IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Onion-LO: Why Does LiDAR Odometry Fail Across Different LiDAR Types and Scenarios?

Xiaolong Cheng[1], Keke Geng[1], Zhichao Liu[1], Tianxiao Ma[1], and Ye Sun[1]

***Abstract*—LiDAR odometry is a fundamental technology for autonomous navigation. However, existing LiDAR-based odometry methods typically demand extensive manual parameter tuning and remain prone to instability when deployed across varying LiDAR types and environments. This letter focuses on the essence of point clouds and introduces a fast, highly adaptable, and robust LiDAR odometry framework named Onion-LO. Onion-LO demonstrates strong compatibility with various LiDAR types and reliable operation across diverse scenarios. This is facilitated by an onion-like point cloud processing structure termed Onion Ball. The Onion Ball supports multi-threaded implementation, efficiently executing point cloud distribution analysis, segmentation, and downsampling. In addition, we design an adaptive optimization strategy for local map management and iterative optimization, which effectively enhances the system's robustness and accuracy. Extensive experiments on five datasets demonstrate that Onion-LO outperforms existing state-of-the-art methods regarding localization accuracy and robustness. Additional evaluations across 11 LiDAR sensors and 8 diverse scenarios further confirm its strong generalization capability. Our method is designed for practical deployment and supports real-time operation on onboard processors. We open-source the code on *https://github.com/huashu996/Onion-LO*.**

***Index Terms*—SLAM, localization, LiDAR odometry**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) is the technology that enables the robot to know its position and mapping the surrounding environment. LiDAR odometry is widely used in autonomous navigation due to its high measurement accuracy and reliable depth perception. Many existing LiDAR odometers [1], [2], [3], [4] are mostly only stable in certain scenarios and with certain LiDAR types, and some methods [5], [6] require heavy parameter adjustment before application. Meanwhile, the advent of new-generation solid-state LiDAR sensors has significantly broadened the range of application scenarios, enabling advanced tasks such as high-altitude mapping [7], 3D reconstruction [8]. The new scanning modes of solid-state LiDAR and their application scenarios bring new challenges for the odometry system. Concurrently, the trajectory of future research is moving towards multi-robot collaborative heterogeneous systems [9], [10], [11], placing greater demands on the adaptability and robustness of algorithms.

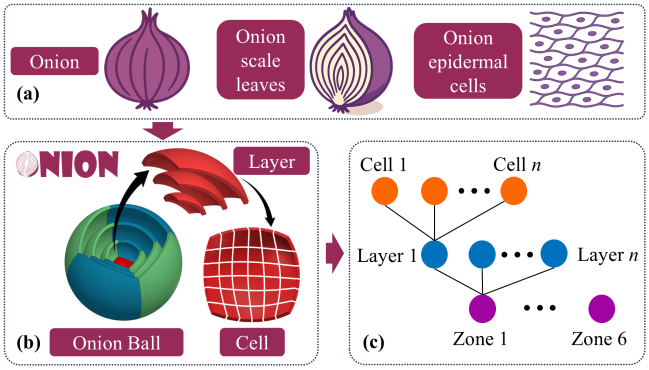

Fig. 1. (a) The schematic structure of a real onion. (b) The onion ball structure diagram. (c) The data structure of the Onion Ball.

This paper returns to the root of the LiDAR sensor to design a novel and versatile framework. The proposed framework is designed to accommodate LiDAR sensors with diverse resolutions and scanning modes. To develop a generalizable LiDAR odometry framework, it is essential to first understand the underlying causes of system failure when LiDAR types or application scenarios change. **A) Design concept.** Some LiDAR SLAM frameworks are tailored to the specific characteristics of certain LiDAR types or scenarios, which limits their generalizability. **B) Point cloud distribution.** Changes in LiDAR sensors or environments ultimately lead to variations in point cloud distribution. When these distributions differ significantly, the system's original modules or parameters are no longer suitable. **C) Parameter adjustment.** Many systems require the adjustment of numerous parameters, which remain fixed during operation and cannot adapt to changing scenarios. The reasons above make many LiDAR odometry systems not capable of out-of-the-box operation.

To solve the above issues, our framework is structured around an onion-like architecture known as the "Onion Ball," as illustrated in Fig. 1. The Onion Ball is a spherical structure, divided into 6 zones. Each region is further segmented into layers at specific intervals, with each layer subdivided into cells based on voxel size. This structure can be adapted to various LiDAR types and can quickly analyze the point cloud distribution. Moreover, it can achieve discrete point removal,

Manuscript received: April, 25, 2025; Revised July, 1, 2025; Accepted August, 2, 2025.

This paper was recommended for publication by Editor S. Behnke upon evaluation of the Associate Editor and Reviewers' comments. This work is supported in part by the National Natural Science Foundation under Grant no. 52272414 and 51905095, Jiangsu Graduate Innovative Research Program under Grant no. SJCX24_0072, National Key R&D Program of China 2023YFD2000303, and Yanetze River Delta Science and Technology Innovation Alliance Collaborative Research Project 2023C51GG1600. (Corresponding author: Keke Geng.)

[1]Xiaolong Cheng, Keke Geng, Zhichao Liu, Tianxiao Ma, Ye Sun are with the Department of Mechanical Engineering, Southeast University, Nanjing, 201804, China (email: {230238059, jsgengke, 230248037, 230230347, 240240374}@seu.edu.cn).

