CURL-SLAM: Continuous and Compact LiDAR Mapping

Kaicheng Zhang[1], Shida Xu[1], Yining Ding[2], Xianwen Kong[2] and Sen Wang[1]

***Abstract*—This paper studies 3D LiDAR mapping with a focus on developing an updatable and localizable map representation that enables continuity, compactness and consistency in 3D maps. Traditional LiDAR Simultaneous Localization and Mapping (SLAM) systems often rely on 3D point cloud maps, which typically require extensive storage to preserve structural details in large-scale environments. In this paper, we propose a novel paradigm for LiDAR SLAM by leveraging the Continuous and Ultra-compact Representation of LiDAR (CURL) introduced in [1]. Our proposed LiDAR mapping approach, CURL-SLAM, produces compact 3D maps capable of continuous reconstruction at variable densities using CURL's spherical harmonics implicit encoding, and achieves global map consistency after loop closure. Unlike popular Iterative Closest Point (ICP)-based LiDAR odometry techniques, CURL-SLAM formulates LiDAR pose estimation as a unique optimization problem tailored for CURL and extends it to local Bundle Adjustment (BA), enabling simultaneous pose refinement and map correction. Experimental results demonstrate that CURL-SLAM achieves state-of-the-art 3D mapping quality and competitive LiDAR trajectory accuracy, delivering sensor-rate real-time performance (10 Hz) on a CPU. We will release the CURL-SLAM implementation to the community.**

***Index Terms*—Mapping, Map Representation, SLAM, LiDAR**

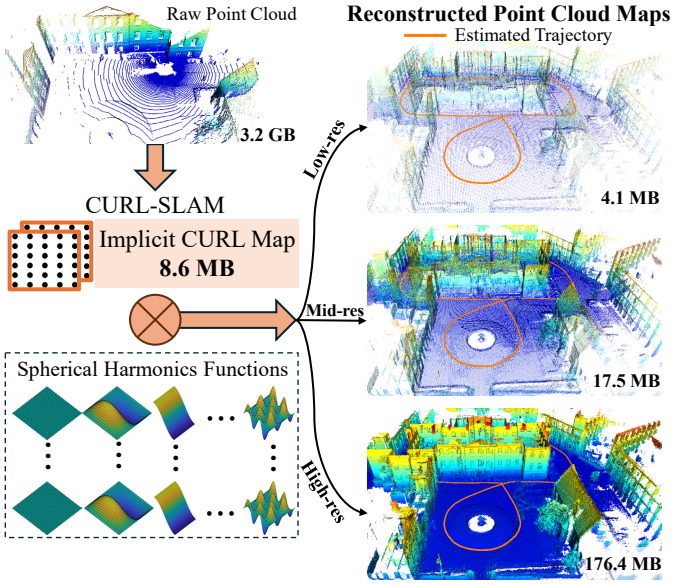


Fig. 1: Globally consistent 3D maps reconstructed by CURL-SLAM. Point cloud maps with different resolutions are continuously reconstructed using the same CURL map which is ultra-compact (0.26% of the 3.2 GB raw point clouds).

I. Introduction

3D mapping is pivotal for Simultaneous Localization and Mapping (SLAM) and autonomous robots. The design and performance of 3D mapping algorithms are significantly affected by their map representations. Among the various 3D map representations, point clouds stand out as the most prevalent due to their simplicity and extensive algorithms and tools available for processing, such as the Point Cloud Library (PCL). However, most point cloud based odometry/SLAM systems, e.g., LOAM [2] and its variants [3], [4], downsample their point clouds for mapping due to computational and storage efficiency, resulting in a potential loss of map resolution and 3D structural details. At the same time, storing all points is not ideal, as map size grows exponentially, leading to significant processing overheads and ample storage space.

Other map representations, such as surfels and meshes, are popular in several SLAM systems [5], [6]. Surfel-based representations benefit SLAM by enabling efficient incremental updates and correcting local distortions. However, the level of detail in surfel-based maps heavily depends on the number of surfels used, which makes large-scale mapping more challenging. Mesh representations, widely used in 3D modeling, offer notable memory efficiency for 3D mapping, especially in structured environments. They also enhance SLAM systems by supporting dense surface reconstruction [6]–[8]. However, achieving high accuracy in mesh maps often requires introducing a large number of vertices. Furthermore, updating a mesh map can be computationally expensive for a real-time SLAM system, as it involves re-establishing vertex connectivity.

Recently, various neural implicit map representations have emerged, typically modeling the spatial environment as a continuous function over coordinates. These methods enable continuous reconstruction, occupancy prediction [9], and shape completion using Signed Distance Functions (SDFs) [10]. Such implicit representations offer smoother, scalable, and memory-efficient reconstructions compared to traditional point-cloud maps. These advantages provide the potential for more accurate, globally consistent, and easily updatable SLAM systems. Several LiDAR-based SLAM systems have adopted neural implicit representations, such as NeRF-LOAM [11], LONER [12], and PIN-SLAM [13]. Notably, PIN-SLAM achieves state-of-the-art LiDAR odometry performance and is

[1] Department of Electrical and Electronic Engineering, Imperial College London, W12 0BZ, UK
k.zhang23@alumni.imperial.ac.uk
{s.xu23, sen.wang}@imperial.ac.uk
[2]School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, EH14 4AS, UK {yd2007, x.kong}@hw.ac.uk
Corresponding author: Sen Wang

among the few complete LiDAR SLAM systems capable of generating globally consistent maps in real-time. A globally consistent map effectively eliminates ghosting artifacts, particularly in revisited regions within large-scale environments. However, these neural implicit approaches generally require substantial computational resources, typically GPU acceleration, limiting their practicality for many robotic applications.

The CURL representation [1] was recently introduced as a method to implicitly encode 3D LiDAR points using spherical harmonics functions. This approach provides a highly compact map representation while enabling continuous reconstruction. Its extension, CURL-MAP [14], further explored the potential of the CURL representation for odometry and mapping. Despite these developments, CURL-MAP is still limited in its ability to achieve consistent large-scale mapping, due to its lack of loop closure and local-window optimization, which are necessary for handling drift in large environments.

In this paper, we present CURL-SLAM, the first full LiDAR SLAM system based on the CURL representation, capable of generating an ultra-compact, updatable, and globally consistent 3D map. CURL-SLAM enables real-time pose estimation and continuous 3D mapping on a CPU at a 10 Hz LiDAR frame rate. Our main contributions are as follows:

- A novel LiDAR SLAM system that allows continuous 3D mapping with adjustable reconstruction resolutions from a single compact CURL map. CURL-SLAM formulates pose estimation as a unique optimization problem specifically tailored for CURL, distinguishing it from widely used Iterative Closest Point (ICP) based techniques.
- A rigorous formulation of local Bundle Adjustment (BA) that jointly optimizes both the CURL map and poses, enhancing map consistency and accuracy. To facilitate effective BA optimization, we derive the analytical Jacobian matrices in detail.
- Extensive experimental evaluations demonstrating CURL-SLAM's state-of-the-art mapping performance, particularly in terms of map accuracy and compactness.

As the example shown in Fig. 1, CURL-SLAM's unique use of the CURL representation enables the recovery of accurate 3D maps at various resolutions from a single CURL implicit map, whose size is only 0.26% of the raw point clouds. We will release our code as open-source for the community[1].

Extension to CURL-MAP [14]: The proposed CURL-SLAM addresses the limitations of CURL-MAP [14], with a specific focus on achieving globally consistent maps. To this end, this paper introduces novel CURL-based local BA and loop closure mechanisms, representing significant extensions to CURL-MAP [14]. Furthermore, we propose new methods for patch generation, data association and height-augmented image formation, replacing the quasi-conformal parametrization used in CURL-MAP [14]. These advancements dramatically reduce the computational overhead of constructing and updating CURL-based maps in large-scale environments, allowing for efficient map creation and real-time updates on CPU-based systems. Additionally, the experiments have been completely redesigned, featuring evaluations against three new state-of-the-art algorithms to provide robust benchmarking.

[1]https://github.com/SenseRoboticsLab/CURL-SLAM

II. Related Work

A. 3D Map Representation

Traditional point cloud maps inherently suffer from high memory consumption, susceptibility to sensor noise, and difficulty in maintaining consistency over large-scale environments. Non-point cloud representations, including continuous implicit maps and neural-based representations, offer smoother, scalable, and memory-efficient reconstructions. These properties enable more accurate, globally consistent, and easily updatable SLAM systems. However, neural-based implicit representations typically require significant computational resources, such as GPU acceleration, which can limit their practicality for real-time applications.

1) Continuous Representation: SDF is widely utilized as continuous implicit map representations in robotics and computer vision due to their ability to encode spatial information efficiently. SDF can be categorized into two main types: Truncated Signed Distance Function (TSDF) and Euclidean Signed Distance Function (ESDF) [15].

TSDF maps are commonly employed in range mapping, especially for RGB-D and LiDAR sensors. To improve mapping efficiency, the weighted distance from each voxel to the nearest surface along the sensor ray is used to update the TSDF map [16]–[19]. On the other hand, ESDF maps, which store the Euclidean distance to its nearest surface for each voxel, are preferred in path planning applications [20]. To generate ESDF maps, several works [21]–[23] have proposed methods to convert from TSDF representations. Most TSDF implementations require GPU acceleration and suffer from memory efficiency limitations. VDBFusion [24] has been introduced to make it run on a CPU more efficiently. However, due to the nature of SDF maps being stored as discretized values within voxel grids, their accuracy is highly dependent on the grid resolution, which can lead to significant memory requirements for dense detailed mapping. Nevertheless, SDF-based representations offer notable benefits for SLAM systems, including incremental and continuous mapping capabilities, high accuracy in surface reconstruction, and efficient integration of sensor measurements over time.

With neural networks, the function of SDF maps can be continuously represented, eliminating the need for voxel grids and offering a more compact and scalable representation [10], [25]. Occupancy Networks [9] offer a probabilistic approach to querying occupancy at any point in Euclidean space, enhancing the flexibility of implicit representations. However, extracting the surface implicitly encoded by neural networks requires initializing a voxel grid or using a ray-casting method to produce a signed distance field or an occupied grid. Recent advancements like Neural Radiance Fields (NeRF) [26] and 3D Gaussian Splatting (3DGS) [27] enhance implicit map representations with rendering new view points. However, these methods need to be executed on GPUs and face challenges in accurately representing 3D geometry in large-scale environments.

Kernel-based non-parametric point cloud modelling techniques, such as Gaussian Processes [28] and Hilbert Space methods [29], have also been explored. However, these approaches typically demand high computational resources, limiting their practicality for real-time applications. Gaussian Mixture Models (GMMs) have also been applied to continuous 3D reconstruction and are effective in generating compact maps [30], capable of running on low-power CPUs to produce continuous occupancy representations. However, they do not focus on highly detailed reconstruction. In contrast, [31] and [32] employ 4D Gaussians to encode point clouds with intensity, achieving compact storage and detailed reconstruction, but at the cost of longer GPU training times, which poses challenges for real-time use.

FS-SLAM [33] inspired our approach by utilizing Fourier Series functions to encode the map, achieving continuous and compact reconstructions. However, FS-SLAM is limited to 2D representations and pillar-like environments, significantly restricting its practical applicability to general 3D mapping tasks. In contrast, our method extends this concept to general 3D environments by leveraging spherical harmonics as the underlying functional representation. This enables continuous, detailed, and computationally efficient reconstructions suitable for broader real-time SLAM applications.

Specifically, CURL-SLAM integrates spherical harmonics with a spatial hashing grid, efficiently clustering point clouds into manageable patches. This reduces computational complexity and memory consumption compared to neural implicit and GMM-based representations. Unlike many neural or occupancy-based methods, our method operates efficiently on CPUs without requiring GPU acceleration, thus balancing computational efficiency and reconstruction fidelity at arbitrary resolutions.

2) Compact Representation: For traditional compact representations, occupancy maps, e.g., OctoMap [34] and Supereight [19], are widely adopted due to their compactness and memory efficiency by employing octrees for spatial partitioning. Wavemap [35] further compresses maps by storing wavelet coefficients within the octrees. Recently, GMM-based occupancy map representations [30]–[32] also achieve compact memory footprint by only storing Gaussian parameters along with their weights.

For learning-based methods that cooperate with the aforementioned occupancy and SDF maps, memory usage can be further reduced by relying solely on network parameters to represent 3D geometry. For occupancy maps, Occupancy Networks [9] utilize neural networks to implicitly encode the occupancy probability function independently of voxel size, thereby achieving compact representations. However, occupancy maps face challenges in recovering structural details in large-scale scenarios, as the grid size inherently limits their resolution. Similarly, SDF maps encoded by neural networks [10], [36]–[39] can produce smoother surfaces, but they also encounter similar issues since the surface needs to be extracted from voxelized signed distance fields, which limits detail fidelity.

Compact representations significantly benefit SLAM systems by enabling efficient memory usage, facilitating scalable mapping of extensive environments, and supporting rapid map updates crucial for real-time operations. However, many existing compact methods, particularly those using neural networks, require GPU acceleration or exhibit limited reconstruction accuracy. In contrast, CURL-SLAM combines compactness and efficiency by encoding surfaces using spherical harmonics and organizing spatial information through a hashing grid. This approach provides detailed and scalable reconstructions while maintaining computational efficiency suitable for real-time SLAM on CPUs.

B. LiDAR Odometry and SLAM

In prevailing LiDAR odometry and mapping frameworks, such as LOAM [2], LeGO-LOAM [3], LIO-SAM [4], CT-ICP [40], KISS-ICP [41], and FAST-LIO2 [42], the focus has largely been on using pose optimization to enhance the accuracy of point cloud maps. These methods achieve high localization accuracy and robustness in real-time performance. Furthermore, BALM [43] and BALM 2.0 [44] achieve locally consistent maps on small-scale environments, while HBA [45] was proposed for achieving globally consistent maps. All these methods rely on further trajectory optimization to improve consistency. Although point cloud maps are simple and straightforward, they inherently contain sensor noise, consume large amounts of memory, and are challenging to update.

