost applies exponential penalty ($\beta = 5$ by default) for the lower bound and linear penalty for the upper bound. This is because, in practice, violation of the lower bound is usually more critical, *e.g.*, if $\underline{b}_j = 0$ and the model predicts negative SDF values. We do not include Eikonal regularization [44] because we observed that it has limited impact on accuracy while making the optimization slower.

B. Hierarchical Feature Initialization for Local SLAM

In practice, the bulk of the computational cost in Problem 1 is incurred by the optimization over the high-dimensional grid features F. To address this challenge, we propose a method that leverages the structure of the multiresolution grid to learn to initialize F from sensor observations. While prior works such as Neuralangelo [21] advocate for coarse-to-fine training strategies, a crucial gap remains since the features at each level are still optimized from scratch, *e.g.*, from zero or random initialization. Our key intuition is that, at any level, a much more effective initialization can be obtained by accounting for optimization results from the previous levels.

In the following, we use F_l to denote the subset of latent features at level l, and $F_{1:l}$ denote all latent features up to and including level l. We consider the problem of initializing F_l

Algorithm 1 HIERARCHICAL LOCAL SLAM

function $\{T_k^s\}_k, F$ = HIERARCHICALLOCALSLAM
  for level $l = 1, 2, \ldots, L$ **do**
    Initialize features at level l: $F_l \leftarrow E_{\phi_l}(r_{1:l-1}(x))$.
  end for
  From the initialized values, jointly update features F and poses $\{T_k^s\}_k$ by minimizing (2).
  return $\{T_k^s\}_k$ and F.
end function

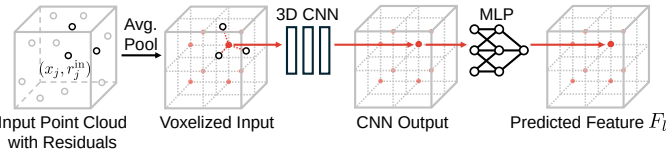


Fig. 3: Illustration of the level-l encoder E_{ϕ_l}. Input point cloud with residuals $\{x_j, r_j^{\text{in}}\}_j$ is voxelized via averaging pooling and processed by a 3D CNN. The CNN outputs at all vertices are then transformed via a shared MLP to predict the target feature grid F_l.

given fixed submap poses and coarser features $F_{1:l-1}$. This amounts to solving the following subproblem of Problem 1,

$$\min_{F_l} \quad \sum_{k=1}^{n} \sum_{j=1}^{m_k} c_j \left(h(T_k^s x_j^k; F_{1:l-1}, F_l, 0_{l+1:L}) \right), \tag{8}$$

where we explicitly expand F into the (known) coarser features $F_{1:l-1}$, the target feature to be initialized F_l, and finer features (assumed to be zero). During initialization, we do not consider pose optimization and thus drop the trust-region regularization in Problem 1.

To develop our approach, we first present theoretical analysis and derive a closed-form solution to (8) in a special linear-least-squares case. Leveraging insight from the closed-form solution in the linear case, we then develop a learning approach to initialize the grid features at each level, applicable to the general (nonlinear) problem in (8).

Special case: linear least squares. Consider the special case where the decoder D_θ in Definition 1 is a linear function. Further, assume that the cost function c_j in (8) is quadratic, *e.g.*, $c_j(h(x_j)) = (h(x_j) - y_j)^2$. For instance, this would correspond to using squared norm for the SDF cost in (4). Under these assumptions, problem (8) is a linear least squares problem, for which we can obtain a closed-form solution from the normal equations, as shown next.

Proposition 1 (Linear least squares)**.** *With linear decoder D_θ and quadratic costs $c_j(h(x_j)) = (h(x_j) - y_j)^2$, the optimal solution to (8) is:*

$$F_l^\star = E(r_{1:l-1}(x)) := -\left[J^\top J\right]^\dagger J^\top r_{1:l-1}(x), \tag{9}$$

where $x = \{T_k^s x_j^k\}$ and $y = \{y_j\}$ collect all observed points and labels in two vectors, $J = \partial h(x; F)/\partial F_l$ is the Jacobian matrix evaluated at x, and $r_{1:l-1}(x)$ are the residuals of prior levels, represented in vector form as,

$$r_{1:l-1}(x) = h(x; F_{1:l-1}, 0_{l:L}) - y. \tag{10}$$

Observe that the residual vector $r_{1:l-1}(x)$ is mapped to the least-squares solution $F_l^\star$ by a linear function, which we denote as $E(\cdot)$.

Proof: Please refer to Appendix B. ■

Proposition 1 reveals an interesting structure of the optimal initialization $F_l^\star$: namely, it can be interpreted as a function of the prior levels' residuals $r_{1:l-1}(x)$. We will build on this insight to approach the problem in the general case.

General case: learning hierarchical initialization. We take inspiration from Proposition 1 to develop a learning-based solution for the general case, where the decoder is nonlinear (*e.g.*, an MLP) and the measurement costs are generic functions. Motivated by the previous insight, we propose to replace the linear mapping E in Proposition 1 with a neural network E_{ϕ_l} to approximate $F_l^\star$ from the residuals $r_{1:l-1}(x)$,

$$F_l^\star \approx E_{\phi_l}(r_{1:l-1}(x)), \tag{11}$$

where ϕ_l are the neural network parameters. We refer to E_{ϕ_l} as an *encoder* due to its similarity to an encoding module used by prior works such as Convolutional Occupancy Networks [45] and Hierarchical Variational Autoencoder [46]. In this work, we train a separate encoder E_{ϕ_l} to initialize the feature grid F_l at each level l. Given the learned encoder networks, we apply them to initialize the multiresolution feature grid progressively in a coarse-to-fine manner, before jointly optimizing all levels together with the robot trajectories, as shown in Algorithm 1.

Next, we present the details of our encoder network for neural SDF reconstruction. The input to the encoder is represented as a point cloud. For each 3D position $x_j \in \mathbb{R}^3$ in the submap frame, we use the following measurement residuals to construct an initial feature vector $r_j^{\text{in}} = \begin{bmatrix} r_{j,1}^{\text{in}} & r_{j,2}^{\text{in}} & r_{j,3}^{\text{in}} \end{bmatrix}^\top \in \mathbb{R}^3$:

$$r_{j,1}^{\text{in}} = \begin{cases} h(x_j) - y_j, & \text{if } x_j \text{ near surface,} \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

$$r_{j,2}^{\text{in}} = \begin{cases} \max(h(x_j) - \bar{b}_j, 0), & \text{if } x_j \text{ in free space,} \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

$$r_{j,3}^{\text{in}} = \begin{cases} \max(\underline{b}_j - h(x_j), 0), & \text{if } x_j \text{ in free space,} \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

The first feature $r_{j,1}^{\text{in}}$ corresponds to the SDF residual (4). The remaining two features correspond to the residuals of upper and lower bounds used to compute (7). The initial point features are pooled onto a 3D voxel grid with the same resolution as the target feature grid at level l. Each 3D vertex stores the average residual features from points nearby. A vertex's feature is set to be zero if there are no nearby points. This voxelized input is then passed through a small 3D convolutional neural network (CNN). Finally, the CNN outputs at all vertices are passed through a shared MLP to predict the target feature grid F_l. Fig. 3 shows a conceptual illustration, and Fig. 5 in the experiments shows example inputs and predictions on a real-world dataset. Similar to the

decoder, we train the encoders offline using submaps from multiple environments. In particular, when training the level l encoder E_{ϕ_l}, we use a training loss based on (8),

$$\min_{\phi_l} \sum_{s\in S}\sum_{j\in J_s} c_j \left(h(x_j^s; F_{1:l-1}^s, E_{\phi_l}(r_{1:l-1}^s), 0_{l+1:L})\right), \quad (15)$$

where S contains the indices of all training submaps, J_s contains the indices of all points in submap s, and F^s denotes the features for submap s. During training, we use noisy poses within the submaps to compute x_j^s to account for the possible pose estimation errors at test time.