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

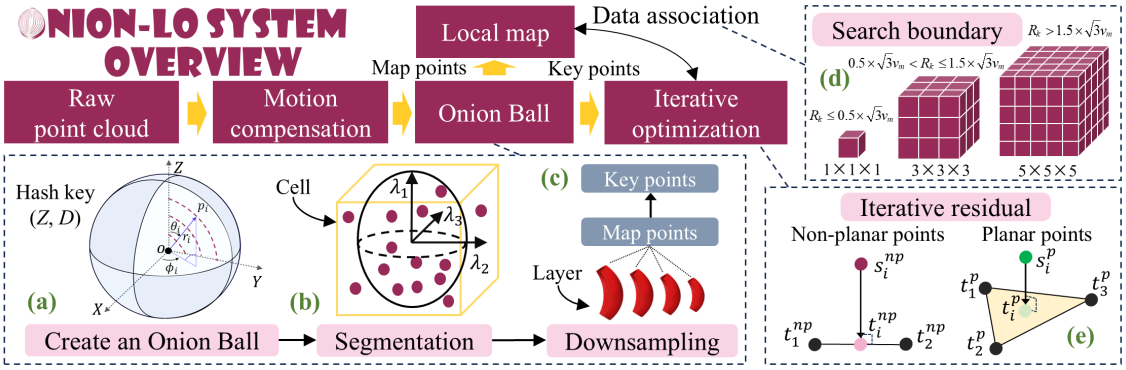

Fig. 2. The Onion-LO system overview. (a) The Onion Ball stores point clouds by partitioning the space based on the zone Z and the distance D. (b) Points are classified into planar and non-planar types based on the cell covariance. (c) Each layer is downsampled in parallel to extract key points and map points. (d) The search range is selected based on the search radius R_k. (e) Different residuals are selected based on point types for iterative optimization.

point cloud segmentation, and downsampling. This front-end processing of the point cloud by Onion Ball enables robust key points to be sent to the back-end for iterative optimization. The main contributions of this work are categorized into the following four aspects:

- We present a novel and generic LiDAR odometry framework named "Onion-LO", which can cope with a wide range of LiDAR types and scenarios without the need for cumbersome system parameter configurations.
- A powerful point cloud front-end processing structure, "Onion Ball", is proposed. This structure is designed based on the inherent characteristics of LiDAR sensors and is capable of fast, multi-threaded point cloud distribution analysis, downsampling, and segmentation.
- An adaptive back-end processing pipeline is proposed, which dynamically adjusts optimization thresholds and local map storage parameters in response to the current point cloud distribution.
- We conduct extensive experiments on both public and private datasets to validate the effectiveness of Onion-LO and demonstrate the advantages of the multi-layered Onion Ball structure.

II. Related Work

LiDAR odometry has been a highly active research area and serves as a core technology for autonomous navigation systems. Its development has closely paralleled the advancement of LiDAR hardware. The early 3D LiDAR odometers [12], [1], [13], [14], [15] are designed around the scanning pattern of the mechanical LiDAR. Zhang et al [12] proposed a method for judging edge and planar points based on the curvature of the adjacent points. Shan et al [1] divide ground points based on the altitude difference between adjacent LiDAR scan lines. The design concepts of the feature point extraction method and the ground segmentation method described above are overly dependent on the scanning type of the mechanical LiDAR. Subsequently, the emergence of new solid-state LiDAR brings many new challenges to LiDAR odometry, with a small field of view (FoV) and non-repeatable scanning, making most classical frameworks difficult to adapt. To allow the LOAM framework to be applied to non-repetitive scanning LiDAR. Lin et al [3] optimize the feature extraction front-end and the iterative optimization back-end based on LOAM, which makes the algorithm stable on a new LiDAR type. However, this algorithm cannot be adapted to both mechanical LiDAR and solid-state LiDAR. Lili-om [16] is a tightly-coupled LiDAR-inertial SLAM for both solid-state and mechanical LiDARs, but they developed a separate feature extraction module for the Livox Horizon LiDAR. With recent advancements, an increasing number of non-repetitive scanning LiDAR types have emerged. However, most of the existing feature extraction methods [3], [14], [15] are designed for specific LiDAR scanning patterns, making them difficult to generalize across devices with multiple scanning modes.

Compared to feature-based approaches, raw point cloud-based LiDAR odometry provides enhanced adaptability and portability, as it does not rely on specific environmental structures or LiDAR scanning patterns. Thus, several new approaches [17], [18], [19], [20] use raw points for iterative optimization instead of feature points. CT-ICP [5] utilizes continuous time frames to handle transient changes in high-speed movements, enabling strong performance in drone and autonomous driving scenarios. However, it requires extensive parameter tuning before deployment. KISS-ICP [17] uses a simple and classical point-to-point iterative closest point (ICP) to realize the adaptation of multiple LIDARs and scenarios. The overall system is controlled by only 7 parameters, but these parameters cannot be adaptively adjusted while the system is running. This means that KISS-ICP is not suitable for missions with variable scenarios, such as transitions between indoor and outdoor environments or drone takeoffs and landings. FAST-LIO2 [19] and Point-LIO [20] both use iterative Kalman filtering to tightly couple the IMU to the raw point cloud, so they are suitable for different kinds of LiDAR. Their parameters are also fixed at runtime, but the IMU addition makes the system stable in a wide range of scenarios. Zheng et al proposed Traj-LO [18], which tightly couples LiDAR point geometry information with kinematic constraints on trajectory smoothing to achieve accurate and robust position estimation. The Traj-LO achieves LIO-like performance using only LiDAR and adapts to a wide range of LiDAR, as well as multi-LiDAR. In pursuit of a unified LiDAR odometry model applicable across diverse sensors and scenarios, recent years have witnessed the emergence of learning-based approaches [21], [22], [23]. Nevertheless, their strong dependence on high-performance GPUs presents significant challenges for practical deployment.