Non-point cloud map-based SLAM frameworks have been proposed to address these limitations. For instance, ElasticFusion [5], to some extent, paved the way for vision-based map-centric methodologies by maintaining a deformation node list on a map instead of a pose graph. This idea has been extended to LiDAR [46], [47], this surfel-based representation benefits SLAM by facilitating incremental updates and correcting local distortions through the deformation graph. However, the increasing computational complexity with deformation nodes makes detailed mapping challenging for large-scale environments.

Recently, several LiDAR SLAM systems employing simplified representations have been proposed, including SegMap [48], Quadric Representation [49], VoxelMap [50], VoxelMap++ [51], and SLIM [52]. These frameworks use simplified geometric features for map representation, significantly benefiting SLAM systems by reducing storage requirements and computational complexity, thus enabling faster pose estimation. However, this simplification primarily facilitates efficient localization rather than accurate and detailed 3D reconstruction.

Mesh based SLAM systems, such as PUMA [6], SLAMesh [7], and ImMesh [8], explicitly encode surface geometry into mesh structures. These methods benefit SLAM systems by providing dense, detailed geometric reconstructions beneficial for robust localization and precise scene analysis. However, real-time performance is limited in PUMA due to computational demands, and while SLAMesh and ImMesh achieve real-time operations, they do not incorporate loop closure. Furthermore, handling loop closure in mesh-based systems typically involves complicated procedures for disconnecting and re-establishing vertex connections.

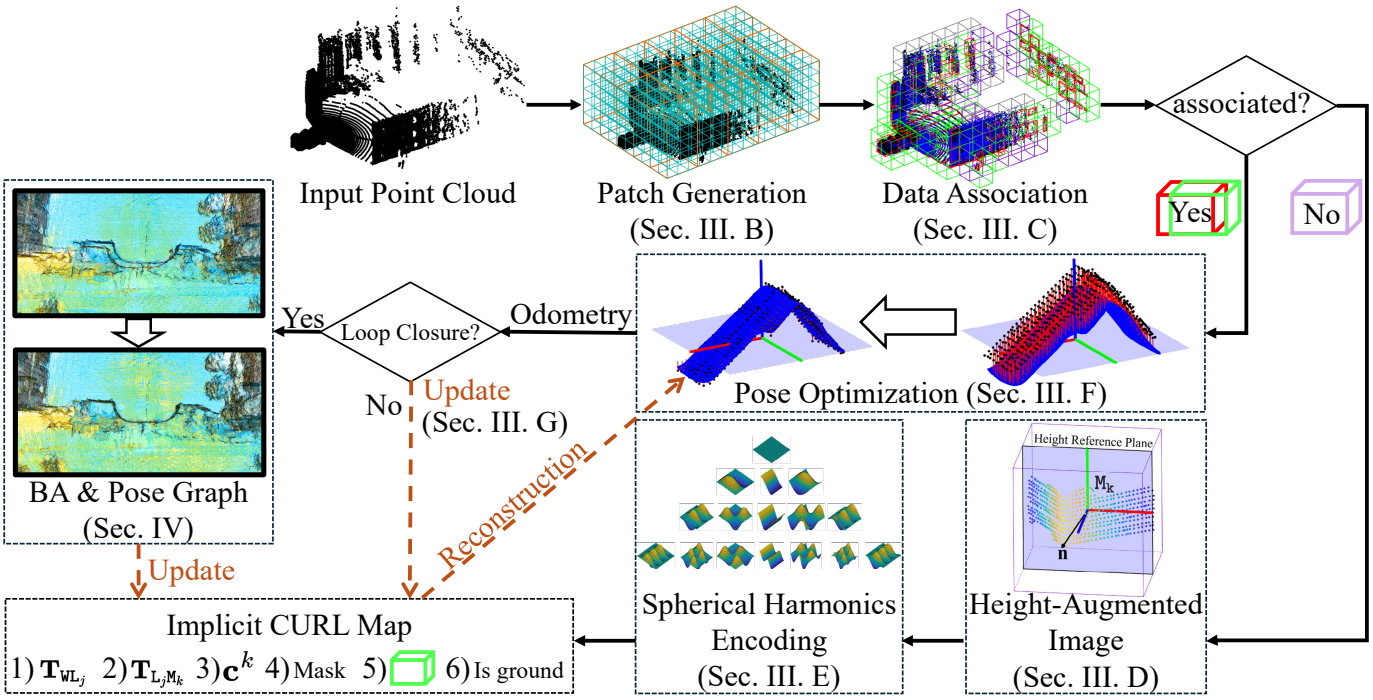


Fig. 2: Pipeline of the proposed CURL-SLAM method.

LiDAR SLAM systems employing implicit map representations have garnered attention in recent years. IMLS-SLAM [53] utilizes implicit moving least squares for map representation, beneficial for accurate map-based localization, but faces high computational demands. Incremental neural implicit SLAM frameworks, such as LONER [12] and NeRF-LOAM [11], leverage neural representations to facilitate high-fidelity reconstruction and robust pose estimation, though only LONER achieves real-time performance. Our previous work, CURL-MAP [14], encodes an implicit map using spherical harmonics functions, allowing for continuous reconstruction and real-time odometry. However, none of these methods incorporate loop closure, preventing the creation of globally consistent maps. PIN-SLAM [13] bridges this gap by using neural points to compactly encode SDF, achieving state-of-the-art performance with the ability to build a globally consistent map while performing real-time odometry on GPUs.

Despite these advancements, building a SLAM system that is capable of building globally consistent, compact and continuous maps in real-time using CPUs remains a considerable challenge. The proposed CURL-SLAM addresses these limitations by providing detailed implicit reconstructions using spherical harmonics functions combined with spatial hashing grids, achieving computational efficiency and global consistency without the need for GPU resources.

III. CURL-based Odometry and Mapping

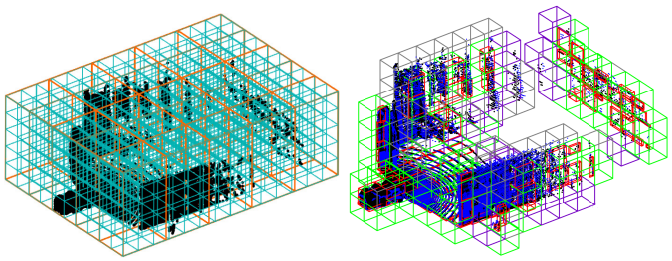
The full pipeline of our CURL-SLAM method is shown in Fig. 2. The SLAM system comprises four main modules: data association, pose optimization, map encoding, and local BA with pose graph optimization, which together enable the construction of a globally consistent map. Instead of using the popular planar/edge features detected from LiDAR point clouds or ICP methods, like LOAM [2] and its variants, our method implicitly encodes point clouds as spherical harmonics functions, using the CURL representation proposed in [1], for odometry and mapping. In this section, we describe the odometry and mapping modules of CURL-SLAM.

A. Notation

A typeface $\mathbb{A}$ denotes a set, bold capital letters $\mathbf{A}$ represent matrices, bold lowercase letters $\mathbf{a}$ denote vectors, and unbolded letters a represent scalars. $\mathbf{T}_{\mathtt{AB}} \in \mathbb{R}^{4\times4}$ represents the transformation from the coordinate frame $\mathtt{B}$ to the coordinate frame $\mathtt{A}$, while ${}_{\mathtt{A}}\mathbf{p}_i \in \mathbb{R}^{3\times1}$ refers to the i-th point in the coordinate frame $\mathtt{A}$. The transformation of a point from frame $\mathtt{A}$ to frame $\mathtt{B}$ is defined as ${}_{\mathtt{B}}\mathbf{p}_i = \mathbf{T}_{\mathtt{BA}}\ {}_{\mathtt{A}}\mathbf{p}_i$, omitting the need for explicit homogeneous coordinates for simplicity.

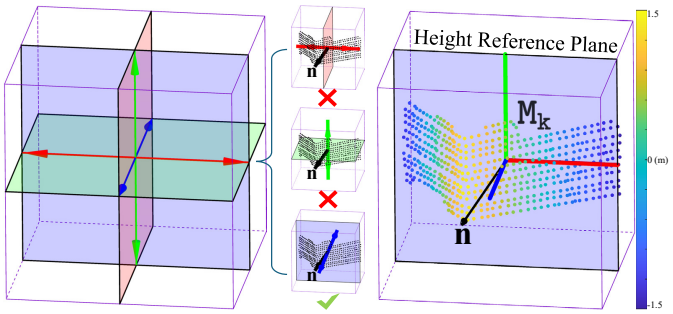
B. Patch Generation

Since ground and non-ground points from LiDAR scans provide different constraints to pose estimation and vary in their complexity for mapping, the current LiDAR scan j is pre-segmented into ground and non-ground points. We use patchwork++ [54] in this work due to its efficiency. Meanwhile, the scan is segmented into a set of raw scan patches $\{M^k\} \subseteq \mathbb{M}_j$ along with their corresponding points $\{{}_{\mathtt{L}_j}\mathbf{P}^k \in \mathbb{R}^{3\times\psi}\}$ in j-th LiDAR frame $\mathtt{L}_j$, using a simple yet efficient volumetric clustering method, i.e., the LiDAR's local 3D space is pre-divided into fixed-size cubic voxels and the set of LiDAR points that fall within the same voxel is clustered as a patch, as the blue cubes shown in Fig. 3(a).



(a) Patch generation. (b) Patch association.

Fig. 3: Patch generation and association; (a) Blue and orange grids show the cubic voxels and the regions of a scan; (b) Selected successful associations between raw scan patches (red boxes) and submap patches (green boxes) for pose estimation and submap updates. Purple boxes represent new patches to be inserted into the submap.



(a) Projection plane selection (b) Patch frame $\mathtt{M}_k$

Fig. 4: (a) Three axis-aligned orthogonal reference planes and their normals in a voxel. $\mathbf{n}$ is the normal vector of the patch points which used to select the height reference plane; (b) Patch frame $\mathtt{M}_k$ defined on the height reference plane.

C. Data Association of Patches

For each patch M^k, it is necessary to determine whether it is associated with a patch in the current submap $\mathbb{S}$ as odometry is computed solely based on the current submap. The details of how this submap is constructed will be introduced later. Assume $\mathbf{T}_{\mathtt{WL}_j}$, the transformation from frame $\mathtt{L}_j$ to the world frame $\mathtt{W}$ is known, the patch points ${}_{\mathtt{L}_j}\mathbf{P}^k$ can be transformed into $\mathtt{W}$ via $\mathbf{T}_{\mathtt{WL}_j}$. Once being transformed into $\mathtt{W}$, the world frame aligned bounding box of M^k is then computed by finding its minimum and maximum values along the Euclidean axes of $\mathtt{W}$ shown as the red boxes in Fig. 3(b). Intersection over Union (IoU) is employed to identify potential associations from M^k's nearby patches in $\mathbb{S}$. If M^k has multiple inlier associations, the associated submap patches in $\mathbb{S}$ with the highest IoU score is selected.

To reduce the computational complexity of the subsequent modules, $\mathbb{M}_j$ of a scan is spatially refined within regions, illustrated as the orange grids in Fig. 3(a). Only the top β_{ng} and β_g associations which have the highest IoU scores for non-ground and ground patches, respectively, are selected from each region for both pose estimation (Section III-F) and submap update (Section III-G), shown as the red and green boxes in Fig. 3(b), and rest of the associations in black boxes are used for submap update only.

Purple boxes in Fig. 3(b) are those M^k without associations from the submap going to be inserted in the submap after generating their CURL representation which has the following attributes: 1) $\mathbf{T}_{\mathtt{WL}_j}$; 2) $\mathbf{T}_{\mathtt{L}_j\mathtt{M}_k}$; 3) spherical harmonics coefficients $\mathbf{c}^k$ of M^k (Section III-E); 4) mask; 5) axis-aligned bounding box in $\mathtt{W}$; 6) ground or non-ground label. $\mathbf{T}_{\mathtt{L}_j\mathtt{M}_k}$ is the transformation from M^k's patch frame $\mathtt{M}_k$ to the local LiDAR frame. Then we describe how to compute the 2) and 3) attributes before formulating the CURL-based pose optimization for 1).

D. Height-Augmented Image

Before converting the patch's points into the CURL representation, the patch frame $\mathtt{M}_k$ needs to be defined to produce a masked height-augmented image. In our previous work [14], conformal mapping parametrization [55] was employed for height-augmented image generation. In this paper, we propose a simplified yet efficient method that omits the quasi-conformal mapping calculation, thereby accelerating the optimization for odometry (Section III-F) and BA (Section IV-B). This approach also simplifies the submap update process (Section III-G).

1) Patch Frame Definition for $\mathbf{T}_{\mathtt{L}_j\mathtt{M}_k}$: Three axis-aligned orthogonal reference planes can be defined at the centre of a cubic voxel, as shown in Fig. 4(a). To select the plane that reserves the maximum information of the patch points for generating a height-augmented image, the normal vector $\mathbf{n}$ of the patch is calculated as

$$\begin{aligned} &\mathbf{n} = \mathbf{v}_{\text{min}}, \ \text{s.t.} \ \mathbf{A}\mathbf{v}_{\text{min}} = \lambda_{\text{min}}\mathbf{v}_{\text{min}} \\ &\text{with} \ \mathbf{A} = \frac{1}{\psi}\sum_{i=1}^{\psi}\left(\mathbf{T}_{\mathtt{WL}_j}\ {}_{\mathtt{L}_j}\mathbf{p}_i - \bar{\mathbf{p}}\right)\left(\mathbf{T}_{\mathtt{WL}_j}\ {}_{\mathtt{L}_j}\mathbf{p}_i - \bar{\mathbf{p}}\right)^T \end{aligned} \tag{1}$$

where λ_{min} is the minimum eigenvalue of matrix $\mathbf{A}$, $\bar{\mathbf{p}} = \frac{1}{\psi}\sum_{i=1}^{\psi}\mathbf{T}_{\mathtt{WL}_j}\ {}_{\mathtt{L}_j}\mathbf{p}_i$, and ${}_{\mathtt{L}_j}\mathbf{p}_i \in {}_{\mathtt{L}_j}\mathbf{P}^k$ denotes the i-th point of patch M_k. The orthogonal reference plane whose normal vector is most closely aligned with $\mathbf{n}$ is then selected as the height reference plane, as illustrated in Fig. 4(a). The origin of the patch frame $\mathtt{M}_k$ in Fig. 4(b) is located at its centre, while its z axis is perpendicular to the plane. Notably, we observed that when the number of patch points is small, selecting the correct height reference plane during the creation of a new submap patch is unstable. Therefore, instead of immediately fixing the height reference plane, points from subsequent scans are accumulated. Once the number of points exceeds a certain threshold, we re-determine and fix the height reference plane.

