PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency

Yue Pan, Xingguang Zhong, Louis Wiesmann, *Graduate Student Member, IEEE*, Thorbjörn Posewsky, Jens Behley, *Member, IEEE*, and Cyrill Stachniss, *Member, IEEE*

***Abstract*—Accurate and robust localization and mapping are essential components for most autonomous robots. In this paper, we propose a simultaneous localization and mapping (SLAM) system for building globally consistent maps, called point-based implicit neural (PIN)-SLAM, that is based on an elastic and compact PIN map representation. Taking range measurements as input, our approach alternates between incremental learning of the local implicit signed distance field and the pose estimation using a correspondence-free, point-to-implicit model registration to the current local map. Our implicit map is based on sparse optimizable neural points, which are inherently elastic and deformable with the global pose adjustment when closing a loop. Loops are also detected using the neural point features. Extensive experiments validate that PIN-SLAM is robust to various environments and versatile to different range sensors, such as light detection and ranging (LiDAR) and RGB color and depth (RGB-D) cameras. PIN-SLAM achieves pose estimation accuracy better or on par with the state-of-the-art LiDAR odometry or SLAM systems and outperforms the recent neural implicit SLAM approaches while maintaining a more consistent, and highly compact implicit map that can be reconstructed as accurate and complete meshes. Finally, thanks to the voxel hashing for efficient neural points indexing and the fast implicit map-based registration without closest point association, PIN-SLAM can run at the sensor frame rate on a moderate graphics processing unit (GPU).**

***Index Terms*—Deep learning, localization, mapping, simultaneous localization and mapping (SLAM).**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) based on range sensors, such as light detection and ranging (LiDAR) sensors or RGB color and depth (RGB-D) cameras, is a fundamental building block for autonomous mobile robots [15], [48], [78], [97]. Various map representations [4], [12], [76] and scan registration algorithms [5], [52], [64] have been proposed over the last decades for efficient and high-fidelity representations of the environment as well as robust and accurate localization of the robot within the mapped environment.

The recent advance in neural implicit representation has shown several advantages over the classical explicit map representation widely used nowadays. Instead of explicitly storing properties in a grid map, one can train a neural network to fit the observations in a scene. This allows for querying properties, such as signed distances [53], occupancy probability [43], colors [44], intensities [22], and semantics [99], at an arbitrary location within the scene implicitly using the neural network. As a memory-efficient continuous representation, these kinds of implicit neural maps support efficient rendering [47], surface reconstruction [100], collision avoidance [51] and path planning [91]. To enable the model to have a high representation capacity, current approaches often learn the neural implicit representation using optimizable local latent features instead of using a single globally shared neural network [51], [71]. The local latent features can be structured by a regular 3-D grid [47], an axis-aligned tri-plane [55] with 2-D grids, or a set of irregular points [89]. Consequently, several SLAM systems based on neural implicit representation have been proposed, mainly for RGB-D cameras operating indoor [24], [65], [71], [92], [101] but also for LiDAR sensors operating outdoor [11]. However, these SLAM approaches lack support for direct loop closure corrections and are not able to build globally consistent maps of larger scenes. This is mainly due to the usage of regular grid-based local feature embeddings, which are not elastic and resilient to loop corrections.

In this article, we investigate the problem of realizing a SLAM system using an implicit neural map representation that supports globally consistent mapping. We opt to use a neural point-based implicit representation, which has two main advantages over grid-based representations: the flexibility of spatial distribution and the elasticity for transformations. Recent work [65] only makes use of the first advantage for neural RGB-D SLAM at the cost of scalability and inefficient neighborhood querying. Instead, we exploit the second and more important advantage to build a globally consistent map that can be corrected while online mapping after closing a loop, which is essential for large-scale LiDAR-based SLAM. Our approach alternates between mapping, i.e., online learning of the local implicit signed distance map given poses, and odometry, i.e., tracking the next scan's

Manuscript received 18 May 2024; accepted 26 June 2024. Date of publication 2 July 2024; date of current version 16 September 2024. This work was supported in part by the European Union under Grant 101070405 (DigiForest), and in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob under Grant STA 1051/5-1 within the FOR 5351 – 459376902 (AID4Crops). This article was recommended for publication by Associate Editor A. Nuechter and Editor J. Civera upon evaluation of the reviewers' comments. *(Corresponding author: Yue Pan.)*

The authors are with the University of Bonn, 53113 Bonn, Germany, also with the Department of Engineering Science, University of Oxford, OX1 2JD Oxford, U.K., and also with the Lamarr Institute for Machine Learning and Artificial Intelligence, 44227 Dortmund, Germany (e-mail: yue.pan@igg.uni-bonn.de).

Data is available online at https://github.com/PRBonn/PIN_SLAM.

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3422055, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3422055

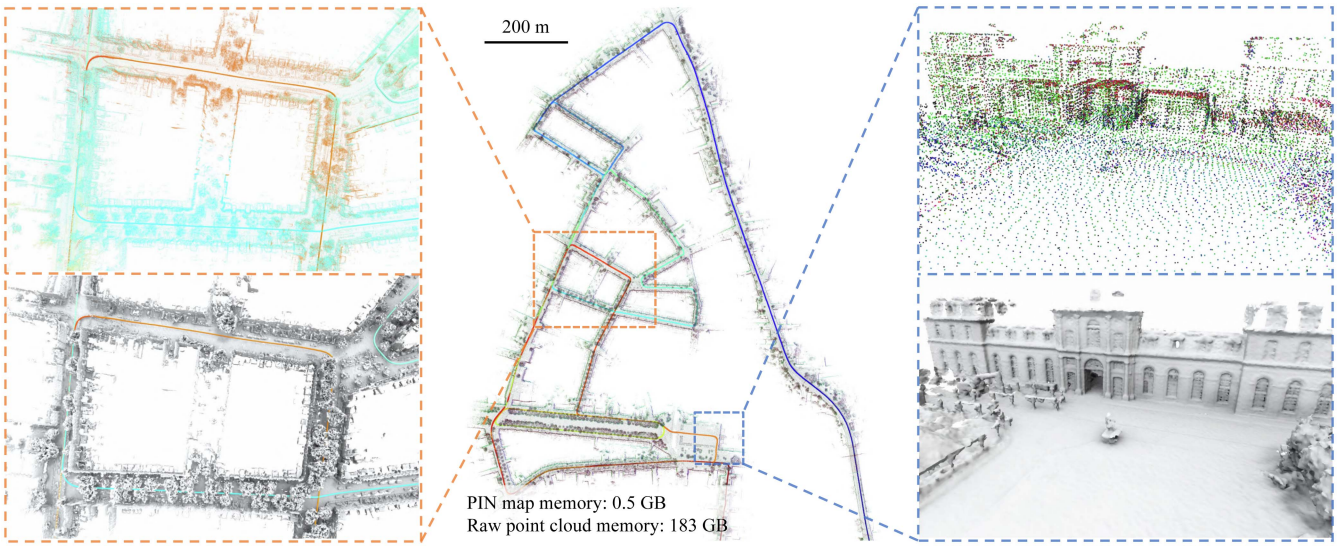


Fig. 1. We present PIN-SLAM, a novel LiDAR SLAM system using an elastic PIN map representation. Depicted in the middle, we show a large-scale globally consistent neural point map built with our approach using about 20 000 LiDAR scans recorded with a car without using any information from a GNSS, IMU, or wheel odometry. We can query the SDF value at an arbitrary position from the neural point map and reconstruct surface meshes. The point colors represent the neural point feature after online optimization. On the left, we show the consistent neural points (top) and mesh (bottom) of a region traversed by the car multiple times indicated by the dashed orange box. The colors of the neural points (top) represent timesteps when the point was added to the map. On the right, we show the high-fidelity mesh (bottom) of a building reconstructed from the neural point map (top) of the region indicated by a dashed blue box.

pose given the current local map, additionally with the ability to correct the drift and keep a globally consistent map after closing a detected loop. For the mapping part, we adapt the training data sampling strategy and loss function used in our previous work [100] but replace the octree-based multiresolution feature grid with a set of neural feature points. The signed distance function (SDF) and additional properties, such as color or semantics, at a query position are predicted by interpolating among the surrounding neural points. For each neural point, we concatenate its optimizable latent feature with the query position's coordinates in the neural point's coordinate system. This concatenated feature is then decoded by a globally shared multilayer perceptron (MLP) as the prediction for that neural point. For efficient and scalable neighborhood querying for the interpolation, we use a voxel hashing data structure to index the neural points by keeping no more than one active neural point in each voxel. This enables our system to run at the sensor frame rate (10 Hz) regardless of the scale of the mapped scene. To tackle the problem of "catastrophic forgetting" in incremental learning, instead of using the less stable regularization-based strategy [100], we use a local data pool replay and local map update strategy in a sliding window manner.

For the odometry part, we extend our previous work Loc-NDF [86] to accomplish scan-to-implicit map registration efficiently without the effort of point correspondence association using a second-order optimization to estimate the ego-motion incrementally. We propose an additional robust kernel based on the regularity of the predicted SDF to further improve the registration accuracy and robustness.

To correct the drift accumulated by the odometry, we use the optimized neural point local map for global loop closure detection and correction. Pose graph optimization (PGO) is conducted after a loop verification to correct the drift and the inherently elastic neural points are transformed along with the poses of their associated frames to form a globally consistent map, from which a consistent SDF and mesh can be generated.

The main contribution of this article is a novel neural SLAM system, called point-based implicit neural (PIN)-SLAM, based on a PIN map representation that supports building large-scale globally consistent maps online, as shown in Fig. 1. To the best of the authors' knowledge, PIN-SLAM is the first full-fledged implicit neural SLAM system including odometry, loop closure detection, and globally consistent implicit mapping.

In sum, we make four key claims, which will be backed up in the experiments as follows:

1) Our SLAM system achieves localization accuracy better or on par with state-of-the-art LiDAR odometry/SLAM approaches and is more accurate than recent implicit neural SLAM methods on various datasets using different range sensors.
2) Our method can conduct large-scale globally consistent mapping with loop closure thanks to the elastic neural point representation.
3) Our map representation is more compact than the previous counterparts and can be used to reconstruct accurate and complete meshes at an arbitrary resolution.
4) Our correspondence-free scan-to-implicit map registration and the efficient neural point indexing by voxel hashing enable our algorithm to run at the sensor frame rate on a single NVIDIA A4000 graphics processing unit (GPU).

II. Related Work

A. LiDAR Odometry and SLAM

SLAM using range sensors, such as LiDAR sensors, has been an active research topic for the last decades. A LiDAR SLAM system often consists of odometry, mapping, and optionally the

loop closure correction. At the core of such a system are the map representation and the point cloud registration algorithm.

The earlier works on 2-D LiDAR SLAM [15], [18], [29] adopt a probabilistic occupancy grid map and use 2-D scan matching based on iterative closest point (ICP) algorithm [5] or particle filters [15], [46] for ego-motion estimation.

For 3-D LiDAR odometry and mapping, similar to the feature-matching-based methods popularized in visual SLAM, the seminal work LiDAR odometry and mapping (LOAM) [97] proposes to extract sparse planar or edge feature points from the scan point cloud and registers them to the last frame or the feature point map using ICP. LOAM inspired multiple follow-up works [36], [52], [56], [67], [68], [82] based on more sophisticated feature point extraction [68], [82], classification and registration optimization schemes [36], [52] or the fusing with inertial measurements [56], [67], leading to faster, more accurate and more robust odometry estimation.

Recently, CT-ICP [10] and KISS-ICP [78] achieved strong LiDAR odometry performance without feature point extraction. They register and append voxel-downsampled point cloud to a local point cloud map supporting efficient neighborhood search using either point-to-plane or point-to-point metrics. Combined with a motion model providing initial poses, these feature-free "direct" methods [10], [13], [31], [78] are comparably robust in structure-less scenarios and can take advantage of denser observations for optimization. The same idea has been applied to LiDAR-inertial odometry [88], [90] at a high operating frequency. Our method belongs to the "direct" algorithms but avoids the time-consuming and nondifferentiable correspondence association procedure of ICP.

Except for the point-based map used by aforementioned methods, some other works estimate the ego-motion by conducting scan registration to other explicit map representations, such as 3-D surfels [4], [7], triangle meshes [62], [76], and voxelized Gaussian distributions [30], [93], as well as the implicit map representations, such as moving least square model [12], Gaussian process [63], and grid-based implicit neural SDF [11]. In contrast, our method uses a PIN SDF map representation, which combines the best of both worlds: it retains the flexibility and elasticity characteristic of point-based methods while offering continuity and compactness simultaneously thanks to the neural representation.

There exists also learning-based LiDAR odometry methods [33], [80], [81], which realize registration between adjacent frames using end-to-end pose supervision. However, they typically cannot generalize well to unseen test sets and do not make use of the map for odometry. Similar to recent work neural radiance field (NeRf)-LOAM [11], our method exploits neural networks to fit the SDF online using the input point clouds, thus requiring no pretraining and generalizing well in various scenarios.

Besides incremental pose estimation, loop closure detection and correction are necessary for building a globally consistent, long-term SLAM system. Various approaches have been proposed to first conduct LiDAR place recognition and then estimate the transformation between the queried and retrieved scan using geometry-based [26], [27] or learning-based [41], [42], [75] scan-wise global descriptor matching as well as local feature matching and verification [40]. However, previous methods often rely on the raw point cloud from a sparse single scan for descriptor generation and matching. In contrast, our method makes use of the neural point features in the local maps for loop closure detection. The usage of the local map makes our method robust to occlusions and sensors with a narrow field of view. Besides, reusing the online-optimized neural point features can avoid the generalizability concern of offline-trained feature extractors and save additional computation for local feature extraction.

B. Implicit Neural Map Representation

Over the past decades, explicit map representations have been widely used in the robotics community for localization [74], planning, and exploration [69]. These methods explicitly represent the scene using point cloud [97], surfels [4], triangle meshes [76], or voxel grids storing either the occupancy probabilities [15] or a truncated signed distance function (TSDF) [48]. Various approaches have been proposed to improve the scalability and efficiency of the map data structure [19], [49], [77] and to realize the incremental integration from sensor measurements [50] for the dense volumetric mapping. These explicit representations often discretize the scene with a fixed spatial resolution. Another line of work adopts a continuous implicit representation, such as Gaussian processes [87] or reproducing kernel Hilbert maps [58]. However, these approaches currently do not scale well to 3-D data.

Recently, implicit neural representations have been proven effective in modeling radiance fields [44] and geometric (occupancy or distance) fields [43], [53] using fairly simple neural networks. These representations have demonstrated notable success in various applications, including novel view synthesis, 3-D surface reconstruction, and shape completion. Following the seminal works on NeRF [44], DeepSDF [53], and occupancy networks [43], numerous works target improving the scalability of the implicit neural representation while boosting the time and memory efficiency. Instead of representing the whole scene with a single MLP, more recent methods exploit a hybrid representation by jointly optimizing explicitly stored local latent features and a shallow MLP. These works propose to store the optimizable local latent features in various data structures such as multi-resolution dense voxel grids [55], sparse voxel hashing grids [35], [47], octree nodes [72], permutohedral lattices [61], axis-aligned tri-plane 2-D grids [6], or as an unordered point sets [89].

The advances in efficient training of scalable implicit neural representations open up an avenue to implicit neural mapping and SLAM systems. Compared to traditional methods, implicit neural map representations have several attractive properties, such as more compact storage, better noise smoothing capabilities, and stronger inpainting and hole-filling ability for sparse or occluded observations. In the realm of mapping and SLAM from a stream of RGB-D data, several works propose to use a single MLP [2], [51], [71] and a hybrid representation combining grid-based local latent features and a shallow MLP [21], [24], [83], [92], [101] to model the radiance or geometric field of the scene and simultaneously track the camera pose. These approaches have shown comparable tracking accuracy compared to the classic visual odometry methods and can generate a more complete and compact map, which can be reconstructed as a 3-D mesh.

Although with fewer works, a related trend can be seen in LiDAR mapping and localization research. IR-MCL [32] and LocNDF [86] propose to localize the robot within an implicit

neural distance map built by laser scans. To scale up the implicit neural representation to large-scale outdoor LiDAR data, SHINE-Mapping [100] stores local features in octree-based sparse voxels. Toward a SLAM system, LONER [23] integrates the incremental neural mapping into a LiDAR odometry front-end using ICP. Recently, NeRF-LOAM [11] proposed an implicit neural LiDAR odometry and mapping system using an online optimizable octree-based feature grid. Besides grid-based representations, the PIN representation stores latent features in a set of neural points. It has been applied to efficient NeRF fitting [89], dynamic NeRF [1], adaptive surface reconstruction [34], and RGB-D SLAM [65]. Although also using neural points, the recent work point-SLAM [65] targets only indoor RGB-D SLAM and does not support loop closure and globally consistent mapping. Point-SLAM takes advantage of the flexibility of point-based representation for the adaptive scene encoding, i.e., allocating more features to structures that require more details. We, instead, make use of the elasticity property of neural points for building a globally consistent map online by transforming the neural points on loop closures.