LiDAR odometry is evolving toward simpler frameworks and broader application scenarios. In this context, we propose

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

a streamlined and generalizable point cloud processing structure that is independent of LiDAR scanning types and environmental characteristics. The Onion-LO is designed to operate seamlessly across various platforms and can be used in changing scenarios, high altitude mapping, underground garages, handheld mapping, unmanned vehicles, etc. The Onion Ball is plug-and-play and can directly replace the front end of existing frameworks.

III. MATH

In this section, we describe the components of the Onion-LO system. The system pipeline is illustrated in Fig. 2. The raw point cloud is first de-distorted and processed by the Onion Ball. Onion Ball segments each cell into planar and non-planar points and calculates the onion factor of the current frame. Then, multi-threaded downsampling of each layer is performed to obtain the map points and key points. The map points are stored in the local map and used for matching with key points. Finally, robust poses are obtained by adaptive point-to-point iterative optimization.

A. Onion Ball and Onion Factor

The point cloud collected by the LiDAR in the k frame is P_k. Using the constant velocity model to compensate motion of point cloud P_k to get the de-distorted point cloud P_k^*. The derivation of the constant velocity model is described in this literature [17]. Then, construct the Onion Ball O_k for the current frame based on the point cloud P_k^*. This structure constructs a hash table using the 'Zone (Z)' and 'Distance (D)' as the key to achieve spatial partitioning for 3D point clouds, shown in Fig. 2(a). Specifically, each point $p_i \in P_k^*$ is assigned to one of six principal direction zones $(\pm X, \pm Y, \pm Z)$, based on its elevation θ_i and azimuth angles ϕ_i in spherical coordinates. Then, the layers for each zone are divided based on the distance r_i from each point to the origin, expressed as

$$\begin{aligned} &r_i = \sqrt{x_i^2 + y_i^2 + z_i^2} \quad \theta_i = \arccos\left(\frac{z_i}{d_i}\right) \quad \phi_i = \arctan 2\left(y_i, x_i\right) \\ &Z_j = \left(\pm X, \pm Y, \pm Z\right) \quad D_j = \left\lfloor \frac{r_i}{R} \right\rfloor \end{aligned} . \tag{1}$$

The distance is further discretized into layers according to the onion thickness R. To ensure adequate segmentation performance for distant points, the R is set to 5 m. Each layer $L_j\left(Z_j, D_j\right)$ corresponds to a unique hash key that maps to a set of points within its region, enabling efficient representation and rapid access. Owing to the inherent density variation in LiDAR point clouds, being denser at close range and sparser at greater distances, the voxel size v_j^v for partitioning each layer into cells should be adaptively adjusted based on the distance from the sensor, and can be expressed as

$$v_j^v = RD_j \sin\left(\delta\right), \tag{2}$$

where δ is the onion resolutions. The δ is set to 3° to suppress isolated points and ensure compatibility, as most LiDAR sensors exhibit angular resolutions finer than 3°.

We define the onion factor F_k to reflect the characteristics of the current scene, and the calculation steps are as follows. In one layer $L_j\left(Z_j, D_j\right)$, the voxel size of cells is v_j^v. Counting the number of points in each cell, defining the cell with a number less than 3 as a discrete cell, and eliminating. The volume V_j of the layer L_j can be obtained by superimposing all remaining cell volumes $v_j^v \times v_j^v \times v_j^v$. The total volume V_k of the current point cloud P_k is obtained by summing up the volumes of all the layers. Maintaining a reasonable number of key points helps ensure stable and efficient iterative optimization. The expected number of key points $N_{\exp}^{key}$ is defined as a configurable system parameter to guide the optimization process. To achieve a good trade-off between real-time performance and localization accuracy, the parameter $N_{\exp}^{key}$ is set to 1000, as determined through extensive empirical testing on diverse datasets. Then, the onion factor F_k of the current frame can be expressed as

$$F_k = \sqrt[3]{\frac{V_k}{N_{\exp}^{key}}}, \tag{3}$$

which reflects the resolution of the current key points. The onion factor is key to enabling adaptive parameter tuning.

B. Segmentation

The traditional feature extraction method is environment-dependent and difficult to adapt to different LiDAR modes. Using raw point clouds keeps geometric details and handles complex scenes better, but lacks robustness. In our previous work [24], a voxel classifier was proposed to segment the point cloud into linear, planar, and spherical points. However, this method applies a uniform voxel size to the entire point cloud, resulting in overly coarse segmentation for near points and excessively fine segmentation for distant points.

