Mesh-LOAM: Real-time Mesh-Based LiDAR Odometry and Mapping

Yanjin Zhu, Xin Zheng, and Jianke Zhu *Senior Member, IEEE*

***Abstract*—Although having achieved real-time performance on mesh construction, most of the existing LiDAR odometry and meshing methods have difficulties in dealing with cluttered scenes due to relying on explicit meshing techniques. To overcome these limitations, we propose a real-time mesh-based LiDAR odometry and mapping approach for large-scale scenes through implicit reconstruction and a parallel spatial-hashing scheme. In order to efficiently reconstruct the triangular meshes using the implicit function, we suggest an incremental voxel meshing strategy that depends on a novel voxel map fusing scans through a single traversal of the current frame. Moreover, we introduce a scalable partition module to compress space. By taking advantage of the rapid access to triangular meshes, we design a robust odometry method with location and feature-based data association to estimate the poses between the input point clouds and the recovered triangular meshes. The experimental results on four public datasets demonstrate the effectiveness of our proposed approach in recovering both accurate motion trajectories and environmental mesh maps.**

***Index Terms*—LiDAR Odometry and Mapping, Point-to-Mesh ICP, Triangular Mesh, Parallel Spatial Hashing.**

I. Introduction

LiDAR odometry and mapping (LOAM) serves as a fundamental component in robotic applications [1], [2]. It aims to estimate the 6-DoF poses of moving sensors while continuously constructing the surrounding environment. Generally, a representative and compact map structure can not only enhance the real-time capability of odometry but also enable seamless integration with downstream tasks. Unfortunately, the prevalent map representations, such as point clouds [3], [4], [5], grid maps [6], [7] and surfels [8], [9] cannot fulfill these requirements. Specifically, storing point clouds in large-scale scenes requires a substantial amount of memory. Although grid maps and surfels have more compact map representations, they lack connectivity among their elemental units. Such discontinuity poses significant challenges to directly employ them in robotic applications, which usually requires complicated and resource-intensive post-processing techniques [10].

In contrast, triangular mesh comprises a collection of vertices and triangular facets, which offers a concise and accurate depiction of outdoor environments. It is widely adopted in dense map generation [12], pose estimation [13], [14], loop closure [15], and collision detection [16]. Mesh map provides a smooth continuous surface representation, which enhances robot navigation and decision making in various environments. In spite of some previous studies like [13], [14], focusing on mesh-based LOAM, real-time meshing in large-scale environments remains a challenging problem. For instance, implicit reconstruction like Poisson surface reconstruction [13] is utilized to construct detailed and accurate smoothing surface meshes of outdoor scenes. However, its performance falls short of real-time requirements. Recent studies [14], [17] make use of either efficient hierarchical sparse data structures or techniques to achieve real-time meshing. They employ explicit methods like Delaunay triangulation [17] or the Gaussian Process [14] to incrementally construct a mesh map. However, these explicit approaches are often noise-sensitive and typically demand high-quality input data. Moreover, their complexity increases when the target scenario is complicated or deviates from the Manhattan assumptions.

To address the above limitations, we introduce Mesh-LOAM, a robust LiDAR odometry and meshing approach

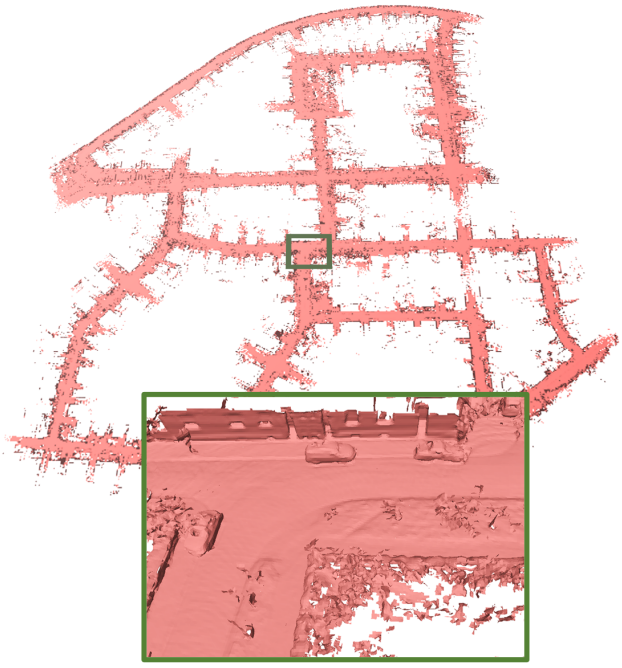


Fig. 1: Our odometry and mapping result on sequence '00' of the KITTI odometry dataset [11]. Our proposed Mesh-LOAM approach accurately estimates the poses of a moving vehicle while recovering a precise and compact mesh of the large-scale outdoor scene. The whole algorithm runs at around 54 frames per second.

All authors are with the College of Computer Science and Technology, Zhejiang University, Zheda Road 38th, Hangzhou, China.
E-mail: {yanjinzhu,xinzheng,jkzhu}@zju.edu.cn
Jianke Zhu is the Corresponding Author. This work is supported by National Natural Science Foundation of China under Grants (62376244).

based on an efficient voxel map, which improves the geometrical accuracy of LiDAR-based mapping and obtains low drift of pose estimation simultaneously. Specifically, we adopt an implicit function for the reconstruction, which is robust to noise and able to handle the cluttered scenarios. For the accurate and efficient reconstruction of large-scale scenes, we present a novel voxel map under a parallel spatial-hashing scheme for 3D meshing, in which the passive computational model for SDF value and scalable partition module facilitate the efficient computation. Fig. 1 shows an example result of our presented Mesh-LOAM method with a triangular mesh map on the KITTI odometry dataset.

The main contributions of this paper are as follows: 1) a novel voxel map for 3D meshing that allows for accurate and efficient implicit reconstruction of large-scale scenes; 2) a robust point-to-mesh LOAM approach using the proposed voxel map, which achieves competitive results among the state-of-the-art methods; 3) experiments across diverse outdoor datasets demonstrate that Mesh-LOAM obtains the accurate pose estimation and recovers the promising triangular meshes in real-time.

II. RELATED WORK

A. Point-based LiDAR Odometry and Mapping

Point-based LOAM method performs localization with point clouds or dense features [8], [18] in LiDAR-based systems. Pioneering work in LiDAR odometry and mapping has been achieved by LOAM [3], where edge features and plane features are extracted to calculate point-to-line and point-to-plane distances. Many subsequent works [4], [19], [20], [21] inspired by LOAM concentrate on enhancing the efficiency and accuracy of LiDAR odometry. FLOAM [4] is an optimized framework for LiDAR-based Simultaneous Localization and Mapping (SLAM) and provides a good trade-off between performance and computational cost. Recently, KISS-ICP [5] achieves the promising performance using a robust point-to-point ICP. While these approaches achieve reliable odometry, they rely on scattered point clouds as mapping representation and overlook the structured information.

In addition to point cloud map, Deschaud [19] proposes an improved point cloud matching method with Implicit Moving Least Squares (IMLS) surface representation. Behley and Stachniss [8] introduce a surfel-based map to represent the large-scale environment and perform projective data association of the current scan to the surfel maps. Yuan et al. [18] present an efficient, probabilistic adaptive voxel mapping method for online LiDAR odometry. Although these representations are more compact than point clouds, the lack of connectivity in their maps prevents smooth integration with subsequent tasks. To this end, triangular meshes are selected as our mapping representation, which has the merit of a continuous surface map.

Lin et al. [17] introduce a novel meshing framework and utilize Voxel Map [18] for the localization. The localization module operates independently from their mapping process. In contrast, our proposed approach performs localization against the triangular meshes and enables localization and meshing to benefit each other.

B. Mesh-based LiDAR Odometry and Mapping

This paper emphasizes mesh-based LiDAR odometry and mapping methods, which employ meshes to represent the surrounding environment and facilitate localization. Vizzo et al. [13] demonstrate the applicability of triangular meshes for incremental scan registration in pose estimation. They represent the map as triangular meshes estimated using Poisson surface reconstruction and utilize ray casting data association for point-to-mesh ICP with an octree. However, this incurs significant computational overhead and limits its capability to handle large rotational movements. To avoid implicit reconstruction in large scenarios with memory and speed limitations, Ruan et al. [14] use Gaussian process regression [22] and construct the mesh by directly connecting the adjacent vertices. They achieve real-time performance by making use of a cell-based hash map that can be efficiently processed in parallel. Although SLAMesh is a novel real-time localization and meshing system, it becomes significantly complicated in modeling complex geometries. Furthermore, the accuracy of these point-to-mesh methods is inferior to approaches [4], [5] that directly implement point-to-point or point-to-plane constraints. To address these limitations, our proposed approach leverages a novel voxel map with a spatial-hashing scheme and a passive computational model to obtain real-time implicit reconstruction. By registering LiDAR scans to triangular meshes, our method achieves odometry accuracy comparable to that of the state-of-the-art methods [4], [5].