The aforementioned neural implicit RGB-D and LiDAR SLAM systems still have two major limitations. First, most of them [11], [65], [71], [101] cannot run at the sensor frame rate, limiting their applications for online robotics applications. This is mainly due to the map data structure, computationally demanding differentiable rendering-based optimization for mapping, or an inefficient gradient descent optimization for pose estimation. Second, the map representations of previous neural implicit SLAM approaches do not support loop closure correction, and thus cannot build globally consistent maps, especially in outdoor long-term robotics missions. The reason for this is the usage of the regular latent feature grids that are not elastic to pose corrections introduced by loop closures. Once a loop is corrected and the poses are updated, previous methods need to re-allocate the feature grids and retrain the entire map, which is computationally demanding and clearly prevents online applications. To solve the first challenge, we realize an efficient SLAM system by adopting direct point-wise SDF supervision in a local map during mapping and a second-order on-manifold optimization for odometry estimation. To deal with the second limitation, our approach exploits the elastic and deformable point-based implicit representation to avoid grids and thus the remapping after loop correction.

C. Global Consistency in SLAM

Global consistency is a desired property for maps. The solutions to globally consistent mapping can be divided into remapping-based, submap-based, and point-based methods.

The remapping-based method, such as Bundle Fusion [8], accomplishes online globally consistent 3-D reconstruction using on-the-fly surface reintegration in a bundle adjustment (BA) manner at the cost of substantial computational efforts.

The submap-based methods accumulate the observations as a submap and assume the submap is a locally defined rigid body. The pseudoglobal consistency is maintained by optimizing a graph linking submaps and their associated poses through submap-to-submap registrations. The submap can use different representations, such as feature points [37], [52] and TSDF [59], [84]. The usage of submaps decreases the number of nodes and edges in PGO, thus saving computational effort. However, these methods have issues of ambiguity in the overlapping region of the adjacent submaps and determining appropriate criteria for submap division to balance the rigidity and efficiency.

The point-based methods represent the map as a fully deformable point set or surfels, each associated with a frame pose. Whelan et al. [85] proposed elastic fusion for globally consistent RGB-D SLAM using surfel map with deformation graph, which is further scaled for outdoor surfel-based LiDAR SLAM [4], [7]. These methods can handle globally consistent mapping without the need for submap division. However, in contrast to volumetric mapping, point-based methods are less suitable for online 3-D reconstruction and path planning. This is attributed to their sparsity and the absence of a direct representation of free or unknown space.

Only a few recent works are trying to solve the loop closure correction and globally consistent mapping using implicit neural map representation without storing the raw observation data and conducting remapping after the pose update. IMT-Mapping [95] uses the SE(3)-equivalent grid-based feature for implicit mapping. When a loop is corrected, IMT-Mapping can directly get the updated grid features by interpolating the transformed SE(3)-equivalent features. Following the submap-based explicit mapping systems [52], [59], [84], MIPS-Fusion [73] and NF-Atlas [94] employ an implicit submap with MLP or octree-based feature grid and adjust the submaps when a loop is corrected to keep the consistency.

In contrast to previous methods, our PIN map belongs to the point-based globally consistent mapping approach. It avoids the nontrivial submap division and overlapping disambiguation while mapping a continuous SDF that enables efficient 3-D reconstruction.

III. Our Approach to Globally Consistent Implicit Neural SLAM

Our approach PIN-SLAM primarily addresses large-scale LiDAR SLAM for global map consistency, with applicability to other range sensors, such as RGB-D cameras.

Notations: In the following, we denote the transformation of a point $\boldsymbol{p}_A$ in coordinate frame A to a point $\boldsymbol{p}_B$ in coordinate frame B by $\mathsf{T}_{BA} \in \mathrm{SE}(3)$, such that $\boldsymbol{p}_B = \mathsf{T}_{BA}\boldsymbol{p}_A$. Note that we use the homogeneous coordinate conversion of points before and after the affine transformation, but will not include this operation explicitly in the following derivations. We let $\boldsymbol{R}_{BA} \in \mathrm{SO}(3)$ and $\boldsymbol{t}_{BA} \in \mathbb{R}^3$ denote the corresponding rotational and translational part of transformation T_{BA}.

We denote the sensor coordinate frame at timestep t as C_t. Frame C_t is associated to the world coordinate frame W by a pose $\mathsf{T}_{WC_t} \in \mathrm{SE}(3)$. We assume the transformation T_{WC_0} from the first frame C_0 to the world frame W is a constant, as either the identity matrix or the extrinsic calibration matrix. We denote the position of frame C_t as $\boldsymbol{t}_{WC_t}$ and the accumulated travel distance up to frame C_t as $D(t)$, which is given by

$$D(t) = \sum_{n=1}^{t} \left\| \boldsymbol{t}_{C_{n-1}C_n} \right\|_2 . \tag{1}$$

Overview: Before the detailed explanation of our methodology, we provide a brief overview of PIN-SLAM's pipeline. For each point cloud $\mathcal{P} = \{\boldsymbol{p} \in \mathbb{R}^3\}$ measured at timestep t, we simultaneously estimate the pose T_{WC_t} of the sensor frame and update the PIN map $\mathcal{M}$. For every timestep, as shown in Fig. 2, PIN-SLAM performes the following steps.

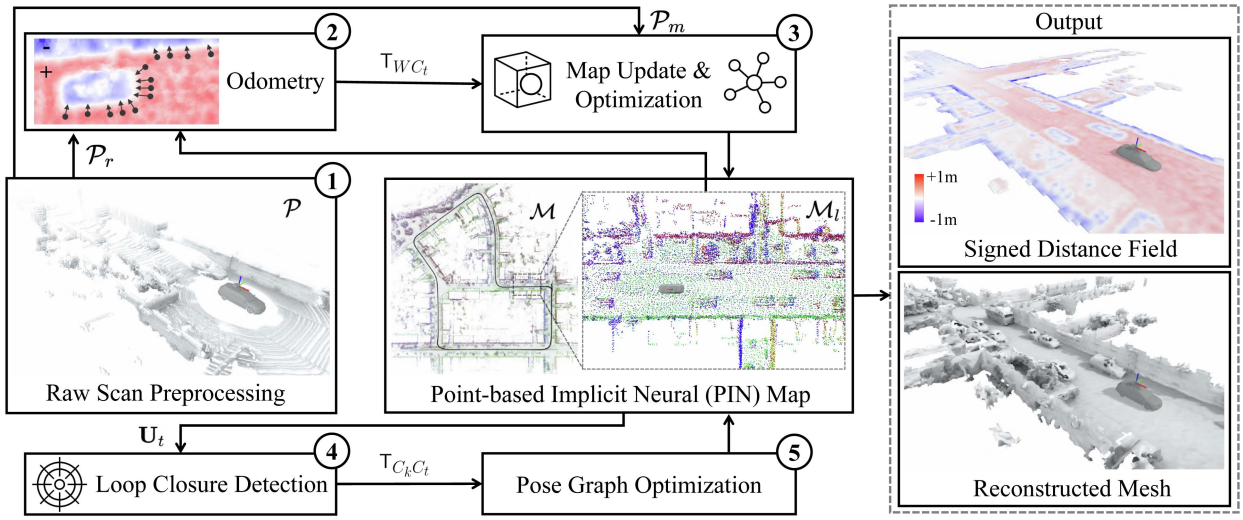


Fig. 2. Pipeline overview of PIN-SLAM. Starting from a point cloud $\mathcal{P}$ scanned at timestep t, (1) a point cloud for registration $\mathcal{P}_r$ and a point cloud for mapping $\mathcal{P}_m$ are voxel-downsampled. (2) We align $\mathcal{P}_r$ to the implicit SDF of current local map $\mathcal{M}_l$ to estimate the global pose of current frame T_{WC_t}. (3) T_{WC_t} is then used to transform $\mathcal{P}_m$ into the map coordinate system. With the transformed $\mathcal{P}_m$, we update the PIN map $\mathcal{M}$ and optimize the neural point features in the local map $\mathcal{M}_l$ by online incremental learning. (4) We generate polar context descriptor $\mathbf{U}_t$ using the current local map $\mathcal{M}_l$ and search for loop closures by comparing $\mathbf{U}_t$ to descriptors generated for previous frames. Once a loop between frame C_t and C_k is detected, we add the transformation $\mathsf{T}_{C_kC_t}$ as a loop edge of the pose graph and then (5) conduct the PGO. The position and orientation of the neural points in $\mathcal{M}$ are transformed along with their associated frames after the PGO, leading to a globally consistent map. With the PIN map, we can query the SDF value at an arbitrary position during or after the SLAM task for path planning and mesh reconstruction.

1) *Preprocessing:* We voxel-downsample the input point cloud $\mathcal{P}$ into the point cloud for registration $\mathcal{P}_r$ and the point cloud for mapping $\mathcal{P}_m$ (Section III-B).
2) *Odometry:* We estimate the global pose T_{WC_t} by registering the point cloud $\mathcal{P}_r$ to the implicit SDF of the local map $\mathcal{M}_l$. The odometry transformation $\mathsf{T}_{C_{t-1}C_t}$ is added as an edge in the pose graph $\mathcal{G}$ (Section III-C).
3) *Mapping:* We filter the dynamic points in $\mathcal{P}_m$ based on the map $\mathcal{M}$. Then, we sample along the ray from the sensor to each point in $\mathcal{P}_m$ to get the training samples $\mathcal{D}$ and transform them to the world frame using T_{WC_t}. We use the close-to-surface samples $\mathcal{D}_s \subset \mathcal{D}$ to initialize new neural points, append them to the map $\mathcal{M}$ and reset the local map $\mathcal{M}_l$ centered at the current position $\boldsymbol{t}_{WC_t}$. We update the training sample pool $\mathcal{D}_p$ with the training samples $\mathcal{D}$ of the current frame. Then we optimize the neural point features in the local map $\mathcal{M}_l$ using the samples in the pool $\mathcal{D}_p$ with direct SDF supervision by gradient descent. After that, we allocate the updated local map $\mathcal{M}_l$ back to the global map $\mathcal{M}$ (Section III-D).
4) *Loop closure detection:* We generate a local polar context descriptor $\mathbf{U}_t$ using the local map $\mathcal{M}_l$. Then, we search for a potential loop closure by comparing the feature distances between the descriptors of the current frame and candidate frames. Once a loop closure candidate between frame C_t and C_k is detected, we verify it by registering the point cloud $\mathcal{P}_r$ of the current frame to the local map $\mathcal{M}_l$ centered at the sensor position $\boldsymbol{t}_{WC_k}$ of frame C_k. We add the loop transformation $\mathsf{T}_{C_kC_t}$ resulting from the registration as an edge in the pose graph $\mathcal{G}$ if the registration succeeds (Section III-E).
5) *Pose graph optimization:* Once a loop closure edge is added, we optimize the pose graph $\mathcal{G}$. The position and orientation of each neural point in the global map $\mathcal{M}$ are transformed along with its associated frames after the optimization to keep the global consistency. We then transform the training sample pool $\mathcal{D}_p$ and reset the local map $\mathcal{M}_l$ accordingly after loop correction (Section III-F).

For the first timestep, we initialize the map using only the first scan as done in the mapping step. During or after the SLAM, we can query the SDF value at an arbitrary position for mesh reconstruction via the marching cubes algorithm [39].

Next, we will explain the basics, data structure, and training process of the proposed PIN map representation (Section III-A) and then explain each step of the pipeline in more detail.

A. Neural Point-Based Map Representation

1) Map Representation: We define the proposed PIN map as a set of neural points

$$\mathcal{M} = \{\boldsymbol{m}_i = (\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{f}_i^g, t_i^c, t_i^u, \mu_i) \mid i = 1, \ldots, N\} \quad (2)$$

where each neural point $\boldsymbol{m}_i$ is defined in the world frame W by a position $\boldsymbol{x}_i \in \mathbb{R}^3$ and a quaternion $\boldsymbol{q}_i \in \mathbb{R}^4$ representing the orientation of its own coordinate frame. Each neural point stores the optimizable latent feature vectors $\boldsymbol{f}_i^g \in \mathbb{R}^{F_g}$ representing the local geometry. In addition, we keep track of the creation timestep t_i^c, the last update timestep t_i^u and the stability μ_i for each neural point to determine whether a neural point is active or inactive, stable or unstable. We link each neural point $\boldsymbol{m}_i$ to the sensor pose $\mathsf{T}_{WC_{t_i^m}}$ at the mean of the neural point's creation and last timestep $t_i^m = \lfloor (t_i^c + t_i^u)/2 \rfloor$ to directly manipulate the map by updating the sensor poses.

As shown in Fig. 3, similar to the autodecoder architecture of DeepSDF [53], we predict the SDF value s at a query position $\boldsymbol{p}$ conditioned on K nearby neural points. For each neural point

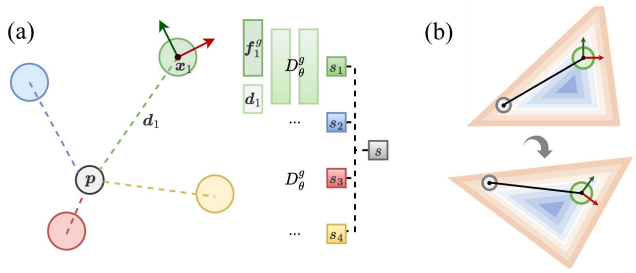


Fig. 3. Diagram of SDF querying in our PIN map simplified in 2-D. (a) Point in gray is the query position $\boldsymbol{p}$ while the other points are the neighboring neural points. Each neural point predicts the SDF value s_i at the query position by feeding the neural point feature $\boldsymbol{f}_i^g$ and the query point's position $\boldsymbol{d}_i$ under the neural point's coordinate system through a globally shared decoder D_θ^g. Then, the predictions are weighted as the final prediction s according to the distances from the neural points to the query position. (b) Orientation of each neural point defines its local coordinate system, ensuring the relative coordinate $\boldsymbol{d}_i$ and thus the SDF querying invariant to rigid-body transformation.

$\boldsymbol{m}_j$ in the K-neighborhood $\mathcal{N}_p$ of the query position $\boldsymbol{p}$, we first concatenate the latent feature vector $\boldsymbol{f}_j^g$ and the relative coordinate $\boldsymbol{d}_j$. Here, $\boldsymbol{d}_j$ represents the query position in the coordinate system of the neural point $\boldsymbol{m}_j$. Then, we feed the concatenated vector to a globally shared geometric decoder D_θ^g, a shallow MLP containing M_{mlp} hidden layers with N_{mlp} neurons, to predict the SDF value s_j as

$$s_j = D_\theta^g(\boldsymbol{f}_j^g, \boldsymbol{d}_j) \tag{3}$$

where the relative coordinate $\boldsymbol{d}_j$ is given by

$$\boldsymbol{d}_j = \boldsymbol{q}_j(\boldsymbol{p} - \boldsymbol{x}_j)\boldsymbol{q}_j^{-1}. \tag{4}$$

As shown in Fig. 3(b), the concatenation of relative coordinate $\boldsymbol{d}_j$ makes the prediction invariant to the local translation and rotation, thus enabling the elasticity of the map after loop closure correction, which will be further explained in Section III-F.

The predicted SDF values s_j of the K nearest neural points at the query position $\boldsymbol{p}$ are then interpolated as the final prediction $s = S(\boldsymbol{p})$ by inverse distance weighting, given by

$$S(\boldsymbol{p}) = \sum_{j \in \mathcal{N}_p} \frac{w_j}{\sum_{k \in \mathcal{N}_p} w_k} s_j \tag{5}$$

where the weights w_j are defined as

$$w_j = \|\boldsymbol{p} - \boldsymbol{x}_j\|^{-2}. \tag{6}$$

Likewise, we define the stability $\mu = H(\boldsymbol{p})$ at the query position $\boldsymbol{p}$ as a distance-weighted mean of the stability μ_j of the K nearby neural points by switching s_j with μ_j in (5).

Intuitively, the final prediction can be regarded as a voting of the individual prediction from each neighboring neural point $\boldsymbol{m}_j$ or an ensemble of multiple locally defined DeepSDF-like autodecoder models. In contrast with the relatively deep decoder used in DeepSDF [53], which needed to be pretrained to represent the object globally, we represent the local geometries mainly in the locally defined latent feature space of neural points and use a shallow decoder as a general interpreter to map from the feature space to SDF values.