To solve this problem, we proposed a planar points segmentation model based on the Onion Ball structure, which divides the point cloud into planar and non-planar points. We choose the larger of the horizontal resolution δ_h and vertical resolution δ_v of the LiDAR as the segmentation resolution δ_s. This approach helps ensure a more uniform distribution of points within each voxel across all directions. For any one layer L_j, the voxel size of the segmentation cell can be expressed as

$$v_j^s = RD_j \sin\left(3\delta_s\right). \tag{4}$$

For each segmentation cell $c_i^j \in L_j$, only those containing more than 4 points are processed, which effectively eliminates noisy points. Then, the geometry covariance matrix $\sum_g \in \mathbb{R}^{3\times3}$ and geometry eigenvalues $\lambda_g^1, \lambda_g^2, \lambda_g^3$ are calculated. In addition to mitigating point cloud degradation property, intensity features are extracted in planar points. The intensity covariance $\sum_i \in \mathbb{R}^{1\times1}$ and intensity eigenvalues λ_i^1 can be computed based on the point cloud intensity. The judgment condition for point cloud classification can be described as

$$\begin{cases} Planar & \lambda_g^1 > th_g\lambda_g^3 \,\&\&\, \lambda_g^2 > th_g\lambda_g^3 \,\&\&\, \lambda_i^1 < th_i \\ Planar-intensity & \lambda_g^1 > th_g\lambda_g^3 \,\&\&\, \lambda_g^2 > th_g\lambda_g^3 \,\&\&\, \lambda_i^1 > th_i \\ Non-planar & else \end{cases}, \tag{5}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

where th_g is the geometry judgment threshold and is set to 10, th_i is the intensity judgment threshold is set to 50. This point cloud segmentation method is independent of both the scene and the LiDAR types. The segmentation results of different LiDAR types are shown in Fig. 3, planar points are visualized in green, planar intensity features are visualized in yellow, while non-planar points are visualized in red.

C. Downsampling

Point cloud downsampling is crucial for enabling real-time performance in the system. Among various downsampling methods, voxel-based downsampling is widely adopted in LiDAR SLAM methods [17], [25], [5] due to its ability to provide uniform sampling. However, these algorithms all require manually setting the downsampling size, and the downsampling size is fixed at runtime, which is unwise. To address the above issues, we designed a downsampling module aimed at capturing an optimal number of key points in any scene. To efficiently leverage the multi-level structure, we perform downsampling on each layer individually. For the layer L_j, the layer volume is V_j, the expected key points number N_j and the downsampling voxel size of this layer are

$$N_j = \frac{V_j}{V_k} N_{\exp}^{key}, \quad v_j^d = \sqrt[3]{\frac{V_j}{N_j}}. \tag{6}$$

Then, the key points P_j^k of the current layer are obtained. The local map requires higher resolution, so we define the expected number of map points is $N_{\exp}^{map} = 5N_{\exp}^{key}$. The map points P_j^{map} of the current layer are obtained by the same downsampling operation as above. Overlay the results of all layers to get the key points P_k^{key} and map points P_k^{map}. Our downsampling strategy ensures that the number of extracted key points remains around $N_{\exp}^{key}$, avoiding both excessive and insufficient sampling.

D. Local Map Management

We use hash tables to store voxels with a voxel size v_m like in KISS-ICP [17]. The distinction lies in two aspects: one is the use of a hierarchical storage strategy, and the other is the implementation of a distance constraint rather than relying on the maximum storage capacity to manage the local map. We store planar and non-planar points separately, so that planar key points $P_k^p \in P_k^{key}$ match the planar local map M_p and non-planar key points $P_k^{np} \in P_k^{key}$ match the non-planar local map M_{np} during data association. The onion factor F_k reflects the resolution of current frame key points, and a higher resolution local map is better for point cloud alignment. Based on extensive real-world testing, we set the save resolution of the local map to $0.5F_k$. Within each voxel block, a new point is accepted only if no other points exist within a $0.5F_k$ neighborhood. This strategy enables the local map to adaptively employ higher resolution in confined environments and lower resolution in open areas, thereby achieving real-time resolution adjustment based on environmental complexity.

E. Iterative Optimization

The iterative optimization process can be divided into two main steps: data association and residual calculation. We introduce the onion factor F_k into data association and residual calculation so that the iterative optimization parameters are changed in real time according to the current point cloud. For data association, it is common to enforce a maximum distance between corresponding points, normally set to a constant value of 1 m or 2 m [17], [15], [26]. Using a fixed search radius is neither efficient nor robust, as extreme variations in scene scale can lead to failure in finding suitable correspondences, ultimately leading to failure. For the above considerations, we define the search radius as $R_k = 3F_k$. A small search radius limits the search to the voxel block containing the source point, whereas a larger search radius expands the search range to neighboring voxel blocks centered around it, as illustrated in Fig. 2(d). We use point-to-point residuals for iterative optimization. The corresponding point is defined as the orthogonal projection of the source point onto the fitted line (for non-planar points) or plane (for planar points), shown in Fig. 2(e). The complete iterative optimization process is as follows.