C. Offline LiDAR Reconstruction

Various kinds of implicit functions, i.e., radial basis function and its variants [23], [24], tangent plane [25], signed distance function (SDF) [26], truncated signed distance function (TSDF) [27], [28], are utilized to approximate the underlying surface. These implicit methods, including Implicit Moving Least Squares (IMLS), have proven to be computationally intensive or constrained by small scale, which hinders their deployment in real-time applications with a large number of LiDAR scans.

Alternatively, voxel-based methods make use of the voxel structure to retain the completeness of point set information. Initially, Curless and Levoy [27] incrementally construct a TSDF map at the object level. Fuhrmann et al. [29] advance the reconstruction scale to the building level by constructing a hierarchical signed distance field. Nevertheless, it takes considerable time to obtain the reconstruction results. On the other hand, Kühner et al. [12] present a volumetric depth fusion using TSDF for large-scale outdoor mapping. However, they require multiple passes over the same scene for competitive reconstruction results. Since these approaches scale to outdoor scene reconstruction through hierarchical data structures, it is hard for them to achieve real-time performance.

Learning-based methods [26], [30], [31] typically incorporate shallow neural networks to model large-scale 3D scenes. Ortiz et al. [26] introduce a neural SDF reconstruction method. Despite achieving encouraging mapping results, these methods possess certain limitations. For instance, they struggle to achieve real-time performance and rely on prior knowledge of

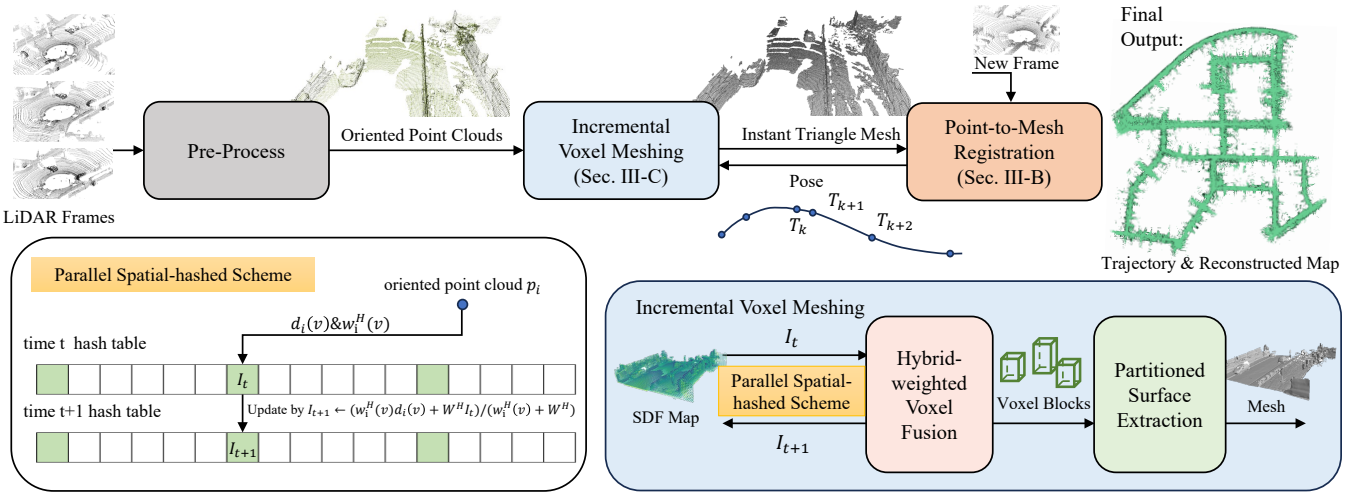


Fig. 2: Overview of our proposed Mesh-LOAM approach. Firstly, the oriented point clouds are obtained by estimating normals via Principal Component Analysis (PCA). Secondly, we employ incremental voxel meshing to generate instant triangular meshes. Finally, point-to-mesh odometry is performed for the new LiDAR frames and the reconstructed triangular meshes to estimate the pose $\mathbf{T}_k$. The hybrid-weighted voxel fusion is based on a parallel spatial-hashing scheme, which updates the SDF value I_{t+1} of the oriented point cloud p_i by weighting the distance $d_i(v)$ and the previous SDF value I_t in the corresponding voxel. Parallel retrieval for height-adaptive voxel blocks allows partitioned surface extraction.

the target scene. Additionally, the size of bounding box needs to be set appropriately. Our proposed approach addresses these challenges by introducing the sparse voxel fusion method and partitioned height-adaptive voxel blocks.

D. Online LiDAR Reconstruction

Oleynikova et al. [32] employ voxel hashing [33] as the underlying data structure to enable dynamically growing maps, incrementally estimating Euclidean Signed Distance Fields (ESDFs) from TSDFs. However, it may perform inferior in reconstructing large-scale scenes. Vizzo et al. [28] develop a CPU-based flexible and effective mapping system. Niedzwiedzki et al. [34] introduce an incremental triangular mesh generation algorithm. Nevertheless, these methods rely on off-the-shelf odometry methods for trajectory information.

III. Mesh-based LiDAR Odometry and Mapping

A. Overview

In this section, we suggest a real-time approach to large-scale mesh-based LiDAR odometry and mapping. Fig. 2 illustrates the overview of our proposed pipeline. Firstly, we present point-to-mesh odometry with a location and feature-based data association module to estimate poses. Secondly, we propose an efficient voxel meshing method that takes advantage of sparse voxels to incrementally reconstruct surface meshes. Finally, we introduce a simple and effective parallel spatial hash-based scheme to efficiently retrieve the voxel.

B. Point-to-Mesh LiDAR Odometry

We adopt a point-to-mesh registration similar to Puma [13] and SLAMesh [14], which can be utilized to improve the accuracy of the odometry. Note that Puma [13] employs ray casting to build point-mesh correspondences while SLAMesh [14] is based on locations for data association. Instead, our proposed Mesh-LOAM approach introduced an effective location and feature-based strategy. The management of triangular meshes is described in Section III-C and Section III-D. Since scan-to-model matching usually performs better than the classical scan-to-scan matching [8], [9], [20], our mesh representations are computed from consecutively accumulated scans.

1) Planar Feature Selection: In the proposed point-to-mesh registration framework, we select planar points to facilitate accurate pose estimation. The process involves estimating the surface normal of a 3D point $\mathbf{p} = (x, y, z)^\top$ by fitting a local plane within the current LiDAR scan. In this context, we assume a plane in 3D Euclidean space is characterized by a normal vector $\mathbf{n} = (n_x, n_y, n_z)^\top$ and a scalar value d, satisfying the condition $\mathbf{p}^\top \mathbf{n} = d$. The computation of the normal vector for each point is achieved by solving the following equation:

$$e = \sum_{i=1}^{k} (\mathbf{p}_i^\top \mathbf{n} - d)^2 \quad s.t. \quad \mathbf{n}^\top \mathbf{n} = 1. \tag{1}$$

Minimizing the above constrained least squared problem can be formulated into performing principal component analysis on the covariance matrix constructed from the neighborhoods of each point $\mathbf{p}_i$. The covariance matrix C is computed by

$$C = \frac{1}{k} \sum_{i=1}^{k} (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^\top, \tag{2}$$

where $\bar{\mathbf{p}} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{p}_i$, $k \geq 5$. The eigenvectors $\mathbf{v_i}$ are estimated by $C\mathbf{v_i} = \lambda_i \mathbf{v_i}, i \in \{0, 1, 2\}$, where eigenvalues $\lambda_0, \lambda_1, \lambda_2$ are in the ascending order. The normal vector is

aligned with the eigenvector corresponding to the smallest eigenvalue λ_0. For consistency in the direction of normal vectors, we apply $\mathbf{n} = -\mathbf{n}$ if $\mathbf{n}^\top(\mathbf{p}_i - \mathbf{p}_v) < 0$, where $\mathbf{p}_v$ refers to the viewpoint.

To select the salient planar points, the surface curvature $c_{\mathbf{p}_i}$ of point $\mathbf{p}_i$ serves as a measure for evaluating the smoothness of the local surface. The surface curvature is defined as $c_{\mathbf{p}_i} = \frac{\lambda_0}{\lambda_0+\lambda_1+\lambda_2}$, where $\lambda_0 < \lambda_1 < \lambda_2$. We choose the planar feature point whose curvature is below the user-defined threshold c_{th}.

2) Data Association: The objective of our odometry approach is to estimate the 6-DoF pose $\mathbf{T}_k \in \mathbb{R}^{4\times4} \in \mathrm{SE}(3)$ of the current frame k with respect to the global mesh model. To facilitate the smoothness of the robot's motion, we use the constant velocity model to initialize the predicted pose $\widehat{\mathbf{T}}_k$. Given the previous estimated poses $\mathbf{T}_{k-2}$ and $\mathbf{T}_{k-1}$, the predicted pose of current frame is computed by $\widehat{\mathbf{T}}_k = \mathbf{T}_{k-1}(\mathbf{T}_{k-2}^{-1}\mathbf{T}_{k-1})$.