2) Map Data Structure: For fast neural point indexing and neighborhood search, we maintain a voxel hashing data structure with a fixed voxel resolution v_p and a hash table size T. This structure serves to organize the neural points, ensuring that each voxel contains no more than one active neural point. In line with prior work [47], [78], we use a spatial hashing function $\kappa = h(\boldsymbol{p}_W)$ to map from a position $\boldsymbol{p}_W \in \mathbb{R}^3$ to an entry $\kappa \in \mathbb{Z}$ in the hash table. This entry stores either the index i of a neural point $\boldsymbol{m}_i$ in the map $\mathcal{M}$ or the default value -1, indicating that the entry is not yet occupied.

In contrast to point-SLAM [65], for efficient proximity search during the SDF prediction at a query position $\boldsymbol{p}$, we use the voxel structure to find the neural points in the voxel-defined neighborhood $\mathcal{N}_p^v$ containing $N_n \times N_n \times N_n$ voxels centered at the corresponding voxel of $\boldsymbol{p}$ with constant time access. We sort the distances and take the K nearest neural points in $\mathcal{N}_p^v$ to get the K-neighborhood $\mathcal{N}_p$ for SDF prediction. The larger N_n is, the larger the receptive field would be while the computation time for neighborhood search would increase. If the number of neighboring neural points K_n is $1 \leq K_n < K$, we conduct SDF interpolation among the K_n points.

3) Map Initialization and Update: We initialize or update neural points at each timestep using the measured point cloud. For a point $\boldsymbol{p}_W$ in the world frame W measured at timestep t, we will initialize a new neural point $\boldsymbol{m}_k$, append it to the map $\mathcal{M}$ and set the hashing entry's value as the new neural point's index, i.e., $\kappa \leftarrow k$ only under the following three circumstances.

1) The hashing entry is not yet occupied, i.e., $\kappa = -1$.
2) There is a hash collision, i.e., the position $\boldsymbol{x}_\kappa$ of currently stored neural point $\boldsymbol{m}_\kappa$ in the voxel of the added point $\boldsymbol{p}_W$ is far away from $\boldsymbol{p}_W$ due to hash collision.
3) The stored neural point $\boldsymbol{m}_\kappa$ is no longer active, i.e., $D(t) - D(t_\kappa^u) > d_l$, meaning the travel distance from the last updated timestep t_κ^u of the neural point to the current timestep t is larger than a travel distance threshold d_l. This is used to differentiate between the historical observation and the present observation at the same position in the world frame upon revisiting.

In such cases, we initialize a new neural point $\boldsymbol{m}_k$ to have a position $\boldsymbol{x} = \boldsymbol{p}_W$, an identity quaternion $\boldsymbol{q} = (1, 0, 0, 0)$, a zero feature vector $\boldsymbol{f}^g = \boldsymbol{0}$, creation and last update timestep as $t^c = t^u = t$, and the stability as $\mu = 0$. In the latter two cases, the originally stored neural point is replaced by the new one and no longer indexed by the voxel hashing map. However, they are kept in the global neural point map for the sake of loop closure correction (cf., Section III-E) and globally consistent map adjustment (cf., Section III-F).

4) Local Map: Our method distinguishes between an active local map $\mathcal{M}_l$ and a global map $\mathcal{M}$. The registration described in Sections III-C and III-E are performed using the local map $\mathcal{M}_l$ to avoid the alignment to the inconsistent historical observations caused by odometry drift.

In previous works, a local map is often defined using either a spatial window [52], [78] or a temporal window [4], [85]. We propose to use both of them. In practice, the speed of the robot may vary a lot during the operation. For instance, the robot may stop at a position for a long time. Since the drift of the odometry is often proportional to the accumulated travel distance $D(t)$ instead of the time span, we propose to use a travel distance threshold instead of the timestep threshold [4], [85] to determine the local temporal window. Consequently, we define a local map $\mathcal{M}_l$ centered at the sensor position $\boldsymbol{t}_{WC_t}$ at timestep t. This map includes all neural points within both the spatial window $\|\boldsymbol{x}_i - \boldsymbol{t}_{WC_t}\|_2 < r_l$ and the temporal window $D(t) - D(t_i^c) <$

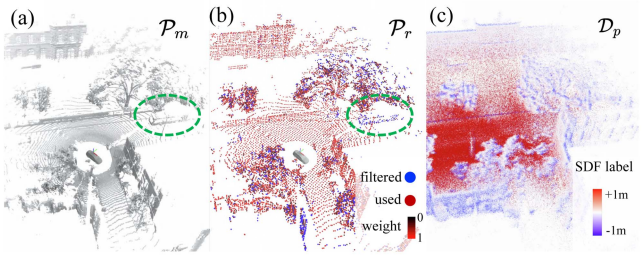


Fig. 4. Example during the operation of PIN-SLAM at a timestep: (a) Point cloud for mapping $\mathcal{P}_m$. A moving bus is highlighted in the green circle. (b) Sparser point cloud for registration $\mathcal{P}_r$, colorized according to the point-wise registration weight from black to red. The points in blue are filtered, mainly lying at dynamic objects, rough vegetation, and newly observed regions. (c) Points from the training sample pool $\mathcal{D}_p$ for map optimization, colorized according to their SDF target values from blue to red.

d_l. Here, r_l represents the radius of the local spatial window, and d_l denotes the previously mentioned travel distance threshold.

Next, we will explain how we incrementally optimize the neural point features in the local map $\mathcal{M}_l$ using a sliding window-like training sample pool $\mathcal{D}_p$.

5) Map Training Samples: At each timestep t, we take samples along the rays from the voxel-downsampled egocentric scan $\mathcal{P}_m = \{\boldsymbol{p} \in \mathbb{R}^3\}$ to collect training data. On each ray $\boldsymbol{r} = \boldsymbol{p}/\|\boldsymbol{p}\|_2$ from the sensor to the measured point $\boldsymbol{p}$, we can represent a sampled point $\boldsymbol{u}$ in the sensor frame C_t uniquely with its depth d along the ray, given by: $\boldsymbol{u} = d\boldsymbol{r}$. We take the endpoint ($d = \|\boldsymbol{p}\|_2$) and N_s points close to the surface with the depth sampled from a Gaussian distribution $d_s \sim \mathcal{N}(\|\boldsymbol{p}\|_2, \sigma_s^2)$ with the endpoint as the mean and σ_s as the standard deviation. In addition, we sample N_f points uniformly in the free space in front of the surface with the depth $d_f \sim U(\zeta_{\min}\|\boldsymbol{p}\|_2, \|\boldsymbol{p}\|_2 - 2\sigma_s)$ and N_b points in the truncated free space behind the surface with the depth $d_b \sim U(\|\boldsymbol{p}\|_2 + 2\sigma_s, \|\boldsymbol{p}\|_2 + d_b)$, where $\zeta_{\min}$ is the minimum sample depth ratio and d_b represents the maximum sample range behind surfaces. For each sample point $\boldsymbol{u}$, we take the projective signed distance along the ray as its SDF target value $\hat{s} = \|\boldsymbol{p}\|_2 - d$. Although the projective distance would always be an overestimation, it is computed quickly and is a good approximation when the sample point is close to the surface.

We define the training samples $\mathcal{D}$ at timestep t as N_t sample positions and their target values from all the rays, given by

$$\mathcal{D} = \{(\boldsymbol{u}_j, \hat{s_j}, t) \mid j = 1, \ldots, N_t\} \tag{7}$$

where $N_t = M_t(N_s + N_f + N_b + 1)$ and M_t is the number of points (rays) in point cloud $\mathcal{P}_m$. We associate the training samples with the corresponding sensor frame C_t by recording the timestep t of each sample so that we can later transform each sample point into the world frame.

To tackle the "catastrophic forgetting" problem in incremental mapping, we maintain a training sample pool $\mathcal{D}_p$ by appending the samples $\mathcal{D}$ from each timestep for replaying the historical samples, as shown in Fig. 4(c). We filter $\mathcal{D}_p$ at each timestep t by removing the samples lying outside a local sliding window with a radius r_p, centered at the current sensor position $\boldsymbol{t}_{WC_t}$, given by

$$\left\|\boldsymbol{u}_W^i - \boldsymbol{t}_{WC_t}\right\|_2 > r_p \tag{8}$$

where $\boldsymbol{u}_W^i = \mathsf{T}_{WC_{t_i}}\boldsymbol{u}_i$ is the sample position in world frame and $r_p = r_l - \frac{\sqrt{3}}{2}N_n v_p$ is set to enforce that the training samples would only affect the neural points in the local map, considering the local map radius r_l, neighborhood voxel counts N_n, and the voxel size v_p. In addition, to deal with the case when the robot stops, if the number of samples in $\mathcal{D}_p$ exceeds N_p, we randomly keep N_p samples.

6) Map Training Losses: We want to train our neural points to predict SDF value on arbitrary locations and these neural point features need to be trained ideally during the incremental mapping process. During incremental mapping, we randomly sample from the training sample pool $\mathcal{D}_p$ in batch for training. For the training of the SDF, we use a similar loss function as our previous work [100], which combines the binary cross entropy (BCE) loss and the Ekional regularization loss.

At a sample position $\boldsymbol{u}_W^i = \mathsf{T}_{WC_{t_i}}\boldsymbol{u}_i$ in the world frame, we map both the SDF prediction $s_i = S(\boldsymbol{u}_W^i)$ from the nearby neural points and target value $\hat{s}_i$ to the range of $[0, 1]$ by a scaled sigmoid function $\Phi_s(s) = 1/(1 + e^{s/\sigma_t})$ as

$$o_i = \Phi_s(s_i), \quad \hat{o}_i = \Phi_s(\hat{s}_i) \tag{9}$$

and then the BCE loss is calculated for N samples as

$$\mathcal{L}_{\text{BCE}} = \frac{1}{N}\sum_{i=1}^{N} \hat{o}_i \log(o_i) + (1 - \hat{o}_i)\log(1 - o_i). \tag{10}$$

The BCE loss enables fast convergence of the SDF [100]. It accomplishes the logistic regression of the projective SDF value and realizes the soft truncation of signed distance close to the surface. We can adjust the truncation smoothness using the scale factor σ_t in the scaled sigmoid function Φ_s.

One necessary property of SDF is the Ekional equation [16], i.e., $\|\nabla S(\boldsymbol{x})\|_2 = 1$. Therefore, we use the Ekional loss $\mathcal{L}_{\text{eik}}$ to enforce the regularity and validity of the fitted SDF

$$\mathcal{L}_{\text{eik}} = \frac{1}{N}\sum_{i=1}^{N} \left(\left\|\nabla S\left(\boldsymbol{u}_W^i\right)\right\|_2 - 1\right)^2. \tag{11}$$

In line with Neuralangelo [35], we calculate the numerical gradient by manually adding perturbations instead of using the analytical gradient based on automatic differentiation. This enables the backpropagation updates beyond the local hash grid that stores neural points, resulting in a smoother SDF gradient for the calculation of $\mathcal{L}_{\text{eik}}$. The gradient component at a position $\boldsymbol{x}$ on the x-axis is given by

$$\nabla_x S(\boldsymbol{x}) = \frac{S(\boldsymbol{x} + \boldsymbol{\epsilon}_x) - S(\boldsymbol{x} - \boldsymbol{\epsilon}_x)}{2\epsilon} \tag{12}$$

where $\boldsymbol{\epsilon}_{\boldsymbol{x}} = (\epsilon, 0, 0)^\top$ is the perturbation vector on the x-axis and ϵ is the perturbation step size used on every axis. The gradient on the y- and z-axes can be calculated likewise.

The final loss is then formulated as a weighted sum of BCE and Ekional term, given by

$$\mathcal{L} = \mathcal{L}_{\text{bce}} + \lambda_e \mathcal{L}_{\text{eik}} \tag{13}$$

where λ_e is the weight for Ekional loss. During training, we minimize the loss by optimizing latent features of the involved neural points in the local map. For each sample point $\boldsymbol{u}_i$, we also update the last update timestep t_j^u and stability μ_j of each

neural point $\boldsymbol{m}_j$ in the neighborhood $\mathcal{N}_p$ of $\boldsymbol{u}_i$, given by

$$t_j^u \leftarrow \max(t_j^u, t_i), \quad \mu_j \leftarrow \mu_j + \frac{w_j}{\sum_{j\in\mathcal{N}_p} w_j} \tag{14}$$

where t_i is the timestep of this training sample, and the weight w_j can be calculated as (6).

Next, we will explain how we use the proposed map representation, namely, the PIN map, to realize a LiDAR SLAM system toward globally consistent mapping.

B. Preprocessing

For each egocentric point cloud $\mathcal{P}$ measured at timestep t, we voxel-downsample $\mathcal{P}$ into the point cloud for mapping $\mathcal{P}_m$ with a smaller voxel size v_m, and the point cloud for registration $\mathcal{P}_r$ with a larger voxel size v_r. During the downsampling, we keep exactly one point, namely, the one whose coordinate is the closest to the voxel center.

Typically for LiDAR sensors, the point cloud of one sweep (frame) consists of the continuously scanned points measured across the acquisition time. Therefore, for each scan frame, we first need to revert the distortions of the point cloud caused by the sensor motion. In line with Vizzo et al. [78], we use a constant velocity model for motion prediction. Based on this model, we first deskew the points $\mathcal{P}_r$ for registration before the odometry estimation by interpolating the motion prediction with the pointwise timestamp. We deskew $\mathcal{P}_m$ with the more accurate ego-motion from odometry afterward.

C. Odometry Estimation

To achieve efficient and robust odometry, we propose a correspondence-free, scan-to-implicit map registration method based on second-order optimization under multiple weighting strategies, which is based on the approach proposed by Wiesmann et al. [86] targeting localization in known maps.

Our goal is to align the source point cloud $\mathcal{P}_r$ at timestep t to the neural SDF of the local PIN map $\mathcal{M}_l$ by finding the transformation $\mathsf{T}^* \in \mathrm{SE}(3)$ minimizing the least square error of the SDF prediction at the transformed points, given by

$$\mathsf{T}^* = \underset{\mathsf{T}}{\operatorname{argmin}} \sum_{\boldsymbol{p}\in\mathcal{P}_r} S\left(\mathsf{T}\boldsymbol{p}\right)^2 \tag{15}$$

which can be solved by a Levenberg–Marquardt optimization taking the initial guess predicted by a constant velocity model.

We denote the 6DOF transformation parameters as the Lie algebra $\boldsymbol{\xi} = [\boldsymbol{t}, \boldsymbol{\Theta}] = \log(\mathsf{T})$, where $\boldsymbol{t}$ is the translation vector and $\boldsymbol{\Theta} = \log(\boldsymbol{R})$ is the axis-angle representation of the rotation matrix $\boldsymbol{R}$. The Jacobian of the transformed point $\boldsymbol{p}_i' = \mathsf{T}\boldsymbol{p}_i$ with regards to $\boldsymbol{\xi}$ is given by

$$\boldsymbol{J}_i = \left[\frac{\partial S\left(\boldsymbol{p}_i'\right)}{\partial \boldsymbol{t}}, \frac{\partial S\left(\boldsymbol{p}_i'\right)}{\partial \boldsymbol{\Theta}}\right] = \left[\boldsymbol{g}_i^{\mathsf{T}}, \left(\boldsymbol{p}_i' \times \boldsymbol{g}_i\right)^{\mathsf{T}}\right] \tag{16}$$

where $\boldsymbol{g}_i = \nabla S(\boldsymbol{p}_i')$ is the distance gradient at $\boldsymbol{p}_i'$, which can be queried from the implicit neural distance field of PIN map by automatic differentiation. In our experiments, we discovered that using this analytical gradient for odometry estimation enhances robustness compared to utilizing the smoother but less precise numerical gradient employed for the Ekional regularization in mapping, see (12).

Intuitively, the registration can be solved by knowing, in which direction, given by $\boldsymbol{g}$ and how much, given by $S(\boldsymbol{p}')$, we have to go. This is conducted without knowing the explicit point-to-point correspondences. The *de facto* optimization target in previous neural implicit SLAM approaches [65], [71], [92], [101] is the depth rendering loss, which can be seen as related to the point-to-point metric with projection-based data association. Our method optimizes the point-to-model SDF loss, which is similar to the point-to-plane metric in ICP with the closest surface-based data association. The latter method has a faster convergence rate and more robust optimization according to the study by Rusinkiewicz, and Levoy [64].