For the k th frame point cloud P_k, the planar key points $P_k^p \in P_k^{key}$ and non-planar key points $P_k^{np} \in P_k^{key}$ are transferred to the global coordinate frame

$$\begin{aligned} S_k^p &= \left\{ s_i^p = \mathrm{T}_{k-1}\mathrm{T}_{pre,k}\, p_i^p \mid p_i^p \in P_k^p \right\} \\ S_k^{np} &= \left\{ s_i^{np} = \mathrm{T}_{k-1}\mathrm{T}_{pre,k}\, p_i^{np} \mid p_i^{np} \in P_k^{np} \right\} \end{aligned}, \tag{7}$$

where T_{k-1} is the global pose of the last frame, $\mathrm{T}_{pre,k}$ is the prediction pose of the current frame calculated by the constant velocity model, the S_k^p denotes planar source points, the S_k^{np} denotes non-planar source points. Then, for each planar source point $s_i^p \in S_k^p$, find the three closest points t_1^p, t_2^p, t_3^p in the planar local map M_p within the search range. The normal vector n_p of the plane is

$$n_p = \frac{\left(t_2^p - t_1^p\right) \times \left(t_3^p - t_1^p\right)}{\left\| \left(t_2^p - t_1^p\right) \times \left(t_3^p - t_1^p\right) \right\|}. \tag{8}$$

The corresponding point t_i^p for the source point s_i^p on this plane is

$$t_i^p = s_i^p - \left[\left(s_i^p - t_1^p\right) \cdot n_p \right] \cdot n_p. \tag{9}$$

For each non-planar source point $s_i^{np} \in S_k^{np}$, find the two closest points t_1^{np}, t_2^{np} in the non-planar local map M_{np}, the direction vector n_l of the line is

$$n_l = \frac{t_2^{np} - t_1^{np}}{\left\| t_2^{np} - t_1^{np} \right\|}. \tag{10}$$

The orthogonal projection of the source point s_i^{np} onto this line is

$$t_i^{np} = t_1^{np} + \left[\left(s_i^{np} - t_1^{np}\right) \cdot n_l \right] \cdot n_l. \tag{11}$$

Subsequently, planar and non-planar points are fused into the target points T_q for iterative optimization. For the q th iteration, the source points S_q and target points T_q are

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE I
ALL PARAMETERS OF OUR APPROACH

Parameter		Value
Onion Ball	Onion resolution δ	3°
	Onion thickness R	5 m
	Onion segmentation resolution δ_s	max (δ_h , δ_v)
Iterative Optimization	Expected number of key points $N_{\exp}^{key}$	1000
	Local map voxel size v_m	2 m
	Convergence criterion γ	5×10^{-4}

obtained. The confidence weight w_c for each pair of points is calculated by

$$w_c(e)=\left(\frac{F_k/3.0}{F_k/3.0+e}\right)^2, \tag{12}$$

where e is the distance residual of the point pair. Points with large residuals are assigned lower weights, effectively reducing the influence of dynamic objects and mismatched points during the iterative optimization process. Finally, the problem can be transformed into a least squares solution is expressed as

$$\Delta \mathrm{T}_q=\arg\min_{\mathrm{T}}\sum w_c\left(\left\|\mathrm{T}S_q-T_q\right\|_2\right), \tag{13}$$

where $\Delta \mathrm{T}_q$ is the correction pose of q th iteration. Iterative convergence is determined when the variation falls below γ. After iterative optimization, the corrected pose $\Delta \mathrm{T}_k$ and the current pose T_k can be expressed as

$$\Delta \mathrm{T}_k=\prod_q \Delta \mathrm{T}_q, \qquad \mathrm{T}_k=\Delta \mathrm{T}_k \mathrm{T}_{pre.k} \mathrm{T}_{k-1}. \tag{14}$$

F. Parameter

Leveraging LiDAR's inherent properties, our system uses only 6 parameters, which are listed in Table I. The first three parameters represent the fundamental attributes of the Onion Ball, while the remaining three parameters influence the speed of the iterative optimization process, ensuring both simplicity and generalizability. The onion structure maintains a proper number of key points and adjusts system parameters via the computed onion factor based on the environment. In comparison, Suma [15] has 49 parameters, CT-ICP [5] has 30 parameters, PIN-SLAM [23] has 30 parameters, Traj-LO [18] has 15 parameters. Requiring parameter reconfiguration when the scene or LiDAR type changes is impractical and inefficient. Our approach provides true out-of-the-box usability, requiring minimal configuration. Throughout all experiments in this paper, only the onion segmentation resolution δ_s is modified to accommodate different LiDAR sensors. Since replacing the LiDAR during operation is impractical, this design is justified. The remaining parameters typically require no adjustments, only being tuned when necessary for resource-limited processors.

IV. EXPERIMENT

In this section, we present comprehensive experiments on 4 public datasets and 1 private dataset to assess the performance of Onion-LO. We compare the proposed method with state-of-the-art LiDAR Odometry (LO) and LiDAR-Inertial Odometry (LIO) techniques. Additionally, we evaluate its runtime and memory usage to demonstrate computational efficiency. All experiments are conducted on a desktop PC with an Intel i7-14700K CPU, 64 GB RAM, and an NVIDIA GeForce RTX 4060 Ti GPU.

A. Dataset

We compare Onion-LO with state-of-the-art algorithms on 4 popular public datasets: the KITTI odometry dataset [27], the NTU VIRAL dataset [28], the NCLT dataset [29], and the Hilti SLAM Challenge Dataset [30]. To highlight the algorithm's strong adaptability to non-repetitive scanning solid-state LiDAR, we additionally collected the SEU-AG dataset to enhance the diversity of our experimental evaluation.

KITTI odometry dataset [27] is very popularly used in the autonomous vehicle community. The dataset contains traffic scenarios from urban environments to highways. The point cloud is collected using a Velodyne HDL64E 3D laser scanner, and the ground truth is obtained from RTK GPS/INS.