For the incoming frame $\mathcal{P}_k$, point clouds $\mathbf{p}_i \in \mathcal{P}_k$ are transformed to the estimated global coordinate system $\mathbf{p}_w = \widehat{\mathbf{T}}_k\mathbf{p}_i \in \mathcal{P}_w$ based on the predicted pose $\widehat{\mathbf{T}}_k$. For every triangular facet $\mathcal{S}_i$, we calculate its normal vectors $\mathbf{n}_{\mathcal{S}_i}$ by taking the cross product of its vertices.

During each iteration of the point-to-mesh ICP process, the correspondences between the point cloud $\mathbf{p}_w$ and triangular mesh $\mathcal{S}$ are established through the nearest neighbor search. Moreover, $\mathbf{v}_i$ represents the point on the triangular facet $\mathcal{S}_i$. To ensure the correct correspondences, two criteria are employed to filter outliers. Firstly, the point-to-mesh distance $|\mathbf{n}_{\mathcal{S}_i}^\top(\mathbf{p}_w - \mathbf{v}_i)|$ is utilized to determine the corresponding facet for the point $\mathbf{p}_w$. Besides, a threshold c_s is set to constrain the absolute value of the cosine similarity $\frac{|\mathbf{n}_{\mathcal{S}_i}^\top\mathbf{n}_w|}{\|\mathbf{n}_{\mathcal{S}_i}\|\cdot\|\mathbf{n}_w\|}$ between the normals of the point and facet. Given the uncertainty in the orientation of the normal vectors, the absolute value of the cosine similarity is utilized to assess the alignment between their normals. Based on this metric for selecting correspondences, we choose matches whose absolute value of the cosine similarity exceeds the predefined threshold c_s.

3) Pose Optimization: To achieve more effective convergence during the optimization process, we concentrate on estimating the relative pose $\mathbf{T}_{icp,k}$ instead of directly computing the global pose $\mathbf{T}_k$. $\mathbf{T}_{icp,k}$ represents the deviation between prediction frame $\mathcal{P}_w$ and global triangular meshes. Therefore, we aim to minimize the point-to-mesh error in the following:

$$\mathbf{T}_{icp,k} = \underset{\mathbf{T}_{icp,k}}{\arg\min} \sum_{(\mathbf{p}_w,\mathbf{v}_{\mathcal{S}_i})\in\mathcal{C}} \|\mathbf{n}_{\mathcal{S}_i}^\top(\mathbf{T}_{icp,k}\mathbf{p}_w - \mathbf{v}_{\mathcal{S}_i})\|_2, \quad (3)$$

where $\mathcal{C}$ is the set of appropriate correspondences between point and meshes, $\mathbf{n}_{\mathcal{S}_i}$ is the normal of triangular facet, and $\mathbf{v}_{\mathcal{S}_i}$ is a point cloud on the triangular facet $\mathcal{S}_i$. Our triangular mesh is the zero set of $I^{\mathcal{P}_k}$ obtained by linear interpolation. When considering each point-to-mesh residual separately, any point on the triangular facet can be chosen as $\mathbf{v}_{\mathcal{S}_i}$. However, arbitrary selection of point $\mathbf{v}_{\mathcal{S}_i}$ may result in slow convergence of the optimization objective $\mathbf{T}_{icp,k}$. To ensure the convergence of the optimization process, we choose the point cloud on the plane as the $\mathbf{v}_{\mathcal{S}_i}$.

Equ. 3 indicates a nonlinear least squared minimization problem, which is typically solved through an iterative Gauss-Newton algorithm. The error term is defined as $e = \mathbf{n}_{\mathcal{S}_i}^\top(\mathbf{T}_{icp,k}\mathbf{p}_w - \mathbf{v}_{\mathcal{S}_i})$, and the Jacobian $\mathbf{J}$ is computed as follows:

$$\mathbf{J} = \frac{\partial e}{\partial \mathbf{T}} = \mathbf{n}_{\mathcal{S}_i}^\top \begin{bmatrix} \mathbf{I} & -[\mathbf{T}_{icp,k}\mathbf{p}_w]_\times \end{bmatrix}, \quad (4)$$

where $[\cdot]_\times$ represents the skew symmetric matrix of a vector. For each iteration j, the pose increment $\Delta\mathbf{T}_{icp,k}^j \in \mathbb{R}^6$ is computed by $\Delta\mathbf{T}_{icp,k}^j = (\mathbf{J}^\top\mathbf{J})^{-1}\mathbf{J}^\top e$, with $\mathbf{T}_{icp,k}^{j-1}$ representing the pose $\mathbf{T}_{icp,k}$ from iteration $j-1$. The relative pose $\mathbf{T}_{icp,k}^j$ is then updated as $\mathbf{T}_{icp,k}^j = \mathrm{Exp}(\Delta\mathbf{T}_{icp,k}^j)\mathbf{T}_{icp,k}^{j-1}$, where $\mathrm{Exp}(\cdot)$ is mapping function from $\mathbb{R}^6$ to $\mathrm{SE}(3)$ [35]. When the increment is below a predefined threshold, the optimization converges and the final relative pose is $\mathbf{T}_{icp,k} = \mathbf{T}_{icp,k}^j$, where j means the final iteration. Then, the global pose is obtained by $\mathbf{T}_k = \mathbf{T}_{icp,k}\widehat{\mathbf{T}}_k$, which transforms the current scan into the global coordinate system for incremental voxel meshing.

C. Incremental Voxel Meshing

To achieve real-time mapping for large-scale environments, we propose a two-stage incremental voxel meshing method. Firstly, an efficient hybrid-weighted voxel fusion scheme is introduced, which uses sparse voxels to preserve global map information. Secondly, we leverage height-adaptive voxel blocks to compress space and efficiently extract surface mesh.

1) Implicit Function: We take the Implicit Moving Least Squares (IMLS), an implicit function $f : \mathbb{R}^3 \to \mathbb{R}$, to approximate the underlying surface. The recovered surface is implicitly defined as the zero level-set $Z(f)$. For each voxel $\mathbf{v} = (x, y, z)^\top$, the IMLS [24] function is defined as below

$$I^{\mathcal{P}_k}(\mathbf{v}) = \frac{\sum_{\mathbf{p}_i\in\mathcal{P}_k} w_i^H(\mathbf{v})(\mathbf{n}_i^\top(\mathbf{v} - \mathbf{p}_i))}{\sum_{\mathbf{p}_i\in\mathcal{P}_k} w_i^H(\mathbf{v})}, \quad (5)$$

where $\mathcal{P}_k$ is the point cloud map of the current frame k. $\mathbf{p}_i \in \mathcal{P}_k$ is a point. $w_i^H(\mathbf{v})$ is hybrid weight, and $\mathbf{n}_i = (n_x, n_y, n_z)^\top$ is the normal of point $\mathbf{p}_i$.

Despite its advantages in scalability to large datasets and robustness to input noise [24], reconstructing large-scale scenes by IMLS usually leads to significant computational overhead. To accelerate the IMLS calculation, an improved solution is to construct a k-d tree for $\mathcal{P}_k$. Unfortunately, frequently building a k-d tree is computationally intensive with the time complexity of $O(N \log N)$ per tree construction, where N is the number of point clouds in $\mathcal{P}_k$. To this end, our voxel fusion accelerates the IMLS calculation without requiring additional data structures for $\mathcal{P}_k$.

2) Voxel Representation: We make use of sparse voxels to retain the information of accumulative point clouds, including a position, normal, SDF value, weight, and frame index. The voxel is defined as the minimal unit with the same size in space. Given the volumetric resolution (r_x, r_y, r_z), a point $\mathbf{p} = (p_x, p_y, p_z)^\top \in \mathcal{P}_w$ in the global coordinate system, corresponds to a voxel as below

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} \lfloor p_x/r_x \rfloor \\ \lfloor p_y/r_y \rfloor \\ \lfloor p_z/r_z \rfloor \end{pmatrix}. \quad (6)$$

(a) Active way (b) Passive way

Fig. 3: Computational models for SDF estimation in either (a) Active way or (b) Passive way. The green points denote the point clouds. The red point is the voxel. The blue circle represents the neighborhood centered around the voxel.

To distinguish our computational model for SDF estimation from others, we refer to our voxel as "passive voxel".

Different from actively searching for points around itself [19], our proposed approach emphasizes point-to-voxel increments. Fig. 3 illustrates the computational models to estimate the signed distance function (SDF) values in active and passive ways. Specifically, the voxels $\{\mathbf{v}_0^A, ..., \mathbf{v}_M^A\}$ in active way search the neighboring points to calculate the implicit function, which leads to substantial computation in searching among enormous point clouds $\{\mathbf{p}_0, ..., \mathbf{p}_N\}$ with the time complexity of $O(NM)$. As the scene expands, the number of voxels M increases. In contrast to actively finding the neighbors, our passive voxels $\{\mathbf{v}_0^P, ..., \mathbf{v}_M^P\}$ receive the SDF increments $\mathbf{n}_i^\top(\mathbf{v} - \mathbf{p}_i)$ from the surrounding points, which avoids the extensive searches. A point can determine the corresponding voxel via Equ. 6. In other words, the passive voxel representation can be built by searching the nearest points at the time complexity of $O(1)$. Consequently, the overall time complexity of our proposed scheme is $O(N)$, where N is the total number of scattered points. Our proposed method reduces the time complexity from $O(NM)$ to $O(N)$ and improves efficiency without the extra efficient representation of the point clouds.