We then approximate the Hessian matrix $\boldsymbol{H}$ as $\boldsymbol{H} = \boldsymbol{J}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{J}$ and calculate the gradient of the target function as $\boldsymbol{g} = \boldsymbol{J}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{b}$, where $\boldsymbol{J} \in \mathbb{R}^{N_r\times 6}$ is the Jacobian, $\boldsymbol{P} \in \mathbb{R}^{N_r\times N_r}$ is the weight matrix, and $\boldsymbol{b} \in \mathbb{R}^{N_r}$ is the residual vector. We filter points that have fewer than K neural points in their N_p^v neighborhood and set each row of $\boldsymbol{J}$ using the N_r remaining points as (16). For each iteration, the increment of the transformation parameters $\delta\boldsymbol{\xi}$ is given by

$$\delta\boldsymbol{\xi} = \left(\boldsymbol{H} + \lambda_d \operatorname{diag}\left(\boldsymbol{H}\right)\right)^{-1}\boldsymbol{g} \tag{17}$$

where λ_d is the damping parameter for Levenberg–Marquardt optimization. The termination criterion is determined by a threshold γ_c for the applied correction $\delta\boldsymbol{\xi}$ and a maximum number of iterations τ_c.

To robustify the optimization under noisy measurements and underfitted map, we apply the Geman-McClure (GM) robust kernel for both the SDF residual $S(\boldsymbol{p}_i')$ and the SDF gradient anomaly ϵ_i to down-weight the potential measurement noise, dynamic objects, and underfitted details of the implicit map. The pointwise robust kernel weights are given by

$$w_i^r = \left(\frac{\kappa_r}{\kappa_r^2 + S(\boldsymbol{p}_i')^2}\right)^2 \tag{18}$$

$$w_i^g = \left(\frac{\kappa_g}{\kappa_g^2 + \varepsilon_i^2}\right)^2 \tag{19}$$

where κ_r and κ_g are the scale parameters of the robust kernel. The gradient anomaly ε_i is defined as the deviation of the distance gradient from the Ekional equation, given by

$$\varepsilon_i = \left|\, \|\nabla S\left(\boldsymbol{p}_i'\right)\|_2 - 1\right|. \tag{20}$$

To apply the robust kernel to the optimization, we scale each diagonal element of the weight matrix $\boldsymbol{P}$ with the point-wise robust kernel weights $w_i^r w_i^g$. An example of the filtering and point-wise weight of $\mathcal{P}_r$ is shown in Fig. 4(b).

After convergence, we additionally calculate the Eigenvalues $\boldsymbol{\lambda}$ of the Hessian matrix $\boldsymbol{H}$ for a degeneracy check [96]. The registration will be regarded as a success if the average residual $\bar{\boldsymbol{b}} > b_s$, the ratio of valid points $\alpha > \alpha_s$, and the minimum Eigenvalue $\min(\boldsymbol{\lambda}) > \lambda_s$ for the last iteration, where b_s, α_s, and λ_s are the thresholds. If the optimization succeeds, i.e., all checks are fulfilled, we take $\mathsf{T}_{WC_t} = \exp(\boldsymbol{\xi})$ as the estimated pose of the current frame C_t for mapping (Section III-D). Otherwise, we directly use the initial guess, i.e., the constant velocity estimate, but skip the mapping step to avoid introducing wrong measurements.

For a potential PGO, we add the odometry transformation $\mathsf{T}_{C_{t-1}C_t}$ as an odometry edge connecting pose node C_{t-1} and C_t in the pose graph $\mathcal{G}$.

D. Mapping and Local BA

With the sensor poses T_{WC_t} estimated by the odometry at each timestep, we can update and optimize the PIN map $\mathcal{M}$ using the point cloud for mapping $\mathcal{P}_m$.

First, we filter the dynamic objects from $\mathcal{P}_m$ and use only the static part for mapping. Inspired by Dynablox [66], we make use of the assumption that a measured point lying in the stable free space can be regarded as a dynamic point. Thereby, we filter points fulfilling $(S(\boldsymbol{p}_W) > \gamma_d) \wedge (H(\boldsymbol{p}_W) > \gamma_\mu)$, where $S(\boldsymbol{p}_W)$ and $H(\boldsymbol{p}_W)$ are the SDF and stability prediction at the measured point $\boldsymbol{p}_W = \mathsf{T}_{WC_t}\boldsymbol{p}$ in the world frame. γ_d and γ_μ are the free space distance and stability threshold, respectively. Only the static points remain for mapping.

As we explained in Section III-A, we get training samples $\mathcal{D}$ by sampling along the rays from the sensor to the static points in $\mathcal{P}_m$. We initialize new neural points in the map $\mathcal{M}$ using the close-to-surface sample points $\mathcal{D}_s$ transformed by current pose T_{WC_t}. We then reset the local map $\mathcal{M}_l$ centered at the current position $\boldsymbol{t}_{WC_t}$, and append current training samples $\mathcal{D}$ to the training sample pool $\mathcal{D}_p$. Next, we conduct PIN map optimization by gradient descent using the training samples from $\mathcal{D}_p$ in batches. When the SLAM starts, we jointly optimize the neural point feature $\boldsymbol{f}^g$ in the local map $\mathcal{M}_l$ and the weights of the MLP decoder D^g_θ for the first F_{mlp} timesteps. After F_{mlp} timesteps, we freeze D^g_θ and optimize only the neural point features to avoid catastrophic forgetting due to the ever-changing decoder while incremental mapping.

The odometry estimation relies on pairwise scan-to-map registration and always optimizes only the latest pose. It ignores the multiview consistency of multiple scans in the local sliding window, thus pruning to rapidly accumulated drift. Inspired by the BA technique widely employed in photogrammetry and computer vision, Liu et al. [38] introduced a LiDAR BA algorithm. This algorithm aims to refine odometry by simultaneously optimizing the scene geometry and a collection of sensor poses. The target of this optimization is to minimize inconsistencies among explicit geometric features, such as edges and planes.

We propose an implicit local BA approach making use of the implicit neural map without the tedious extraction of explicit geometric features. For every F_{ba} timesteps, we jointly optimize the sensor poses at the past N_{ba} timesteps, and the neural point features in the local map $\mathcal{M}_l$, taking the current state as the initial guess. We optimize the L2 SDF regression loss by gradient descent using only the endpoint samples from the training sample pool $\mathcal{D}_p$ to achieve a more consistent local map and poses. The loss function is given by

$$L_{\text{ba}} = \sum_{\tau \in \mathcal{T}_{\text{ba}}} \sum_{\boldsymbol{p}_i \in \mathcal{P}_m^\tau} S\left(\mathsf{T}_{WC_\tau}\boldsymbol{p}_i\right)^2 \tag{21}$$

where $\mathcal{T}_{\text{ba}} = \{t - N_{\text{ba}} + 1, \ldots, t-1, t\}$ are the timesteps used for local BA. t is the current timestep and $\mathcal{P}_m^\tau$ is the point cloud for mapping at timestep τ.

After the map optimization and local BA, we assign the updated local map $\mathcal{M}_l$ back to the global map $\mathcal{M}$.

E. Loop Closure Detection

Detecting loop closures is essential to correct the accumulated drift of the odometry for globally consistent mapping.

First, we use a distance-based criterion for identifying local loop closures. Specifically, we assess whether the positions $\boldsymbol{t}_t$ and $\boldsymbol{t}_{t_i}$ of the current scan at C_t and a historical scan at C_{t_i} (where $t_i < t$ and $D(t) - D(t_i) > d_l$) meet the condition

$$\|\boldsymbol{t}_t - \boldsymbol{t}_{t_i}\|_2 < d_{\text{loop}}. \tag{22}$$

If there is no local loop candidate, we search for global loop closures by comparing frame-wise descriptors. We propose a neural point feature enhanced local map descriptor for typical driving scenarios. This descriptor is used for global loop detection and provides a semimetric localization as the initial guess for the relative loop transformation. To address the place recognition challenges arising from diverse scan patterns and occlusions caused by varied viewpoints and dynamic objects, we use the local map as the processing unit for loop detection, instead of relying on a single scan. Therefore, at each timestep t, with a latency of F_{lat} timesteps waiting for sufficient map training using the most recent observations, we generate a local context descriptor $\mathbf{U}_t$ using neural points in the local map $\mathcal{M}_l$.

Similar to Scan Context [27], the neural points in the local map are mapped into discretized 2-D polar-coordinate bins by a bird-eye-view projection with the ring resolution H_r and sector resolution H_s. Instead of encoding the maximum point height in each bin to create the descriptor [27], we encode the average neural point features optimized by the self-supervised online mapping and keep a F_g dimensional vector in each bin, thus resulting in a descriptor $\mathbf{U}_t \in \mathbb{R}^{H_r \times H_s \times F_g}$. We average the sector dimensions of $\mathbf{U}_t$ to get a retrieval key $\mathbf{R}_t \in \mathbb{R}^{H_r \times F_g}$ as a rotation-invariant descriptor of the local map for fast global descriptor retrieval. We employ the mean of ring-wise feature cosine distance as the comparative metric and select the closest historical frame C_k as a loop candidate. Then, the rough relative rotation between the local map centered at the current and candidate loop frame can be estimated by shifting along the sector dimension of $\mathbf{U}_t$ and taking the shift number n^* with the minimum bin-wise feature cosine distance d_c^*, which is calculated as the mean of $H_r \cdot H_s$ bins. If $d_c^* < d_{\text{mc}}$, where d_{mc} is the cosine distance threshold, we accept the loop candidate and use the relative rotation around z-axis by $\frac{2n^*\pi}{H_s}$ as the rotation part of the semimetric localization.

Furthermore, to deal with the deficiency in translational invariance, we follow Scan Context++ [26] to use a polar context augmentation to enable the loop with large relative translation at the revisit. Specifically, for each scan position $\boldsymbol{t}_{WC_t}$, we generate $2V_a$ local map context descriptors at virtual positions $\boldsymbol{\eta}_t^j$ shifted from $\boldsymbol{t}_{WC_t}$ by increments of equal length d_v along a direction perpendicular to the motion path $\delta\boldsymbol{t} = \boldsymbol{t}_{WC_t} - \boldsymbol{t}_{WC_{t-1}}$, results in virtual scan positions

$$\boldsymbol{\eta}_t^j = \boldsymbol{t}_{WC_t} + \frac{j d_v}{\|\delta\boldsymbol{t}\|_2} \boldsymbol{R}_z\left(90^\circ\right)\delta\boldsymbol{t}, \quad j = -V_a, \ldots, V_a \tag{23}$$

where $\boldsymbol{R}_z(90^\circ)$ denotes the rotation around z-axis by 90°. We use these descriptors to query the most similar historical descriptors. We use the one resulting in the minimum cosine distance and take $\boldsymbol{\eta}_t - \boldsymbol{t}_{WC_t}$ as the translation part of semimetric localization results. We refer to Kim et al. [26] for more details regarding retrieval and augmentation.

When a candidate loop is detected, we register the current scan $\mathcal{P}_r$ to the local map $\mathcal{M}_l$ centered at the sensor position $\boldsymbol{t}_{WC_k}$ of the loop frame C_k as described in Section III-C. For scan-to-map registration of local loops, we initialize the transformation with the identity matrix. For global loops, we use the results obtained from semimetric localization as the initial

guess for the transformation. If the registration succeeds, we take the transformation $\mathsf{T}_{C_k C_t}$ resulting from the registration as the loop closure edge connecting node C_k and C_t in the pose graph $\mathcal{G}$. Otherwise, the candidate loop will be rejected.

F. Globally Consistent Implicit Neural Map Adjustment

Once a loop is detected and verified by scan-to-map registration, we conduct PGO taking the current poses as the initial guess and using constant information matrices. To avoid the almost redundant optimization, which would take up significant time, we disable the loop detection for F_{loop} timesteps after conducting a PGO. After performing PGO, the neural points in the global map $\mathcal{M}$ will move along with their associated frames. We apply a transformation to both the position and orientation of each neural point, as detailed in Section III-A. The transformation is determined according to the associated sensor frame $C_{t_i^m}$ of the neural point as the mean of its creation and last update timestep $t_i^m = \lfloor (t_i^c + t_i^u)/2 \rfloor$. We denote the pose difference of frame C_t before and after the PGO as $\delta\mathsf{T}_t = \mathsf{T}_{WC_t'}^{-1}\mathsf{T}_{WC_t}$ and the quaternion for the rotation part of $\delta\mathsf{T}_t$ as $\delta\boldsymbol{q}_t$. We can update the position $\boldsymbol{x}_i$ and orientation $\boldsymbol{q}_i$ of the neural point as

$$\boldsymbol{x}_i \leftarrow \delta\mathsf{T}_{t_i^m}\boldsymbol{x}_i, \quad \boldsymbol{q}_i \leftarrow \delta\boldsymbol{q}_{t_i^m}\boldsymbol{q}_i. \tag{24}$$

After transforming the neural points, we recreate the voxel hash map and reset the local map to ensure correct neural point indexing. If there are multiple neural points in one voxel, we keep the one whose stability μ is higher to avoid redundant map memory consumption in the revisiting region. To enable incremental mapping after PGO, we also transform the samples in the training sample pool $\mathcal{D}_p$ according to the optimized poses of their associated frames.

IV. Extension to RGB-D or Metric-Semantic SLAM

As a proof of concept, we show how we can extend our approach to RGB-D and metric-semantic SLAM by extending the environment model with color or semantic information.

If the input point cloud contains point-wise RGB values or semantic classification probabilities predicted by any off-the-shelf segmentation models [45], we can additionally predict the RGB value $\boldsymbol{c} \in \mathbb{R}^3$ and the L-class semantic probability $\boldsymbol{v} \in \mathbb{R}^L$ at $\boldsymbol{p}$ by each nearby neural point $\boldsymbol{m}_j$ in the same way as the SDF value prediction

$$\boldsymbol{c}_j = D_\theta^c\left(\boldsymbol{f}_j^c, \boldsymbol{d}_j\right), \quad \boldsymbol{v}_j = D_\theta^s\left(\boldsymbol{f}_j^s, \boldsymbol{d}_j\right) \tag{25}$$

where D_θ^c and D_θ^s are the globally shared color and semantic decoder, and $\boldsymbol{f}_j^c$ and $\boldsymbol{f}_j^s$ are the latent feature of neural point $\boldsymbol{m}_j$ assigned for color and semantics. We can interpolate the final RGB prediction $\boldsymbol{c} = C(\boldsymbol{p})$ and semantic prediction $\boldsymbol{v} = V(\boldsymbol{p})$ in the same way as (5). To train the color and semantic field, we set the color target values $\hat{\boldsymbol{c}}_j$ and semantic pseudolabels $\hat{\boldsymbol{v}}_j$ for the $N_s + 1$ close-to-surface samples as the same value at the endpoint. We then add the L1 loss $\mathcal{L}_{\text{col}}$ for color regression with a weight λ_c or the L-class cross-entropy loss $\mathcal{L}_{\text{sem}}$ for semantic segmentation with a weight λ_s to (13) using the close-to-surface training samples

$$\mathcal{L}_{\text{col}} = \frac{1}{N_c}\sum_{i=1}^{N_c}\left\|\hat{\boldsymbol{c}}_i - \boldsymbol{c}_i\right\|_1 \tag{26}$$

$$\mathcal{L}_{\text{sem}} = -\frac{1}{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{L} p_i^j \log \hat{p}_i^j \tag{27}$$

where $\hat{\boldsymbol{c}}_i$ and $\boldsymbol{c}_i = C(\boldsymbol{u}_W^i)$ are the color label and prediction while p_i^j is the semantic prediction probability for class j and $\hat{p}_i^j$ is the one-hot probability for the semantic pseudolabel.

For odometry estimation explained in Section III-C, with the point-wise color available, we add the photometric term into the optimization with a weight of w_c to provide more constraints. In this case, we additionally minimize the photometric difference at the transformed position $\boldsymbol{p}_i'$ between the observed value $\boldsymbol{c}_i$ and the predicted value $C(\boldsymbol{p}_i')$ from the PIN map as

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{\boldsymbol{p}_i \in \mathcal{P}_r} S\left(\boldsymbol{p}_i'\right)^2 + w_c \left\| C\left(\boldsymbol{p}_i'\right) - \boldsymbol{c}_i \right\|_2^2. \tag{28}$$

The Jacobian for the photometric term $\boldsymbol{J}_i^c$ can be calculated likewise the geometric term as (16), by replacing the SDF gradient $\boldsymbol{g}$ with the color gradient defined by $\nabla C(\boldsymbol{p}_i')$

$$\boldsymbol{J}_i^c = \left[\nabla C\left(\boldsymbol{p}_i'\right)^\mathsf{T}, \left(\boldsymbol{p}_i' \times \nabla C\left(\boldsymbol{p}_i'\right)\right)^\mathsf{T}\right]. \tag{29}$$

We then append $\boldsymbol{J}_i^c$ to Jacobian matrix $\boldsymbol{J}$ for the iterative optimization as (17).

During or after the mapping, we can colorize the mesh reconstructed from the SDF by querying the color or semantic value at the position of the mesh vertices.

V. Experimental Evaluation

The main focus of this work is a LiDAR SLAM system for building globally consistent maps using a PIN map representation.