C. Extension to Incremental Processing

The local SLAM optimization formulated in (2) can be performed in an incremental manner. We describe an implementation inspired by PIN-SLAM [29] and present corresponding evaluation on outdoor datasets in Sec. VI-D. At each time step k, the received depth image or LiDAR scan introduces new cost terms $\{c_j\}_{j=1}^{m_k}$ in (2). We then alternate between tracking and mapping to update the estimated robot pose and the neural implicit submap. During tracking, we only optimize the current robot pose T_k^s and keep the submap features F fixed. Specifically, we only use surface observations and further perform voxel downsampling (voxel size 0.6 m in Sec. VI-D). We estimate the robot motion by minimizing (2) with respect to T_k^s, where the pose regularization term is disabled. Following PIN-SLAM, a Geman-McClure robust kernel is also applied to improve robustness against outlier measurements. During mapping, we fix all pose estimates $T_{1:k}^s$ and only optimize the submap features F. Voxel downsampling is similarly applied but with a smaller voxel size (0.08 m in Sec. VI-D). In the incremental setting, the encoder initialization is disabled. Instead, we use the latest frame and 10 evenly spaced historical frames to update the submap features F by minimizing (2).

V. Global Submap Alignment and Fusion

As the robot navigates in a large environment or for an extended time, its onboard pose estimation will inevitably drift. To achieve globally consistent 3D reconstruction (*e.g.*, after loop closures), it is imperative to accurately align and fuse the submaps in the global frame. Many state-of-the-art systems, such as MIPS-Fusion [1] and Vox-Fusion++ [10], employ approaches that align submaps using learned SDF values. However, this is computationally expensive and susceptible to noise. In this section, we address this limitation by developing a hierarchical method for submap alignment and fusion, which attains significant speed-up by directly performing optimization using the features from the multiresolution submaps.

Hierarchical Submap Alignment. Consider the problem of aligning a collection of n_s submaps, each represented as a multiresolution feature grid from Sec. IV. For each submap $u \in [n_s]$, we aim to optimize the submap base pose in the world frame, denoted as $T_u^w \in \mathrm{SE}(3)$. The key intuition for our approach is that, for any pair of submaps to be well aligned, their implicit feature fields should also be aligned in the global frame. We present our *hierarchical* and *correspondence-free* approach to exploit this intuition. Our method performs alignment by progressively including features at finer levels. In the following, let $f_l^u(x) \in \mathbb{R}^d$ denote the level-l interpolated feature at query position x in submap u. Furthermore, let $f_{1:l}^u(x)$ denote the result after concatenating features up to and including level l, *i.e.*, $f_{1:l}^u(x) = \bigoplus_{l'=1}^{l} f_{l'}^u(x) \in \mathbb{R}^{ld}$.

Algorithm 2 Hierarchical Submap Alignment

```
function {T_u^w}_u = SubmapAlignment
    Initialize submap poses {T_u^w}_u.
    for level l = 1, 2, ..., L do
        Update {T_u^w}_u by solving (17) at level l for k_{f,l} iters.
    end for
    Update {T_u^w}_u by solving (19) for k_s iters.
    return {T_u^w}_u.
end function
```

Consider a pair of overlapping submaps $u, v \in [n_s]$. Let $\{z_{l,i}^u\} \subset \mathbb{R}^3$ denote the vertex positions at level l in submap u. Using these vertices, we define the following pairwise cost to align features,

$$c_l^{\text{feat}}(T_u^w, T_v^w) = \sum_{i\in I_l^{uv}} \mathbf{d}\big(f_{1:l}^u(z_{l,i}^u), f_{1:l}^v((T_v^w)^{-1}(T_u^w)z_{l,i}^u)\big). \quad (16)$$

In (16), I_l^{uv} denotes the indices of level-l vertices in submap u that lie within the overlapping region of the two submaps. Intuitively, the right-hand side of (16) compares feature vectors interpolated from the two submaps. The first feature comes from the source grid u evaluated at its grid vertex position $z_{l,i}^u$. To evaluate the corresponding feature in the target grid v, we use the submap base poses to transform the vertex position, *i.e.*, $z_{l,i}^v = (T_v^w)^{-1}(T_u^w)z_{l,i}^u$ before querying the target feature grid. Finally, $\mathbf{d}$ denotes a distance metric in the space of implicit features. In our implementation, we use the L2 distance, *i.e.*, $\mathbf{d}(f, f') = \|f - f'\|_2^2$ as we find it works well empirically. Sec. VI-E presents an ablation study on alternative choices of $\mathbf{d}$.

Given the pairwise alignment costs defined in (16), MISO performs joint submap alignment by formulating and solving a problem similar to pose graph optimization. Let $\mathcal{E}$ denote the set of submap pairs with overlapping regions. Then, we jointly optimize all submap poses $\{T_u^w\}_u \subset \mathrm{SE}(3)$ as follows.

Problem 2 (Level-l submap alignment)**.** Given n_s submaps with current base pose estimates $\{\widehat{T}_u^w\}_u$, solve for updated submap base poses via,

$$\min_{\{T_u^w\}_u \subset \mathrm{SE}(3)} \sum_{(u,v)\in\mathcal{E}} c_l^{\text{feat}}(T_u^w, T_v^w) + \sum_{u=1}^{n_s} \rho(\widehat{T}_u^w, T_u^w), \quad (17)$$

where ρ is the trust-region regularization defined in (3).

Similar to local SLAM, we solve (17) using PyTorch [43] where the poses are updated by optimizing local corrections (represented in exponential coordinates) to the initial pose estimates $\{\widehat{T}_u^w\}_u$. Our formulation naturally leads to a sequence of alignment problems that include features at increasingly fine levels. We propose to solve these problems sequentially, using

solutions from level l as the initialization for level $l+1$; see lines 3-5 in Algorithm 2.

The hierarchical, feature-based method presented above achieves robust and sufficiently accurate submap alignment. To further enhance accuracy, we may finetune the submap pose estimates during a final alignment stage using predicted SDF values. Since only a few iterations are needed in typical scenarios, this approach allows us to preserve computational efficiency compared to other methods that directly use SDF for alignment. We define the following SDF-based pairwise alignment cost for submap pair (u, v),

$$c^{\text{sdf}}(T_u^w, T_v^w) = \sum_{j \in J^{uv}} \left(h^u(x_j^u; F^u) - h^v((T_v^w)^{-1} T_u^w x_j^u; F^v) \right)^2. \tag{18}$$

In (18), J^{uv} contains the indices of observed points that are in the intersection region of the two submaps. For each observation j, $x_j^u \in \mathbb{R}^3$ denotes its position in the frame of submap u. Compared to (16), in (18) we minimize the squared difference of the final SDF predictions from both submaps. Using this in the pose-graph formulation leads to an SDF-based submap alignment.

Problem 3 (SDF-based submap alignment)**.** Given n_s submaps with base pose estimates $\{\widehat{T}_u^w\}_u$, solve for updated submap base poses via,

$$\min_{\{T_u^w\}_u \subset \text{SE}(3)} \sum_{(u,v)\in\mathcal{E}} c^{\text{sdf}}(T_u^w, T_v^w) + \sum_{u=1}^{n_s} \rho(\widehat{T}_u^w, T_u^w), \tag{19}$$

where ρ is the trust-region regularization defined in (3).

In Algorithm 2, the SDF-based submap alignment is performed at the end to finetune the submap base poses (see line 6). In Sec. VI-E, we demonstrate that the combination of feature-based and SDF-based submap alignment yields the best performance in terms of both robustness and computational efficiency.