For simplicity, unless specified, we will use the term "voxel" as "passive voxel" in the following.

3) Hybrid-weighted Voxel Integration: Our voxel emphasizes the incremental fusion from point to itself. The point-to-voxel increment is computed as follows:

$$d_i(\mathbf{v}) = \mathbf{n}_i^\top(\mathbf{v} - \mathbf{p}_i), \tag{7}$$

where $d_i(\mathbf{v})$ is the distance between the voxel and local surface formed by point $\mathbf{p}_i$.

For point $\mathbf{p}_i$, we make use of a hybrid weight that describes the distance and normal feature similarity. To this end, the hybrid weight $w_i^H(\mathbf{v})$ is defined as follows:

$$w_i^H(\mathbf{v}) = e^{-\frac{||\mathbf{v}-\mathbf{p}_i||^2}{h}} + \lambda_n \mathbf{n}_i^\top \mathbf{n}_\mathbf{v}, \tag{8}$$

where h is the range parameter. λ_n is the weight. $\mathbf{n}_i$ is the normal of point $\mathbf{p}_i$, and $\mathbf{n}_\mathbf{v}$ is the normal of voxel $\mathbf{v}$.

Our real-time LiDAR mapping scheme can be viewed as fitting surfaces from a continuous stream of point clouds. For point clouds $\{\mathbf{p}_0, ..., \mathbf{p}_n\}$ related to the voxel $\mathbf{v}$, SDF value in the voxel is updated as in [27]:

$$I(\mathbf{v}) \leftarrow I(\mathbf{v}) + \frac{w_n^H(\mathbf{v})(d_n(\mathbf{v}) - I(\mathbf{v}))}{W(\mathbf{v}) + w_n^H(\mathbf{v})}, \tag{9}$$

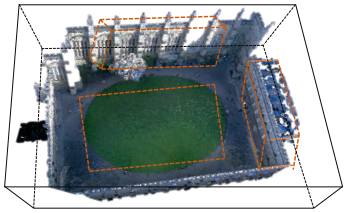


Fig. 4: The black cuboid represents a predetermined bounding box, while the orange cuboid is our scalable voxel block.

$$W(\mathbf{v}) \leftarrow W(\mathbf{v}) + w_n^H(\mathbf{v}), \tag{10}$$

where $I(\mathbf{v})$ and $W(\mathbf{v})$ are the SDF value and weight of the voxel, respectively. $d_n(\mathbf{v})$ is the distance between the voxel and local surface formed by point $\mathbf{p}_n$, and $w_n^H(\mathbf{v})$ is its weight.

Due to the sparsity of point clouds, the individual influence of each point on its corresponding voxel is insufficient. To this end, we extend the influence of point cloud to the surrounding voxels. In particular, we define an influence region centered on the selected voxel as a cube, where the side length is l times the voxel size. l is a small constant.

The hybrid-weighted voxel integration focuses on point-to-voxel increments, making itself suitable for parallel acceleration. Additionally, the fusion completes the update by traversing all points only once and allows for the safe removal of data from previous frames. This not only significantly improves efficiency but also reduces memory consumption.

4) Partitioned Meshing: Our sparse voxels are distributed throughout space. To acquire the voxel information within a specific space, a complete traversal of the range is required. Our approach allows quick access at each position, including voxel and empty space. However, it is inefficient and unnecessary to check a large number of empty locations. To reduce the empty space, we divide the 3D volume into voxel blocks with the same length L_{vb}, same width W_{vb}, and variable height H_{vb}. Conceptually, the voxel block contains a certain amount of voxels. The voxel block $\mathcal{B}$ is a scalable cuboid, located around the reconstructed surfaces, as depicted in Fig. 4, and dynamically adjusts its height to match the local scene in 3D space. The height is computed as follows:

$$H_{vb} = \max_{\mathcal{B}} z - \min_{\mathcal{B}} z. \tag{11}$$

Height-adaptive voxel blocks are designed to compress space and eliminate the requirement for a predefined bounding box.

The block-based Signed Distance Function (SDF) map provides quick access, which enables to extract the triangular mesh in parallel by the marching cubes algorithm [36]. Given the similarity of neighboring frames, performing repetitive surface extraction on each scan may require a substantial computational cost. Consequently, we extract the explicit surface at intervals t_s. We suggest that t_s is constrained by the total number of voxel blocks, making it a dynamic parameter that does not require manual setting. By exploiting the benefits of our proposed hybrid-weighted voxels fusion method and easily accessible blocked-based SDF map, we can progressively reconstruct the surface meshes.

D. Parallel Spatial-hashing Scheme

To achieve parallelization of voxel operations, we employ a simple and efficient spatial hash-based scheme. Besides, our proposed voxel deletion scheme facilitates long-term reconstruction and ensures the mesh quality involved with little influence.

1) Parallel Hash Table: The world coordinates of voxels are mapped to their respective hash codes using a spatial hash function. As in [37], a spatial hashing function $f : \mathbb{R}^3 \to \mathbb{R}$ is defined for a voxel position $\mathbf{v} = (x, y, z)$:

$$H(x, y, z) = (x \cdot p_1 \oplus y \cdot p_2 \oplus z \cdot p_3) \bmod n, \tag{12}$$

p_1, p_2, p_3 are large prime numbers, which are set to 73856093, 19349663, 83492791, respectively. The value n is the size of hash table.

Due to its compactness and flexibility, hash-based voxel representation is a common solution for processing and analyzing 3D scenes, however, it is usually implemented on CPUs. To maximize the efficiency, our spatial-hashing scheme is implemented on GPU. Unfortunately, the proposed hybrid voxel fusion scheme leads to data races [38] on GPU. To tackle this issue, we employ the distributed locks to ensure that threads with serial operations are implemented mutually exclusive writing to memory. This achieves parallel operations among voxels while supporting serial operations within them.

2) Collision Resolving Strategy: Collisions in the hash table may result in numerous invalid values of the reconstructed mesh, thereby under-utilizing the storage space. To deal with hash collisions, we employ open addressing strategies, such as robin hood hashing [39], linear probing, and quadratic probing. In our experiments, linear probing is empirically chosen as the collision resolving strategy.

3) Voxel Insertion: The position of a voxel serves as its unique identification. To insert a new hash item, we calculate its hash code by Equ. 12. If the corresponding entry in the hash table is empty, we insert the voxel directly. If the entry is already occupied, we utilize a collision resolving strategy to obtain a new mapping value. Then, we iterate the above steps until the voxel is inserted. Algorithm 1 describes the parallel insertion of voxels through atomic operations on mutexes.

4) Voxel Deletion: A key observation is that the initial scene information typically remains unchanged with the extension of the scene. In this perspective, we propose a voxel deletion scheme that converts the invariant information stored in the hash table into explicit mesh surfaces and removes the voxels that contain outdated information. Due to the removal of voxels, consecutive new voxels can be safely inserted. Our proposed voxel deletion scheme enables the continuous reconstruction of the large-scale surface mesh within the limited memory. To delete an existing voxel, we first retrieve the position of its corresponding hash item. Then, we clean up this hash item.

5) Voxel Update: Since the sequential LiDAR scans are processed, voxel information needs to be frequently updated. To retrieve and update a voxel, we calculate its hash code through Equ. 12 and locate the corresponding entry in the hash table. Then, we compare the unique identification, voxel position, of two voxels. If they are the same, we find the targeted voxel and update it as described in Section III-C. Otherwise, we obtain a new mapping value by a collision resolving scheme and iteratively repeat the above steps until the position matches.

Algorithm 1 Voxel Insertion Algorithm

```
// for each point p = (x, y, z)^⊤ simultaneously do
// Assigned the hash value of the voxelized point p to variable s
s = Hash(V(x, y, z)) ;
for i ← 1 to T_max do
  blocked ← true;
  while blocked is true do
    // Check if Mutex(s) is unlocked and atomically lock itself
    if Mutex(s) == unlock and Mutex(s) ← lock then
      if voxel v_s is empty then
        insert the voxel;
      else if KEY(v_s) == V(x, y, z) then
        exit;
      end if
      Mutex(s) ← unlock;
      blocked ← false;
    end if
  end while
  s ← s + f(i); // use collision resolving strategy f(i)
end for
```

6) Voxel Block Implementation: For a voxel block $\mathbf{v}_b = (x_b, y_b, z_b)^\top$, we construct a hash function $f : \mathbb{R}^2 \to \mathbb{R}$ using the first two components of the vector. Specifically $H(x, y) = (x \cdot p_1 \oplus y \cdot p_2) \bmod n$, similar to Equ. 12. The robin hood hashing [39] is used to resolve the collisions. Voxel blocks are able to effectively compress space, which can be used in scenes with unknown volume. According to the voxel deletion scheme, we delete those expired voxels so that the expired voxel blocks can be removed.