2) Weighted Height-Augmented Image with Mask: After defining the patch frame $\mathtt{M}_k$, a height-augmented image can be created by projecting all LiDAR points in the patch M^k on to the height reference plane. A discrete grid of a resolution of $W \times W$ is associated with the reference plane, corresponding to the width and height of the square augmented image $\mathbf{H}$, as illustrated in Fig. 5(b). For simplicity, the 2D coordinates $[x, y]^T \in \mathbf{H}$ are defined in the patch frame $\mathtt{M}_k$. A binary mask stores valid pixel positions by coding grey pixels as zero and colored regions as one.

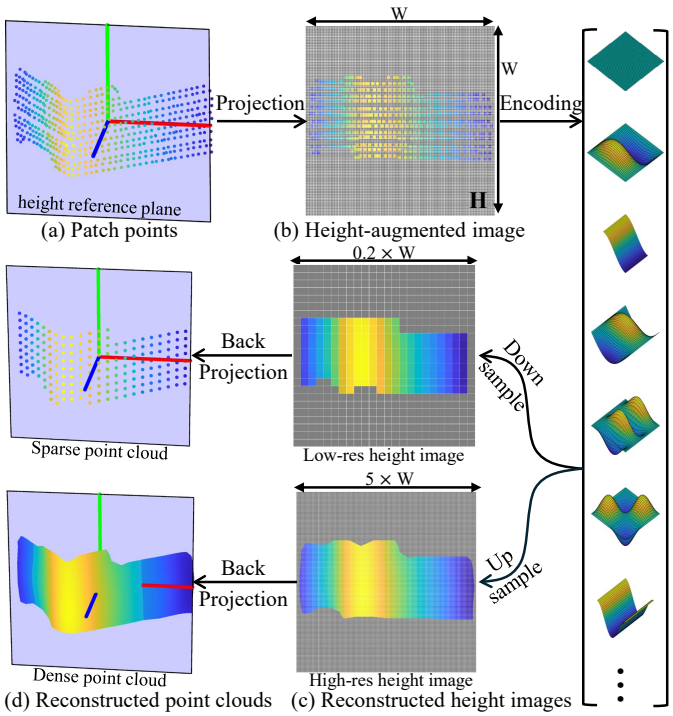


Fig. 5: (a) and (b) illustrate the patch points and the corresponding height-augmented image with a resolution of $W \times W$, respectively. The grey pixels indicate the masked area. After encoding, sparser (down-sampling) and denser (up-sampling) points can be reconstructed. Height values are color-coded using the color bar in Fig. 4(b).

Meanwhile, a weight image $\mathbf{W}$ with the same resolution is initialized based on the range measurement d in the LiDAR frame $\mathtt{L}_j$, as e^{-2d^2/σ^2}, where $\sigma = 50$ is used in our experiment. Basically indicates that points further away are assigned less weight. The pixels of the weight image corresponding to the invalid locations on the height image are initialized to zero. This weight image will be used to update $\mathbf{H}$ in Section III-G.

E. Spherical Harmonics Encoding

The height-augmented image $\mathbf{H}$ is encoded as spherical harmonics which is a series of spherical functions defined on the surface of a sphere [1], [56]. The spherical harmonics expansion function with the maximum degree L is defined as

$$f(\theta,\phi) \approx \sum_{l=0}^{L} \sum_{m=-l}^{l} c_{l,m} Y_{l,m}(\theta,\phi), \ \ (\theta \in [0,\pi], \phi \in [0,2\pi]) \quad (2)$$

where $f(\theta,\phi)$ in spherical coordinate (θ,ϕ) is represented by a linear combination of the real-form spherical harmonics function $Y_{l,m}(\theta,\phi)$ with degree l and order m, defined as

$$Y_{l,m}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} P_{l,|m|}(cos\theta) N_m(\phi) \quad (3)$$

where $P_{l,m}(\cdot)$ is the associated Legendre polynomials [57], and $N_m(\phi)$ is defined in Appendix A. We define $\mathbf{c} = [c_{0,0}, c_{1,-1}, \ldots, c_{l,m}, \ldots, c_{L,L}] \in \mathbb{R}^{(L+1)^2 \times 1}$ as the spherical harmonics coefficients.

To encode discrete observations using this representation, suppose we have ς observations sampled on the sphere. Let $\mathbf{f} = [f(\theta_1,\phi_1), \ldots, f(\theta_\varsigma,\phi_\varsigma)]^\top \in \mathbb{R}^{\varsigma \times 1}$ be the vector of observed values. According to (3), we construct the spherical harmonics basis matrix $\mathbf{Y} \in \mathbb{R}^{\varsigma \times (L+1)^2}$, where each row corresponds to the evaluation of all spherical harmonics functions at one observation point. This results in the following linear system:

$$\mathbf{Y}\mathbf{c} = \mathbf{f} \quad (4)$$

The spherical harmonics coefficients $\mathbf{c}$ that best approximate the function values can then be obtained by solving this system via least squares.

To encode $\mathbf{H}$ using spherical harmonics, we first establish a bijective mapping between the coordinates (x, y) in $\mathbf{H}$ and the spherical coordinates (θ, ϕ). However, the north and south poles in spherical coordinates correspond to singularities, where the azimuthal angle ϕ becomes undefined due to all meridians converging at a single point. As a result, these poles do not preserve a one-to-one mapping. To avoid this issue and maintain bijectivity, we define the following adjusted coordinate transformation:

$$\begin{aligned} \theta(y) &= \frac{y - 0.5s}{s} \cdot \pi \cdot \eta + \frac{\pi}{2} \cdot (1-\eta) \\ \phi(x) &= \frac{x - 0.5s}{s} \cdot 2\pi \cdot \eta + \pi \cdot (1-\eta) \end{aligned} \quad (5)$$

where s is the cubic voxel length for patch generation, and η is a scaling factor (fixed as 0.8 in our experiment). Iterative Residual Fitting method [56] is used to compute the spherical harmonics coefficients for balanced accuracy and efficiency. With the spherical harmonics coefficients, continuous patch reconstructions can be achieved by reconstructing height-augmented image with a width resolution ω. Fig. 5(c) and 5(d) show an example of continuously reconstructing the patch points in Fig. 5(a) using 5-degree spherical harmonics and $\omega = 0.2 \times W$, as down-sampling, and $\omega = 5 \times W$, as up-sampling, respectively.

After finishing the spherical harmonics encoding of a patch, its coefficients $\mathbf{c}$ in the CURL representation can be used to continuously reconstruct and update the submap. Next, we discuss how the CURL map can be utilized for pose estimation.

F. Frame-to-Submap Pose Optimization as Odometry

The LiDAR pose of the current scan relative to the previous scan is formulated as a frame-to-submap pose optimization problem. It minimizes the errors between the projected heights of the raw scan patches and the heights of their corresponding submap patches reconstructed from the CURL's spherical harmonics functions.

Suppose there are K patches in $\mathbb{M}_j$ associated with the submap patches and ${}_{\mathtt{L}_j}\mathbf{p}_i^k \in {}_{\mathtt{L}_j}\mathbf{P}^k$ denotes the ith point in patch M^k. The objective function to be minimized is formulated as

$$\begin{aligned} e(\boldsymbol{\xi}) &= \sum_{k=1}^{K} \sum_{i} \left|\left| \mathsf{P}_z(\mathsf{Q}) - \mathbf{I}^k(\boldsymbol{\mu}) \right|\right|^2 \\ \text{with } \mathsf{Q} &= \mathbf{T}_{\mathtt{WM}_k}^{-1} \mathbf{T}_{\mathtt{WL}_{j-1}} \mathsf{Exp}(\boldsymbol{\xi}) \ {}_{\mathtt{L}_j}\mathbf{p}_i^k \end{aligned} \quad (6)$$

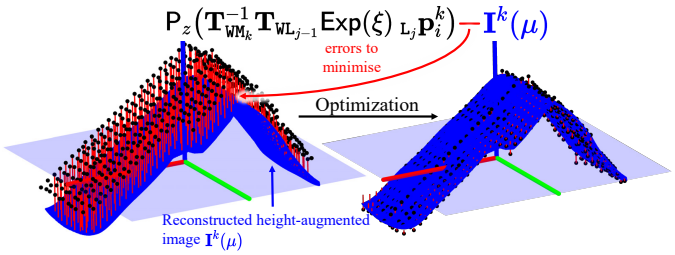


Fig. 6: Pose optimization on errors between the projected heights and the reconstructed heights.

where $\boldsymbol{\xi} \in \mathfrak{se}(3)$ is the Lie algebra of the 6-DoF relative pose from $\mathtt{L}_j$ to $\mathtt{L}_{j-1}$, $\mathsf{Exp}(\cdot)$ is the exponential map, $\mathsf{P}_z([x,y,z]^T) = z$, and $\mathbf{I}^k(\boldsymbol{\mu})$ is the height of the corresponding point reconstructed using the spherical harmonics:

$$\begin{aligned} \mathbf{I}^k(\boldsymbol{\mu}) &= \sum_{l=0}^{L} \sum_{m=-l}^{l} c_{l,m}^k Y_{l,m}\big(\theta(y), \phi(x)\big) \\ \text{with } \boldsymbol{\mu} &= \mathsf{P}_{xy}(\mathsf{Q}) \end{aligned} \tag{7}$$

where $c_{l,m}^k \in \mathbf{c}^k$ is the spherical harmonics coefficient of patch M^k, and $\mathsf{P}_{xy}([x,y,z]^T) = [x,y]^T$. Note $\mathbf{T}_{\mathtt{WM}_k}$ can be derived from the patch's attributes defined in Section III-C. Hence, (6) computes the sum of errors between the projected heights of the patches in the current LiDAR scan j and the heights of the reconstructed submap patches. Intuitively, referring to Fig. 6, if the projected heights are considered as pixel intensities, this is similar to minimizing photometric errors in visual odometry and SLAM systems. [58]–[60]. Since $e(\boldsymbol{\xi})$ is non-linear with respect to $\boldsymbol{\xi}$, its Jacobian matrix is required. The Jacobian matrix of the residual $r_{k,i}(\boldsymbol{\xi})$ inside $||\cdot||^2$ of (6) is

$$\mathbf{J} = \frac{\partial r_{k,i}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \mathbf{J}^{\mathsf{P}_z}\mathbf{J}^{\mathsf{Q}} - \mathbf{J}^{\mathbf{I}^k}\mathbf{J}^{\mathsf{P}_{xy}}\mathbf{J}^{\mathsf{Q}} \tag{8}$$

and according to (30) in Lemma 1

$$\mathbf{J}^{\mathsf{Q}} = \begin{bmatrix} \mathbf{R}_{\mathtt{WM}_k}^{-1}\mathbf{R}_{\mathtt{WL}_{j-1}}\mathsf{Exp}(\boldsymbol{\theta}) & -\mathbf{R}_{\mathtt{WM}_k}^{-1}\mathbf{R}_{\mathtt{WL}_{j-1}}\mathsf{Exp}(\boldsymbol{\theta})[{}_{\mathtt{L}_j}\mathbf{p}_i^k]_\times \end{bmatrix} \tag{9}$$

here, $\boldsymbol{\theta}$ denotes the rotational component of the $\boldsymbol{\xi}$, which is able to be written as $\boldsymbol{\xi} = [\boldsymbol{\rho}, \boldsymbol{\theta}]^\top \in \mathbb{R}^6$, where $\boldsymbol{\rho}$ represents the translational part. More details of these Jacobian terms are given in the Appendix A. The Levenberg-Marquardt method is used to solve the optimization problem.

G. Correction of Spherical Harmonics Coefficients

After the pose optimization, we update the spherical harmonics coefficients of the submap patches based on the optimal pose $\boldsymbol{\xi}^*$. Using $\boldsymbol{\xi}^*$ and $\mathbf{T}_{\mathtt{WM}_k}$, we first transform the point cloud of the current scan to the corresponding patch frame. Then, following the same method described in Section III-D2, we obtain a height-augmented image $\hat{\mathbf{H}}$ and a corresponding weight $\hat{\mathbf{W}}$. The updated height image $\mathbf{H}$ is then computed as shown in (10):

$$\mathbf{H} = (\mathbf{H} \circ \mathbf{W} + \hat{\mathbf{H}} \circ \hat{\mathbf{W}}) \oslash (\mathbf{W} + \hat{\mathbf{W}}) \tag{10}$$

where $\circ$ and $\oslash$ denote element-wise multiplication and division, respectively.

According to (4), the vector form $\mathbf{f} = \mathbf{Y}\mathbf{c}$ allows us to recover the projected heights of a patch using its existing spherical harmonics coefficients $\mathbf{c}$. Moreover, by vectorizing $\mathbf{H}$ and removing invalid pixels, we obtain a vector $\mathbf{h}$ representing the valid height values. The residuals between the observed heights and the reconstructed heights can then be computed as:

$$\mathbf{r} = \mathbf{h} - \mathbf{Y}\mathbf{c}. \tag{11}$$

The updated spherical harmonics coefficients $\mathbf{c}^*$ can be calculated using:

$$\mathbf{c}^* = \mathbf{c} + (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{r}. \tag{12}$$

Since computing the inverse of the matrix is computationally expensive and there are nearly a thousand patches to update for each scan, we accumulate data points over numerous scans for one correction and update them simultaneously using parallel processing. This approach significantly reduces the computational burden while maintaining real-time performance.

IV. Loop Closure and CURL-based Bundle Adjustment

A CURL map with estimated odometry poses can be derived from the previous section. Given the map and poses, we now propose a local bundle adjustment method that is embedded within the loop closure process, jointly optimizing the implicit spherical harmonics encoding of the map patches and the associated keyframe poses.