NTU VIRAL [28] is a visual-inertial-LiDAR dataset for autonomous aerial systems, which is collected by a DJI M600 Hexacopter flying at low altitude over the NTU campus. The drone carries two 16-channel OS1 gen1 LiDARs, two uEye 1221 LE cameras, and a VectorNav VN1003 rugged IMU.

NCLT dataset [29] is a large-scale, long-term autonomous dataset for robotics research, collected at the University of Michigan's North Campus. The dataset is collected through repetitive exploration of the campus, covering many challenging elements, including indoor and outdoor environments, dynamic objects, and hallways. The point cloud is collected by Velodyne HDL-32E LiDAR.

HILTI SLAM Challenge Dataset [30] is a multi-sensor SLAM dataset designed for complex real-world scenarios. It includes a variety of representative environments, such as indoor offices, laboratories, and construction sites. The dataset features challenging conditions such as point cloud degradation, textureless surfaces, and significant variations in illumination. In this work, we use only the Livox MID-70 LiDAR data from the dataset to evaluate the adaptability of our method to solid-state LiDAR sensors.

SEU-AG dataset focuses on non-repeat scanning LiDAR, collected by an intelligent electric vehicle and a DJI M350-RTK on our campus. The point cloud is collected using a LIVOX-AVIA LiDAR. The dataset contains 12 sequences, including autonomous driving scenarios, underground garage scenarios, and high-altitude downward-looking scenarios.

B. Experimental Setup

We compare Onion-LO with state-of-the-art methods, including conventional methods CT-ICP [5], A-LOAM [1], KISS-ICP [17], Traj-LO [18], and learning-based methods ELONet [21], PWCLONet [22], PIN-SLAM [23]. For the NCLT dataset, the many LiDAR-only methods are not stable, so we introduce the LIO algorithm FAST-LIO2 [19] and LIO-SAM [31] for comparison. To highlight the advantages of the Onion Ball, all comparison methods were evaluated under identical experimental conditions. Especially, parameters related to point cloud downsampling, feature extraction, local map storage, and iterative optimization were kept unchanged.

[1] https://github.com/HKUST-Aerial-Robotics/A-LOAM

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE II
RTE RESULTS (%) ON KITTI ODOMETRY BENCHMARK

Approach	Type	00	01	02	03	04	05	06	07	08	09	10	avg
CT-ICP [5]	dense points	**0.49**	0.76	**0.52**	0.72	0.39	**0.25**	0.27	0.31	0.81	0.49	**0.48**	**0.49**
A-LOAM[1]	feature points	0.93	0.86	0.83	0.98	0.67	0.55	0.49	0.45	1.37	0.88	1.15	0.83
KISS-ICP [17]	dense points	0.53	0.71	0.53	0.66	0.35	0.31	**0.26**	0.33	0.84	0.49	0.57	0.50
Traj-LO [18]	dense points	0.50	0.81	**0.52**	0.67	0.40	**0.25**	0.27	**0.30**	0.81	**0.45**	0.55	0.50
PWCLONet [22]	supervised	0.78	0.67	0.86	0.76	0.37	0.45	0.27	0.60	1.26	0.79	1.69	0.77
ELONet [21]	supervised	0.83	**0.55**	0.71	**0.49**	**0.22**	0.34	0.36	0.46	1.14	0.78	0.80	0.61
PIN-SLAM [23]	neural implicit	0.55	0.68	0.54	0.76	**0.22**	0.30	0.35	0.34	**0.80**	0.54	0.50	0.51
Ours	dense points	0.52	0.92	0.56	0.66	0.32	0.30	**0.26**	0.31	0.81	0.50	0.49	0.51

The **bold** values represent the best results in each sequence.

TABLE III
RMSE OF ATE (M) ON THE NTU VIRAL DATASET

Approach	eee01	eee02	eee03	nya01	nya02	nya03	sbs01	sbs02	sbs03	avg
CT-ICP [5]	7.947	0.332	9.638	0.372	0.443	0.288	×	0.327	1.474	3.170
A-LOAM[1]	0.257	0.309	0.233	0.209	**0.174**	0.162	0.259	**0.212**	0.554	0.263
KISS-ICP [17]	2.515	1.581	1.144	0.885	3.028	1.207	1.091	1.133	1.041	1.513
Traj-LO [18]	0.208	0.123	**0.193**	0.209	0.186	0.191	0.214	0.219	0.234	0.197
PIN-SLAM [23]	3.543	1.249	1.254	0.213	0.196	0.172	×	1.342	0.373	1.042
Ours	**0.206**	**0.117**	0.206	**0.204**	0.186	**0.161**	**0.211**	0.217	**0.197**	**0.189**

The **bold** values represent the best results in each sequence, '×' denotes that the algorithm fails in this sequence.

TABLE IV
RMSE OF ATE (M) ON THE NCLT DATASET

Sequence data	Duration	**Ours**	KISS-ICP [17]	FAST-LIO2 [19]	LIO-SAM [31]	PIN-SLAM [23]
20120108	1 hr:25 min:35 sec	**2.13**	×	7.52	21.66	16.38
20120115	1 hr:52 min:19 sec	2.88	**1.47**	2.19	×	27.92
20120429	43 min:18 sec	1.43	**0.92**	1.84	5.67	9.37
20120511	1 hr:25 min:5 sec	**1.27**	×	2.48	×	7.42
20120615	55 min:10 sec	**1.14**	2.11	2.23	×	6.63
20130110	17 min:4 sec	**0.96**	4.60	1.47	5.08	2.24
avg		**1.63**	2.27	2.95	10.80	11.66

The **bold** values represent the best results in each sequence, '×' denotes that the algorithm fails in this sequence.