IV. EXPERIMENTS

In this section, we present the details of our experiments and evaluate our proposed Mesh-LOAM on four public large-scale datasets. Moreover, we demonstrate both quantitative and qualitative results compared to the state-of-the-art approaches. Additionally, we assess the effectiveness of our proposed point-to-mesh odometry, passive computational model as well as the voxel deletion scheme.

A. Implementation Details

Our method is implemented in C++ with CUDA and primarily utilizes GPU, except for data transmission. All experiments are conducted on a PC with an Intel Core i7-9800X CPU @ 3.80GHz and an NVIDIA GeForce RTX 2080Ti graphics card with 11GB GPU RAM. We select the normals by the curvature threshold $c_{th} = 0.1$. We set the voxel size as $r_x = r_y = r_z = 0.1$. In the case of the influence region for voxels, the side length of the cube is set to $l = 3$ times the size of the voxel. We compute the hybrid-weighted IMLS using the parameters $h = 0.05$ and $\lambda_n = 0.2$. We set the maximum iteration $T_{max} = 10$ in the voxel operations. For every voxel block, we set the length $L_{vb} = 2$ and width $W_{vb} = 2$. When choosing the correct correspondences to

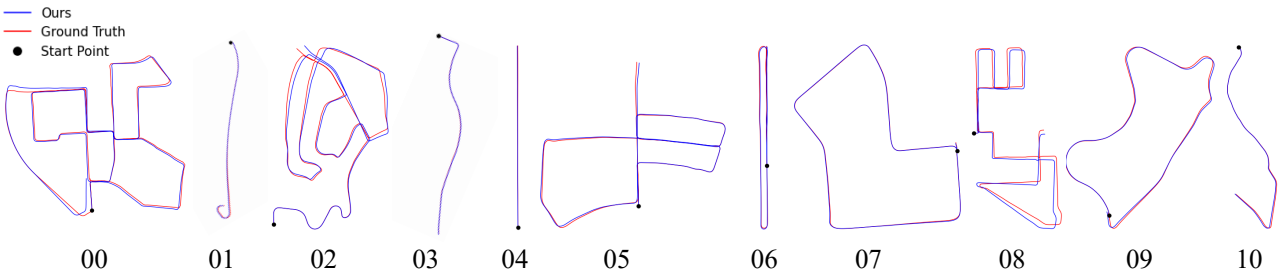


Fig. 5: Trajectories estimated by our Mesh-LOAM method on the KITTI sequences '00' to '10'.

TABLE I: Comparisons of Relative pose error (RPE) on KITTI Odometry Benchmark. The first row reports the relative translation error (%), and the second row shows the relative rotation error (degree/100m).

Map type	Method	00	01	02	03	04	05	06	07	08	09	10	Mean
Surfel	SuMa [8]	0.70 0.30	1.70 0.30	1.10 0.40	0.70 0.50	0.40 0.30	0.50 0.20	0.40 0.20	0.40 0.30	1.00 0.40	0.50 0.30	0.70 0.30	0.74 0.32
	SuMa++ [9]	0.64 0.22	1.60 0.46	1.00 0.37	0.67 0.46	0.37 0.26	0.40 0.20	0.46 0.21	0.34 0.19	1.10 0.35	0.47 0.23	0.66 0.28	0.70 0.29
NDT	Litamin2 [40]	0.70 0.28	2.10 0.46	0.98 0.32	0.96 0.48	1.05 0.52	0.45 0.25	0.59 0.34	0.44 0.32	0.95 0.29	0.69 0.40	0.80 0.47	0.88 0.38
Point Cloud	DLO [41]	2.73 1.09	14.56 0.83	3.06 1.06	2.13 1.11	1.90 0.82	1.89 0.86	2.26 0.84	2.30 1.11	2.43 0.98	2.91 1.13	3.25 1.21	3.58 1.00
	FLOAM [4]	0.73 0.44	0.95 0.16	0.83 0.34	1.47 0.35	0.50 0.26	0.63 0.42	0.48 0.31	0.71 0.66	1.03 0.40	0.66 0.30	0.94 0.42	0.81 0.37
	KISS-ICP [5]	**0.51** **0.19**	0.72 0.11	0.53 0.15	0.66 **0.16**	0.35 0.14	**0.30** **0.14**	**0.26** 0.08	0.33 0.16	**0.81** **0.18**	0.49 0.13	0.56 0.18	**0.50** 0.15
	Ours (point-to-plane)[1]	0.64 0.23	2.32 0.17	0.58 0.19	0.63 0.25	**0.31** 0.11	0.39 0.18	0.36 0.08	0.35 0.25	0.91 0.28	0.53 0.16	0.78 0.24	0.71 0.19
Mesh	Puma [13]	1.46 0.68	3.38 1.00	1.86 0.72	1.60 1.10	1.63 0.92	1.20 0.61	0.88 0.42	0.72 0.55	1.44 0.61	1.51 0.66	1.38 0.84	1.55 0.74
	SLAMesh [14]	0.77 0.35	1.25 0.30	0.77 0.30	0.64 0.43	0.50 0.13	0.52 0.30	0.53 0.22	0.36 0.23	0.87 0.27	0.57 0.25	0.65 0.42	0.68 0.29
	Ours	0.53 0.20	**0.64** **0.11**	**0.52** **0.15**	**0.63** 0.20	0.41 **0.07**	0.34 0.15	0.35 **0.07**	**0.31** **0.15**	0.82 0.19	**0.47** **0.13**	**0.56** **0.16**	0.51 **0.14**

[1] The point-to-plane loss is used to estimate the LiDAR poses.

compute point-to-mesh residuals, the threshold c_s is set to 0.98.

B. Datasets

KITTI [11]: The KITTI odometry is a large-scale outdoor dataset collected by Velodyne HDL-64E S2 LiDAR, containing various street environments. Our experiments were conducted on Sequence '00' to '10', which consist of a total number of 23,201 scans. We made use of the KITTI odometry dataset to evaluate the odometry accuracy and map quality of our proposed approach. To examine the effectiveness of our method, we also carried out an ablation study of point-to-mesh odometry, the passive computational model, and the voxel deletion scheme.

HILTI 2021 [42]: To evaluate our proposed Mesh-LOAM, we utilized the Hilti 2021 dataset that contains a series of real-world indoor sequences, including offices, labs, and construction environments, as well as outdoor sequences from construction sites and parking areas. The handheld sensor platform is used to record the data, which provides millimeter-accurate ground truth data for each sequence. It includes an Ouster OS0-64 LiDAR, which collects long-range point cloud data with a 360° field of view (Fov) at a rate of 10 Hz. Since most of these sequences provide 3-DoF poses, we utilize the absolute trajectory error (ATE) as the primary evaluation metric for odometry.

Mai City [13]: To qualitatively and quantitatively evaluate our proposed method, we conducted experiments on the Mai City dataset, which is a synthetic collection generated from an urban-like scenario with a virtual Velodyne HDL-64 LiDAR.

Newer College [43]: The Newer College dataset is collected by a handheld device with a multi-beam 3D LiDAR at Oxford University providing a detailed millimeter-accurate 3D map.

In our experiments, we quantitatively evaluate the quality of recovered mesh on the two latter datasets with accessible ground truth mesh information.

C. Odometry Evaluation

To examine the performance of LiDAR odometry, we employ the widely-used KITTI odometry dataset [11] to compare our proposed method with the state-of-the-art LiDAR-only approaches that employ different types of maps. Fig. 5 plots the estimated trajectories from the sequence '00' to '10', including urban, countryside, residential, and highway environments.

TABLE II: Comparisons of Absolute trajectory error (m) on KITTI Odometry Benchmark.

Map type	Method	00	01	02	03	04	05	06	07	08	09	10	Mean
Surfel	SuMa [8]	**2.9**	13.8	8.4	0.9	0.4	**1.2**	0.4	0.5	2.8	2.9	1.3	3.2
NDT	Litamin2 [40]	5.8	15.9	10.7	0.8	0.7	2.4	0.9	0.6	**2.5**	2.1	1.0	3.9
Point Cloud	FLOAM [4]	5.0	3.2	8.6	0.7	0.3	3.4	0.5	0.6	3.1	1.6	1.2	2.6
	KISS-ICP [5]	5.7	30.7	17.8	3.4	1.1	1.9	0.9	0.8	4.8	3.7	2.3	6.6
	Ours (point-to-plane)[1]	5.5	3.1	**7.0**	0.5	1.5	1.7	8.9	0.5	3.0	1.8	0.9	3.1
Mesh	Puma [13]	6.6	32.6	18.5	2.2	0.9	3.3	2.4	0.9	6.3	3.9	4.4	7.5
	SLAMesh [14]	5.5	10.9	13.2	0.8	0.3	3.7	0.7	0.8	5.1	**1.0**	1.1	3.9
	Ours	5.3	**3.0**	7.4	**0.5**	**0.3**	1.7	**0.3**	**0.4**	3.3	1.7	**0.9**	**2.3**

[1] The point-to-plane loss is used to optimize the estimated poses.