Due to the inevitable error accumulation in the frame-to-submap odometry, loop closure is required to correct these errors and maintain a globally consistent map. Upon a loop closure is detected, the standard approach typically involves inserting a pose constraint, obtained from ICP alignment, into the pose graph. However, this method does not account for the consistency of other overlapping regions beyond the current frame. While a global bundle adjustment could potentially achieve a consistent map by optimizing all poses and map points simultaneously, it is computationally expensive and demands substantial resources, making it impractical for real-time applications. In this section, a local bundle adjustment is performed for the overlapping regions between the current CURL-based submap and historical CURL-based submaps. This refined information is then integrated into the global pose graph optimization, providing a balance between accuracy and computational efficiency.

A. Loop Closure

Most existing systems apply loop closure only to a pose graph, neglecting map consistency due to the substantial storage and computation requirements of point cloud maps. To address these issues, we introduce a submap strategy in CURL-SLAM, followed by a detailed loop closure integrated with local BA for map consistency.

1) Construction of the Submap Graph: Since all map patches are tied to specific keyframes—through an attribute that links them to the keyframe pose (to be detailed in Section IV-B1)—we create submaps by assigning keyframe labels, where keyframes sharing the same label belong to the same submap. When initializing a submap, the patches observed by the first keyframe of that submap are assigned the same label. For each subsequent keyframe, we monitor the number of observed patches that retain this label. If this number falls below a specified threshold (fixed as 50 in our experiment), a new submap is created, and the new submap is recognized as a neighbor of the previous one. Only map patches belonging to the current and neighbor submaps are used as data association for pose estimation and map update. After a loop is detected and pose graph optimization is performed, the corresponding historical submaps are treated as neighbors of the current submap, enabling data association across the current and historical submaps.

2) Loop Closure for Consistent Mapping: Both a basic loop closure detection method and the scan context approach [61] are employed to detect loop closures. The basic loop closure method identifies historical candidates by searching within a circular region of a small radius that does not belong to its neighboring submaps. Once potential candidates are identified, we reconstruct the point cloud map from the historical submap patches observed by the candidate's historical keyframe. Using the initial odometry estimation, the ICP algorithm is then applied between the current scan and the reconstructed point cloud.

If the ICP successfully converges with an acceptable distance between the two point clouds, we consider the loop closure to be successfully detected. This constraint is then used to update the pose graph. Next, for scans from the current submap that are associated with patches from the historical submap, we perform the local BA, which will be introduced in Section IV-B, to optimize the keyframe trajectories of both the current and historical submaps. Once the BA is completed, these constraints are added to the pose graph for further optimization, and the patches from the current submap that overlap with those from the historical submap are merged. We can see that in Fig. 7 local BA is able to improve the map consistency.

Since our patches are bound to keyframe positions, once the pose graph optimization is completed, the patches' locations in $\mathtt{W}$ are updated. Fig. 8 illustrates that two submaps are consistently merged after loop closure and map update.

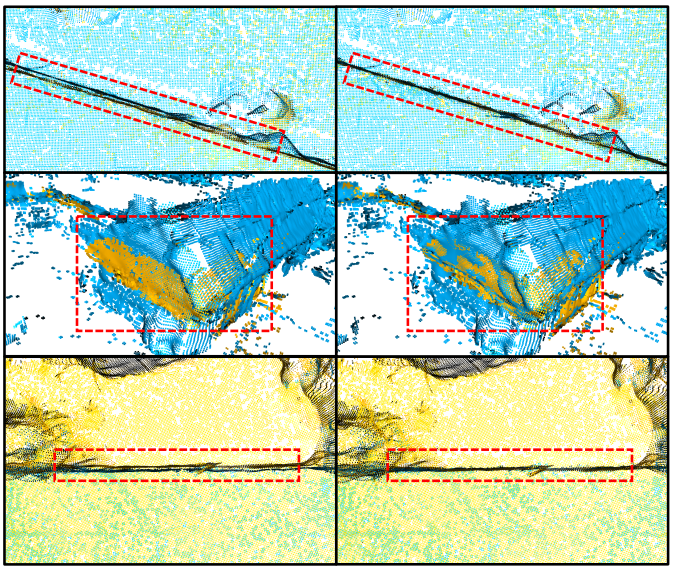


Fig. 7: Reconstructed maps using pose graph optimization only (left) and the proposed local BA (right). Yellow: current submap. Blue: historical submap.

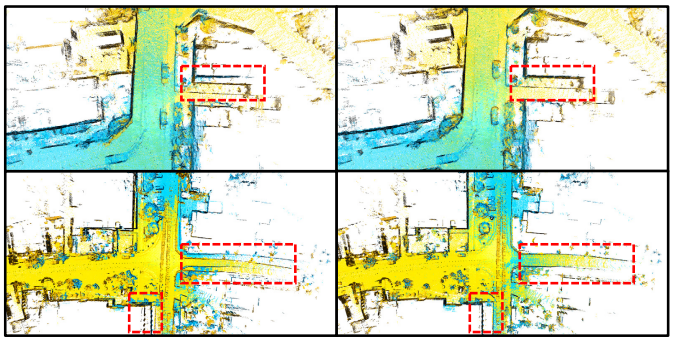


Fig. 8: Reconstructed submaps before (left) and after (right) loop closure. Yellow: current submap. Blue: historical submap.

B. Local Bundle Adjustment

We now describe the details of the local BA used in our system. Since our map is implicitly represented by patches' spherical harmonics functions, our CURL-based local BA is formulated to simultaneously optimize the poses and the spherical harmonics coefficients.

1) Patch-Keyframe Association: To efficiently update the map when the pose has been refined, we associate patches of the map with a keyframe. As stated in Section III-C, each patch has an attribute $\mathbf{T}_{\mathtt{WL}_j}$, which represents its corresponding keyframe pose. When the keyframe pose is updated, the patch's world coordinates are automatically adjusted accordingly. A new keyframe is added whenever the pose has translated a certain distance or rotated beyond a specified degree. Moreover, instead of rigidly binding the patch to its original keyframe, we dynamically associate the patch with the keyframe whose position is the nearest.

2) BA Formulation: Suppose all the scans of the keyframes within a local window belong to the current submap have been divided into raw scan patches. Let $\mathbb{M}_{ba}$ be a set of K raw scan patches that have been successfully associated with a set of N historical submap patches and are intended for use in BA. Unlike in Section III-F, which describes a surjective relationship between the patches from the keyframe scan and the patches from the submap, here multiple keyframe patches may be associated with the same submap patch. Consequently, we have $N \leq K$. The objective of our BA is to simultaneously optimize the keyframe poses within the local window and the spherical harmonics coefficients of the associated submap

patches. The objective function is derived as

$$e_{ba}(\mathbf{T}_{\mathtt{WL}_q}, \mathbf{T}_{\mathtt{WL}_j}, \mathbf{c}^n) = \sum_{k=1}^{K} \sum_{i} ||\mathsf{P}_z(\mathsf{Q}) - \mathbf{I}^n(\boldsymbol{\mu})||^2 \quad \text{with } \mathsf{Q} = \mathbf{T}_{\mathtt{M}_n\mathtt{L}_q} \mathbf{T}_{\mathtt{WL}_q}^{-1} \mathbf{T}_{\mathtt{WL}_j} \, {}_{\mathtt{L}_j}\mathbf{p}_i^k \tag{13}$$

where $\mathbf{c}^n$ represents the vectorized spherical harmonics coefficients of the submap patch n, and it is encoded in $\mathbf{I}^n(\cdot)$ to reconstruct the height using (7). $\mathbf{T}_{\mathtt{WL}_q}$ denotes the pose of the keyframe q with which the historical submap patch is associated. $\mathbf{T}_{\mathtt{WL}_j}$ represents the pose of the keyframe j from the current submap, whose scan contains the point ${}_{\mathtt{L}_j}\mathbf{p}_i^k$. From (13), we observe that only Q differs from (6). Alougth both of the Q are referring to the same concept—transforming a point from the LiDAR frame to the corresponding patch frame, the primary difference is that we decompose $\mathbf{T}_{\mathtt{WM}_n}^{-1}$ into $\mathbf{T}_{\mathtt{M}_n\mathtt{L}_q}\mathbf{T}_{\mathtt{WL}_q}^{-1}$ here. Consequently, $\mathbf{T}_{\mathtt{WL}_q}$, as well as $\mathbf{T}_{\mathtt{WL}_j}$, can be optimized during the BA.

The Jacobian matrix of the residual $r_{k,i}(\mathbf{T}_{\mathtt{WL}_q}, \mathbf{T}_{\mathtt{WL}_j}, \mathbf{c}^n)$ inside $||\cdot||^2$ of (13) is given by:

$$\begin{aligned}\mathbf{J}^{r_{k,i}} &= \begin{bmatrix}\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{r_{k,i}} & \mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{r_{k,i}} & \mathbf{J}_{\mathbf{c}^n}^{r_{k,i}}\end{bmatrix} \\ &= \mathbf{J}^{\mathsf{P}_z}\begin{bmatrix}\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{\mathsf{Q}} & \mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{\mathsf{Q}} & \mathbf{0}\end{bmatrix} - \begin{bmatrix}\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{\mathbf{I}^n} & \mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{\mathbf{I}^n} & \mathbf{J}_{\mathbf{c}^n}^{\mathbf{I}^n}\end{bmatrix}\end{aligned} \tag{14}$$

Here, $\mathbf{J}^{\mathsf{P}_z}$ is presented in (25), and according to (31) in Lemma 2, $\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{\mathsf{Q}}$ is given by:

$$\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{\mathsf{Q}} = \begin{bmatrix}-\mathbf{R}_{\mathtt{M}_n\mathtt{L}_q} & \mathbf{R}_{\mathtt{M}_n\mathtt{L}_q}\left[\mathbf{T}_{\mathtt{WL}_q}^{-1}\mathbf{T}_{\mathtt{WL}_j}\,{}_{\mathtt{L}_j}\mathbf{p}_i^k\right]_\times\end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{15}$$

Similarly, according to (30) in Lemma 1, $\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{\mathsf{Q}}$ is given by:

$$\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{\mathsf{Q}} = \begin{bmatrix}\mathbf{R}_{\mathtt{M}_n\mathtt{L}_q}\mathbf{R}_{\mathtt{WL}_q}^{-1}\mathbf{R}_{\mathtt{WL}_j} & -\mathbf{R}_{\mathtt{M}_n\mathtt{L}_q}\mathbf{R}_{\mathtt{WL}_q}^{-1}\mathbf{R}_{\mathtt{WL}_j}\left[{}_{\mathtt{L}_j}\mathbf{p}_i^k\right]_\times\end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{16}$$

Furthermore, unlike the pose optimization (8) in Section III-F, which utilizes the numerical Jacobian $\mathbf{J}^{\mathbf{I}^k}$ in (26), here we derive the analytical Jacobian matrix directly with respect to both the poses and the spherical harmonics coefficients for fast computation. These are denoted by $\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{\mathbf{I}^n}$, $\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{\mathbf{I}^n}$, and $\mathbf{J}_{\mathbf{c}^n}^{\mathbf{I}^n}$, allowing for the simultaneous adjustment of both keyframe poses and the map. The Jacobian matrix with respect to the keyframe pose $\mathbf{T}_{\mathtt{WL}_q}$ is

$$\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{\mathbf{I}^n} = \sum_{l=0}^{L}\sum_{m=-l}^{l} c_{l,m}^n \mathbf{J}^{Y_{l,m}} \begin{bmatrix}\mathsf{U}\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{\mathsf{Q}} \\ \mathsf{V}\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_q}}^{\mathsf{Q}}\end{bmatrix} \in \mathbb{R}^{1\times 6}, \tag{17}$$

the Jacobian matrix with respect to the keyframe pose $\mathbf{T}_{\mathtt{WL}_j}$ is

$$\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{\mathbf{I}^n} = \sum_{l=0}^{L}\sum_{m=-l}^{l} c_{l,m}^n \mathbf{J}^{Y_{l,m}} \begin{bmatrix}\mathsf{U}\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{\mathsf{Q}} \\ \mathsf{V}\mathbf{J}_{\mathbf{T}_{\mathtt{WL}_j}}^{\mathsf{Q}}\end{bmatrix} \in \mathbb{R}^{1\times 6}. \tag{18}$$

and the Jacobian matrix with respect to the coefficients $\mathbf{c}^n$ can be easily derived based on (7), given by

$$\mathbf{J}_{\mathbf{c}^n}^{\mathbf{I}^n} = \begin{bmatrix}Y_{0,0} & Y_{1,-1} & \dots & Y_{l,m} & \dots & Y_{L,L}\end{bmatrix} \in \mathbb{R}^{1\times(L+1)^2} \tag{19}$$

Here, $\mathsf{U} = \mathbf{J}^{\phi}\mathbf{J}^{\mathsf{P}_x}\mathbf{J}^{\mathsf{P}_{xy}}$, $\mathsf{V} = \mathbf{J}^{\theta}\mathbf{J}^{\mathsf{P}_y}\mathbf{J}^{\mathsf{P}_{xy}}$, and $\mathbf{J}^{\phi}$ and $\mathbf{J}^{\theta}$ can be found in Appendix A. $\mathbf{J}^{Y_{l,m}}$ is the most complex term, as defined in (32) in Lemma 3 of Appendix B. As illustrated in Fig. 9, the tangent vector calculated by our analytical gradients (represented as red arrows) demonstrates strong alignment with the tangent vector calculated by numerical gradients (depicted as green arrows). This alignment substantiates the correctness of our derived Jacobian matrices and the optimization process.

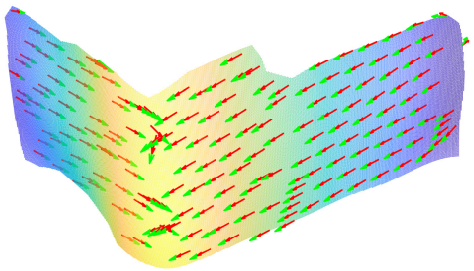


Fig. 9: Comparison of numerical (green) and analytical (red) tangent vector plotted on a reconstructed point cloud.