We present our experiments to show the capabilities of our method. The results of our experiments also support our key claims, which are as follows:

1) Our SLAM system achieves localization accuracy better or on par with state-of-the-art LiDAR odometry/SLAM approaches and is more accurate than recent implicit neural SLAM methods on various datasets using different range sensors.
2) Our method can conduct large-scale globally consistent mapping with loop closure thanks to the elastic neural point representation.
3) Our map representation is more compact than the previous counterparts and can be used to reconstruct accurate and complete meshes at an arbitrary resolution.
4) Our correspondence-free scan-to-implicit map registration and the efficient neural point indexing by voxel hashing enable our algorithm to run at the sensor frame rate on a single NVIDIA A4000 GPU.

A. Experimental Setup

1) Datasets: We extensively test our method on various datasets summarized in Table I. For driving scenes, we test on KITTI, MulRAN, and our self-collected dataset called IPB-car. KITTI odometry dataset [14] comprises 22 sequences of LiDAR scans acquired by a Velodyne HDL64 LiDAR mounted on a car driving through Karlsruhe, Germany. Reference poses are available on sequences 00-10. MulRAN dataset [25] is collected

TABLE I
CHARACTERISTIC OF THE DATASETS USED FOR EVALUATION

dataset	sensor	scenario	# seqs.	# frames
KITTI [14]	64-beam LiDAR	outdoor, car	22	43k
MulRAN [25]	64-beam LiDAR	outdoor, car	9	64k
IPB-Car	64/128-beam LiDAR	outdoor, car	4	43k
Newer College [57]	64/128-beam LiDAR	outdoor/indoor, handheld	7	53k
Hilti-21 [17]	64-beam LiDAR	outdoor/indoor, handheld	6	15k
Replica [70]	synthetic RGB-D	indoor, handheld	8	16k

using an Ouster OS1-64 LiDAR mounted on a car traversing the roads of Daejeon, South Korea. Note that for the MulRAN dataset, the LiDAR is blocked by the Radar sensor, resulting in a smaller FoV than KITTI. For KITTI and MulRAN, the poses measured by GNSS-INS are regarded as the reference for evaluation. We additionally collected another challenging robot car dataset in Bonn, Germany with an OS1-64 LiDAR in 2020 and an OS1-128 LiDAR in 2023. We generate the reference poses by fusing GNSS-INS, LiDAR odometry [78], loop closure constraints, and scan-to-map constraints between OS1-128 and a global map obtained by a geo-referenced terrestrial laser scanner (TLS) in a factor graph.

To further test our approach on handheld LiDARs, which have a less constant motion profile, we adopt the Newer College and Hilti-21 dataset. Newer College dataset [57] contains two longer sequences acquired by a handheld OS1-64 LiDAR and a couple of shorter sequences collected by an OS0-128 LiDAR on the campus of Oxford University, U.K. The reference poses are obtained by aligning each scan to the survey grade point cloud map measured by TLS. We also take the survey grade map as the reference model for mapping quality evaluation. The Hilti 2021 SLAM challenge dataset [17] comprises indoor sequences of offices, labs, and basements, as well as outdoor sequences of construction sites. These sequences were captured using a handheld OS0-64 LiDAR. The reference trajectories are measured by either a total station tracking system or a motion capture system.

Aside from LiDAR datasets, we adopt the Replica synthetic RGB-D dataset [70], which is a popular benchmark for recent neural RGB-D SLAM methods to show that our approach can also achieve precise pose estimation using RGB-D images.

2) Parameters and Implementation Details: We list the parameter setting of our approach in Table II. All the length-based parameters used in our method are set adaptively according to the maximum used measurement range $r_{\max}$ of the sensor. The maximum range is $r_{\max} = 80\,\text{m}$ for KITTI, MulRAN, and self-recorded IPB-Car datasets and $r_{\max} = 60\,\text{m}$ for Newer College, and Hilti-21 datasets. For RGB-D datasets, we have $r_{\max} = 8\,\text{m}$. Our model is implemented mainly in PyTorch [54]. For the online training of PIN map, we use Adam optimizer [28] with a learning rate of 0.01 and a batch size of 16384 for 15 iterations. Note that we train for 600 iterations at the first frame for the map initialization. To improve training efficiency, we only use 1/10th of all training samples to calculate the numerical gradient for the Ekional loss $\mathcal{L}_{\text{eik}}$. For local BA, the poses are represented as Lie algebra tensors with PyPose [79]. We use a learning rate of 0.01 for the neural point features, 0.0001 for the poses, and a batch size of 16 384 for 80 iterations. For PGO, we employ GTSAM [9] and optimize the pose graph using Levenberg–Marquardt optimization with a maximum of 50 iterations.

TABLE II
HYPERPARAMETERS OF OUR APPROACH

type	symbol	value	description
PIN map	v_p	$0.005r_{\max}$	voxel hashing map resolution
	F_g	8	dimensions of neural point latent features
	$M_{\text{mlp}}, N_{\text{mlp}}$	$2, 64$	MLP level and neuron count
	N_n	5	neighborhood voxel count on each axis
	K	6	neighborhood neural point number
	σ_t	$0.001r_{\max}$	sigmoid function scale factor
	ϵ	$0.002r_{\max}$	perturbation step for numerical gradient
	λ_e, λ_c, λ_s	$0.5, 0.5, 1.0$	weight for Ekional, color, and semantic loss
	r_l	$1.05r_{\max}$	radius of local map
	d_l	$4r_l$	travel distance threshold of local map
	γ_d, γ_μ	$0.008r_{\max}, 4.0$	dynamic filtering threshold for SDF and stability
sampling	v_m	$0.001r_{\max}$	downsample voxel size for mapping
	σ_s	$0.003r_{\max}$	close-to-surface sampling standard deviation
	$\zeta_{\min}$	0.3	minimum sample depth ratio
	d_b	$4\sigma_s$	maximum sample range behind the surface
	N_s, N_f, N_b	$4, 2, 1$	surface, front, and behind free space sample count
	N_p	2×10^7	maximum sample count in training sample pool
odometry	v_r	$0.0075r_{\max}$	downsample voxel size for registration
	w_c	0.01	photometric tracking weight
	κ_r, κ_g	$0.005r_{\max}, 0.1$	GM kernel scale of residual and gradient anomaly
BA	F_{ba}	20	implicit BA frequency
	N_{ba}	50	count of poses optimized during BA
loop	F_{loop}	20	PGO frequency
	d_{loop}	$0.025r_{\max}$	distance threshold for local loop
	H_r, H_s	$20, 60$	ring and sector resolution of polar descriptor
	d_{mc}	0.3	cosine distance threshold for global loop
	V_a	6	polar descriptor augmentation count
	d_v	$0.02r_{\max}$	polar descriptor augmentation translation step

B. Localization Accuracy Evaluation

In this section, we compare the pose estimation performance of PIN-SLAM with state-of-the-art odometry/SLAM systems.

1) LiDAR Odometry Evaluation on KITTI: We first evaluate the pure odometry accuracy of our approach on the competitive KITTI odometry benchmark [14]. We denote the LiDAR odometry version of our approach as PIN-LO, which disables the loop closure detection correction module of PIN-SLAM. Since the LiDAR scans in KITTI odometry dataset are already deskewed, we disable the motion compensation for our method. We compare PIN LiDAR odometry against various LiDAR odometry systems using different map representations, such as feature points [52], [68], [82], denser voxel-downsampling points [10], [31], [78], normal distribution transformation [93], surfels [4], IMLS model [12], and triangle meshes [62], [76]. Since our approach is based on online learning of a neural implicit map, we additionally compare our method against the learning-based LiDAR odometry systems, including those using supervised learning [33], [80], [81] and Nerf-LOAM [11], which also utilizes a neural implicit map. Note that those systems based on supervised learning are trained on sequences 00-,06 and tested on sequences 07–10 while our method and Nerf-LOAM do not rely on any pretraining. We use the average relative translational error (ARTE) [14] as the metric for odometry drift evaluation. Due to the nondeterministic nature of online training in our method, the results may have slight variations with different random seeds. Therefore, we opt to report the mean and standard deviation of the metrics calculated from ten different runs with varying random seeds on the KITTI dataset. This allows a fair comparison with other methods and provides a better understanding of the stochastic nature of PIN-SLAM. As shown in Table III, PIN LiDAR odometry outperforms all the learning-based methods and most of the nonlearning-based methods. It attains an average translation error of 0.51% with a standard deviation of 0.02%, which is on par with the two open-source state-of-the-art LiDAR odometry KISS-ICP [78] and CT-ICP [10] using a voxel-downsampled point cloud map.

TABLE III
LIDAR ODOMETRY PERFORMANCE COMPARISON ON *KITTI* LIDAR DATASET WITH MOTION COMPENSATED POINT CLOUD WITH THE AVERAGE RELATIVE TRANSLATIONAL DRIFTING ERROR (%)

Method	Type	00	01	02	03	04	05	06	07	08	09	10	**Avg.**	11-21
LeGO-LOAM [68]	feature points	2.17	13.4	2.17	2.34	1.27	1.28	1.06	1.12	1.99	1.97	2.21	2.49	-
F-LOAM [82]	feature points	0.78	1.43	0.92	0.86	0.71	0.57	0.65	0.63	1.12	0.77	0.79	0.84	0.72
MULLS [52]	feature points	0.56	0.64	0.55	0.71	0.41	0.30	0.30	0.38	**0.78**	**0.48**	0.59	0.52	0.65
VG-ICP [31]	dense points	2.16	2.38	0.99	0.67	0.55	0.45	**0.24**	0.99	1.74	0.95	0.95	1.10	-
CT-ICP [10]	dense points	**0.49**	0.76	0.52	0.72	0.39	**0.25**	0.27	**0.31**	0.81	0.49	**0.48**	**0.50**	**0.59**
KISS-ICP [78]	dense points	0.52	**0.63**	**0.51**	0.66	0.36	0.31	0.26	0.33	0.82	0.51	0.56	**0.50**	0.61
SuMa-LO [4]	surfels	0.73	1.71	1.06	0.66	0.38	0.50	0.42	0.39	1.02	0.48	0.71	0.73	1.39
Litamin-LO [93]	normal distribution	0.78	2.10	0.95	0.96	1.05	0.55	0.55	0.48	1.01	0.69	0.80	0.88	-
IMLS-SLAM [12]	implicit model	0.50	0.82	0.53	0.68	0.33	0.32	0.33	0.33	0.80	0.55	0.53	0.52	0.69
Puma [76]	mesh	1.46	3.38	1.86	1.60	1.63	1.20	0.88	0.72	1.44	1.51	1.38	1.55	-
SLAMesh [62]	mesh	0.77	1.25	0.77	**0.64**	0.50	0.52	0.53	0.36	0.87	0.57	0.65	0.68	-
LONet [33]	supervised	1.47†	1.36†	1.52†	1.03†	0.51†	1.04†	0.71†	1.70	2.12	1.37	1.80	1.33	-
PWCLONet [81]	supervised	0.78†	0.67†	0.86†	0.76†	0.37†	0.45†	0.27†	0.60	1.26	0.79	1.69	0.77	-
ELONet [80]	supervised	0.83†	0.55†	0.71†	0.49†	0.22†	0.34†	0.36†	0.46	1.14	0.78	0.80	0.61	1.92
Nerf-LOAM [11]	neural implicit	1.34	2.07	-	2.22	1.74	1.40	-	1.00	-	1.63	2.08	1.69	-
PIN-LO	neural implicit	0.55 (±0.02)	0.68 (±0.13)	0.54 (±0.02)	0.76 (±0.02)	**0.22** (±0.02)	0.30 (±0.01)	0.35 (±0.01)	0.34 (±0.02)	0.80 (±0.02)	0.54 (±0.06)	0.50 (±0.05)	0.51 (±0.02)	0.64 -

The numbers with † are the results of the training set for learning-based methods and are not included in the ranking. - indicates the number is not reported. We highlight the best results in bold and the second best in underscored. For the results of our method PIN-LO, we report the mean and standard deviation (in brackets) calculated from ten runs with different random seeds.

TABLE IV
SLAM PERFORMANCE COMPARISON (ATE RMSE [M]) ON *KITTI* LIDAR DATASET WITH MOTION COMPENSATED POINT CLOUD

Method	00*	01	02*	03	04	05*	06*	07*	08*	09*	10	**Avg.***	Avg.
SuMa [4]	1.1	14.6	8.0	1.0	0.3	0.7	0.6	1.1	3.7	1.2	1.4	2.3	3.1
MULLS [52]	1.1	**1.9**	5.4	0.7	0.9	1.0	0.3	0.4	2.9	2.1	1.1	1.9	1.6
Litamin2 [93]	1.3	15.9	3.2	0.8	0.7	0.6	0.8	0.5	**2.1**	2.1	1.0	1.5	2.4
SC-LeGO-LOAM [27], [68]	2.3	19.7	5.3	1.6	0.4	1.2	1.0	1.5	5.9	2.0	1.7	2.7	5.3
HLBA [37]‡	**0.8**	**1.9**	5.1	0.6	0.8	0.4	**0.2**	**0.3**	2.7	1.3	1.1	1.5	1.4
SC-F-LOAM [27], [82]‡	1.3	4.7	3.3	0.7	0.3	1.2	0.4	0.5	3.0	1.3	1.6	1.6	1.7
SC-KISS-ICP [27], [78]‡	1.0	3.7	**1.9**	**0.4**	0.3	0.3	0.3	**0.3**	2.2	**1.0**	**0.8**	**1.0**	**1.1**
PIN-LO	5.6 (±0.3)	4.3 (±1.7)	9.3 (±0.7)	0.7 (±0.1)	**0.1** (±0.0)	1.7 (±0.1)	0.5 (±0.0)	0.5 (±0.0)	3.0 (±0.2)	1.8 (±0.4)	**0.8** (±0.1)	3.2 (±0.3)	2.6 (±0.2)
PIN-SLAM	**0.8** (±0.1)	4.3 (±1.7)	2.1 (±0.4)	0.7 (±0.1)	**0.1** (±0.0)	**0.3** (±0.0)	0.4 (±0.0)	**0.3** (±0.0)	**2.1** (±0.4)	1.2 (±0.1)	**0.8** (±0.1)	**1.0** (±0.1)	1.2 (±0.2)

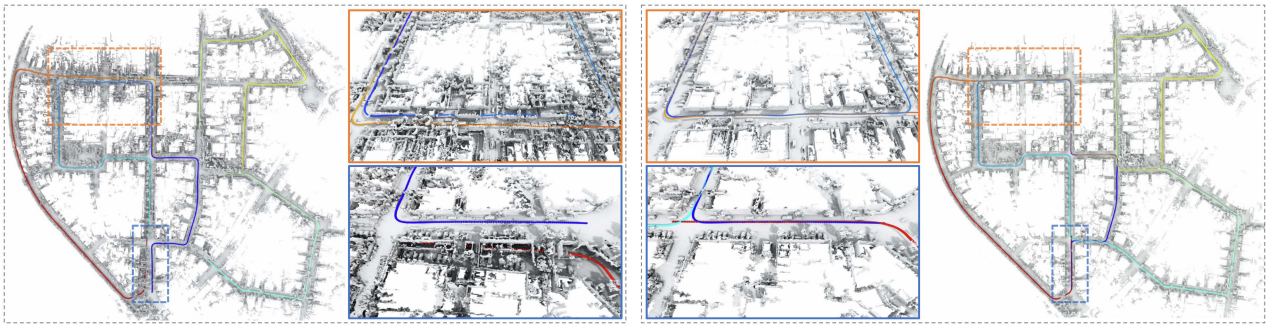
* Indicates the sequence is with loops, Avg.* denotes the average metric on sequences with loops. We highlight the best results in bold and the second best in underscored. ‡ indicates the method conducts offline PGO. For the results of our method PIN-LO and PIN-SLAM, we report the mean and standard deviation (in brackets) calculated from ten runs with different random seeds.

Our method demonstrates superior performance compared to Nerf-LOAM [11], the sole baseline using also an implicit neural map representation. This achievement is attributed to our better SDF training in the close-to-surface free space and the more robust point-to-SDF registration using Levenberg–Marquardt optimization combined with robust kernels instead of the gradient descent used by Nerf-LOAM.

In addition, as shown in the last column of Table III, our method performs among the best LiDAR odometry/SLAM systems on the hidden sequences 11–21 of the KITTI odometry leaderboard and is the best-ranked learning-based method.

2) LiDAR SLAM Evaluation: We proceed to evaluate the full system of PIN-SLAM, including the loop closure correction module. We test PIN-SLAM quantitatively and compare it against state-of-the-art LiDAR SLAM/odometry approaches on four public datasets and one self-recorded dataset collected by different robot platforms in various scenarios. Instead of using the ARTE, we adopt the root-mean-square error (RMSE) of the absolute trajectory error (ATE) [98] with Umeyama trajectory alignment as the localization accuracy metric since it can better reflect the global consistency of the estimated pose. For comparison, we focus on methods that either disclose their achieved ATE RMSE on our utilized datasets in their papers or release open-sourced code that supports our utilized datasets.