TABLE III: ATE (m) on the Hilti SLAM Challenge Dataset

Approach	Sensor	RPG	Base1	Base4	Lab	Cons2	Camp2
SuMa [8]	Ouster OS0-64	0.262	2.244	0.286	0.045	1.642	×[1]
FLOAM [4]		2.775	0.914	0.287	0.182	11.515	8.946
KISS-ICP [5]		0.187	0.294	**0.119**	0.073	0.835	5.052
Puma [13]		×	×	×	×	×	×
SLAMesh [14]		**0.165**	0.175	0.330	**0.048**	0.339	0.653
Ours		0.173	**0.165**	0.267	0.049	**0.083**	**0.113**

[1] '×' denotes the failed registration in this sequence.

TABLE IV: ATE (m) on the MaiCity and Newer College Dataset

Approach	Sensor	Mai01	Sensor	NCD-QUAD
SuMa [8]	Velodyne HDL-64	0.061	Ouster OS-1	×[1]
FLOAM [4]		0.062		17.620
KISS-ICP [5]		×		0.100
Puma [13]		0.058		0.573
SLAMesh [14]		0.017		0.337
Ours		**0.016**		**0.079**

[1] '×' denotes the failed registration in this sequence.

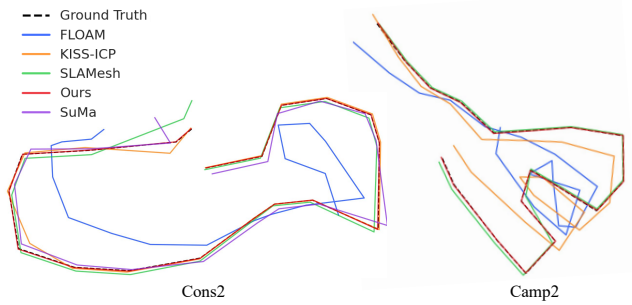


Fig. 6: Comparisons of trajectories on the outdoor sequences of the Hilti SLAM challenge dataset.

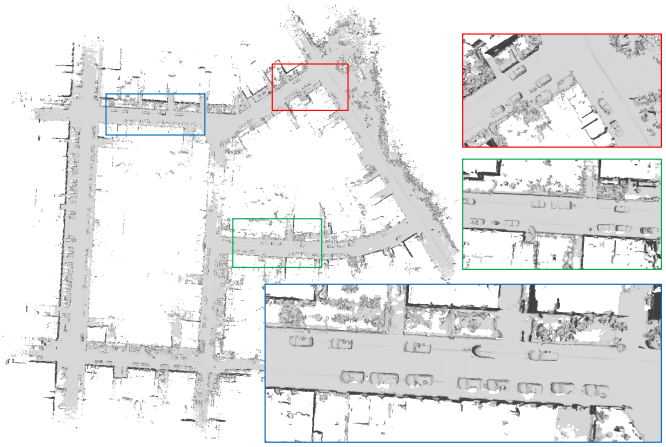
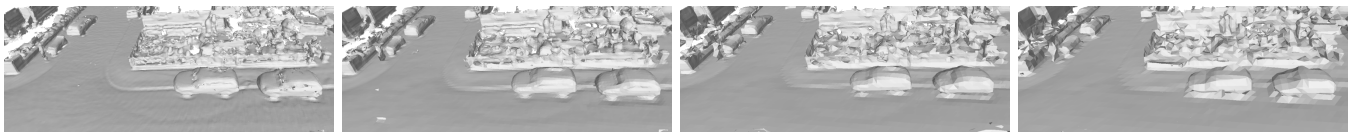


Fig. 7: The mesh result of our proposed Mesh-LOAM approach with local details on KITTI sequence '07'.

Fig. 8: The left figure shows a view of the scene. The right one is our odometry and mapping result on the construction environment of the Hilti 'cons2' sequence.

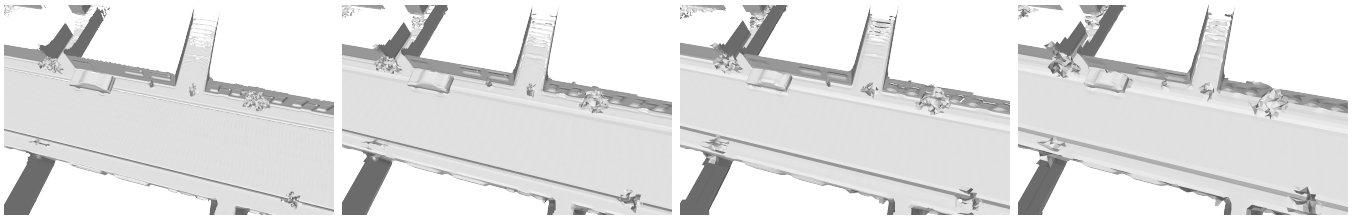
Following the evaluation criteria described in [11], we use the relative translational error in % and the relative rotational error in degrees per 100m to evaluate the accuracy of LiDAR odometry. We conducted experiments for DLO [41], KISS-ICP, and FLOAM based on their experimental setups. The other results are directly from their published papers. Table I shows that our approach achieves the promising results with 0.51% drift in translation error and 0.15 deg/100m in rotation error in average 11 sequences, which is slightly inferior to KISS-ICP with 0.50% error in translation and better than surfel-based, NDT-based and other mesh-based methods. Note that KISS-ICP is the state-of-the-art LiDAR-only odometry method, which performs the best among the published results. As shown in Table I, our results on sequences '04' '05' '06' are worse than KISS-ICP. This could be attributed to the 0.2m search interval used in the experiment, which hinders finding the suitable correspondences. Accordingly, it may prevent the convergence. Our point-to-mesh odometry outperforms other methods with various kinds of maps.

To make a comparison on the entire shape of trajectories, we employ ATE as a performance metric to evaluate the odometry results on the KITTI odometry dataset. We export ATE results [40] of SuMa and Litamin2. Since SuMa++ requires extra semantic information, thus excluded from the ATE comparison. Besides, we conduct experiments for other methods based on their experimental setups. Table II demonstrates that



0.1 0.2 0.3 0.4

Fig. 9: Mapping results on Sequence '07' of the KITTI dataset with various resolutions from 0.1m to 0.4m.



0.1 0.2 0.3 0.4

Fig. 10: Mapping results on the MaiCity dataset with various resolutions from 0.1m to 0.4m.

TABLE V: Mapping results of different methods on MaiCity [13] and Newer College [43] datasets in terms of map accuracy, completion, Chamfer-L_1 distance, completion ratio, and F-score.

Method	MaiCity					Newer College				
	Comp. ↓	Acc. ↓	C-L_1. ↓	Comp.Ratio ↑	F-score (10cm) ↑	Comp. ↓	Acc. ↓	C-L_1. ↓	Comp.Ratio ↑	F-score (20cm) ↑
VDB Fusion [28]	6.9	1.3	4.5	90.2	94.1	12.0	6.9	9.4	91.3	92.6
Puma [13]	32	1.2	16.9	78.8	87.3	15.4	7.7	11.5	89.9	91.9
SHINE-Mapping [30]	3.2	**1.1**	2.9	95.2	95.9	10.0	6.7	8.4	93.6	93.7
SLAMesh [14]	7.5	3.7	6.1	89.2	90.6	13.7	11.4	12.6	83.5	82.3
Ours	**2.5**	1.2	**2.4**	**96.3**	**97.4**	**9.6**	**6.7**	**8.2**	**94.2**	**94.1**

'↓' indicates that a lower value is preferable for the metric, while '↑' signifies the higher value is great for the metric.

our proposed Mesh-LOAM approach achieves 2.3m ATE on average and exhibits the smallest deviation from the ground truth trajectory in the majority of sequences.

We quantitatively evaluate the performance of our odometry method on the Hilti 2021 dataset [42], which is more challenging. Since most of sequences in the Hilti 2021 dataset provide 3-DoF ground truth poses, we use the absolute trajectory error (ATE) in $\%$ as evaluation criteria. We only use data collected from Ouster OS0-64 liDAR in our experiments. We compared our method against Surfel-based method SuMa [8], FLOAM [4], the state-of-the-art method KISS-ICP [5], mesh-based methods Puma [13] and SLAMesh [14]. We conducted the experiments using their own implementations. Due to the lack of publicly available implementation, Litamin2 [40] is not compared. As shown in Table III and Fig. 6, our proposed method achieves the best performance in most outdoor and indoor scenes on the handheld Hilti 2021 dataset. Puma fails in this dataset. SLAMesh performs slightly better than ours in the 'RPG' and 'Lab' sequences. Especially, SuMa, FLOAM, and KISS-ICP perform poorly on the 'camp2' sequence, while our method still achieves the lowest drift in such challenging sequence.