TABLE V
RMSE OF ATE (M) ON THE NON-REPETITIVE LIDAR DATASET

Approach	Unmanned Aerial Vehicle (SEU-AG)						Unmanned Ground Vehicle (SEU-AG)						Handheld Device (HILTI 2021)					
	A01	A02	A03	A04	A05	A06	G01	G02	G03	G04	G05	G06	RPG	Base1	Base4	Lab	Cons2	Camp2
KISS-ICP [17]	3.66	4.42	11.79	5.18	2.96	2.58	20.06	**0.61**	11.04	×	1.61	×	3.72	6.85	0.29	×	21.65	3.61
Traj-LO [18]	1.76	×	41.62	×	×	2.65	**0.95**	1.13	3.89	0.76	**0.30**	0.82	**0.22**	0.31	**0.06**	×	**0.13**	**0.05**
Livox mapping[2]	×	×	×	×	×	×	14.90	18.26	1.25	0.65	40.48	2.35	4.14	0.21	0.92	×	11.25	2.11
PIN-SLAM [23]	×	×	13.8	×	×	×	17.86	2.63	5.49	4.73	1.23	1.42	5.83	×	1.12	×	×	3.42
Ours	**1.37**	**4.18**	**1.76**	**1.43**	**2.74**	**1.72**	1.19	**0.78**	**0.77**	**0.49**	0.59	**0.52**	1.62	**0.17**	**0.06**	×	0.51	0.14

The **bold** values represent the best results in each sequence, '×' denotes that the algorithm fails in this sequence.

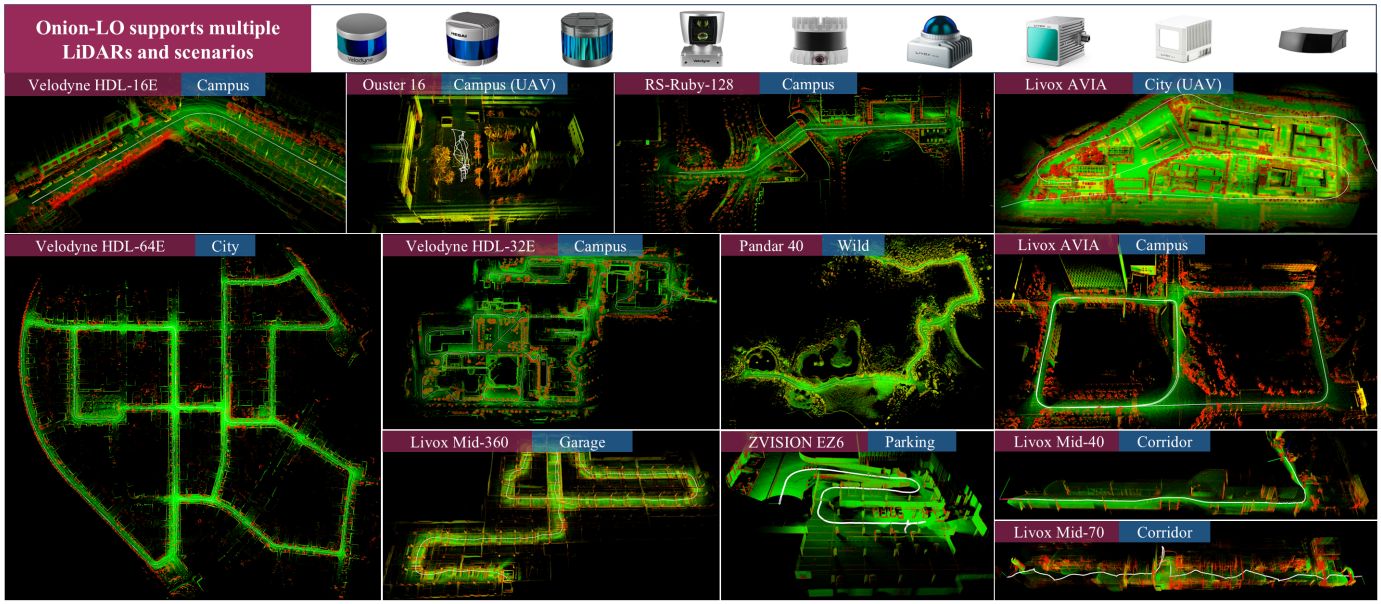

Fig. 3. The segmentation maps generated by Onion-LO. The numbers above each subfigure correspond to the indices in Table VI. The purple boxes are LiDAR types, and the blue boxes are scenario types. Planar points are green, intensity features in planar points are yellow, and non-planar points are red.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

For quantitative analysis, we use Relative Translation Error (RTE) on the KITTI dataset, and employ the evo toolkit [32] to obtain the Root Mean Square Error (RMSE) of Absolute Trajectory Error (ATE) on other datasets.