Submap Fusion. So far, we addressed the problem of aligning submaps in the global frame to reduce estimation drift. In some applications, there is an additional need to extract a global representation (*e.g.*, a SDF or mesh) of the entire environment from the collection of local submaps. In MISO, we achieve this by using the average feature from all submaps to decode the global scene. For any submap u, let $f^u(x^u)$ denote the output of its multiresolution feature field evaluated at a position $x^u \in \mathbb{R}^3$ in the submap frame. Given any query coordinate in the world frame $x^w \in \mathbb{R}^3$, we first compute the weighted average of all submap features,

$$f^w(x^w) = \Big(\sum_{u=1}^{n_s} w_u(x^w)\Big)^{-1} \sum_{u=1}^{n_s} w_u(x^w) f^u((T_u^w)^{-1} x^w), \tag{20}$$

where each submap is associated with a binary weight $w_u(x^w)$ computed using its bounding box,

$$w_u(x^w) = \begin{cases} 1 & \text{if } x^w \text{ is inside submap } u\text{'s bounding box,} \\ 0 & \text{otherwise.} \end{cases}$$

The final prediction is obtained by passing the average feature to the decoder network,

$$h^w(x^w) = D_\theta(f^w(x^w)). \tag{21}$$

In summary, the proposed scheme achieves submap fusion via an averaging operation in the implicit feature space.

Optionally, the fused prediction in (21) allows one to finetune the estimation by jointly optimizing all submap features and pose variables using global bundle adjustment:

$$\min_{\substack{F^u,\ T_u^w \in \text{SE}(3),\\ \{T_k^u\}_k \subset \text{SE}(3),\ \forall u \in [n_s]}} \sum_{u=1}^{n_s} \sum_{k=1}^{n_u} \sum_{j=1}^{m_k} c_j\big(h^w(T_u^w T_k^u x_j^k)\big). \tag{22}$$

In (22), each c_j is the same cost term induced by a local measurement as in Sec. IV. Each observed local position x_j^k is transformed to the world frame to evaluate the reconstruction. The integers n_s, n_u, m_k denote the number of submaps, the number of robot poses in submap u, and the number of measurements made at robot pose k, respectively. In Sec. VI-D, we show that this global bundle adjustment step allows the method to further improve the reconstruction quality on outdoor datasets.

VI. Evaluation

In this section, we evaluate MISO using several publicly available real-world datasets. Our results show that MISO achieves superior computational efficiency and accuracy compared to state-of-the-art approaches during both local SLAM and global submap alignment and fusion.

A. Experiment Setup

We used four datasets in our experiments: Replica [49], ScanNet [47], FastCaMo-Large [1], and Newer College [50]. Among these datasets, Replica is used to pre-train the encoders and decoder networks offline; see Appendix A for details. Then, the pre-trained weights are used to evaluate MISO on the real-world ScanNet dataset and the large-scale FastCaMo-Large dataset without additional fine-tuning. Lastly, we present a larger scale evaluation on sequences from the outdoor Newer College dataset.

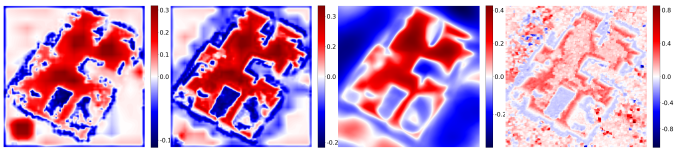
For quantitative evaluations, we compare the multiresolution submaps in MISO against the MLP-based representation from iSDF [11] and the neural-point-based representation from PIN-SLAM [29], using default parameters from their open-source code. In the following, we refer to these two baselines as iSDF and Neural Points, respectively. When evaluating the performance of submap alignment, we introduce two baseline techniques from state-of-the-art submap-based systems. The first is the correspondence-based method from MIPS-Fusion [1], hereafter referred to as MIPS. The second is the correspondence-free method from Vox-Fusion++ [10], hereafter referred to as VFPP. In addition, we compare against an ICP-based method introduced by Choi *et al.* [51] and implemented in Open3D [52]. Given the raw surface points observed in the submaps, this baseline first aligns pairs of submaps via point-to-plane ICP on voxel-downsampled point

TABLE I: Evaluation of local mapping quality for different methods on ScanNet [47]. MISO is optimized for 20 epochs and the baselines iSDF and Neural Points are optimized for 100 epochs. For each scene, we report optimization time (sec), Chamfer-L1 error (cm), and F-score (%) computed using a threshold of 5 cm. Best and second-best results are highlighted in **bold** and underline, respectively.

Scene	0000			0011			0024			0207		
Method	Time↓	C-l1 ↓	F-score ↑	Time↓	C-l1 ↓	F-score ↑	Time↓	C-l1 ↓	F-score ↑	Time↓	C-l1 ↓	F-score ↑
iSDF [11] (GT pose)	67.71	**4.77**	77.87	25.89	6.35	67.32	39.80	**4.98**	73.94	26.17	7.32	59.65
Neural Points [29] (GT pose)	57.87	12.23	40.89	22.41	9.62	49.87	32.84	10.63	46.34	21.29	10.20	44.26
MISO (GT pose)	**1.49**	4.97	**81.04**	**0.80**	**6.07**	**71.23**	**0.95**	5.70	**73.98**	**0.70**	**6.31**	**69.28**
MISO (noisy pose)	7.96	5.36	78.43	4.11	6.85	65.21	4.79	5.89	71.61	3.35	6.86	64.59

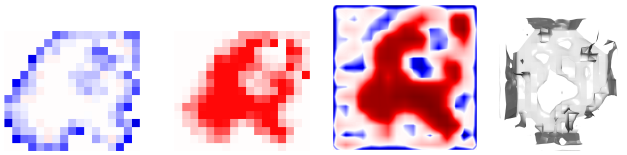
TABLE II: Comparison between local mapping and SLAM on ScanNet [47] using colored ICP odometry as initial guess. For each scene, we report translation RMSE (cm), rotation RMSE (deg), and Chamfer-L1 error (cm).

Scene	0000			0011			0024			0207		
Method	Tran err. ↓	Rot err. ↓	C-l1 ↓	Tran err. ↓	Rot err. ↓	C-l1 ↓	Tran err. ↓	Rot err. ↓	C-l1 ↓	Tran err. ↓	Rot err. ↓	C-l1 ↓
Color ICP [48] + Mapping	40.85	10.17	15.81	17.25	5.35	8.94	18.98	5.04	10.89	19.47	6.25	12.94
Color ICP [48] + SLAM	**12.64**	**5.56**	**10.1**	**8.45**	**3.14**	**7.88**	**9.55**	**3.68**	**8.85**	**11.24**	**4.25**	**9.63**

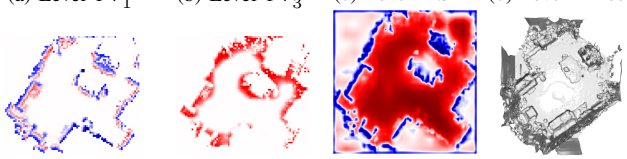


(a) MISO (init) (b) MISO (opt) (c) iSDF (d) Neural Points

Fig. 4: Visualization of estimated SDF at a fixed height on ScanNet scene 0207. MISO performs SLAM using noisy poses. iSDF and Neural Points use ground truth poses and only perform mapping.



(a) Level 1 r_1^{in} (b) Level 1 r_3^{in} (c) Level 1 SDF (d) Level 1 mesh



(e) Level 2 r_1^{in} (f) Level 2 r_3^{in} (g) Level 2 SDF (h) Level 2 mesh

Fig. 5: Inputs and output predictions from learned hierarchical encoders on ScanNet scene 0024. Red and blue show positive and negative residual or SDF values, respectively.

clouds. The aligned submaps are then fused in the global frame via outlier-robust pose graph optimization. Whereas the submaps used by the neural approaches (including ours) have a fine-level resolution of 0.1 m, we allow the ICP baseline to use a higher resolution of 0.02 m and the other parameters are set to default. For MISO, we implement each submap as a two-level multiresolution feature grid with spatial resolutions [0.5 m, 0.1 m] for indoor and [1.0 m, 0.2 m] for outdoor scenes. The feature dimension at each level is set to $d = 4$. All methods are implemented using PyTorch [43]. All experiments are run on a laptop equipped with an Intel i9-14900HX CPU, an NVIDIA GeForce RTX 4080 GPU, and 12 GB of GPU memory.