3) Regularization: Although the above BA is well formulated, in challenging cases where only a few points from a historical submap patch are observed in the local window scans, optimizing the spherical harmonics coefficients may degrade due to insufficient data constraints. To address this problem, we introduce the following regularization term to stabilize the optimization process:

$$e_{reg}(\mathbf{c}^n) = \sum_{n=1}^{N} \left\| \tilde{\mathbf{h}} - \mathbf{Y}\mathbf{c}^n \right\|^2, \tag{20}$$

where $\tilde{\mathbf{h}}$ represents the reconstructed height image values at valid pixels before $\mathbf{c}^n$ is optimized, and $\mathbf{Y}$ denotes the matrix of spherical harmonics basis functions.

Thus, the final objective function for the local BA is

$$e = e_{ba} + \kappa e_{reg}, \tag{21}$$

where $\kappa = 1$ is used as the weight for the regularization term. Again, Fig. 7 presents the reconstructed maps using pose graph optimization only and the proposed BA. It is evident that the proposed BA significantly improves the map alignment between the current and the historical submaps by jointly optimizing the poses and the CURL coefficients, enhancing the map consistency.

V. Implementation Details

To achieve real-time performance on CPUs, apart from the methodology design, we carefully select appropriate data structures and introduce several strategies to accelerate our implementation tailored for the proposed CURL-SLAM.

1) Hashing Voxel for Data Association: Our data association method involves identifying overlapping bounding boxes from the map. However, iterating through each patch in the map can be computationally expensive. To mitigate this challenge, we adopt the approach proposed in [62], which employs a spatial hashing grid in the $\mathtt{W}$ frame. This grid structure stores pointers to patches, thereby improving both memory efficiency and lookup performance.

For any 3D point $\mathbf{p}$ located within a cubic voxel with side length, s, a unique hash key corresponding to the voxel can be computed as

$$\begin{aligned} \text{key} &= [i\ j\ k]^T = \lceil \mathbf{p}/s \rceil, \\ \text{hash} &= (i \cdot P_1) \oplus (j \cdot P_2) \oplus (k \cdot P_3), \end{aligned} \tag{22}$$

where $\lceil\rceil$ is the ceil function, $\oplus$ is bitwise XOR, and P_1, P_2, P_3 are large prime numbers, chosen as 73856093, 19349669, 83492791 in our experiment. If the axis-aligned bounding box of a patch intersects with the voxel, a pointer to the patch's information is stored in the voxel. By storing patch pointers in the hashing voxel, we achieve efficient access to all the CURL attributes described in Section III-C. For a query axis-aligned bounding box of a patch, the hash keys for the voxel intersected with the query bounding box can be computed immediately. This allows us to efficiently retrieve the relevant map patches and use their bounding boxes to perform a more precise intersection detection, thereby enhancing both the accuracy and performance of the data association process. Once the relevant patches are identified from the map, the insertion and removal of patches in the hashing voxel structured map are highly efficient, further reducing the computational cost of updating the map.

2) Accelerating Odometry Performance: Through experimentation, it is found that the reconstruction step in (7) for pose optimization in odometry consumes the majority of the computational time. Additionally, the map patches observed by the current scan are highly likely to be observed in subsequent scans. Therefore, we reconstruct the height images and compute their gradients in advance. The precomputed reconstructed images and gradients are then efficiently updated during the patch update process, as described in Section III-G with parallel processing. By precomputing these values, we significantly reduce the computational overhead during pose optimization, enhancing real-time performance only using CPUs while maintaining accuracy.

3) Managing RAM Utilization: Although our map representation is compact, the previously mentioned precomputed reconstructed and gradient images can impose a significant memory burden during runtime. To manage memory usage effectively, we clear this information for map patches that are out of the LiDAR's line of sight and dynamically recover it when these patches re-enter the LiDAR's field of view. This strategy ensures that memory resources are optimally allocated, maintaining system performance while reducing unnecessary memory consumption.

4) Spherical Harmonics Lookup Table: Repeatedly computing the values of the spherical harmonics functions $\mathbf{Y}$ for every map patch update is computationally inefficient. To address this, we precalculate a lookup table for the spherical harmonics functions, along with their corresponding Jacobian matrices, at a high resolution. This table is then stored for efficient lookup during subsequent updates. By utilizing this precomputed lookup table, the computational cost associated with map patch update is considerably reduced, thereby improving the overall CURL-SLAM efficiency.

VI. Experiments

We evaluate the proposed CURL-SLAM in terms of both mapping and trajectory performance. Given that the primary focus of CURL-SLAM is on mapping rather than localization, our evaluation emphasizes the mapping results.

A. Setup

1) Datasets: We use both the Newer College dataset [2] [63] and the FusionPortable dataset [3] [64] for map evaluation due to the availability of survey-level ground truth maps obtained using a Leica BLK360. Both datasets employ 128-channel LiDAR sensors: the Newer College dataset uses an Ouster OS0-128 sensor with a 90° vertical field of view (FoV), while the FusionPortable dataset uses an Ouster OS1-128 sensor with a 45° vertical FoV.

2) Benchmarking Methods: In the following experiments, we present our results in comparison with SLAMesh [7], PIN-SLAM [13] and HBA [45], the state-of-the-art SLAM systems that use different mapping representations. SLAMesh, which uses a mesh-based representation, builds a mesh map online on a CPU. In contrast, PIN-SLAM incorporates a neural representation with loop closure to produce a globally consistent map, operating in real-time on certain GPUs but not on CPUs. To evaluate with a traditional point cloud-based method, we select HBA, an offline global hierarchical BA method for point clouds, yielding optimal trajectory accuracy with LiDAR data. In our experiments, we use PIN-SLAM's trajectory as the input for HBA, following the finding in [13] that HBA with PIN-SLAM's trajectory performs better than the combination of HBA and MULLS [65]. Since the map accuracy of standard LiDAR SLAM methods, such as LOAM [2] and LeGO-LOAM [3], depends solely on pose accuracy, and HBA has already achieved optimal results in this regard, we exclude these methods from comparison.

For HBA, we followed the guidelines in their paper, setting the layer number to 3 for sequences with fewer than 5×10^3 poses, and to 4 for sequences with more than 5×10^3 poses. For SLAMesh, all experiments used the meshing parameters provided for the KITTI dataset, as SLAMesh does not account for LiDAR channels due to the absence of feature extraction. For PIN-SLAM, we primarily used the parameters from `run_ncd_128.yaml` and adjusted certain parameters for successful runs on specific sequences, such as `cloister` and `stairs`. Additionally, we used a 20 cm voxel size for PIN-SLAM's mesh reconstruction, as described in their paper.

3) CURL-SLAM Setting: In our experiments, for mapping parameters the default voxel side length is set to $s = 1.5\,\text{m}$ for generating both ground and non-ground patches (Section III-B). The default degree of the spherical harmonics for ground patches is set to 2, while for non-ground patches it is set to 5, as ground patches generally exhibit less geometric variation. Additionally, the default width resolution of reconstructed height-augmented images is set to $\omega = 30$ for odometry and default map reconstruction, with $s = 1.5\,\text{m}$, resulting

[2] https://ori-drs.github.io/newer-college-dataset/multi-cam/

[3] https://fusionportable.github.io/dataset/fusionportable/#publications

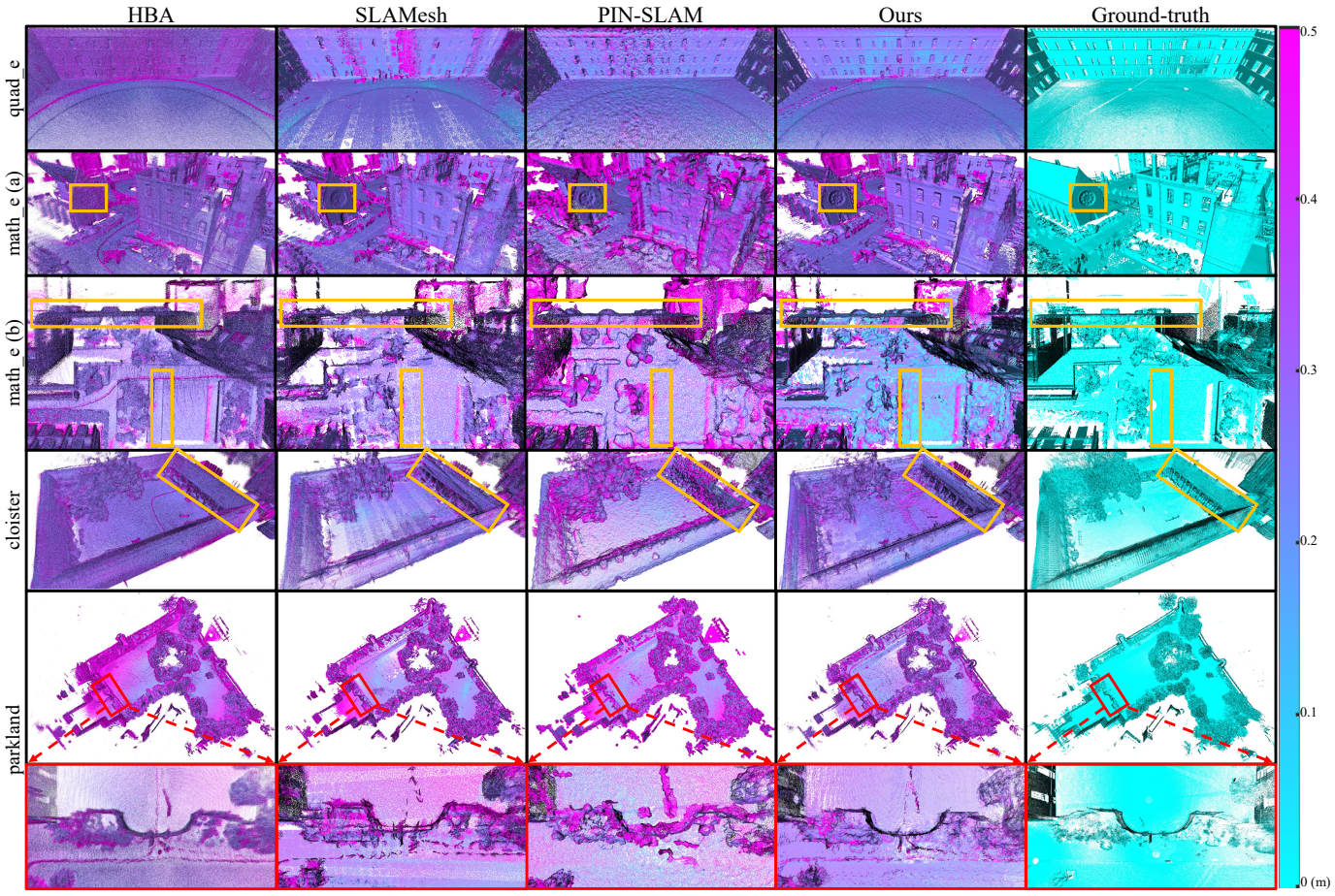
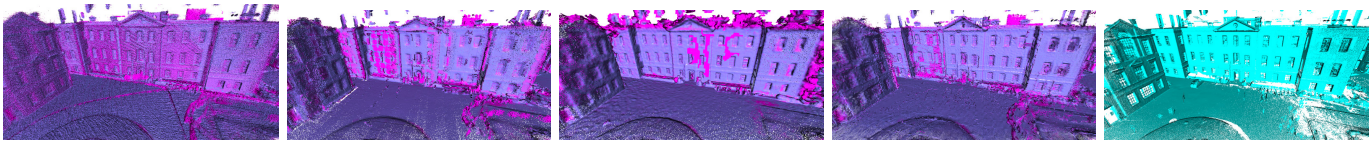


Fig. 10: Reconstructed maps on the Newer College Dataset.



(a) HBA (b) SLAMesh (c) PIN-SLAM (d) Ours (e) Ground-truth

Fig. 11: Reconstructed maps in a region in `math_e` scanned for a longer time and visited multiple times.

in a point interval on the patch of $s/\omega = 5\,\text{cm}$. The spherical harmonics coefficients of a map patch are updated using the accumulated residuals from every 5 data associations. The region number is set to 25, and the data association parameters are set to $\beta_{ng} = \beta_g = 30$. To speed up the performance, initial scan are down-sampled using 0.2 m voxel before entering our pipeline. All mapping parameters are kept consistent across all experiments. For pose estimation, we utilize all associated patches to enhance robustness, particularly on challenging sequences marked with an asterisk (*) in TABLE VII.

All results of CURL-SLAM were conducted on a laptop equipped with an Intel i7-10875H CPU.

B. Evaluation on Map Reconstruction

In this section, we present both qualitative and quantitative evaluation results on the map reconstruction. To assess the accuracy of a reconstructed map, we first align the map generated by each SLAM method with the ground truth map using ICP algorithm.

For the qualitative evaluation, we sample the same number of points from the meshes produced by SLAMesh and PIN-SLAM, as well as from our default reconstruction. The HBA point cloud map is the downsampled map using voxels with a side length of 5 cm, matching our distance of point interval on the patch. The points are then colorized based on their distance errors to the nearest neighbor points in the ground-truth maps.

For the quantitative evaluation, both SLAMesh and PIN-SLAM generate mesh maps. Mesh maps are built from HBA's point cloud map and from our reconstructed point cloud map, using the default resolution with the ball-pivoting algorithm [66]. We then uniformly sample an equal number of points from the meshes produced by all methods. This ensures a fair comparison in terms of the map accuracy (M. Acc), map completeness (M. Comp), and L1-Chamfer distance (C-L1),

TABLE I: Map evaluation results on Newer College Dataset

Approach	Metric	Sequence				
		quad_e	math_e	cloister	parkland	stairs
HBA	M. Acc ↓	17.83	16.15	16.32	17.93	9.88†
	M. Comp ↓	158.41	23.14	129.40	113.43	**186.92**†
	C-L1 ↓	88.12	19.65	72.86	65.68	**98.40**†
	F-Score ↑	17.83	68.06	31.11	36.29	**7.66**†
SLAMesh	M. Acc ↓	7.15	10.75	10.80	15.72	N/A
	M. Comp ↓	155.73	18.52	132.22	111.97	N/A
	C-L1 ↓	81.44	14.64	71.51	63.85	N/A
	F-Score ↑	**33.84**	**83.85**	39.37	45.90	N/A
PIN-SLAM	M. Acc ↓	9.48	13.16	11.23/11.18*	17.39	**6.04***
	M. Comp ↓	**155.14**	19.45	**127.28**/127.64*	111.30	195.01*
	C-L1 ↓	82.31	16.31	69.25/69.41*	64.34	100.53*
	F-Score ↑	32.27	75.16	42.38/41.67*	41.70	4.15*
Ours	M. Acc ↓	**6.46**	**9.95**	**9.95**	**13.33**	6.45
	M. Comp ↓	155.72	**18.18**	127.63	**110.11**	193.11
	C-L1 ↓	**81.09**	**14.06**	**68.79**	**61.72**	99.78
	F-Score ↑	33.29	83.43	**42.84**	**50.16**	4.77

The best results are highlighted in **bold**, and the second-best results are underlined. N/A indicates a failure in mapping due to inconsistent regions.
* Results obtained using the recommended parameters.
† HBA fails to produce a consistent map at this sequence although it has some good results.
For the `stairs` sequence, our method used a smaller voxel side length of $s = 0.5$ m due to its narrow space.

as well as the F-score as a percentage (%). Following the evaluation methodology of PIN-SLAM, M. Acc, M. Comp, and C-L1 are reported in centimeters (cm), calculated with a 20 cm error threshold.