First, on the KITTI sequence 00–10, we compare our approach with four state-of-the-art LiDAR SLAM systems enabling loop closure correction, and HLBA [37], a postprocessing method using LiDAR BA with the pose initial guess of MULLS [52]. In line with previous work [37], [93], we report the accuracy at decimeter precision. As shown in Table IV, PIN-SLAM achieves the smallest average RMSE of 1.0 m on the sequences with loops and 1.2 m on all the eleven sequences with a standard deviation of 0.2 m calculated from ten runs with different random seeds. Notably, PIN-SLAM even outperforms the post-processing approach [37] while PIN-SLAM can run online. In addition to SC-LeGO-LOAM, an open-source SLAM system combining LeGO-LOAM [68] with scan context [27], we additionally implement SC-F-LOAM and SC-KISS-ICP, which perform offline PGO using the pose of more recent LiDAR odometry systems F-LOAM and KISS-ICP with the loop closures detected by scan context. We conduct a fine ICP registration between the query and retrieved point cloud to refine the loop transformation initial guess. PIN-SLAM outperforms SC-F-LOAM and achieves comparable localization accuracy to SC-KISS-ICP. Meanwhile, PIN-SLAM can maintain a continuous SDF map online for downstream tasks such as mesh reconstruction or path planning whereas the compared methods typically construct only a sparse point cloud map. We also report the result of PIN LiDAR odometry to show the significant improvement of PIN-SLAM on trajectory global consistency, with the RMSE decreasing from 3.2 to 1.0 m on sequences with loop closures. As shown in Fig. 5, PIN-SLAM manages to correct the drift of PIN LiDAR



(a) (b)

Fig. 5. Comparison of (a) locally consistent mesh with duplicated structures reconstructed by PIN LiDAR odometry—PIN-LO, and (b) globally consistent mesh reconstructed by PIN-SLAM built on KITTI sequence 00 after loop closure corrections—PIN-SLAM. The estimated trajectories are overlaid on the map and colorized according to the timestamp. Details of two revisited regions are highlighted in the boxes.

TABLE V
SLAM PERFORMANCE COMPARISON (ATE RMSE [M]) OF OUR METHOD VERSUS THE STATE-OF-THE-ART LIDAR ODOMETRY (ABOVE) AND LIDAR SLAM METHODS (BELOW) ON *MULRAN* LIDAR DATASET

Method	KA1	KA2	KA3	DC1	DC2	DC3	RS1	RS2	RS3	**Avg.**
F-LOAM [82]	49.16	35.93	38.36	36.89	26.72	31.97	91.85	104.25	119.79	59.44
KISS-ICP [78]	20.81	13.52	14.85	20.26	12.79	13.17	14.35	32.74	75.14	24.18
SuMa [4]	10.45	9.33	10.33	13.00	86.04	12.03	114.89	818.74	367.41	160.25
MULLS [52]	19.72	15.26	6.93	14.95	11.31	6.00	41.25	45.05	44.18	22.74
SC-LeGO-LOAM [27], [68]	5.45	5.49	5.70	6.95	5.49	6.29	19.05	16.04	30.91	11.26
HLBA [37]‡	3.36	3.75	3.53	**5.20**	3.22	2.54	8.92	**7.94**	10.26	5.71
SC-F-LOAM [27], [82]‡	4.74	4.70	4.32	9.67	5.57	3.98	17.72	22.42	24.07	10.80
SC-KISS-ICP [27], [78]‡	3.33	**2.80**	2.65	6.41	3.42	**2.13**	**6.59**	9.45	8.97	5.08
PIN-LO	29.07	24.74	23.05	22.07	12.94	23.42	45.81	54.41	44.41	31.10
PIN-SLAM	**2.25**	2.86	**2.15**	5.25	**2.83**	2.65	8.70	**7.94**	**6.00**	**4.51**

All sequences are with loops. The best result is in bold and the second best is underscored. ‡ indicates the method conducts offline PGO.

TABLE VI
SLAM PERFORMANCE COMPARISON (ATE RMSE [M]) OF THE PROPOSED METHOD VERSUS THE STATE-OF-THE-ART LIDAR ODOMETRY (ABOVE) AND LIDAR SLAM METHODS (BELOW) ON OUR SELF-COLLECTED *IPB-CAR* LIDAR DATASET

Method	2020-0*	2020-1*	2023-0	2023-1*	**Avg.**
F-LOAM [82]	11.75	39.92	92.48	52.70	49.21
KISS-ICP [78]	6.13	15.66	93.70	22.91	34.60
SuMa [4]	7.15	117.91	100.12	57.66	70.71
MULLS [52]	10.79	47.07	**78.10**	68.63	51.15
PIN-LO	5.70	17.42	87.59	21.44	33.04
PIN-SLAM	**3.51**	**6.12**	87.59	**19.27**	**29.12**

* means the sequence is with loops. The best result is in bold and the second best is underscored.

TABLE VII
SLAM PERFORMANCE COMPARISON (ATE RMSE [M]) OF OUR METHOD VERSUS THE STATE-OF-THE-ART LIDAR ODOMETRY (ABOVE) AND LIDAR SLAM METHODS (BELOW) ON *NEWER COLLEGE* HANDHELD LIDAR DATASET

Method	01	02	quad_e	math_e	ug_e	cloister	stairs	**Avg.**
F-LOAM [82]	6.74	✗	0.40	0.26	0.09	7.69	✗	3.04
KISS-ICP [78]	0.62	1.88	0.10	**0.07**	0.33	0.30	✗	0.55
SuMa [4]	2.03	3.65	0.28	0.16	0.09	0.20	1.85	1.18
MULLS [52]	2.51	8.39	0.12	0.35	0.86	0.41	✗	2.11
MD-SLAM [13]	-	1.74	0.25	-	-	0.36	0.34	-
SC-LeGO-LOAM [27], [68]	-	1.30	**0.09**	-	-	0.20	3.20	-
PIN-LO	2.08	5.32	**0.09**	0.09	**0.07**	0.19	0.07	1.13
PIN-SLAM	**0.43**	**0.31**	**0.09**	0.09	**0.07**	**0.15**	**0.06**	**0.17**

All sequences are with loops. The best result is bold and the second best is underscored. ✗ denotes failure. - indicates the number is not reported and unavailable using the open-source code.

TABLE VIII
SLAM PERFORMANCE COMPARISON (ATE RMSE [M]) OF THE PROPOSED METHOD VERSUS THE STATE-OF-THE-ART METHODS ON *HILTI-21* LIDAR DATASET

Method	rpg	lab	base1	base4	cons2	camp2	**Avg.**
F-LOAM [82]	2.78	0.18	0.91	0.29	11.52	8.95	4.10
KISS-ICP [78]	0.22	0.07	0.32	0.11	0.84	1.98	0.58
HDLGraph-SLAM [30]	0.35	0.05	**0.28**	0.37	0.74	0.35	0.36
PIN-SLAM	**0.21**	**0.04**	0.30	**0.08**	**0.41**	**0.11**	**0.19**

Best result is bold and the second best is underscored.

odometry and build a globally consistent map without duplicated structures.

Next, we test PIN-SLAM on longer and more challenging driving scenes, i.e., nine sequences of the MulRan dataset and four sequences of our self-recorded IPB-Car dataset. As shown in Tables V and VI, compared with the state-of-the-art LiDAR odometry, SLAM or offline optimized approaches, PIN-SLAM is able to correct the drift of PIN LiDAR odometry and achieves the best overall localization accuracy.

To further test our method on other motion profiles, we run experiments on Newer College and Hilti-21 datasets, which are mainly collected by a handheld LiDAR in both indoor and outdoor environments. We evaluate PIN-SLAM against existing LiDAR odometry/SLAM systems supporting handheld LiDARs. As given in Tables VII and VIII, PIN-SLAM demonstrates superior performance, yielding the smallest localization error on both datasets and surpassing the compared approaches by a large margin. For the Newer College dataset, half of the compared methods fail on the challenging stairs sequence while

TABLE IX
SLAM PERFORMANCE COMPARISON (ATE RMSE [CM]) OF THE PROPOSED METHOD VS. THE STATE-OF-THE-ART METHODS ON *REPLICA* RGB-D DATASET

Method	r0	r1	r2	o0	o1	o2	o3	o4	**Avg.**
iMAP [71]	3.12	2.54	2.31	1.69	1.03	3.99	4.05	1.93	2.58
NICE-SLAM [101]	1.69	2.04	1.55	0.99	0.90	1.39	3.97	3.08	1.95
Vox-Fusion [92]	0.40	0.54	0.54	0.50	0.46	0.75	0.50	0.60	0.54
Co-SLAM [83]	0.60	1.13	1.43	0.55	0.50	0.46	1.40	0.77	0.86
ESLAM [24]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63	0.63
Point-SLAM [65]	0.61	0.41	0.37	0.38	0.48	0.54	0.72	0.63	0.52
PIN-SLAM	**0.27**	**0.31**	**0.13**	**0.22**	**0.30**	**0.28**	**0.16**	**0.28**	**0.24**
PIN-SLAM w/o BA	0.41	0.36	0.25	0.36	0.33	0.34	0.38	0.73	0.40
PIN-SLAM w/o color	0.38	0.40	0.16	0.39	0.27	0.30	0.52	0.40	0.35

Best result is bold and the second best is underscored.

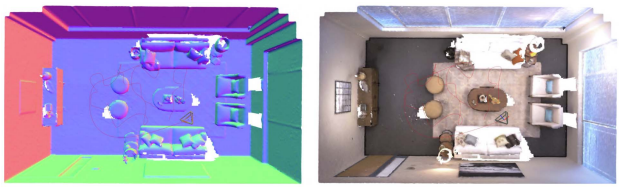


Fig. 6. Qualitative results of PIN-SLAM on Room0 sequence of *Replica* RGB-D dataset. We show the mesh colorized in the normal direction and the mesh colorized by RGB built by PIN-SLAM. The estimated trajectory is shown in red.

our approach achieves the smallest RMSE of 6 cm. Note that the sequences in Hilti-21 dataset are relatively short and the LiDAR is mainly moving in a confined area without explicit loops. Therefore, we do not distinguish an odometry and a SLAM method on Hilti-21.

3) RGB-D SLAM Evaluation: We also conducted experiments to show that PIN-SLAM can also work well taking as input the RGB-D images. We compare the localization (camera pose tracking) accuracy of PIN-SLAM against six state-of-the-art neural implicit RGB-D SLAM approaches on all eight sequences of the popular Replica dataset. Since the camera is moving in a single room that is always covered by the local map, we disable the loop closure detection and correction. As given in Table IX, PIN-SLAM achieves the best camera tracking accuracy on average with an RMSE of only 2.4 mm. The superior performance demonstrates that, with precise depth measurements, our point-to-SDF registration achieves greater accuracy compared to the rendering-based camera tracking utilized in most of the neural implicit SLAM approaches under comparison. As an ablation study, we show that the localization RMSE of PIN-SLAM decreases by 40% and 31% by conducting local BA and using the color information during camera tracking. Moreover, as shown in Fig. 6, PIN-SLAM can reconstruct high-fidelity colorized mesh from the neural point map built by the RGB-D SLAM.

4) Additional Challenging Scenarios: We show qualitative results on various challenging scenarios, such as a cave tunnel from Nebula dataset [60], as shown in Fig. 7. PIN-SLAM manages to build a globally consistent map using the data collected by a quadruped robot moving in the cave. PIN-SLAM is also robust to highly dynamic scenes in ETH DOALS dataset [66] as shown in Fig. 8. PIN-SLAM manages to filter the moving pedestrians and reconstruct a static mesh.

C. Evaluation of the Loop Closure Detection Performance

We follow BEVPlace [41] to evaluate loop closure detection performance on the KITTI dataset. We use the same partition

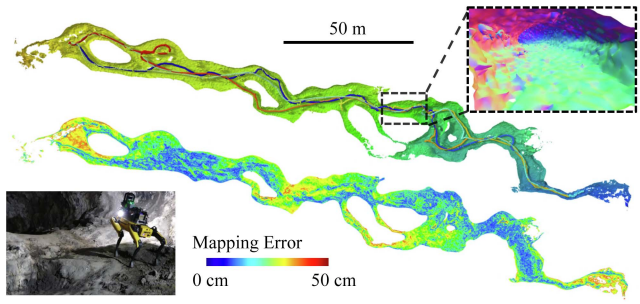


Fig. 7. Qualitative results on the *Nebula* dataset collected by a Spot1 robot with a 32-beam LiDAR moving back-and-forth in the Valentine Cave. We show the estimated trajectory and mesh built by PIN-SLAM on the top. We show the mapping error compared to the survey-grade map measured by a TLS on the bottom.

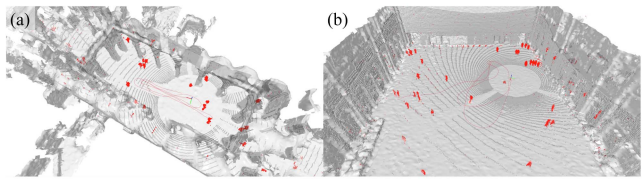


Fig. 8. Qualitative results on the hauptgebaeude and station sequences of *ETH DOALS* dataset. We show the LiDAR scan overlaid on the static mesh built by PIN-SLAM. The dynamic points (in red) are filtered from a LiDAR scan (in gray) online using the PIN map.

TABLE X
LOOP CLOSURE DETECTION RECALL AT TOP-1 OF THE PROPOSED METHOD VERSUS THE STATE-OF-THE-ART NON-LEARNING (ABOVE) AND LEARNING-BASED (BELOW) METHODS ON FOUR SEQUENCES FROM *KITTI* DATASET

Method	00	02	05	06	**Avg.**
Scan Context [27]	89.7	73.9	77.0	86.7	81.8
BVMatch [40]	93.8	78.2	90.2	93.8	89.0
PointNetVLAD [75]	91.6	62.3	76.9	77.8	77.2
OverlapTransformer [42]	96.7	80.1	91.9	95.6	91.1
BEVPlace [41]	**99.7**	**98.1**	99.3	**100.0**	**99.3**
Ours (local map context)	96.0	73.0	98.3	**100.0**	91.8
Ours (full)	99.4	78.4	**100.0**	**100.0**	94.5

The results of the compared methods are reported by BEVPlace [41]. The best result is in bold and the second best is underscored.

of database and query frames as used in BEVPlace. We show the loop detection recall at Top-1 in Table X and compare our approach with several nonlearning and learning-based methods. Even without any offline pretraining, our approach achieves the second-best average recall after BEVPlace, outperforming the other compared methods. We show that using the polar context descriptor of neural points in the local map can improve the loop detection recall over the original scan context [27]. Moreover, leveraging online optimized neural point features can enhance the distinctiveness of the context descriptor. Though our approach does not achieve the best performance in LiDAR place recognition, it proves to be adequate for a highly accurate SLAM system, without the need for pre-training or external map representations.

TABLE XI
3-D RECONSTRUCTION QUALITY OF DIFFERENT METHODS ON QUAD AND MATH INSTITUTE SEQUENCE OF THE *NEWER COLLEGE* DATASET [57]

Method	Pose	Quad				Math Institute			
		Map. Acc. ↓	Map. Comp. ↓	C-l1. ↓	F-score ↑	Map. Acc. ↓	Map. Comp. ↓	C-l1. ↓	F-score ↑
VDB-Fusion [77]	KISS-ICP [78]	14.03	25.46	19.75	69.50	15.21	28.66	21.94	63.35
SHINE [100]		14.87	20.02	17.45	68.85	14.46	34.03	24.24	64.38
NKSR [20]		15.67	36.87	26.27	58.57	15.11	27.10	21.11	65.08
Puma [76]	Own Odometry	15.30	71.91	43.60	57.27	15.81	46.00	30.91	54.95
SLAMesh [62]		19.21	48.83	34.02	45.24	**12.80**	23.50	18.16	75.17
Nerf-LOAM [11]		12.89	22.21	17.55	74.37	-	-	-	-
PIN-SLAM		**11.55**	**15.25**	**13.40**	**82.08**	13.70	**21.91**	**17.80**	**75.49**

The reference model is measured by terrestrial laser scanning with mm-level accuracy. We report completion, accuracy, and chamfer-L1 in cm as well as F-Score in % calculated with a 20 cm error threshold. We highlight the best results in bold and the second best in underscored. - indicates the number is not reported and unavailable using the open-source code.