B. Evaluation on ScanNet Dataset

ScanNet [47] features a collection of real-world RGB-D sequences with accurate camera poses and 3D reconstructions. In the following, we use ScanNet to separately evaluate the proposed local SLAM (Sec. IV) and global alignment and fusion (Sec. V) methods. Joint evaluation is reported in the next subsection on the larger FastCaMo-Large [1] datasets.

Local SLAM evaluation. Our first experiment evaluates the performance of the local optimization approach in MISO (Algorithm 1). Since each scene in ScanNet is relatively small, we represent the entire scene as a single submap. For each scene, we run MISO from initial pose estimates obtained by perturbing the ground truth poses with 3 deg and 5 cm errors. For comparison, we also run MISO and the baseline iSDF and Neural Points methods using ground truth poses, where pose optimization is disabled. Table I reports results on four ScanNet scenes. For each method, we report its GPU time and the mesh reconstruction error against the ground truth, measured in Chamfer-L1 distance and F-score. Both MISO variants are optimized for 20 epochs. For the iSDF and Neural Points baselines, since they do not have access to pre-training, we optimize both for 100 epochs for a fair comparison. As shown in Table I, MISO achieves either the best or second-best reconstruction results on all scenes. Using ground truth poses, MISO achieves superior speed, requiring only 0.7–1.5 sec for optimization. MISO with noisy poses takes longer due to the additional pose estimation but is still significantly faster than the baseline techniques.

Fig. 4 shows a qualitative comparison of the estimated SDF at a fixed height on scene 0207. MISO starts from noisy poses and performs full SLAM, while the baseline methods use ground truth poses and perform mapping only. Fig. 4a shows the initialization produced by the learned encoder (corresponding to line 4 in Algorithm 1), which already captures the scene geometry to a large extent. The remaining errors and missing details are fixed after running 20 optimization epochs, as shown in Fig. 4b. Because iSDF uses a single MLP to represent the scene, its output (see Fig. 4c) is overly smooth, which leads to lower recall and F-score than MISO. Lastly, the SDF prediction from Neural Points (Fig. 4d) is especially

TABLE III: Submap alignment evaluation on ScanNet [47] from initial errors of 5 deg and 0.20 m. For each scene, we report the GPU time (sec), and the final submap rotation error (deg) and translation error (m) compared to ground truth. Results averaged over 10 trials. Best and second-best results are highlighted in **bold** and underline.

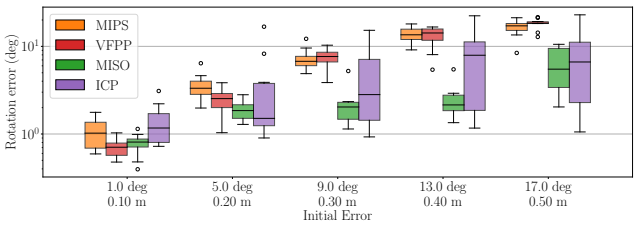
Scene	0011 (159 poses, 4 submaps)			0024 (227 poses, 5 submaps)		
Methods	Time↓	Rot err↓	Tran err↓	Time↓	Rot err↓	Tran err↓
MIPS [1]	59.23	3.93	0.15	123.63	4.52	0.19
VFPP [10]	74.16	2.40	0.11	137.58	**2.24**	**0.08**
MISO	**10.53**	**1.86**	**0.06**	**18.85**	3.14	0.14
ICP [51]	-	2.62	0.19	-	9.18	0.53

noisy in free space, due to the lack of neural point features far away from the surface.

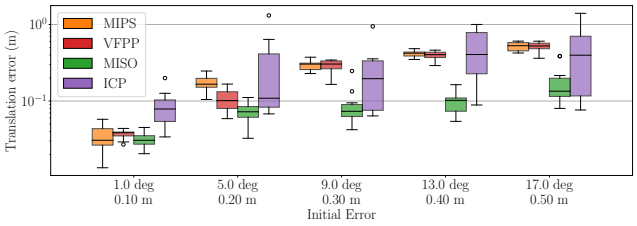
To provide more insight on the performance of the learned hierarchical encoders, Fig. 5 visualizes the inputs and output predictions on scene 0024. For this visualization, we do not use noisy pose estimates. At each level (shown as a row in the figure), we visualize two channels $r_1^{\text{in}}, r_3^{\text{in}}$ of the voxelized input residuals defined in (12) and (14), the predicted SDF, and the predicted 3D mesh. The level 1 encoder, although with a coarse resolution of 0.5 m, already captures the rough scene geometry and free space SDF. On top of this coarse prediction, the level 2 encoder is able to add fine details and recover objects such as the sofa and the table.

Lastly, to evaluate local SLAM under more challenging initial trajectory estimates, we include an experiment with color ICP odometry [48] as the initial guess. When running local SLAM, we let MISO incrementally process a sliding window of 10 frames and disable pose regularization in (2). Table II shows that performing local SLAM substantially improves performance compared to mapping only.

Submap alignment and fusion evaluation. The next set of experiments evaluates the submap alignment and fusion approach in MISO. We run MIPS-Fusion [1] to obtain the submap information. We select scene 0011 and perform local SLAM within all submaps starting from noisy poses with 1 deg and 0.1 m errors. We then evaluate the performance of the proposed alignment (Algorithm 2) and the baseline MIPS [1], VFPP [10], and ICP [51] techniques under increasing submap alignment errors. All methods are run for 100 iterations before evaluating their results. For MISO, we allocate 45 iterations for alignment at each feature level and 10 iterations for final alignment using SDF, which corresponds to setting $k_{f,1} = k_{f,2} = 45$ and $k_s = 10$ in Algorithm 2. All neural methods use the same trust-region-based pose regularization, where the trust-region radius τ in (3) is set based on the initial submap alignment error. For each setting of the initial error shown on the x axis, we perform 10 random trials and show the final results as boxplots in Fig. 6. With a small initial alignment error of 1 deg and 0.1 m, all methods produce accurate alignment results. However, as the initial error increases, both MIPS and VFPP methods quickly fail, showing that solely relying on SDF prediction is very sensitive to the initial guess. The ICP baseline results in higher variance indicating more frequent alignment failures despite the use

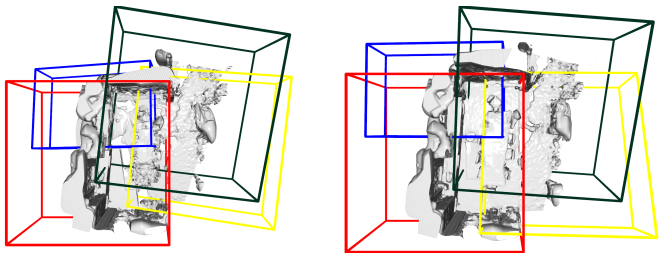


(a) Rotation error (deg)



(b) Translation error (m)

Fig. 6: Evaluation of submap alignment on ScanNet scene 0011 under varying initial errors over 10 trials for each configuration.



(a) Before alignment (b) After alignment

Fig. 7: 3D mesh reconstruction with oriented submap bounding boxes: (a) before submap alignment and (b) after submap alignment on ScanNet scene 0011.

of robust optimization. The proposed hierarchical alignment in MISO is more robust under more severe misalignments and outperforms the baseline methods significantly. Since our experiments are offline, we record the total GPU time and final accuracy. Timing results for ICP are omitted as the method is not implemented on GPU. Table III reports the results on scenes 0011 and 0024. The initial submap alignment error is set to 5 deg and 0.2 m, and results are collected over 10 trials. MISO is significantly faster since the majority of iterations in Algorithm 2 only uses features for alignment and does not need to decode the SDF values. In terms of accuracy, MISO demonstrates competitive performance and achieves either best or second-best results.