1) Newer College: Fig. 10 shows the reconstruction results from sequences of the Newer College dataset. The sequences `quad_e`, `math_e`, and `cloister` represent structured regions, while `parkland` demonstrates results from 2 loops in the woods from the `park` sequence, highlighting an unstructured region.

In the `quad_e` sequence, our method clearly preserves significantly more detail compared to HBA, SLAMesh, and PIN-SLAM. HBA exhibits numerous noise points on its surface, likely due to sensor noise and scans with some levels of pose errors. SLAMesh displays blurred map details, with some white areas on the ground indicating insufficient mesh connectivity. In contrast, PIN-SLAM introduces multiple artificial bumps that do not exist in the actual environment. Nevertheless, in this case, PIN-SLAM does demonstrate an advantage by effectively removing dynamic points.

The `math_e (a)` further accentuates these differences. Both HBA and SLAMesh smooth out height variations, resulting in a loss of structural details. Although PIN-SLAM retains some details, it remains inferior to our method, as artificial bumps reappear on sections of the walls.

`math_e (b)` presents a top view, showing that the wall mapped by HBA suffers from excessive thickness. SLAMesh displays spurious points as artifacts, while PIN-SLAM appears overly smoothed. In contrast, our method preserves greater sharpness. Additionally, ours is the only non-point-based method capable of recovering curb step edges at the bottom of the figures. Furthermore, our method accurately identifies non-observed regions, similarly to HBA, while both SLAMesh and PIN-SLAM fill gaps that should remain unfilled.

In the `cloister` sequence, only our method clearly reconstructs all the pillars, and even the texture of the roof is preserved.

The `parkland` sequence demonstrates that our method has the potential to perform even in unstructured environments like woods. The map produced by SLAMesh contains a ghost shadow in the red-box highlighted region, likely due to the absence of a loop closure detector and the inability to update the map for global consistency. While PIN-SLAM generates a globally consistent map, it struggles with reconstructing thin surfaces, as indicated by the substantial map errors associated. HBA also produces a globally consistent map but with noise points on its surface due to sensor noise and small errors of trajectories as mentioned before which suffers a thick and blurred surface. In contrast, our method not only maintains global consistency but also recovers fine details with sharpness, such as the visible open gate.

The quantitative results in TABLE I further support the observations made in Fig. 10. Although the original point cloud from HBA contains many inlier points, the thick surface makes it difficult to create a well-connected mesh. Consequently, its completeness results are the worst overall. In contrast, our method does not suffer from this issue.

Additionally, for regions visited multiple times, the reconstruction of PIN-SLAM is generally good, albeit slightly smoothed like Fig. 11 shows. However, in areas visited only once, such as some areas in the `math_e` sequence (Figures 10 and 15), the reconstruction quality noticeably drops. This may be due to the larger number of parameters that need to be fitted for the PIN-SLAM's MLPs compared to our CURL coefficients.

2) FusionPortable: The 3D maps reconstructed by the methods on the FusionPortable dataset are shown in Fig. 12. From the reconstructions of the `canteen_night` and `garden_night` sequences, it is clear that our method more accurately identifies most of the unoccupied regions, as well as the point cloud map from the HBA method. In the `garden_night` sequence, it is evident that SLAMesh fails to reconstruct a smooth building wall. Additionally, SLAMesh cannot produce a consistent map in the `corridor` sequence due to the lack of loop closure, whereas both PIN-SLAM and our method successfully reconstruct consistent maps.

For the `building_day` sequence, although SLAMesh achieves the best accuracy according to TABLE II, its reconstruction of the wall is less smooth compared to PIN-SLAM and our method. HBA also produces noisy points on the wall. PIN-SLAM, on the other hand, exhibits larger errors in the woodland area.

According to the quantitative results in TABLE II, our method achieves the highest map reconstruction accuracy across most sequences. Even when it ranks second, its map accuracy is only marginally lower than the best results.

Overall, CURL-SLAM consistently outperforms existing methods across both the Newer College and FusionPortable datasets, achieving the best or second-best performance in nearly all map evaluation metrics. It demonstrates strong capabilities in reconstructing accurate, complete and sharp maps—especially in various environments.

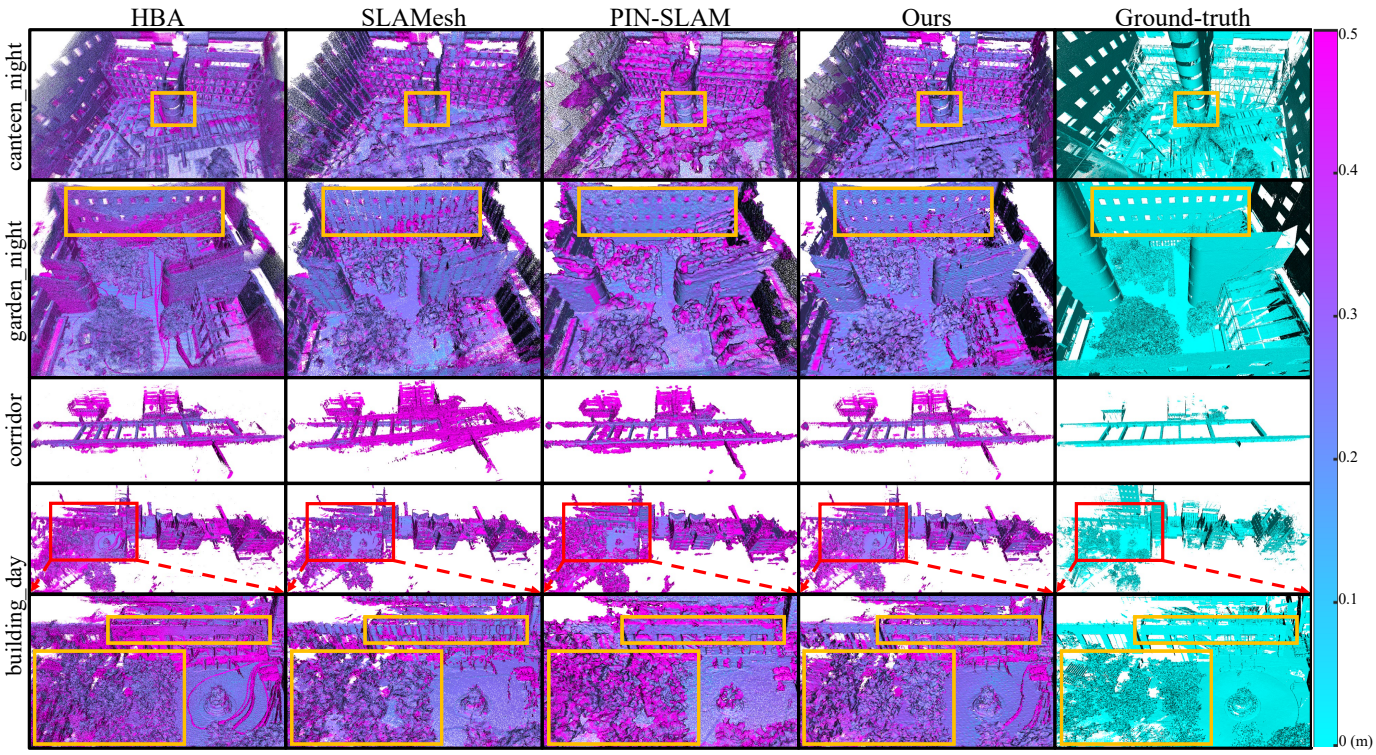


Fig. 12: Reconstructed maps on the FusionPortable Dataset.

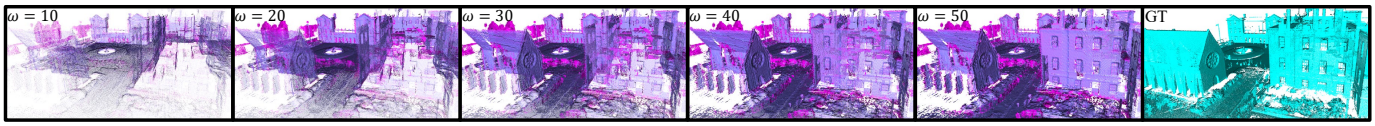
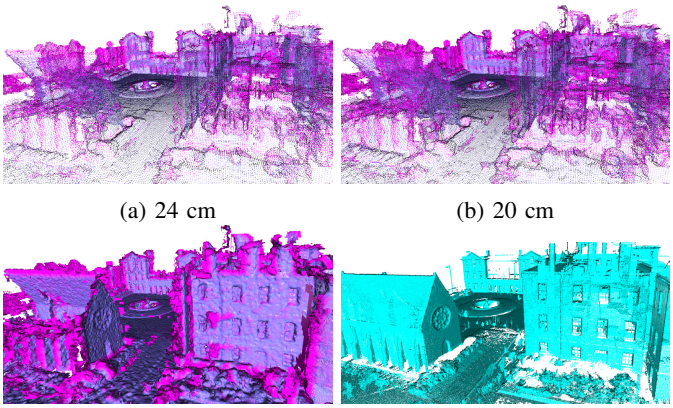


Fig. 13: Continuous reconstruction of CURL-SLAM with ω values corresponding to TABLE III(a).

C. Evaluation on Continuous Map Reconstruction

The proposed CURL-SLAM method is capable of continuous reconstruction by adjusting the parameter ω, as detailed in Section III-E. To evaluate our results on continuous reconstruction, we use the ball-pivoting algorithm to generate meshes from maps reconstructed with increasing resolutions. TABLE III(a) shows that, even when reconstructed at different resolutions, the accuracy of our map remains relatively consistent. This indicates that our continuous reconstruction avoids overfitting or over-sampling, ensuring map reconstruction reflects true 3D scenes and structures. Additionally, as ω increases, our completeness, L1-Chamfer distance, and F-score improve, which is expected. This indicates that our map covers more regions of the ground-truth map. The reconstructed maps corresponding to different ω settings in TABLE III(a) are visualized in Fig. 13.

PIN-SLAM can also extract meshes with varying marching cube resolutions on the SDF map for continuous reconstruction. However, as shown in TABLE III(b), increasing the voxel grid resolution used to discretize the signed distance field for finer mesh extraction leads to reduced map accuracy. This suggests that more points are being sampled from invalid regions by PIN-SLAM, as illustrated in Fig. 14 when comparing with Fig.13.



(a) 24 cm (b) 20 cm

(c) 5 cm (d) Ground-truth

Fig. 14: Continuous reconstruction of PIN-SLAM with voxel sizes in TABLE III(b).

D. Evaluation on Mapping with Different CURL Degrees

Our map accuracy depends on the degree of spherical harmonics coefficients used. The higher the degree, the more geometric variation can be captured. Therefore, we evaluate the relationship between map accuracy and the degree of the CURL's spherical harmonics.

TABLE II: Map evaluation results on the FusionPortable Dataset

Approach	Metric	Sequence							
		canteen_night	garden_night	MCR_fast	MCR_normal	MCR_slow	corridor_day	escalator_day	building_day
HBA	M. Acc ↓	14.43	15.65	21.44	20.38	18.40	13.33	16.66	17.72
	M. Comp ↓	78.23	46.68	31.49	27.42	25.48	**13.00**	54.15	40.39
	C-L1 ↓	46.33	31.17	26.46	23.90	21.94	**13.16**	35.41	29.05
	F-Score ↑	55.78	48.74	31.79	36.88	45.63	**80.27**	51.83	53.21
SLAMesh	M. Acc ↓	10.65	8.32	10.53	9.26	9.36	N/A	**11.24**	**12.67**
	M. Comp ↓	73.18	**34.67**	**5.68**	**5.84**	**6.04**	N/A	**43.51**	34.39
	C-L1 ↓	41.91	**21.50**	**8.11**	**7.55**	**7.70**	N/A	**27.38**	23.53
	F-Score ↑	**69.38**	**83.13**	87.69	**90.14**	89.62	N/A	**78.01**	72.57
PIN-SLAM	M. Acc ↓	12.65	9.69	**9.42**	8.71	8.66	13.99	13.92	15.43
	M. Comp ↓	**72.80**	36.22	7.05	6.79	7.04	13.68	46.18	35.91
	C-L1 ↓	42.73	22.96	8.23	7.75	7.85	13.83	30.05	25.67
	F-Score ↑	63.51	76.94	**88.34**	89.22	89.25	76.70	68.25	65.38
Ours	M. Acc ↓	**9.92**	**7.37**	9.48	**8.69**	**8.13**	**12.24**	11.92	13.19
	M. Comp ↓	73.67	35.69	8.03	7.62	7.43	16.87	45.30	**31.95**
	C-L1 ↓	**41.80**	21.53	8.75	8.16	7.78	14.56	28.61	**22.57**
	F-Score ↑	68.78	81.31	88.27	89.62	**90.49**	77.53	75.94	**72.90**

The best results are highlighted in **bold**, while the second-best results are underlined. N/A indicates a failure in mapping due to inconsistent regions.