To evaluate the effectiveness and generalization capability of our proposed LiDAR odometry approach, we compare it against SuMa, FLOAM, KISS-ICP, Puma, and SLAMesh on the MaiCity and Newer College Dataset. Table IV shows that our proposed method achieves the smallest ATE and outperforms the other four methods. These promising experimental results indicate that our proposed point-to-mesh odometry is robust to noise and capable of finding reliable correspondences.

D. Mapping Evaluation

To demonstrate the effectiveness and generalization capability of our proposed Mesh-LOAM approach, we qualitatively show several odometry and mapping results on two large-scale datasets [11], [42]. Fig. 1 illustrates the result of our proposed Mesh-LOAM with geometric details on the KITTI sequence '00', whose travel length is 3.7km. Moreover, an outdoor large-scale scene on KITTI sequence '07' is presented in Fig. 7, where we can see smoothing and flat road surfaces, as well as a variety of largely intact shaped vehicles. Furthermore, Fig. 8 illustrates a real-world construction environment on the Hilti dataset, depicting smooth floors and recessed rough construction floors. Since the data is collected from a handheld platform, the point cloud coverage is nonuniform, resulting in some blank edges. These visual results indicate that our proposed Mesh-LOAM method is able to predict the accurate 6-DoF poses of a mobile system while recovering a dense mesh of the large-scale outdoor scene.

To visually present the reconstructed mesh, we compare our mapping results on different grid resolutions ranging from 0.1m to 0.4m. As depicted in Fig. 9 and Fig. 10, our method is capable of constructing complete meshes for large-scale outdoor scenes while preserving accurate structures, such as vehicle outlines, shallow curbs, and trees.

To evaluate the mapping quality of our proposed method, we conducted experiments with the voxel size of 0.1m on

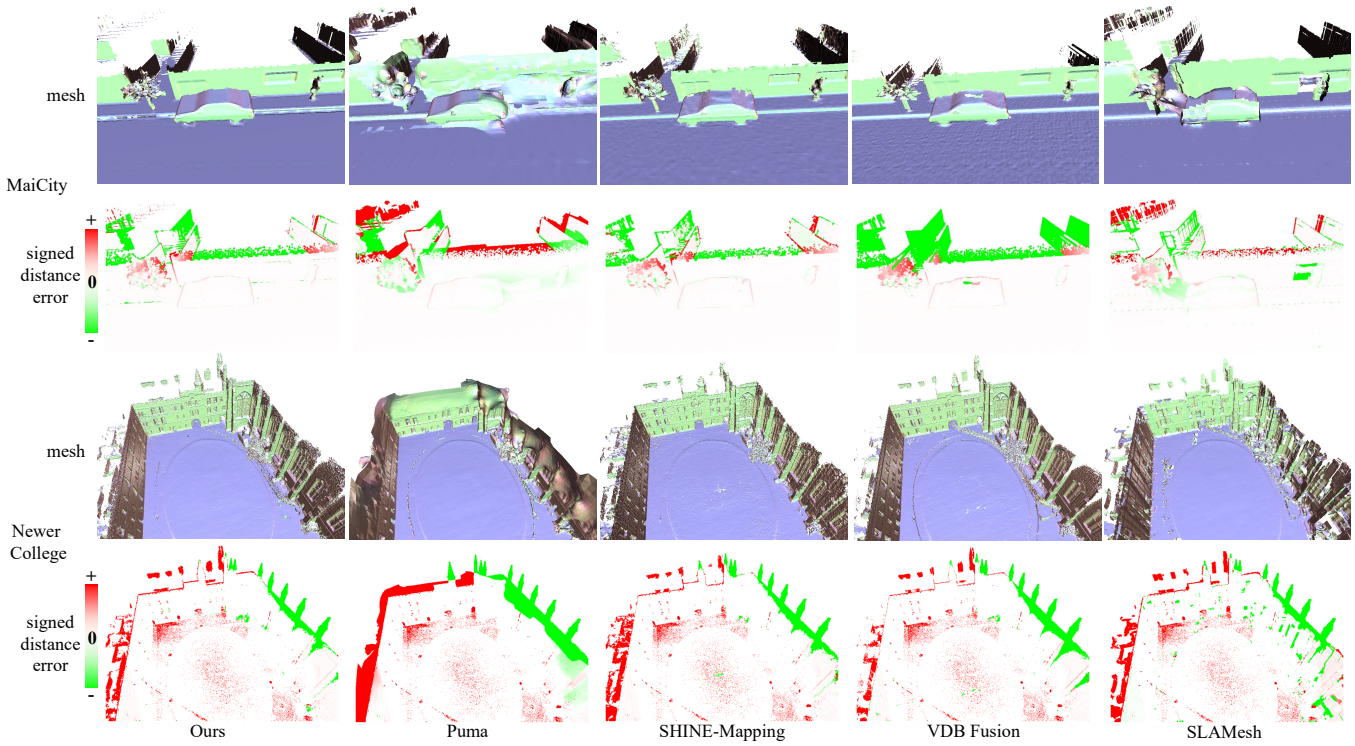


Fig. 11: Qualitative comparison results on the MaiCity and Newer College dataset. The first and third rows show the recovered mesh of different methods, including our proposed approach, Puma [13], SHINE-Mapping [30], VDB Fusion [28] and SLAMesh [14]. The second and fourth illustrates the error map between the reconstructed mesh and the ground truth. The red color represents the positive distance between the resulting mesh and the ground truth, while the green color indicates the negative distance. The brighter the color, the greater the error is.

TABLE VI: The computational cost of each module per frame (ms) with different SDF estimation models.

Computation Model	Preprocess	Odometry	SDF map update	Partitioned Meshing	Total Time
passive	4.85	10.96	2.12	0.56	18.49
active	5.18	11.62	8806.20	1.40	8824.40

TABLE VII: Evaluation of Information Dropout Ratio(%) on the KITTI odometry dataset (Sequence '00' to '10')

Collision Resolving Strategy	Voxel Deletion Scheme	00	02	08	Avg. Dropout Ratio
Linear probing	×	3.7	15.9	16.4	7.4
	✓	1.9e-05	4.8e-05	**1.9e-04**	**5.3e-05**
Quadratic probing	×	3.5	15.8	16.3	7.4
	✓	2.4e-05	6.7e-05	2.0e-04	5.9e-05
Robin hood hashing	×	6.64	17.2	17.6	8.5
	✓	**1.9e-06**	**2.3e-06**	3.0e-04	6.4e-05

the virtual dataset Mai City and the real-world dataset Newer College. We compare our approach against four state-of-the-art methods, including a TSDF fusion-based method VDB Fusion [28], a Poisson regression-based approach Puma [13], a learning-based method SHINE-Mapping [30] and an explicit approach SLAMesh [14]. To compensate for the errors resulting from LiDAR odometry, we directly employ the ground truth pose for all methods and follow the settings of [30] to facilitate fair comparisons. We use five evaluation metrics [44], including accuracy, completion, Chamfer-L_1 distance, completion ratio, and F-score. Table V presents the evaluation results with distance error in cm. It shows the completion Ratio and F-score, expressed as percentages, with a 10cm and 20cm error threshold for the two datasets, respectively. Our proposed approach outperforms the four methods on both two datasets. As shown in Fig. 11, our method recovers the most complete surface mesh while preserving the detailed structures, including vehicle outlines, pedestrian, and roadside trees. Both quantitative and qualitative results on these two datasets demonstrate that our proposed meshing method is capable of recovering the complete and accurate mesh for large-scale outdoor scenes by taking advantage of our proposed hybrid-weighted voxel integration scheme.

E. Ablation Study

Point-to-mesh Odometry. To evaluate our proposed point-to-mesh odometry approach, we optimize the pose by minimizing the point-to-plane loss, and the remaining settings exactly the same. The point-to-plane loss refers to the distance from the source point cloud to the local plane in the global point cloud map. We conducted experiments on the KITTI odometry dataset. As shown in Table I, the KITTI relative translation and rotation errors of point-to-plane odometry are 0.71 % and 0.19 degree/100m, which are worse than our results with point-to-mesh loss. Moreover, Table II also demonstrates our point-to-mesh odometry archives lower ATE than the point-to-plane

TABLE VIII: Evaluation on Computational Time (ms) and Memory Consumption (MB) on the KITTI sequence 07.

	Preprocess	Odometry	Meshing	Total Time	CPU Memory	GPU Memory
Puma [13]	94.00	1000.00	2213.50	3307.50	5766	0
SLAMesh [14]	5.18	9.78	14.31	29.27	3014	0
Ours	4.85	10.96	2.68	18.49	442	2860

one. The point-to-plane loss may find some unreasonable correspondences and increase the pose prediction errors. This not only shows that our point-to-mesh ICP can achieve accurate odometry accuracy but also proves the reconstructed mesh map is beneficial for our odometry.