Fig. 7 visualizes the reconstructed global mesh on scene 0011 before and after submap alignment. To extract the global SDF, we use the approach presented at the end of Sec. V, which uses the average feature from all submaps to decode the scene in the world frame. Along with the mesh, we also show the oriented bounding boxes of all submaps. As shown in Fig. 7a, the initial mesh is very noisy in areas where multiple submaps overlap. Fig. 7b shows that MISO effectively fixes this error and recovers clean geometry (*e.g.*, the table and

TABLE IV: Evaluation on Newer College dataset [50]. For each scene, we report translation RMSE (m), rotation RMSE (deg), Chamfer-L1 error (cm), and F-score (%) with a threshold of 20 cm. Best and second-best results are highlighted in **bold** and underline, respectively.

Scene	Quad (1991 scans)				Maths Institute (2160 scans)			
Method	Tran err. ↓	Rot err. ↓	C-l1 ↓	F-score ↑	Tran err. ↓	Rot err. ↓	C-l1 ↓	F-score ↑
ICP [52] + MISO Mapping	4.06	11.02	42.0	32.58	3.57	15.66	48.87	24.68
KISS-ICP [53] + MISO Mapping	**0.08**	0.98	**13.96**	82.68	0.24	2.4	**20.65**	63.41
PIN-SLAM [29]	0.10	**0.97**	14.74	78.85	**0.15**	**1.37**	20.66	**70.18**
MISO (Odometry)	0.47	1.58	18.53	68.41	0.25	1.57	23.28	57.78
MISO (Full)	0.10	0.98	14.01	**83.0**	0.22	1.57	20.89	62.07

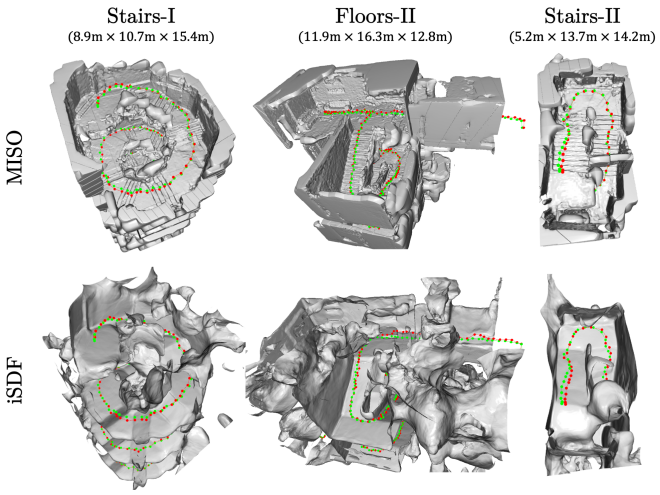


Fig. 8: Qualitative evaluation on the FastCaMo-Large dataset [1]. We show the final reconstructed mesh along with estimated and reference robot trajectories (from [1]) in red and green, respectively.

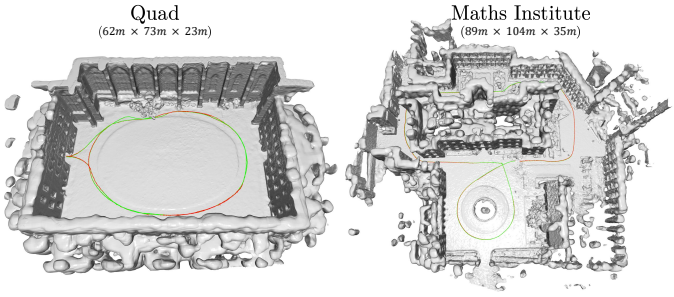


Fig. 9: Qualitative evaluation on the Newer College dataset [50]. We show the final reconstructed mesh along with estimated and reference robot trajectories in red and green, respectively.

chairs) in the global frame.

C. Qualitative Evaluation on FastCaMo-Large Dataset

FastCaMo-Large [1] is a recent dataset with multiple real-world RGB-D sequences. Compared to ScanNet, this dataset features larger indoor environments (up to 200 m^2) and fast camera motions, making it suitable for evaluating the overall accuracy and scalability of our method. Since ground truth is not available, we only focus on qualitative evaluation in this experiment. We compare the performance of MISO against iSDF, where the latter is extended to perform joint optimization over both robot poses and the map. The input pose estimates are obtained by perturbing robot poses estimated by MIPS-Fusion [1] by 3 deg rotation error and 0.05 m translation error in their corresponding submaps, and additionally perturbing all submap base poses by 5 deg rotation error and 0.1 m translation error. The trust region radius used in pose regularization (3) is set correspondingly for both methods. For iSDF, we run optimization for 300 epochs to ensure convergence. For MISO, we run 150 epochs of local SLAM in parallel within all submaps, followed by submap alignment where we skip the coarse level feature alignment as we observe it leads to degraded results on these datasets. Fig. 8 shows qualitative comparisons of the estimated trajectories and meshes on the Floors-II, Stairs-I, and Stairs-II sequences. Fig. 1 shows more visualizations for MISO on the Floors-I sequence. MISO is able to accurately represent these larger-scale scenes while preserving fine details such as staircases and chairs. In comparison, iSDF loses many details and introduces more artifacts in areas where there are fewer observations.

D. Evaluation on Newer College Dataset

In this section, we evaluate MISO on the Quad-Easy and Maths-Easy sequences from the outdoor, LiDAR-based Newer College Dataset [50]. For comparison, we run PIN-SLAM [29], KISS-ICP [53], and point-to-point ICP in Open3D [52] using their open-source implementations and default hyperparameters. For KISS-ICP and point-to-point ICP, we obtain corresponding maps for evaluation by running our mapping pipeline with the pose estimates fixed. For MISO, we first pre-train a decoder for each scene and then use the incremental implementation presented in Sec. IV-C to process each sequence. Point-to-point ICP is used to initialize the tracking optimization at every scan, and a new submap is created every 400 scans. We evaluate two variants of our method in this experiment. The first (Odometry) only performs incremental tracking and mapping. The second (Full) also performs submap alignment and fusion as described in Sec. V, and its results are visualized in Fig. 9. Table IV reports translation RMSE (m), rotation RMSE (deg), Chamfer L1 distance (cm), and F-score (%). In the Quad scene, our method achieves comparable performance with other state-of-the-art methods. In the more challenging Maths Institute scene, MISO performs slightly worse than PIN-SLAM but still on par with KISS-ICP. On both sequences, our results demonstrate that MISO substantially improves over the point-to-point ICP initialization. Further, the comparison between Odometry and Full validates the effectiveness of the proposed

TABLE V: Ablation study on the effect of pre-trained encoder and decoder networks. For each scene, we report F-score (%) and the training loss at epochs 10 and 100. Results are averaged over five runs, and the best outcomes are highlighted in **bold**.

Scene	0011 (10 epoch)		0011 (100 epoch)		0207 (10 epoch)		0207 (100 epoch)	
Method	F-score↑	Loss↓	F-score↑	Loss↓	F-score↑	Loss↓	F-score↑	Loss↓
No-ED	-	0.206	59.3	0.061	-	0.196	58.5	0.056
No-E	52.8	0.139	**69.5**	**0.056**	53.8	0.130	**67.5**	**0.052**
Full	**59.4**	**0.098**	66.3	0.057	**57.8**	**0.089**	66.3	0.052

TABLE VI: Ablation study on hierarchical alignment on ScanNet [47] from large initial errors of 10 deg and 0.25 m. We report the alignment solver time (sec) and the final submap rotation error (deg) and translation error (m) compared to ground truth. Results are averaged over 10 trials. Best results are highlighted in **bold**.