TABLE III: Continuous reconstruction on `math_e`

(a) CURL-SLAM

Metric	ω				
	10	20	30	40	50
M. Acc ↓	9.53	9.46	9.50	9.50	9.53
M. Comp ↓	23.13	20.17	19.21	18.68	18.34
C-L1 ↓	16.33	14.82	14.35	14.09	13.94
F-Score ↑	79.51	83.08	83.83	84.09	84.20

(b) PIN-SLAM

Metric	Voxel Size		
	24 cm	20 cm	5 cm
M. Acc ↓	12.90	13.17	13.51
M. Comp ↓	20.84	19.45	18.13
C-L1 ↓	16.87	16.31	15.82
F-Score ↑	75.05	75.15	75.74

* The greener, the better result.

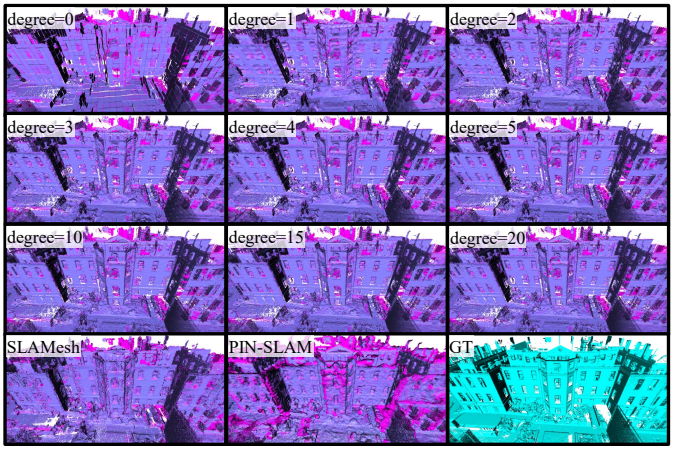


Fig. 15: Reconstruction at different CURL degrees. SLAMesh, PIN-SLAM and Ground-truth results are shown in the last row.

TABLE IV: Map evaluation on CURL degrees (`math_e`).

Metric	CURL Degree								
	0	1	2	3	4	5	10	15	20
M. Acc ↓	16.82	10.33	9.95	9.75	9.61	9.53	9.38	9.39	9.46
M. Comp ↓	30.99	20.26	19.65	19.28	19.05	18.88	18.68	18.60	18.65
C-L1 ↓	82.38	15.29	14.80	14.52	14.33	14.21	14.03	14.00	14.05
F-Score ↑	76.88	81.17	82.46	83.16	83.60	83.89	84.38	84.47	84.40

* The greener, the better result.

TABLE V: Map size evaluation (Megabyte - MB)

	Datasets	HBA	SLAMesh	PIN-SLAM	Ours
Newer College	quad_e	198.8	29.5	10.8	**3.5**
	math_e	274.7	62.9	19.4	**8.6**
	ug_e	71.3	13.8	3.3	**2.0**
	cloister	227.1	41.9	9.1	**6.3**
	stairs	18.1	N/A	4.0	**1.9**
	parkland	807.5	147.8	43.4	**21.8**
FusionPortable	canteen_night	192.6	53.5	11.7	**4.3**
	garden_night	208.2	50.0	9.5	**3.3**
	MCR_fast	33.6	3.6	1.5	**0.47**
	MCR_normal	28.7	3.0	1.3	**0.44**
	MCR_slow	27.8	3.0	1.2	**0.42**
	corridor_day	170.3	49.3	9.4	**6.9**
	escalator_day	298.0	65.7	12.9	**10.0**
	building_day	591.7	112.4	31.4	**17.2**

The best results are highlighted in **bold**, while the second-best results are underlined.

As shown in Fig. 15, at degree 0, all patches are represented as flat planes. As the degree increases, more details are recovered. TABLE IV supports the same finding: with increasing CURL degrees, map accuracy gradually improves. However, the improvement becomes marginal beyond degree 5, indicating that degree 5 captures most of the geometric details for scenarios with similar structural complexity as the sequence `math_e`.

E. Evaluation on Map Size

For each patch, we only need to store a boolean value to indicate whether it is a ground or non-ground patch, which helps to determine the degree it utilizes. Then, the corresponding spherical harmonics coefficients are stored, consuming $(L+1)^2$ double-precision numbers. Additionally, a transformation matrix of size 3×4 floats is stored to represent the information of $\mathbf{T}_{\mathtt{WM}_k}$ for the k-th patch. Finally, a binary mask of size ω^2 bits is required. For a map with N patches,

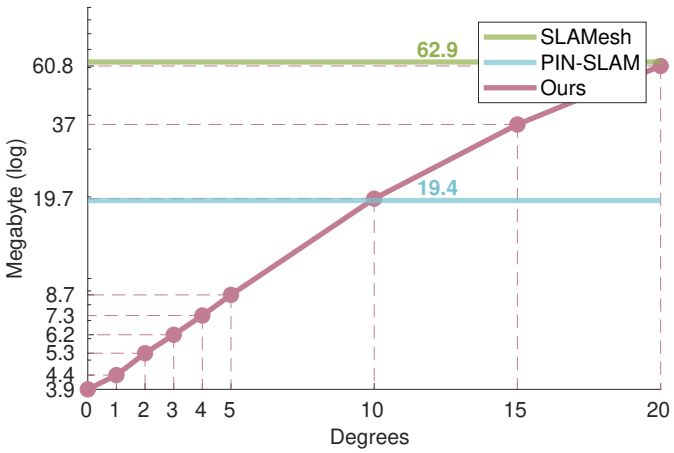


Fig. 16: Map size changes with degrees (`math_e`).

the map size in total can be formulated as follows:

$$\begin{aligned}\text{map_size} =& N \times \Big(\text{sizeof(bool)} + (L+1)^2 \times \text{sizeof(double)} \\ &+ 12 \times \text{sizeof(float)} + \frac{\omega^2}{8}\Big) \text{ bytes}\end{aligned} \tag{23}$$

Therefore, by saving the data in the aforementioned format as a binary file, we can achieve an ultra-compact map. This saved map also supports continuous reconstruction with varying resolutions.

From TABLE V, we observe that our map size is approximately 2.8% of HBA's dense point cloud map, 13.4% of SLAMesh's mesh map and around 51.6% of PIN-SLAM's neural map on average.

Our map size heavily depends on the degree of spherical harmonics used, as the number of coefficients equals $(L+1)^2$. Fig. 16 illustrates how our map size changes with different degrees. It shows that PIN-SLAM's map size is similar to ours when using a degree of 10, while SLAMesh's map size is comparable to our map size with a degree of 20.

F. Evaluation on Estimated Trajectory

1) LiDAR Odometry Evaluation: TABLE VI presents the pure odometry accuracy, obtained by disabling the loop closure modules in both PIN-SLAM and our method. Across most sequences, our method consistently ranks second in performance. The HBA method is excluded from this comparison, as it does not include an odometry module. Additionally, the `stairs` and `MCR_slow` sequences are omitted as they are too short to provide meaningful results.

2) LiDAR SLAM Trajectory Evaluation: TABLE VII demonstrates that the ATE trajectory accuracy of our proposed method consistently ranks in the top two positions in over half of all the sequences, achieving localization accuracy comparable to that of HBA, SLAMesh and PIN-SLAM. Notably, the ATE results reported here are from the same runs used in the map reconstruction evaluation, ensuring consistency across trajectory and mapping performance comparisons. Although the HBA method generally achieves the best trajectory accuracy, it performs the worst in the map accuracy. This observation tentatively suggests that, after a certain level of pose precision

TABLE VI: Odometry evaluation

	Datasets	SLAMesh	PIN-SLAM	Ours
Newer College	quad_e	0.427/1.539	**0.277/1.163**	0.394/1.396
	math_e	**0.252**/0.982	0.265/**0.512**	0.272/0.702
	ug_e	1.000/5.532	**0.383/2.020**	0.881/4.630
	cloister*	0.992/2.610	**0.423/1.042**	0.442/1.706
	parkland*	0.784/0.926	1.735/1.954	**0.637/0.814**
FusionPortable	canteen_night	0.115/0.650	0.154/**0.647**	**0.114**/0.666
	garden_night	0.254/1.384	0.149/0.787	**0.111/0.638**
	MCR_fast	0.246/4.647	**0.176**/3.631	0.193/**3.186**
	MCR_normal	0.295/80.056	**0.290/79.751**	0.337/80.113
	corridor_day	2.019/4.303	**0.398/0.879**	1.087/1.620
	escalator_day	0.233/1.911	**0.180/1.077**	0.243/1.925
	building_day	**0.329**/0.607	0.438/0.733	0.337/**0.601**

Relative errors averaged over trajectories of 100 to 800 m length: relative translational error in % / relative rotational error in degrees per 100 m.
The best results are highlighted in **bold**, while the second-best results are underlined.
* represents the sequences that using all the available associated patches for pose estimation in our method.

TABLE VII: Trajectory evaluation (ATE in °/meter)

	Datasets	HBA	SLAMesh	PIN-SLAM	Ours
Newer College	quad_e	**0.978**/0.087	1.337/**0.084**	0.986/0.104	1.225/0.086
	math_e	0.440/**0.073**	1.010/0.116	**0.420**/0.080	0.708/**0.073**
	ug_e	1.628/**0.074**	4.120/0.262	**1.567/0.074**	5.587/0.180
	cloister*	1.277/0.137	3.140/0.242	**1.249/0.134**	3.200/0.212
	stairs*	10.422/1.078	N/A	**1.392/0.075**	5.320/0.169
	parkland*	3.216/0.288	3.139/0.486	**1.261**/0.345	3.420/**0.271**
FusionPortable	canteen_night	**0.685/0.078**	0.693/**0.078**	0.700/0.109	0.702/0.081
	garden_night	0.660/**0.038**	1.412/0.077	0.758/0.091	**0.640**/0.040
	MCR_fast	**14.087/0.168**	14.539/0.195	14.120/0.173	14.551/0.184
	MCR_normal	28.668/0.116	28.773/0.114	**28.615/0.107**	28.737/0.115
	MCR_slow	17.132/0.088	**16.962/0.061**	17.122/0.077	17.262/0.096
	corridor_day	**1.086/0.125**	8.767/3.000	1.232/0.189	2.570/0.304
	escalator_day	**1.453/0.181**	2.733/0.186	1.461/0.186	4.875/0.187
	building_day	1.473/0.154	**1.284/0.131**	1.568/0.184	1.383/0.141

The best results are highlighted in **bold**, while the second-best results are underlined. N/A indicates a failure.
* represents the sequences that using all the available associated patches for pose estimation in our method.

is reached, additional factors, such as the underlying map representation, may exert a greater influence on mapping quality. A thorough exploration of this hypothesis lies beyond the scope of the present work, but it nonetheless suggests that the map representation is needing for more attention, and partially motivates this work.

SLAMesh, while performing well in several sequences (e.g., `canteen_night`, `building_day`), shows inconsistent performance, particularly due to the lack of loop closure, as evidenced in the `corridor_day` sequence. PIN-SLAM achieves stable and robust results across all sequences, demonstrating consistent performance. Notably, our algorithm requires all associated patches for localization in scenarios with narrow spaces and limited FoV, like `cloister` and `stairs` sequences.

G. Effect of Local Bundle Adjustment

Additionally, to further evaluate the effect of BA, we compare ATE results on the same runs by disabling and enabling local BA for sequences where loop closures are detected. The quantitative results, presented in TABLE VIII, highlight

TABLE VIII: ATE (m) with/without Local BA

Sequences	Pose-Graph Only	With Local BA
math_e	0.072	**0.067**
ug_e	0.078	**0.077**
cloister	0.205	**0.204**
stairs	0.509	**0.372**
parkland	0.240	**0.234**
corridor_day	0.168	**0.167**
escalator_day	0.195	**0.193**
building_day	0.121	**0.120**

The best results are highlighted in **bold**.

the impact of local BA on trajectory accuracy. As shown in TABLE VIII, enabling local BA consistently improves or maintains trajectory accuracy across all evaluated sequences. The improvement is especially notable in more challenging scenarios such as `stairs`, where the ATE is significantly reduced from 0.509 m to 0.372 m. These results demonstrate the effectiveness of local BA in refining trajectory estimates after loop closure, particularly in complex environments with narrow spaces or limited field of view.

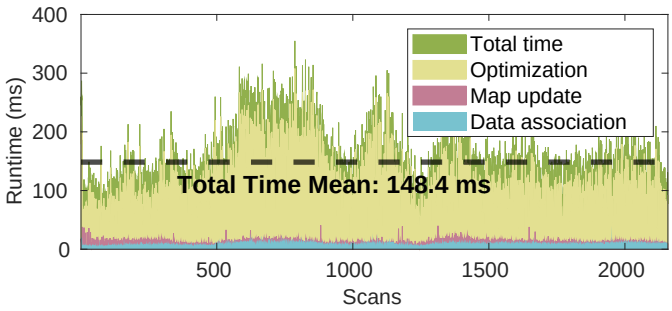
H. Evaluation on Runtime

Fig. 17(a) shows our runtime performance using the default parameters. All modules run on a single thread, except for the map update, which uses 16 threads, achieving 6.74 Hz on a CPU. By setting $\omega = 16$, $\beta_{ng} = 15$, and $\beta_g = 10$ to reduce the patch resolution and the limit on associated pairs, achieving real-time performance of 14.3Hz, as shown in Fig. 17(b), while maintaining similar odometry performance. This test was conducted on a 128-channel LiDAR; fewer channels would further reduce processing time.

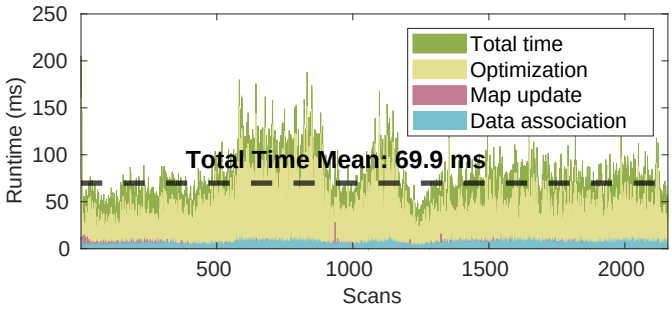
Both PIN-SLAM and our proposed method are capable of achieving globally consistent map reconstruction at multiple resolutions and encoding the map into a compact representation. However, PIN-SLAM is limited to real-time performance only when utilizing certain GPU architectures. Additionally, extracting mesh maps for large-scale regions with PIN-SLAM demands substantial amounts of RAM memory, making it impractical for embedded systems. Although PIN-SLAM can be operated on a CPU, it requires processing times exceeding two seconds per scan, which restricts its applicability in real-time and resource-constrained environments. In contrast, our method offers more flexibility and efficiency, particularly in scenarios where hardware resources are limited.