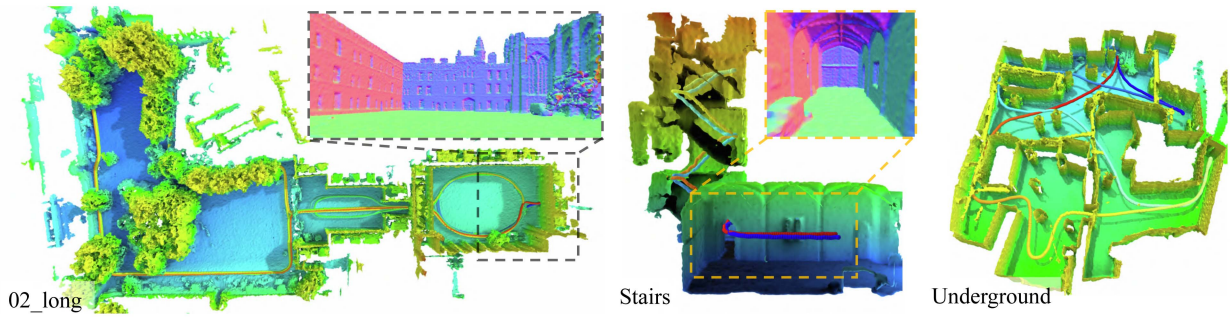


Fig. 9. Globally consistent mesh reconstructed by PIN-SLAM on *Newer College* dataset with multiple loops. The estimated trajectories are overlaid on the map and colorized according to the timestamp. Details of several revisited regions are highlighted in the dashed boxes.

D. Evaluation of the Mapping Performance

1) 3-D Reconstruction Quality: In this section, we show the mapping quality of PIN-SLAM in terms of the 3-D reconstruction quality of the resulting mesh in Table XI. The mesh is reconstructed from the SDF queried at the fixed-size grid using the marching cubes algorithm [39]. For the quantitative evaluation, we use the commonly used 3-D reconstruction metrics [43] calculated between the reconstructed and reference 3-D model, namely, accuracy, completeness, Chamfer-L1 distance, and F-score. We select two scenes from the Newer College dataset [57], namely, Quad from the 02_long sequence and Math Institute from the math_easy sequence. Both scenes have the survey-grade point cloud map measured by TLS available, which is taken as the reference model for evaluation. We compare PIN-SLAM against three state-of-the-art LiDAR SLAM approaches that can reconstruct dense 3-D meshes, namely, Puma [76] based on Poisson surface reconstruction, SLAMesh [62] utilizing Gaussian process reconstruction, and Nerf-LOAM [11] employing implicit neural representation like us. We additionally include three state-of-the-art "mapping with known poses" methods tailored for LiDAR data, namely, VDB-Fusion [77] based on TSDF integration, SHINE-Mapping [100] based on implicit neural scene reconstruction, and the data-driven method NKSR [20] based on neural kernel field. We use the pose estimation of KISS-ICP [78] as the pose input to these three mapping methods. Note that KISS-ICP and PIN-SLAM achieve similar localization RMSE of about 10 cm in both scenes. We use the same voxel size of 20 cm for mesh reconstruction for the compared methods. Table XI lists the obtained results. As can be seen, PIN-SLAM achieves the best mapping quality in terms of completeness, Chamfer distance and F-score in both scenes, indicating that PIN-SLAM can achieve more accurate and more complete reconstruction of the environment than the compared methods. The superior performance of PIN-SLAM owes to the continuous implicit neural representation and the more consistent pose estimation.

2) Map Consistency: Since the global localization accuracy can reflect the mapping consistency, we qualitatively depict the globally consistent mesh reconstructed from PIN map built by PIN-SLAM on Newer College in Fig. 9, KITTI in Fig. 10, MulRAN in Fig. 11, and IPB-Car dataset in Fig. 12. Several regions with multiple loop closures are highlighted.

E. Ablation Studies

In this section, we conduct ablation studies to validate our design choices. We use the average relative translational drifting error (ARTE) on KITTI dataset as the metrics and the same random seed to carry out the studies. We show ARTE obtained on KITTI sequences 00–10 with different design choices for mapping and odometry in Tables XII and XIII, respectively. For the mapping losses, we find out that the BCE loss used by our method is more suitable for SDF training than L1 or L2 loss. Besides, Ekional loss is necessary for the odometry and a weight λ_e of 0.5 is preferable. To calculate the Ekional loss, using the numerical gradient rather than the analytical gradient leads to a smooth and regular SDF in the free space, thus improving localization accuracy. A perturbation step ϵ of 8 cm for numerical gradient calculation is preferable. For odometry estimation, we verify that the usage of GM robust kernel for SDF residual

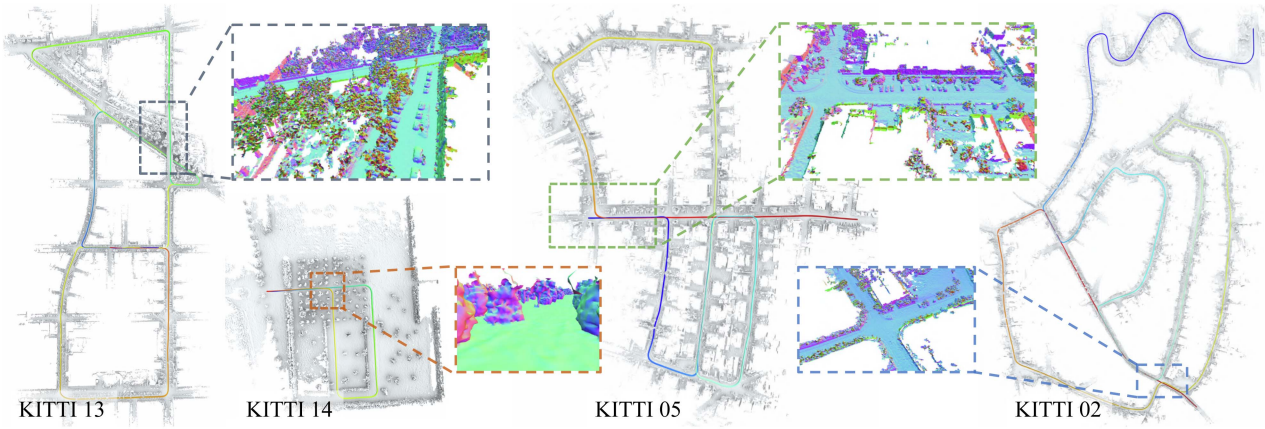


Fig. 10. Globally consistent mesh reconstructed by PIN-SLAM on *KITTI* dataset with multiple loops. The estimated trajectories are overlaid on the map and colorized according to the timestamp. Details of several revisited regions are highlighted in the dashed boxes.

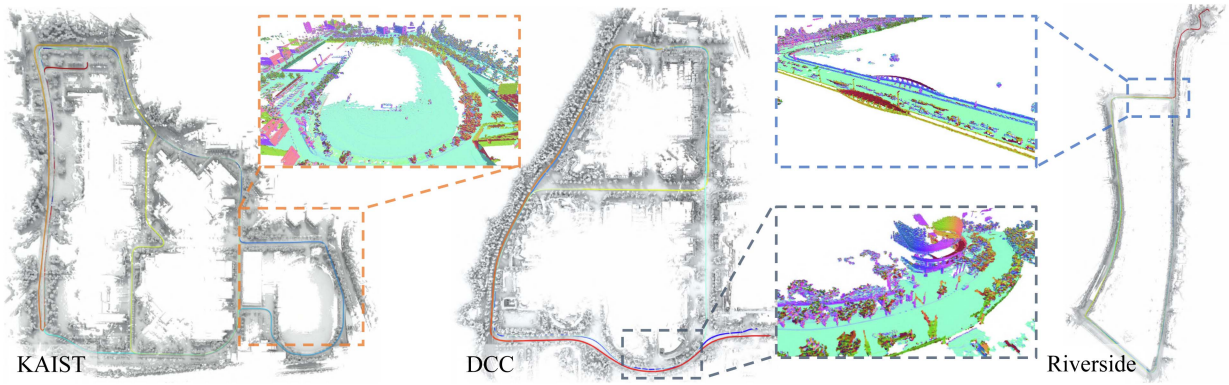


Fig. 11. Globally consistent mesh reconstructed by PIN-SLAM on *MulRAN* dataset with multiple loops. The estimated trajectories are overlaid on the map and colorized according to the timestamp. Details of several revisited regions are highlighted in the dashed boxes.

TABLE XII
ABLATION STUDY: DESIGN CHOICES OF MAP TRAINING LOSSES

SDF loss	Ekional loss λ_e	num. grad.	ϵ [m]	ARTE [%] ↓
BCE	0.5		-	0.57
BCE	1.0	✓	0.08	0.59
BCE	0.2	✓	0.08	0.55
BCE	0.0	-	-	✗
L1	0.5	✓	0.08	1.15
L2	0.5	✓	0.08	✗
BCE	0.5	✓	0.04	0.54
BCE	0.5	✓	0.16	1.48
BCE	0.5	✓	0.08	**0.50**

ARTE represents the average relative translational drifting errors on KITTI sequence 00–10. ✗ denotes failure.
The best results are highlighted in bold.

TABLE XIII
ABLATION STUDY: DESIGN CHOICES OF ODOMETRY ESTIMATION

GM kernel w_r	GM kernel w_g	anl. grad.	ARTE [%] ↓
✗	✗	✓	0.56
✓	✗	✓	0.52
✗	✓	✓	0.55
✓	✓	✗	✗
✓	✓	✓	**0.50**

ARTE represents the average relative translational drifting errors on KITTI sequence 00–10. ✗ denotes failure.
The best results are highlighted in bold.

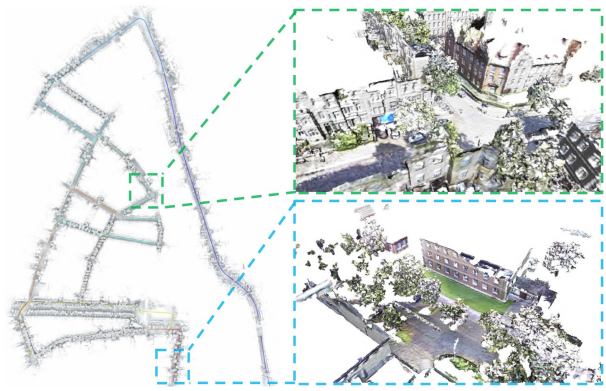


Fig. 12. Qualitative results on the *IPB-Car* dataset. On the left, we show the globally consistent mesh reconstructed by PIN-SLAM. On the right, we show two examples of the colorized mesh reconstructed using our calibrated LiDAR-camera multisensor platform.

and SDF gradient anomaly are both beneficial to localization accuracy. The odometry loses track when using the numerical gradient because it is often less accurate than the analytical one once the SDF has been fitted. Besides, as shown in Table XIV, we find out that a too-large or too-small neural point resolution

TABLE XIV
ABLATION STUDY: AVERAGE RELATIVE TRANSLATIONAL DRIFTING ERROR [%] WITH REGARDS TO DIFFERENT NEURAL POINT RESOLUTIONS v_p

Sequence	Neural point resolution v_p				
	1.0 m	0.8 m	0.6 m	0.4 m	0.2 m
KITTI 00	0.98	0.75	0.85	**0.55**	1.06
KITTI 05	0.36	0.56	0.45	**0.29**	0.50
KITTI 08	0.96	0.88	0.85	**0.83**	1.05

The best results are highlighted in bold.

TABLE XV
MEMORY CONSUMPTION IN MB AND THE COMPRESSION RATIO WITH REGARDS TO THE RAW POINT CLOUD OF DIFFERENT MAP REPRESENTATIONS

Representation	KITTI 00	KITTI 05	KITTI 08	NCD 02
Raw point cloud	13624.2	8284.7	12214.1	26559.0
Surfel map	887.7	512.6	835.7	79.0
Mesh map	2032.9	1317.4	1894.1	1503.7
VDB TSDF map	748.1	434.6	958.6	462.5
SHINE map	160.6	114.2	189.9	117.1
PIN-LO map	137.3	89.6	162.4	110.3
PIN-SLAM map	**102.1 (0.7%)**	**66.3 (0.8%)**	**138.8 (1.1%)**	**76.8 (0.3%)**

The best results are highlighted in bold.

v_p would lead to suboptimal odometry accuracy. Setting v_p as 0.4 m, corresponding to our adaptive setting $0.005r_{\max}$, leads to the smallest ARTE on all three tested sequences.

F. Memory and Computational Resources

1) Memory Requirement: We report the map memory consumption of PIN-SLAM on four representative sequences from KITTI and Newer College dataset in Table XV. We compare the proposed PIN map with several common map representations: surfel map used by SuMa [4], mesh map used by Puma [76], VDB TSDF map used by VDB Fusion [77], and the grid-based sparse hierarchical implicit neural (SHINE) map used by SHINE-Mapping [100] and Nerf-LOAM [11]. We also provide the raw point cloud memory consumption as a reference. For the VDB TSDF map, we use a voxel size of 20 cm. For SHINE map and PIN map, we use the same resolution of 40 cm for the local latent feature grid and neural points. With the continuous neural SDF queried from SHINE or PIN map, we can reconstruct the mesh with a voxel size of 20 cm with a similar quality as the mesh reconstructed from the discrete VDB TSDF map. Besides, for VDB TSDF map and SHINE map, we take the odometry estimation of KISS-ICP [78] as their pose. As shown in Table XV, our PIN map is the most compact representation among the compared methods with a compression ratio of about 0.7% on average. Compared to SHINE map, PIN map using the same resolution and feature dimensions is about 15% more memory efficient even with the additional storage of neural point coordinates and orientations. This is because PIN map does not have a hierarchical tree structure like SHINE map and the neural points are only allocated close to the surface. Notably, compared with PIN LiDAR odometry, the memory consumption of PIN map decreases by about 20% due to the loop correction and map update of PIN-SLAM. The elastic PIN map can deform with the corrected pose and eliminate duplicated neural points representing the same place. This is not possible for the grid-based VDB or SHINE map without the submap scheme or offline remapping.

2) Computational Requirement: In Table XVI, we report the average runtime of PIN-SLAM running on a single NVIDIA Quadro A4000 GPU over the whole 11 sequences of the KITTI dataset. We show the results of two versions of PIN-SLAM, the full version as used in the localization evaluation above and the light version which takes a fewer number of iterations during mapping and odometry. The light version of PIN-SLAM works 38% faster at the cost of a minor degradation (10%) of the localization accuracy. It can operate efficiently at approximately 11 Hz, aligning with the typical sensor frame rate. In Fig. 13, we report the average processing time for each submodule of PIN-SLAM (light). Odometry and map optimization are the two most time-consuming parts, each taking up about 40% of the total run time. When loops are detected and PGO is conducted, the run time per frame occasionally exceeds 200 ms (as the spikes in the figure). Notably, the processing time remains consistent as the frame count increases. In contrast, the only implicit neural LiDAR odometry baseline, Nerf-LOAM [11] takes more than 4 s per frame, which is 30× slower than PIN-SLAM. Moreover, we observe that Nerf-LOAM's processing time increases drastically with an increasing number of scans.

TABLE XVI
COMPARISON OF THE AVERAGE OPERATION SPEED AND THE LOCALIZATION ERROR ON KITTI SEQUENCE 00–10 USING A SINGLE NVIDIA A4000 GPU

Method	Time per frame [s] ↓	FPS [Hz] ↑	ARTE [%] ↓
PIN-SLAM (full)	0.14	7.1	**0.50**
PIN-SLAM (light)	**0.09**	**11.3**	0.56
Nerf-LOAM [11]	4.43	0.2	1.69

ARTE represents the average relative translational error.
The best results are highlighted in bold.

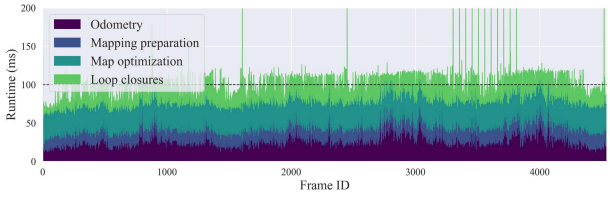


Fig. 13. Processing time for each submodule of the light version of PIN-SLAM on KITTI sequence 00.

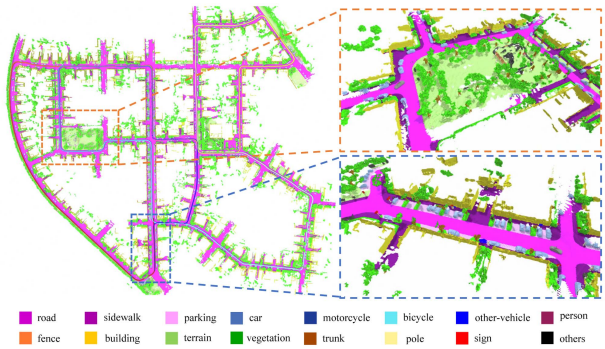


Fig. 14. Example of the metric-semantic map built by PIN-SLAM on KITTI sequence 00 using the semantic labels of SemanticKITTI.