Scene	0011 (4 submaps)			0207 (2 submaps)		
Method	Time↓	Rot err↓	Tran err↓	Time↓	Rot err↓	Tran err↓
Coarse only	**1.20**	4.73	0.16	**0.25**	4.86	0.07
Coarse+Fine	2.54	3.68	0.12	0.50	1.89	0.07
Full	10.13	**2.71**	**0.09**	1.68	**1.11**	**0.05**

submap optimization approach to achieve global consistency.

E. Ablation Studies

We conclude the experiments with ablation studies to provide more insight into several design choices in MISO.

Ablation on encoder and decoder pre-training. The first ablation study investigates the performance improvements brought by pre-training the encoder and decoder networks in MISO. For this, we use the same setup as the local mapping experiments in Table I. For each scene, we report the F-score (%) calculated with a 5 cm threshold and the training loss. The results are presented at epochs 10 and 100 to demonstrate both early-stage and later-stage training performance.

In Table V, we compare the following variants of MISO:

- No-ED: Algorithm 1 without pre-trained decoder or encoder initialization. The grid features are initialized from a normal distribution with standard deviation 10^{-4}. Both the grid features and the decoder are optimized jointly.
- No-E: Algorithm 1 using only pre-trained decoder and no encoder initialization. The grid features are initialized from a normal distribution with standard deviation 10^{-4}.
- Full: default Algorithm 1 using both pre-trained decoder and encoder initialization.

As shown in Table V, our hierarchical initialization scheme significantly speeds up training loss optimization, particularly when both the pre-trained encoder and decoder are used (see rows 1 and 3). Even with only the pre-trained decoder, optimization remains faster than training from scratch (rows 1 and 2). As expected, after sufficient training (*i.e.*, at 100 epochs), all three methods converge in terms of training loss. Notably, using the pre-trained decoder yields better performance than training from scratch. We attribute the slight degradation of Full compared to No-E on scene 0011 (100 epochs) to domain mismatch since our encoders are pre-trained on synthetic Replica scenes only. We do indeed observe Replica-like artifacts in the Full results. With greater realism and diversity in pre-training, we expect Full to further improve. Overall, Table V still shows clear benefits of pre-trained encoders especially at early stages (10 epochs).

Ablation on hierarchical alignment. To investigate the effects of each alignment stage in Algorithm 2, we conduct ablation experiments on ScanNet scenes 0011 and 0207, each containing four submaps. We use the same setup as in the submap alignment experiments. In Table VI, we compare the following variants of MISO:

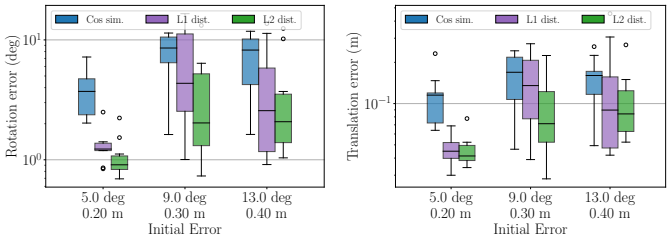
- Coarse only: Algorithm 2 with only coarse grid ($l = 1$) feature-based alignment.
- Coarse+Fine: Algorithm 2 with both coarse grid ($l = 1$) and fine grid ($l = 2$) feature-based alignment.
- Full: default Algorithm 2 with both coarse grid ($l = 1$) and fine grid ($l = 2$) feature-based alignment, and a final alignment stage using predicted SDF values.

In Table VI, one can see that the full hierarchical approach (coarse, fine, and SDF) achieves the smallest rotation and translation errors, whereas using only the coarse alignment yields the largest errors. As expected, the full approach requires more computation time as the last stage based on SDF alignment requires using the decoder to predict SDF values. We do not report the F-score for No-ED at epoch 10 because it fails to produce a mesh due to insufficient training.

Ablation on metric function. We study how different metric functions used in the feature-based alignment stage (*i.e.*, $\mathbf{d}$ in (18)) affect alignment accuracy. Specifically, we compare negative cosine similarity, L1 distance, and L2 distance under three progressively increasing initial rotation and translation errors in scene 0011. Our results in Fig. 10 show that L2 distance generally yields the most accurate submap alignment, while negative cosine similarity tends to produce the largest misalignment.

VII. Limitations

While MISO demonstrates very promising results in our evaluation, it still has several limitations. One direction for improvement is motivated by the use of dense feature grids. While dense grids offer a range of advantages including fast interpolation, better free-space modeling, and the ease of designing encoder networks, their memory cost may present a challenge when scaling the method to very large environments. To address this, combining MISO's hierarchical optimization approach with sparse data structures or tensor decomposition techniques is an exciting direction for future work. In addition, while we have only considered SDF reconstruction in this work, we expect the underlying approach in MISO to be applicable for modeling additional scene properties such as radiance, semantics, and uncertainty. Lastly, in this work, we primarily demonstrated MISO in an offline batch optimization setting, where a static scene is reconstructed given robot odometry and observations. Extending to online and dynamic settings is an important direction for future research.



(a) Rotation error (deg) (b) Translation error (m)

Fig. 10: Ablation study on hierarchical alignment on ScanNet [47]. We compare negative cosine similarity, L1 distance and L2 distance under three different initial rotation and translation errors.

VIII. Conclusion

We presented MISO, a hierarchical back-end for neural implicit SLAM based on multiresolution submap optimization. MISO performs local SLAM by representing each submap as a multiresolution feature grid. To achieve fast optimization and enable generalization, MISO makes use of pre-trained hierarchical encoder and decoder networks, where the encoders serve to initialize the submap features from observations, and the decoder serves to decode optimized features to predict the scene geometry. Furthermore, MISO presents a hierarchical method that sequentially uses implicit features and SDF predictions to align and fuse submaps, achieving globally consistent mapping. Evaluation on the real-world ScanNet, FastCaMo-Large, and Newer College datasets demonstrated MISO's strong performance and superior efficiency.

Acknowledgments

We gratefully acknowledge support from ARL DCIST CRA W911NF-17-2-0181, ONR N00014-23-1-2353, and NSF CCF-2112665 (TILOS).

References

[1] Y. Tang, J. Zhang, Z. Yu, H. Wang, and K. Xu, "MIPS-Fusion: Multi-implicit-submaps for scalable and robust online neural RGB-D reconstruction," *ACM Transactions on Graphics (TOG)*, 2023. 1, 2, 3, 6, 7, 8, 9, 10, 14

[2] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, "Neural fields in visual computing and beyond," *Computer Graphics Forum (CGF)*, 2022. 1, 2

[3] F. Tosi, Y. Zhang, Z. Gong, E. Sandström, S. Mattoccia, M. R. Oswald, and M. Poggi, "How NeRFs and 3D Gaussian Splatting are reshaping SLAM: A survey," *arXiv preprint arXiv:2402.13255*, 2024. 1, 2, 3

[4] C. Estrada, J. Neira, and J. D. Tardós, "Hierarchical SLAM: Real-time accurate mapping of large environments," *IEEE Transactions on Robotics (T-RO)*, 2005. 1

[5] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Transactions on Robotics (T-RO)*, 2005. 1

[6] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *International Conference on Robotics and Automation (ICRA)*, 2010. 1

[7] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural geometric level of detail: Real-time rendering with implicit 3D shapes," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 3

[8] C. Sun, M. Sun, and H.-T. Chen, "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3

[9] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (TOG)*, 2022. 1, 2, 3

[10] H. Zhai, H. Li, X. Yang, G. Huang, Y. Ming, H. Bao, and G. Zhang, "Vox-Fusion++: Voxel-based neural implicit dense tracking and mapping with multi-maps," *arXiv preprint arXiv:2403.12536*, 2024. 1, 2, 3, 6, 7, 9

[11] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "iSDF: Real-time neural signed distance fields for robot perception," in *Robotics: Science and Systems (RSS)*, 2022. 1, 2, 4, 7, 8, 15