Passive Computational Model. Our adopted passive voxel accelerates the SDF map update, theoretically reducing the time complexity from O(MN) to O(N). To evaluate the efficiency improvement from active computation model to passive computational model, we conduct ablation experiments to examine the computational time of each module on the KITTI odometry dataset. For both two implementations, we use GPU parallel acceleration. As shown in Table VI, the passive computational model significantly improves the efficiency of SDF map update.

Voxel Deletion Scheme. In our implementation, open addressing [39], [45] is chosen as the collision resolving strategy. To examine the efficacy of our presented voxel deletion scheme, we conduct ablation studies on the KITTI odometry dataset with different collision resolving strategies, including robin hood hashing, linear probing, and quadratic probing. We introduce the Dropout Ratio in % to describe the rate of lost voxels. The lower the value, the better the information is preserved. We empirically set $2i + 1$ as linear probing and $i^2 + 1$ as quadratic probing.

As shown in Table VII, all three probing methods with our voxel deletion scheme achieve a small dropout ratio on the KITTI odometry dataset, including long sequences '00', '02', and '08'. On the contrary, the results without the voxel deletion scheme perform poorly. The experiment indicates that our proposed scheme can effectively minimize the influence of hash collisions, which supports incremental reconstruction within limited memory and ensures mesh quality with minimal impact. Table VII also shows that linear probing $2i + 1$ with the suggested strategy is appropriate for the mapping task on the KITTI odometry dataset.

F. Computational Efficiency and Memory Consumption

To demonstrate the efficiency of our presented approach, we evaluate the computational time per frame on different steps, including preprocessing, point-to-mesh odometry, and incremental voxel meshing. We compare with mesh-based LOAM methods, Puma [13] and SLAMesh [14] on the KITTI odometry dataset. As shown in Table VIII, Puma is far from real-time. SLAMesh reaches 34 frames per second (fps). Our method runs around 54 fps overall and fulfills the real-time requirement. Our run-time performance is attributed to the passive SDF computational model, the scalable partition module, and an efficient parallel spatial-hashing scheme. The bottleneck is mainly from obtaining appropriate point-to-mesh correspondences in the point-to-mesh odometry, i.e., data association. Although the quick access to the SDF maps allows us to speed up the data association process by retrieving the neighboring meshes in parallel, this process still takes some time.

In addition, we record the peak memory usage with the three methods on the '07' sequence of the KITTI dataset. Puma and SLAMesh are both CPU-based methods. Table VIII summarizes the memory usage. Due to primarily implementing on GPU, our approach has low main memory consumption. To achieve parallel operations on the GPU, we have allocated additional space for efficiency, which takes up 2860 MB of video memory.

V. CONCLUSIONS

This paper proposed a real-time large-scale LiDAR odometry and meshing approach. A novel voxel meshing algorithm was introduced to incrementally reconstruct triangular meshes using a parallel spatial-hashing scheme, which employs only single traversal per LiDAR scan and takes advantage of a scalable partition module. Besides, point-to-mesh odometry was designed to reduce the drifting in pose estimation. We conducted our experiments on four large-scale outdoor datasets, whose promising results demonstrated that our proposed Mesh-LOAM approach achieves low drift for odometry and high-quality 3D reconstruction in real time.

Since mesh extraction is performed on GPU, it requires some GPU RAM. We will explore the mesh simplification technique to reduce memory usage in the future.

REFERENCES

[1] A. Chalvatzaras, I. Pratikakis, and A. A. Amanatiadis, "A survey on map-based localization techniques for autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1574–1596, 2022.

[2] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[3] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[4] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-loam: Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4390–4396.

[5] 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*, vol. 8, no. 2, pp. 1029–1036, 2023.

[6] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[7] Z. Zhou, X. Feng, S. Di, and X. Zhou, "A lidar mapping system for robot navigation in dynamic environments," *IEEE Transactions on Intelligent Vehicles*, 2023.

[8] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." in *Robotics: Science and Systems*, vol. 2018, 2018, p. 59.

[9] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4530–4537.

[10] E. Piazza, A. Romanoni, and M. Matteucci, "Real-time cpu-based large-scale three-dimensional mesh reconstruction," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1584–1591, 2018.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, 2012, pp. 3354–3361.

[12] T. Kühner and J. Kümmerle, "Large-scale volumetric scene reconstruction using lidar," in *2020 IEEE international conference on robotics and automation*, 2020, pp. 6261–6267.
[13] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson surface reconstruction for lidar odometry and mapping," in *2021 IEEE International Conference on Robotics and Automation*, 2021.
[14] 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, ICRA 2023, London, UK, May 29 - June 2, 2023*. IEEE, 2023, pp. 3546–3552.
[15] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: Online surfel-based mesh reconstruction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2494–2507, 2019.
[16] C. Ericson, *Real-time collision detection*. Crc Press, 2004.
[17] 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.
[18] 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*, vol. 7, no. 3, pp. 8518–8525, 2022.
[19] J.-E. Deschaud, "Imls-slam: Scan-to-model matching based on 3d data," in *2018 IEEE International Conference on Robotics and Automation*, 2018.
[20] X. Zheng and J. Zhu, "Efficient lidar odometry for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8458–8465, 2021.
[21] B. Zhou, Y. Tu, Z. Jin, C. Xu, and H. Kong, "Hpplo-net: Unsupervised lidar odometry using a hierarchical point-to-plane solver," *IEEE Transactions on Intelligent Vehicles*, 2023.
[22] J. Ruan, B. Li, Y. Wang, and Z. Fang, "Gp-slam+: real-time 3d lidar slam based on improved regionalized gaussian process map reconstruction," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5171–5178.
[23] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 67–76.
[24] R. Kolluri, "Provably good moving least squares," *ACM Transactions on Algorithms*, vol. 4, no. 2, pp. 1–25, 2008.
[25] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, 1992, pp. 71–78.
[26] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "isdf: Real-time neural signed distance fields for robot perception," in *Robotics: Science and Systems*, 2022.
[27] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.
[28] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "Vdbfusion: Flexible and efficient tsdf integration of range sensor data," *Sensors*, vol. 22, no. 3, p. 1296, 2022.
[29] S. Fuhrmann and M. Goesele, "Fusion of depth maps with multiple scales," *ACM Transactions on Graphics (TOG)*, vol. 30, 2011.
[30] X. Zhong, Y. Pan, J. Behley, and C. Stachniss, "Shine-mapping: Large-scale 3d mapping using sparse hierarchical implicit neural representations," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2023.
[31] 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, pp. 8218–8227.
[32] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 1366–1373.
[33] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 1–11, 2013.
[34] J. Niedzwiedzki, P. Lipinski, and L. Podsedkowski, "Idtmm: Incremental direct triangle mesh mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5416–5423, 2023.
[35] J. Solà Ortega, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," 2018.
[36] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.
[37] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. H. Gross, "Optimized spatial hashing for collision detection of deformable objects." in *Vmv*, 2003, pp. 47–54.
[38] C. Artho, K. Havelund, and A. Biere, "High-level data races," *Software Testing, Verification and Reliability*, vol. 13, pp. 207–227, 2003.
[39] P. Celis, P.-A. Larson, and J. I. Munro, "Robin hood hashing," in *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, 1985, pp. 281–288.
[40] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, "Litamin2: Ultra light lidar-based slam using geometric approximation applied with kl-divergence," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 619–11 625.
[41] K. Chen, B. T. Lopez, A.-a. Agha-mohammadi, and A. Mehta, "Direct lidar odometry: Fast localization with dense point clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2000–2007, 2022.
[42] M. Helmberger, K. Morin, B. Berner, N. Kumar, G. Cioffi, and D. Scaramuzza, "The hilti slam challenge dataset," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7518–7525, 2022.
[43] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The newer college dataset: Handheld lidar, inertial and vision with ground truth," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 4353–4360.
[44] 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, pp. 4460–4470.
[45] P. Flajolet, P. Poblete, and A. Viola, "On the analysis of linear probing hashing," *Algorithmica*, vol. 22, no. 4, pp. 490–515, 1998.

Yanjin Zhu received her bachelor's degree in Software Engineering from Xi'an Jiaotong University in 2021. She is currently a PhD candidate at the College of Computer Science and Technology, Zhejiang University of China. Her research interests include LiDAR odometry and mapping.

Xin Zheng received his bachelor's degree in Mechanical Engineering from Zhejiang University of Technology in 2018. He is currently a PhD candidate at the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include state estimation and LiDAR SLAM.

Jianke Zhu received the master's degree from the University of Macau in Electrical and Electronics Engineering, and the PhD degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong in 2008. He held a postdoctoral position at the BIWI Computer Vision Laboratory, ETH Zurich, Switzerland. He is currently a Professor with the College of Computer Science, Zhejiang University, Hangzhou, China. His research interests include computer vision and robotics. He is a Senior member of the IEEE.