G. Extension on Semantic Mapping

As a proof of concept, we illustrate briefly that our approach is capable of conducting metric-semantic SLAM with global consistency. As shown in Fig. 14, PIN-SLAM manages to build a metric-semantic map using the SemanticKITTI [3] dataset

by integrating the semantic labels as described in Section IV. Note that we only need to add another shallow MLP for the semantic querying. Currently, the semantics do not play a role in odometry estimation and loop closure detection. Considering this is outside the scope of this work but a possible future extension.

VI. Conclusion

In this article, we presented PIN-SLAM, a novel LiDAR SLAM approach to perform globally consistent mapping using a PIN map representation. Our approach alternates between the online incremental learning of the local implicit map and the odometry estimation using a correspondence-free point-to-implicit map registration. We exploit sparse neural points as local feature embeddings, which are inherently elastic and deformable throughout the global pose adjustment when correcting a loop closure. This enables us to effectively maintain the global consistency of both the neural points and the underlying implicit map. We implemented and evaluated our approach on various datasets, provided comparisons to other existing techniques and supported all claims made in this article. Extensive experiments suggest PIN-SLAM achieves better or on-par localization accuracy than previous methods. In addition, it builds a more consistent and compact implicit map, which can be reconstructed into more accurate and complete meshes. Besides, PIN-SLAM can run at the sensor frame rate using a moderate GPU.

Limitations and future work: Future work may use additionally an IMU for a more robust, accurate and efficient SLAM system. Besides, we are currently using a fixed resolution of neural points. We can enhance the reconstruction quality by adaptively distributing and dynamically moving neural points throughout the scene like previous works [34], [65], and use a data structure, such as i-Kdtree [90], enabling efficient neighbor search without using voxel structures. Another direction can be using semantic information to further enhance odometry estimation and loop closure detection.

Acknowledgment

The authors would like to thanks Ignacio Vizzo, Benedikt Mersch, Yibin Wu, Tiziano Guadagnino, and Haofei Kuang for the fruitful discussions and Rodrigo Marcuzzi for the colorized point cloud.

References

[1] J. Abou-Chakra, F. Dayoub, and N. Sünderhauf, "ParticleNeRF: A particle-based encoding for online neural radiance fields," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2024, pp. 5963–5972.

[2] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural RGB-D surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6280–6291.

[3] J. Behley et al., "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Intl. Conf. Comput. Vis.*, 2019, pp. 9296–9306.

[4] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Proc. Robotics: Sci. Syst.*, 2018, pp. 59–68.

[5] P. Besl and N. McKay, "A method for registration of 3D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[6] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "TensoRF: Tensorial radiance fields," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 333–350.

[7] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMa : Efficient LiDAR-based semantic SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4530–4537.

[8] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time globally consistent 3D reconstruction using online surface re-integration," *ACM Trans. Graph.*, vol. 36, no. 3, pp. 1–18, 2017.

[9] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-RIM-CP&R-2012-002, 2012.

[10] P. Dellenbach, J. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP real-time elastic LiDAR odometry with loop closure," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 5580–5586.

[11] J. Deng et al., "NeRF-LOAM: Neural implicit representation for large-scale incremental lidar odometry and mapping," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 8184–8193.

[12] J. Deschaud, "IMLS-SLAM: Scan-to-model matching based on 3D data," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2480–2485.

[13] L. Di Giammarino, L. Brizi, T. Guadagnino, C. Stachniss, and G. Grisetti, "MD-SLAM: Multi-cue direct slam," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 11047–1105.

[14] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[15] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[16] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *Proc. Int. Conf. Mach. Learn.*, 2020, Art. no. 355.

[17] M. Helmberger, K. Morin, B. Berner, N. Kumar, G. Cioffi, and D. Scaramuzza, "The Hilti SLAM challenge dataset," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7518–7525, Jul. 2022.

[18] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1271–1278.

[19] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[20] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams, "Neural kernel surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 4369–4379.

[21] J. Huang, S. S. Huang, H. Song, and S. M. Hu, "Di-fusion: Online implicit 3D reconstruction with deep priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8928–8937.

[22] S. Huang et al., "Neural LiDAR fields for novel view synthesis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 18236–18246.

[23] S. Isaacson, P. C. Kung, M. Ramanagopal, R. Vasudevan, and K. A. Skinner, "LONER: LiDAR only neural representations for real-time slam," *IEEE Robot. Autom. Lett.*, vol. 8, no. 12, pp. 8042–8049, Dec. 2023.

[24] M. M. Johari, C. Carta, and F. Fleuret, "ESLAM: Efficient dense slam system based on hybrid representation of signed distance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17408–17419.

[25] G. Kim, Y. Park, Y. Cho, J. Jeong, and A. Kim, "MulRan: Multimodal range dataset for urban place recognition," in *Proc. IEEE Intl. Conf. Robot. Autom.*, 2020, pp. 6246–6253.

[26] G. Kim, S. Choi, and A. Kim, "Scan context : Structural place recognition robust to rotation and lateral variations in urban environments," *IEEE Trans. Robot.*, vol. 38, no. 3, pp. 1856–1874, Jun. 2022.

[27] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4802–4809.

[28] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–13.

[29] S. Kohlbrecher, S. O. Von, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *Proc. IEEE Int. Symp. Saf. Secur. Rescue Robot.*, 2011, pp. 155–160.

[30] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LiDAR-based system for long-term and wide-area people behavior measurement," *Intl. J. Adv. Robotic Syst.*, vol. 16, no. 2, pp. 1–20, 2019.

[31] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 11054–11059.

[32] H. Kuang, X. Chen, T. Guadagnino, N. Zimmerman, J. Behley, and C. Stachniss, "IR-MCL: Implicit representation-based online global localization," *IEEE Robot. Autom. Lett.*, vol. 8, no. 3, pp. 1627–1634, Mar. 2023.

[33] Q. Li et al., "LO-Net: Deep real-time LiDAR odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8473–8482.

[34] T. Li, X. Wen, Y. S. Liu, H. Su, and Z. Han, "Learning deep implicit functions for 3D shapes with dynamic code clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12840–12850.

[35] Z. Li et al., "Neuralangelo: High-fidelity neural surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 8456–8465.

[36] J. Lin, F. Zhang, and A. Loam_livox, "Robust LiDAR odemetry and mapping LOAM package for livox LiDAR," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 3126–3131.

[37] X. Liu, Z. Liu, F. Kong, and F. Zhang, "Large-scale LiDAR consistent mapping using hierarchical LiDAR bundle adjustment," *IEEE Robot. Autom. Lett.*, vol. 8, no. 3, pp. 1523–1530, Mar. 2023.

[38] Z. Liu, X. Liu, and F. Zhang, "Efficient and consistent bundle adjustment on LiFAR point clouds," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4366–4386, Dec. 2023.

[39] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proc. Int. Conf. Comput. Graph. Interactive Techn.*, 1987, pp. 163–169.

[40] L. Luo, S. Y. Cao, B. Han, H. L. Shen, and J. Li, "BVMatch: LiDAR-based place recognition using birds-eye view images," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 6076–6083, Jul. 2021.

[41] L. Luo et al., "BEVPlace: Learning LiDAR-based place recognition using bird's eye view images," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 8666–8675.

[42] J. Ma, J. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen, "OverlapTransformer: An efficient and yaw-angle-invariant transformer network for LiDAR-based place recognition," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6958–6965, Jul. 2022.

[43] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4460–4470.

[44] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Europ. Conf. Comput. Vis.*, 2020, pp. 405–421.

[45] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet : Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4213–4220.

[46] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A. factored solution to the simultaneous localization and mapping problem," in *Proc. Conf. Advancements Artif. Intell.*, 2002, pp. 593–598.

[47] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, 2022.

[48] R. A. Newcombe et al., "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. Int. Symp. Mixed Augmented Reality*, 2011, pp. 127–136.

[49] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, 2013.

[50] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwar, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.

[51] J. Ortiz et al., "iSDF: Real-time neural signed distance fields for robot perception," in *Proc. Robotics: Sci. Syst.*, 2022, pp. 30–39.

[52] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 11633–1164.

[53] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.

[54] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[55] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 523–540.

[56] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "LINS: A LiDAR-inertial state estimator for robust and efficient navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 8899–8906.

[57] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The newer college dataset: Handheld LiDAR, inertial and vision with ground truth," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 4353–4360.

[58] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1717–1730, 2016.

[59] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robot. Autom. Lett.*, vol. 5, no. 1, pp. 227–234, Jan. 2020.

[60] A. Reinke et al., "LOCUS 2.0: Robust and computationally efficient LiDAR odometry for real-time 3D mapping," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9043–9050, Oct. 2022.

[61] R. A. Rosu and S. Behnke, "PermutoSDF: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 8466–8475.

[62] J. Ruan, B. Li, Y. Wang, and Y. Sun, "Slamesh: Real-time LiDAR simultaneous localization and meshing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 3546–3552.

[63] J. Ruan, B. Li, Y. Wang, and Z. Fang, "GP-SLAM : Real-time 3D LiDAR SLAM based on improved regionalized Gaussian process map reconstruction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5171–5178.

[64] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. Int. Conf. 3-D Digit. Imag. Model.*, 2001, pp. 145–152.

[65] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, "Point-SLAM: Dense neural point cloud-based SLAM," in *Proc. IEEE/CVF Intl. Conf. Comput. Vis.*, 2023, pp. 18433–18444.

[66] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart, "Dynablox: Real-time detection of diverse dynamic objects in complex environments," *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6259–6266, Oct. 2023.

[67] T. Shan et al., "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

[68] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[69] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Proc. Robotics: Sci. Syst.*, 2005, pp. 65–72.

[70] J. Straub et al., "The replica dataset: A digital replica of indoor spaces," 2019, *arXiv:1906.05797*.

[71] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proc. IEEE/CVF Intl. Conf. Comput. Vis.*, 2021, pp. 6229–6238.

[72] T. Takikawa et al., "Neural geometric level of detail: Real-time rendering with implicit 3D shapes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11358–11367.

[73] 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 Trans. Graph.*, vol. 42, no. 6, 2023, Art. no. 246.

[74] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artif. Intell.*, vol. 128, no. 1/2, pp. 99–141, 2001.

[75] A. Uy and G. Lee, "PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4470–4479.

[76] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson surface reconstruction for LiDAR odometry and mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 5624–5630.

[77] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "VDBFusion: Flexible and efficient TSDF integration of range sensor data," *Sensors*, vol. 22, no. 3, pp 1296, 2022.

[78] 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 Robot. Autom. Lett.*, vol. 8, no. 2, pp. 1029–1036, Feb. 2023.

[79] C. Wang et al., "PyPose: A library for robot learning with physics-based optimization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 22024–22034.

[80] G. Wang, X. Wu, S. Jiang, Z. Liu, and H. Wang, "Efficient 3D deep LiDAR odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5749–5765, May 2023.

[81] G. Wang, X. Wu, Z. Liu, and H. Wang, "PWCLO-Net: Deep LiDAR odometry in 3D point clouds using hierarchical embedding mask optimization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15910–15919.

[82] H. Wang, C. Wang, C. Chen, and L. Xie, "F-LOAM: Fast LiDAR odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4390–4396.

[83] H. Wang, J. Wang, and L. Agapito, "Co-SLAM: Joint coordinate and sparse parametric encodings for neural real-time slam," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 13293–13302.

[84] Y. Wang et al., "Elastic and efficient LiDAR reconstruction for large-scale exploration tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 5035–5041.

[85] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Proc. Robotics: Sci. Syst.*, 2015, pp. 3–11.

[86] L. Wiesmann et al., "LocNDF: Neural distance field mapping for robot localization," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 4999–5006, Aug. 2023.

[87] L. Wu, K. M. B. Lee, C. Le Gentil, and T. Vidal-Calleja, "Log-GPIS-MOP: A unified representation for mapping, odometry, and planning," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 4078–4094, Oct. 2023.

[88] Y. Wu, T. Guadagnino, L. Wiesmann, L. Klingbeil, C. Stachniss, and H. Kuhlmann, "LIO-EKF: High frequency LiDAR-inertial odometry using extended kalman filters," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 13741–13747.

[89] Q. Xu et al., "Point-neRF: Point-based neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5428–5438.

[90] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

[91] Z. Yan, H. Yang, and H. Zha, "Active neural mapping," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 10947–10958.

[92] 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 *Proc. Int. Symp. Mixed Augmented Reality*, 2022.

[93] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, "LiTAMIN2: Ultra light LiDAR-based SLAM using geometric approximation applied with kl-divergence," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 11619–11625.

[94] X. Yu et al., "Nf-atlas: Multi-volume neural feature fields for large scale LiDAR mapping," *IEEE Robot. Autom. Lett.*, vol. 8, no. 9, pp. 5870–5877, Sep. 2023.

[95] Y. Yuan and A. Nüchter, "An algorithm for the SE(3)-transformation on neural implicit maps for remapping functions," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7763–7770, Jul. 2022.

[96] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 809–816.

[97] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robotics: Sci. Syst.*, 2014, pp. 1–9.

[98] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 7244–7251.

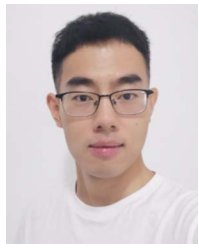
[99] S. Zhi, T. Laidlow, S. Leutenegger, and A. Davison, "In-place scene labelling and understanding with implicit scene representation," in *Proc. IEEE/CVF Intl. Conf. Comput. Vis.*, 2021, pp. 15818–15827.

[100] X. Zhong, Y. Pan, J. Behley, and C. Stachniss, "SHINE-Mapping: Large-scale 3D mapping using sparse hierarchical implicit neural representations," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 8371–8377.

[101] Z. Zhu et al., "NICE-SLAM: Neural implicit scalable encoding for SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12776–12786.

Yue Pan received the B.Sc. degree in geomatics engineering from Wuhan University, Wuhan, China, in 2019, and the M.Sc. degree in geomatics engineering from ETH Zurich, Zurich, Switzerland, in 2022. He is currently working toward the Ph.D. degree in engineering with the Photogrammetry & Robotics Lab headed by Prof. Cyrill Stachniss, University of Bonn, Bonn, Germany.

His research interets include SLAM, 3-D reconstruction, and navigation.



Xingguang Zhong received the B.Sc. degree in mechanical engineering and the M.Sc. degree in mechatronic engineering from the Harbin Institute of Technology, Harbin, China, in 2017 and 2019, respectively. He is currently working toward the Ph.D. degree in engineering with the Photogrammetry & Robotics Lab, University of Bonn, Bonn, Germany, headed by Prof. Cyrill Stachniss.

His research interests include large-scale 3-D reconstruction and autonomous navigation.

Louis Wiesmann (Graduate Student Member, IEEE) received the B.Sc. degree and M.Sc. degree in geodetic engineering in 2017 and 2019, respectively, from the University of Bonn, Bonn, Germany, where he is currently working toward the Ph.D. degree with the Photogrammetry & Robotics Lab headed by Prof. Cyrill Stachniss

His research focuses on localization and mapping in large-scale environments.

Thorbjörn Posewsky received the B.Sc. and M.Sc. degrees in computer science from the University of Paderborn, Paderborn, Germany, in 2012 and 2015, respectively. He is currently working toward the Ph.D. degree in engineering with the Photogrammetry & Robotics Lab headed by Prof. Cyrill Stachniss, University of Bonn, Bonn, Germany.

He is currently a Software Engineer with MicroVision GmbH, Hamburg, Germany. His research focuses on Mapping and SLAM.

Jens Behley (Member, IEEE) received the Dipl.-Inform. in computer science and the Ph.D. in computer science, both from the Department of Computer Science, University of Bonn, Bonn, Germany, in 2009 and 2014, respectively.

Since 2016, he is a Postdoctoral Researcher with the Photogrammetry & Robotics Lab, University of Bonn. He finished his habilitation with the University of Bonn in 2023. His research interests include in the area of perception for autonomous vehicles, deep learning for semantic interpretation, and LiDAR-based SLAM.

Cyrill Stachniss (Member, IEEE) received the Habilitation degree and the Ph.D. degree in computer science from the University of Freiburg, Freiburg im Breisgau, Germany, in 2009 and 2016, respectively.

He is currently a Full Professor with the University of Bonn, Bonn, Germany, University of Oxford, Oxford, U.K., as well as Lamarr Institute for Machine Learning and AI, Dortmund, Germany.

He is the Spokesperson of the DFG Cluster of Excellence PhenoRob with the University of Bonn. His research interets include probabilistic techniques and learning approaches for mobile robotics, perception, and navigation, agricultural robotics, service robotics, and self-driving cars.