[12] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (TOG)*, 2023. 2

[13] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[14] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[15] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM (CACM)*, 2021. 2

[16] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural RGB-D surface reconstruction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[17] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[18] S. Fridovich-Keil, G. Meanti, F. R. Warburg, B. Recht, and A. Kanazawa, "K-Planes: Explicit radiance fields in space, time, and appearance," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[19] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "TensoRF: Tensorial radiance fields," in *European Conference on Computer Vision (ECCV)*, 2022. 2

[20] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-NeRF: Point-based neural radiance fields," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[21] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin, "Neuralangelo: High-fidelity neural surface reconstruction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 4

[22] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[23] C. Jiang, H. Zhang, P. Liu, Z. Yu, H. Cheng, B. Zhou, and S. Shen, "H_2-mapping: Real-time dense mapping using hierarchical hybrid representation," *Robotics and Automation Letters*, 2023. 2

[24] J. Wang, T. Bleja, and L. Agapito, "GO-Surf: Neural feature grid optimization for fast, high-fidelity RGB-D surface reconstruction," in *IEEE International Conference on 3D Vision (3DV)*, 2022. 2

[25] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "iMAP: Implicit mapping and positioning in real-time," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[26] X. Yang, H. Li, H. Zhai, Y. Ming, Y. Liu, and G. Zhang, "Vox-Fusion: Dense tracking and mapping with voxel-based neural implicit representation," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2022. 2

[27] Z. Zhu, S. Peng, V. Larsson, Z. Cui, M. R. Oswald, A. Geiger, and M. Pollefeys, "NICER-SLAM: Neural implicit scene encoding for RGB SLAM," in *International Conference on 3D Vision (3DV)*, 2024. 2

[28] J. Deng, Q. Wu, X. Chen, S. Xia, Z. Sun, G. Liu, W. Yu, and L. Pei, "NeRF-LOAM: Neural implicit representation for large-scale incremental lidar odometry and mapping," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2

[29] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss, "PIN-SLAM: LiDAR SLAM using a point-based implicit neural representation for achieving global map consistency," *IEEE Transactions on Robotics (T-RO)*, 2024. 2, 3, 6, 7, 8, 10, 15
[30] Y. Pan, X. Zhong, L. Jin, L. Wiesmann, M. Popović, J. Behley, and C. Stachniss, "PINGS: Gaussian Splatting Meets Distance Fields within a Point-Based Implicit Neural Map," *arXiv preprint arXiv:2502.05752*, 2025. 2
[31] M. M. Johari, C. Carta, and F. Fleuret, "ESLAM: Efficient dense SLAM system based on hybrid representation of signed distance fields," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
[32] H. Wang, J. Wang, and L. Agapito, "Co-SLAM: Joint coordinate and sparse parametric encodings for neural real-time SLAM," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
[33] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, "GO-SLAM: Global optimization for consistent 3D instant reconstruction," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2
[34] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3D reconstruction with loop closure," in *European Conference on Computer Vision (ECCV)*, 2016. 3
[35] L. Liso, E. Sandström, V. Yugay, L. Van Gool, and M. R. Oswald, "Loopy-SLAM: Dense neural SLAM with loop closures," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3
[36] T. Deng, G. Shen, T. Qin, J. Wang, W. Zhao, J. Wang, D. Wang, and W. Chen, "PLGSLAM: Progressive neural scene represenation with local to global bundle adjustment," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3
[37] H. Matsuki, K. Tateno, M. Niemeyer, and F. Tombari, "NEWTON: Neural view-centric mapping for on-the-fly large-scale SLAM," *IEEE Robotics and Automation Letters (RA-L)*, 2024. 3
[38] S. Liu and J. Zhu, "Efficient map fusion for multiple implicit slam agents," *Transactions on Intelligent Vehicles (T-IV)*, 2023. 3
[39] J. Hu, M. Mao, H. Bao, G. Zhang, and Z. Cui, "CP-SLAM: Collaborative neural point-based SLAM system," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 3
[40] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "NICE-SLAM: Neural implicit scalable encoding for SLAM," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3
[41] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999. 4
[42] D. P. Kingma, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2014. 4, 13
[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 4, 6, 8
[44] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *International Conference on Machine Learning (ICML)*, 2020. 4
[45] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *European Conference on Computer Vision (ECCV)*, 2020. 5
[46] A. Vahdat and J. Kautz, "NVAE: A deep hierarchical variational autoencoder," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 5
[47] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7, 8, 9, 11, 12, 14
[48] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *International Conference on Computer Cision*, 2017. 8, 9
[49] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. De Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019. 7, 13
[50] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-camera lidar inertial extension to the newer college dataset," *arXiv preprint arXiv:2112.08854*, 2021. 7, 10
[51] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Conference on computer vision and pattern recognition*, 2015. 7, 9
[52] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv preprint arXiv:1801.09847*, 2018. 7, 10
[53] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In defense of point-to-point ICP–simple, accurate, and robust registration if done the right way," *IEEE Robotics and Automation Letters (RA-L)*, 2023. 10

APPENDIX A
IMPLEMENTATION AND TRAINING DETAILS

In our experiments, each submap is a multiresolution feature grid that has two levels with spatial resolutions 0.5 m and 0.1 m, respectively. The feature dimension at each level is set to $d = 4$. The decoder network D_θ is a single-layer MLP with hidden dimension 64. For the encoder network E_{ϕ_l} at each level l, we first use a small 3D CNN to process the input voxelized features. Each CNN has 2 hidden layers, a kernel size of 3, and the number of feature channels doubles after every layer. The CNN outputs at all 3D vertices are processed by a shared two-layer MLP with hidden dimension 16 to predict the target feature grid F_l (see Fig. 3). We use ReLU as the nonlinear activation for all networks.

To train the decoder network D_θ offline, we compile an offline training dataset using six scenes from Replica [49]. Within each scene $s \in S$, we randomly simulate 128 camera views with ground truth pose information. During training, all scenes share a single set of decoder parameters θ, and each scene s has its own dedicated grid features F^s. We perform joint optimization using a loss similar to Problem 1, except we use ground truth camera poses and disable pose optimization,

$$\min_{\{F^s\}_s,\theta} \sum_{s\in S}\sum_{k=1}^{n}\sum_{j=1}^{m_k} c_j\left(h(\underline{T}_k^s x_j^k; F^s, \theta)\right), \tag{23}$$

where $\underline{T}_k^s$ denote the ground truth camera poses. We use Adam [42] with a learning rate of 10^{-3} and train for a total of 1200 epochs. During training, we employ a coarse-to-fine strategy where all fine level features are activated after 200 epochs. After training completes, the grid features F^s are discarded and only decoder parameters θ is saved.

The offline training of encoder networks follows a similar setup and uses the same Replica scenes. The design of the training loss is presented in Sec. IV in the main paper. We sequentially train the encoder networks from coarse to fine levels. At level l, we use the trained encoders from previous levels to compute $F^s_{1:l-1}$, which are needed to evaluate the training loss in (15). To account for noisy pose estimates at test time, we simulate pose errors with 1 deg and 1 cm rotation and translation standard deviations. Each encoder is trained using Adam [42] with a learning rate of 10^{-3} for 1000 epochs.