VII. Conclusions

This work proposes a novel LiDAR mapping system, CURL-SLAM, using the CURL representation with the full pipeline of odometry, local bundle adjustment and loop closure. It is capable of continuously generating 3D reconstructions at a desired resolution. To the best of our knowledge, our method achieves the most efficient map size in dense online LiDAR SLAM systems, with the capability of continuous dense 3D reconstruction. Our experiments demonstrate that the proposed CURL-SLAM method achieves the state-of-the-art mapping performance and comparable localization accuracy.



(a) Full version (0.322% and 0.818°).



(b) Light version (0.330% and 0.797°).

Fig. 17: Runtime evaluation of CURL-SLAM on `math_e`, with relative translation and rotation errors in brackets.

Finally, our system can operate in real-time on a single Intel i7-10875H CPU.

In future work, we will investigate the integration of Inertial Measurement Unit with CURL representation, aiming to improve the robustness of pose estimation in environments with limited field of view.

Appendix A

This appendix provides the terms in (3), (8), (17), and (18) that were not given in the text.

$$N_m(\phi) = \begin{cases} \sqrt{2}cos(m\phi), & m > 0 \\ 1, & m = 0 \\ \sqrt{2}sin(|m|\phi), & m < 0 \end{cases} \tag{24}$$

$$\mathbf{J}^{\mathsf{P}_z} = [0 \quad 0 \quad 1] \tag{25}$$

The numerical jacobian for $\mathbf{I}^k$ is

$$\mathbf{J}^{\mathbf{I}^k} = \begin{bmatrix} \mathbf{J}^{\mathbf{I}^k}_x & \mathbf{J}^{\mathbf{I}^k}_y \end{bmatrix} \tag{26}$$

where

$$\mathbf{J}^{\mathbf{I}^k}_x \approx \frac{\mathbf{I}^k([x+\delta, y]^T) - \mathbf{I}^k([x-\delta, y]^T)}{2\delta}$$

$$\mathbf{J}^{\mathbf{I}^k}_y \approx \frac{\mathbf{I}^k([x, y+\delta]^T) - \mathbf{I}^k([x, y-\delta]^T)}{2\delta}$$

and δ is a small constant.

$$\mathbf{J}^{\mathsf{P}_{xy}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{27}$$

$$\mathbf{J}^{\mathsf{P}_x} = [1 \quad 0] \qquad \mathbf{J}^{\mathsf{P}_y} = [0 \quad 1] \tag{28}$$

$$\mathbf{J}^{\phi} = \frac{2\pi\eta}{s} \qquad \mathbf{J}^{\theta} = \frac{\pi\eta}{s} \tag{29}$$

APPENDIX B

Lemma 1. For a transformation matrix $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3) \subset \mathbb{R}^{4\times4}$, which acts on a point $\mathbf{p} \in \mathbb{R}^{3\times1}$, such that $\mathbf{T}\cdot\mathbf{p} = \mathbf{R}\mathbf{q} + \mathbf{t}$. The Jacobian matrix $\mathbf{J}_{\mathbf{T}}^{\mathbf{T}\cdot\mathbf{q}}$ is given by:

$$\mathbf{J}_{\mathbf{T}}^{\mathbf{T}\cdot\mathbf{q}} = [\mathbf{R} \quad -\mathbf{R}[\mathbf{p}]_\times] \tag{30}$$

where $[\cdot]_\times$ is the skew-symmetric matrix of the vector.

Lemma 2. For the transformation matrices $\mathbf{T}_1 = \begin{bmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3) \subset \mathbb{R}^{4\times4}$ and $\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_2 & \mathbf{t}_2 \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3) \subset \mathbb{R}^{4\times4}$, which act on a point $\mathbf{p} \in \mathbb{R}^{3\times1}$, the Jacobian matrix $\mathbf{J}_{\mathbf{T}_2}^{\mathbf{T}_1\mathbf{T}_2^{-1}\cdot\mathbf{p}}$ is given by:

$$\mathbf{J}_{\mathbf{T}_2}^{\mathbf{T}_1\mathbf{T}_2^{-1}\cdot\mathbf{p}} = \begin{bmatrix} -\mathbf{R}_1 & \mathbf{R}_1[\mathbf{T}_2^{-1}\cdot\mathbf{p}]_\times \end{bmatrix} \tag{31}$$

Lemma 3. For function $Y_{l,m}(\theta,\phi)$, its Jacobian matrix $\mathbf{J}^{y_{l,m}}$ is

$$\mathbf{J}^{Y_{l,m}} = \begin{bmatrix} \mathbf{J}_\phi^{Y_{l,m}} & \mathbf{J}_\theta^{Y_{l,m}} \end{bmatrix} \in \mathbb{R}^{1\times2} \tag{32}$$

$$\mathbf{J}_\phi^{Y_{l,m}} = \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}P_{l,|m|}(\cos\theta)\cdot \begin{cases} -m\sqrt{2}\sin(m\phi), & m>0 \\ 0, & m=0 \\ -m\sqrt{2}\cos(m\phi), & m<0 \end{cases} \tag{33}$$

$$\mathbf{J}_\theta^{Y_{l,m}} = \begin{cases} mcot\theta Y_{l,m}(\theta,\phi) - \\ \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}N_m(\phi)P_{l,m+1}(\cos\theta), & m>0 \\ \frac{-\sqrt{l(l+1)}}{N_1(\phi)}Y_{l,1}(\theta,\phi) \ or \\ -\sqrt{\frac{2l+1}{4\pi}}P_{l,1}(cos\theta), & m=0 \\ -mcot\theta Y_{l,m}(\theta,\phi) - \\ \sqrt{\frac{2l+1}{4\pi}\frac{(l+m)!}{(l-m)!}}N_m(\phi)P_{l,1-m}(cos\theta), & m<0 \end{cases} \tag{34}$$

(34) is derived according to (35) from [67].

$$sin\theta\frac{dP_{l,m}}{d\theta} = mcos\theta P_{l,m} - sin\theta P_{l,m+1} \tag{35}$$

REFERENCES

[1] K. Zhang, Z. Hong, S. Xu, and S. Wang, "CURL: Continuous, ultra-compact representation for LiDAR," in *Robotics: Science and Systems*, 2022.

[2] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.

[3] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2018.

[4] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2020.

[5] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Robotics: Science and Systems*, 2015.

[6] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson surface reconstruction for LiDAR odometry and mapping," in *IEEE International Conference on Robotics and Automation*. IEEE, 2021.

[7] J. Ruan, B. Li, Y. Wang, and Y. Sun, "SLAMesh: Real-time LiDAR simultaneous localization and meshing," in *IEEE International Conference on Robotics and Automation*. IEEE, 2023.

[8] J. Lin, C. Yuan, Y. Cai, H. Li, Y. Ren, Y. Zou, X. Hong, and F. Zhang, "ImMesh: An immediate LiDAR localization and meshing framework," *IEEE Transactions on Robotics*, 2023.

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

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

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

[12] S. Isaacson, P.-C. Kung, M. Ramanagopal, R. Vasudevan, and K. A. Skinner, "LONER: LiDAR only neural representations for real-time SLAM," *IEEE Robotics and Automation Letters*, 2023.

[13] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss, "PIN-SLAM: LiDAR SLAM using a point-based implicit neural representation for achieving global map consistency," *IEEE Transactions on Robotics*, 2024.

[14] K. Zhang, Y. Ding, S. Xu, Z. Hong, X. Kong, and S. Wang, "CURL-MAP: Continuous mapping and positioning with CURL representation," in *IEEE International Conference on Robotics and Automation*. IEEE, 2024.

[15] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed Distance Fields: A natural representation for both mapping and planning," in *RSS 2016 Workshop: Geometry and Beyond-representations, Physics, and Scene Understanding for Robotics*, 2016.

[16] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-blox: A scalable and consistent TSDF-based dense mapping approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2018.

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *10th IEEE International Symposium on Mixed and Augmented Reality*, 2011.

[18] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[19] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. Kelly, and S. Leutenegger, "Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters*, 2018.

[20] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, 2019.

[21] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robotics and Automation Letters*, 2019.

[22] A. J. Millane, H. Oleynikova, C. Lanegger, J. Delmerico, J. Nieto, R. Siegwart, M. Pollefeys, and C. C. Lerma, "Freetures: Localization in signed distance function maps," *IEEE Robotics and Automation Letters*, 2021.

[23] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2017.

[24] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "VDBFusion: Flexible and efficient TSDF integration of range sensor data," *Sensors*, 2022.

[25] X. Zhong, Y. Pan, J. Behley, and C. Stachniss, "SHINE-Mapping: Large-scale 3D mapping using sparse hierarchical implicit neural representations," in *IEEE International Conference on Robotics and Automation*, 2023.

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

[27] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, 2023.

[28] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, 2012.

[29] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, 2016.

[30] P. Z. X. Li, S. Karaman, and V. Sze, "GMMap: Memory-efficient continuous occupancy map using Gaussian mixture model," *IEEE Transactions on Robotics*, 2024.

[31] K. Goel and W. Tabib, "GIRA: Gaussian mixture models for inference and robot autonomy," in *IEEE International Conference on Robotics and Automation*, 2024.

[32] K. Goel, N. Michael, and W. Tabib, "Probabilistic point cloud modeling via self-organizing Gaussian mixture models," *IEEE Robotics and Automation Letters*, 2023.

[33] J. Zhao, T. Li, T. Yang, L. Zhao, and S. Huang, "2D laser SLAM with closed shape features: Fourier series parameterization and submap joining," *IEEE Robotics and Automation Letters*, 2021.

[34] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous robots*, 2013.

[35] V. Reijgwart, C. Cadena, R. Siegwart, and L. Ott, "Efficient volumetric mapping of multi-scale environments using wavelet-based compression," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[36] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[37] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "iSDF: Real-time neural signed distance fields for robot perception," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[38] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "NICE-SLAM: Neural implicit scalable encoding for SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern recognition*, 2022.

[39] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, "Point-SLAM: Dense neural point cloud-based SLAM," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[40] J.-E. Deschaud, P. Dellenbach, B. Jacquet, and F. Goulette, "CT-ICP: Real-time elastic LiDAR odometry with loop closure," 2021.

[41] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way," *IEEE Robotics and Automation Letters*, 2023.

[42] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Transactions on Robotics*, 2022.

[43] Z. Liu and F. Zhang, "BALM: Bundle adjustment for LiDAR mapping," *IEEE Robotics and Automation Letters*, 2021.

[44] Z. Liu, X. Liu, and F. Zhang, "Efficient and consistent bundle adjustment on LiDAR point clouds," *IEEE Transactions on Robotics*, 2023.

[45] X. Liu, Z. Liu, F. Kong, and F. Zhang, "Large-scale LiDAR consistent mapping using hierarchical LiDAR bundle adjustment," *IEEE Robotics and Automation Letters*, 2023.

[46] C. Park, P. Moghadam, J. L. Williams, S. Kim, S. Sridharan, and C. Fookes, "Elasticity meets continuous-time: Map-centric dense 3D LiDAR SLAM," *IEEE Transactions on Robotics*, 2021.

[47] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic LiDAR fusion: Dense map-centric continuous-time SLAM," in *IEEE International Conference on Robotics and Automation*. IEEE, 2018.

[48] R. Dube, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research*, 2020.

[49] C. Xia, C. Xu, P. Rim, M. Ding, N. Zheng, K. Keutzer, M. Tomizuka, and W. Zhan, "Quadric representations for LiDAR odometry, mapping and localization," *IEEE Robotics and Automation Letters*, 2023.

[50] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online LiDAR odometry," *IEEE Robotics and Automation Letters*, 2022.

[51] C. Wu, Y. You, Y. Yuan, X. Kong, Y. Zhang, Q. Li, and K. Zhao, "VoxelMap++: Mergeable voxel mapping method for online LiDAR-inertial odometry," *IEEE Robotics and Automation Letters*, 2023.

[52] Z. Yu, Z. Qiao, W. Liu, H. Yin, and S. Shen, "SLIM: Scalable and lightweight LiDAR mapping in urban environments," *IEEE Transactions on Robotics*, 2025.

[53] J.-E. Deschaud, "IMLS-SLAM: Scan-to-model matching based on 3D data," in *IEEE International Conference on Robotics and Automation*, 2018.

[54] S. Lee, H. Lim, and H. Myung, "Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3D point cloud," in *IEEE Robotics and Automation Letters*, 2022.

[55] T. W. Meng, G. P.-T. Choi, and L. M. Lui, "TEMPO: Feature-endowed Teichmüller extremal mappings of point clouds," *SIAM Journal on Imaging Sciences*, 2016.

[56] L. Shen and M. K. Chung, "Large-scale modeling of parametric surfaces using spherical harmonics," in *Third International Symposium on 3D Data Processing, Visualization, and Transmission*. IEEE, 2006.

[57] C. Müller, *Spherical harmonics*. Springer, 2006.

[58] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013.

[59] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation*. IEEE, 2014.

[60] D. Luo, Y. Zhuang, and S. Wang, "Hybrid sparse monocular visual odometry with online photometric calibration," *The International Journal of Robotics Research*, 2022.

[61] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2018.

[62] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. H. Gross, "Optimized spatial hashing for collision detection of deformable objects." in *Vmv*, 2003.

[63] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-camera LiDAR inertial extension to the newer college dataset," *arXiv preprint arXiv:2112.08854*, 2021.

[64] J. Jiao, H. Wei, T. Hu, X. Hu, Y. Zhu, Z. He, J. Wu, J. Yu, X. Xie, H. Huang *et al.*, "FusionPortable: A multi-sensor campus-scene dataset for evaluation of localization and mapping accuracy on diverse platforms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2022.

[65] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in *IEEE International Conference on Robotics and Automation*. IEEE, 2021.

[66] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE transactions on visualization and computer graphics*, 1999.

[67] W. Bosch, "On the computation of derivatives of Legendre functions," *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy*, 2000.