APPENDIX B
PROOFS

Proof of Proposition 1: Without loss of generality, we assume the features F_l is represented as a vector $F_l \in \mathbb{R}^{|F_l|}$. In

the following, all points are expressed in the coordinate frame of the submap. First, consider a single observed point $x_j \in \mathbb{R}^3$ and its label $y_j \in \mathbb{R}$. At each level l, the output feature $f_l(x_j)$ in Definition 1 is a linear function of the features F_l at this level. This means that there exists a matrix $K_l(x_j) \in \mathbb{R}^{d \times |F_l|}$ such that $f_l(x_j) = K_l(x_j)F_l$. When the decoder D_θ is a linear function, the final output from the multiresolution feature grid is a weighted sum of features from all levels, *i.e.*,

$$h(x_j; F) = D_\theta\Big(\bigoplus_{l\in[L]} f_l(x_j)\Big) = \sum_{l=1}^{L} \theta_l^\top f_l(x_j) = \sum_{l=1}^{L} \theta_l^\top K_l(x_j) F_l. \tag{24}$$

In the following, we use l to specifically refer to the target level that we seek to initialize in (8). The measurement residual is,

$$h(x_j; F) - y_j = \underbrace{\sum_{l'=1}^{l-1} \theta_{l'}^\top K_{l'}(x_j) F_{l'} - y_j}_{r_{1:l-1}(x_j)} + \theta_l^\top K_l(x_j) F_l, \tag{25}$$

where we have used the fact that features at levels greater than l are zero. We can concatenate the above equation over all observations $x_1, \ldots, x_N$ as follows,

$$h(x; F) - y = \underbrace{\begin{bmatrix} r_{1:l-1}(x_1) \\ \vdots \\ r_{1:l-1}(x_N) \end{bmatrix}}_{r_{1:l-1}(x)} + \underbrace{\begin{bmatrix} \theta_l^\top K_l(x_1) \\ \vdots \\ \theta_l^\top K_l(x_N) \end{bmatrix}}_{J} F_l. \tag{26}$$

Note that the matrix J is exactly the Jacobian of our model $h(x; F)$ with respect to the level-l features F_l. Substituting the above equation into (8) shows that the problem is equivalent to the following linear least squares,

$$\min_{F_l} \|h(x; F) - y\|_2^2 = \|r_{1:l-1}(x) + JF_l\|_2^2. \tag{27}$$

Thus, the solution is obtained by solving its normal equations, yielding the final expression in (9). ■

APPENDIX C
ADDITIONAL RESULTS

In Fig. 11, we show qualitative SDF visualizations on additional scenes from ScanNet [47] under noisy poses. For each method, a horizontal slice of the estimated SDF is shown where red and blue indicate positive and negative values. These figures further supports the conclusions drawn in the paper, and the reader is referred to Sec. VI for the discussion.

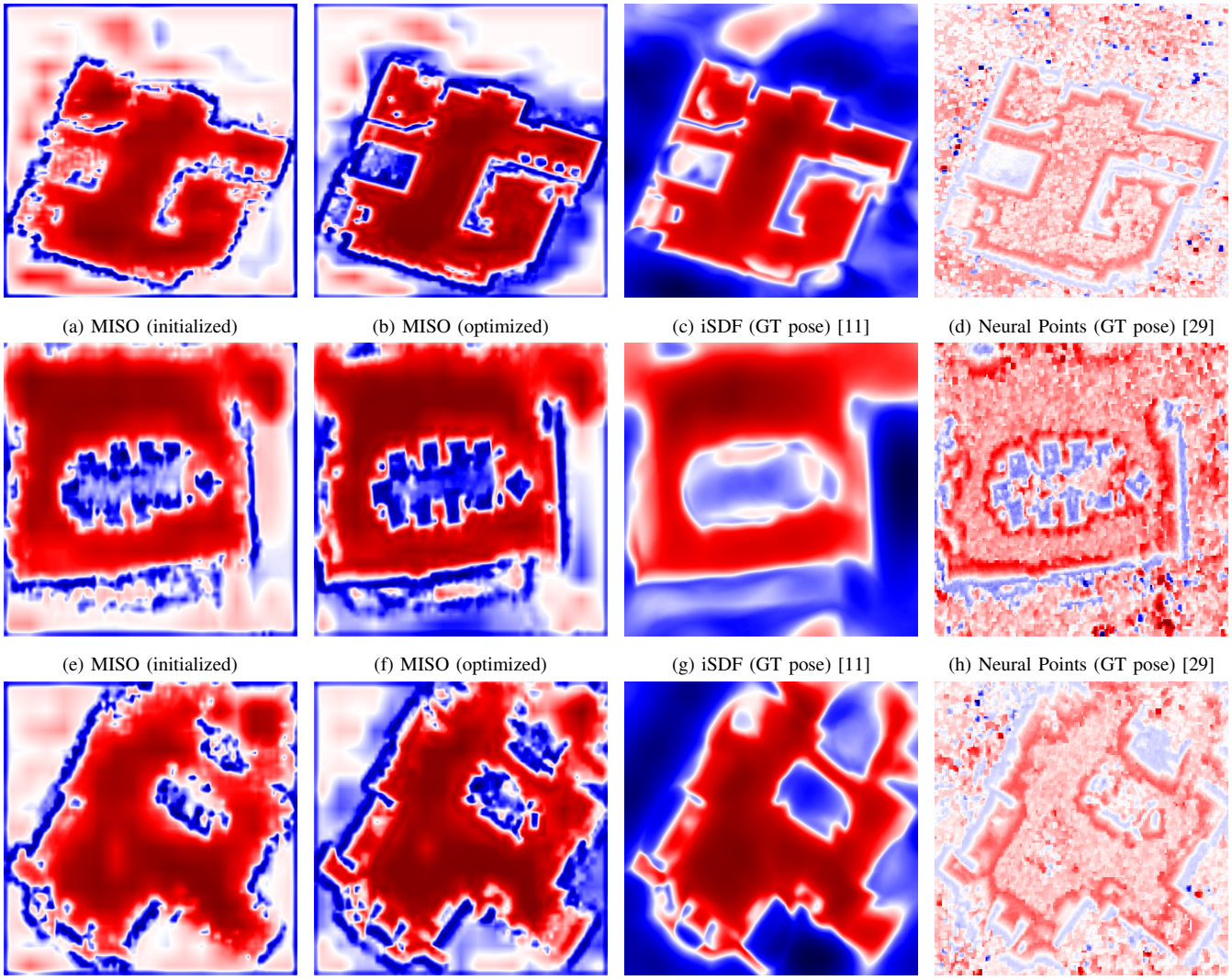
TABLE VII: Number of parameters and GPU memory usage of MISO.

Method	ScanNet (0011)		FastCamo (Stairs-I)		FastCamo (Floors-I)	
	Params	GPU mem	Params	GPU mem	Params	GPU mem
Coarse only	0.14M	2.11GB	0.25M	1.60GB	0.26M	2.64GB
Full	0.70M	2.18GB	9.08M	1.67GB	8.48M	2.71GB

We evaluate MISO 's memory requirements by measuring both the total number of parameters (covering the grid model and the decoder) and the GPU memory usage. The GPU memory usage is measured across the entire environment using PyTorch's `cuda.max_memory_allocated()` function. We test three scenes in Table VII: ScanNet [47] scene 011 and FastCaMo-Large [1] Stairs-I and Floors-I. The sizes of these environments, measured as the dimensions of their axis-aligned bounding boxes, are reported below,

- ScanNet 0011: 6.0 m × 8.2 m × 2.7 m
- FastCaMo-Large Stairs-I: 8.9 m × 10.7 m × 15.4 m
- FastCaMo-Large Floors-I: 26.0 m × 16.7 m × 7.5 m

We consider two configurations: one using only the coarse grid (Coarse only) and another using both the coarse and fine grids (Full). In all scenes, the decoder itself consistently contains about 0.12M parameters.



(a) MISO (initialized) (b) MISO (optimized) (c) iSDF (GT pose) [11] (d) Neural Points (GT pose) [29]

(e) MISO (initialized) (f) MISO (optimized) (g) iSDF (GT pose) [11] (h) Neural Points (GT pose) [29]

(i) MISO (initialized) (j) MISO (optimized) (k) iSDF (GT pose) [11] (l) Neural Points (GT pose) [29]

Fig. 11: Qualitative comparison on additional ScanNet scenes 0000, 0011, and 0024, each shown in a row. For each method, a horizontal slice of the estimated SDF is shown where red and blue indicate positive and negative values.