C. Comparison of Public Datasets

1) KITTI: The results of the KITTI-RAW dataset are listed in Table II. Our method achieves competitive accuracy to KISS-ICP and Traj-LO, with an average RTE of 0.50%. And learning-based methods perform well on the KITTI dataset but do not demonstrate a clear advantage. While Onion-LO does not achieve the best performance on the KITTI dataset, it is designed for general use across diverse scenarios rather than being tuned for a specific dataset or LiDAR. This ensures better generalization in real-world applications.

2) NTU: The results are shown in Table III. The NTU dataset is a different research scenario, leading to many algorithms (CT-ICP, KISS-ICP, PIN-SLAM) that perform well on the KITTI dataset, but exhibit unstable performance on the NTU dataset. This is because changes in LiDAR type and point cloud distribution led to system parameters no longer being optimally configured. PIN-SLAM exhibits large errors in the Z-axis direction, leading to unstable overall performance. The Onion-LO is stable and performs well in all sequences, and the average ATE error is only 0.189 m.

3) NCLT: The NCLT dataset is a long-term dataset, which has high requirements on the robustness and real-time performance of the odometry. Many LiDAR-only methods cannot operate stably to complete the whole sequence. The comparison results are presented in Table IV, only Onion-LO, FAST-LIO2, and PIN-SLAM have no failures in all sequences. KISS-ICP performance is very erratic, which is due to the presence of extremely narrow corridor scenes within the sequence. Our method performs best in the NCLT dataset with an average ATE error of 1.63 m, which validates the stability of our algorithm in long-duration operational tasks.

D. Evaluation on Non-Repetitive LiDAR

To validate the suitability of our method for non-repetitive scanning LiDAR, we do comparative experiments with KISS-ICP [17], Traj-LO [18], Livox_mapping[2], and PIN-SLAM [23] on the SEU-AG and HILTI datasets. The experiment scenarios include high-altitude mapping, autonomous driving scenarios, and handheld scenarios. The results are listed in Table V.

In high-attitude aerial downward-looking scenarios, the point cloud tends to be highly sparse with broad spatial coverage, introducing significant challenges for LiDAR odometry. Livox_mapping fails to operate reliably, as its feature extraction method is not suitable for highly sparse point clouds. Traj-LO and PIN-SLAM struggle with instability due to the lack of altitude-aware adaptation in downsampling and local map storage strategies. Our method maintains stable performance in this scenario due to the Onion Ball structure. Based on Eq. (2) and Eq. (4), larger voxels are used to encapsulate distant points, effectively preventing the algorithm from failing to adapt to sparse long-range data. The map of sequence A05 generated by our algorithm is shown in Fig. 3(10), where it can be observed that the segmentation performance remains unaffected. For ground driving sequences, most algorithms perform well in this common scenario. Sequences 03, 04, and 06 include transitions from outdoor environments into underground parking structures, where KISS-ICP fails due to its inability to adapt to such drastic scene changes.

[2] https://github.com/Livox-SDK/livox_mapping

TABLE VI
MAIN PARAMETERS OF ONION-LO

No.	LiDAR Type	Scene	t_o / *ms*	t / *ms*	N_{key}	F_k / *m*
1	Velodyne HDL-16E	Campus	2.7	31.6	1445	1.6
2	Velodyne HDL-32E	Campus	2.2	42.4	1272	1.5
3	Velodyne HDL-64E	City	8.3	27.1	1035	1.8
4	Ouster 16	Campus	1.4	21.4	1232	1.9
5	Pandar 40	Wild	8.6	32.0	1227	2.9
6	RS-Ruby-128	Campus	9.2	21.9	1213	3.3
7	Livox Mid-40	Corridor	2.4	26.5	903	0.2
8	Livox Mid-70	Corridor	0.9	4.4	712	0.6
9	Livox Mid-360	Garage	1.5	46.3	1217	0.5
10	Livox AVIA	City	1.7	34.3	1202	3.2
11	Livox AVIA	Campus	1.6	23.1	1242	1.5
12	ZVISION EZ6	Parking	9.2	21.7	991	2.6
Average			4.1	27.7	1140	/

TABLE VII
PROCESSING TIME PER LIDAR FRAME

Dataset	Points number	Onion time Mean ± STD (ms)	Total time Mean ± STD (ms)	Max memory usage (MB)
KITTI	133,000	6.8 ± 1.5	31.3 ± 9.7	809.36
NTU	16,384	1.6 ± 0.8	26.0 ± 15.3	142.38
NCLT	69,500	2.1 ± 0.8	47.3 ± 26.0	855.59
HILTI	10,000	0.8 ± 0.2	9.1 ± 6.5	187.36
SEU-A	24,000	1.9 ± 0.8	34.2 ± 14.3	327.01
SEU-G	24,000	2.3 ± 1.2	25.3 ± 12.2	381.58

For handheld mapping scenarios, the performance of our algorithm is second only to Traj-LO. As Traj-LO leverages point cloud partitioning and continuous-time motion modeling to enable precise motion compensation for handheld scanning. Due to many point cloud degradation scenarios in the HILTI dataset, LiDAR-only odometry generally exhibits unstable performance. Our method maintains separate storage for planar and non-planar points and further extracts intensity features from degeneration-prone planar regions, thereby effectively mitigating the impact of point cloud degeneration.

E. Generalization Performance Analysis

To further explore and validate the superiority of the Onion Ball structure. We use 12 sequences to analyze, which are from different scenarios or LiDARs. The segmentation results are shown in Fig. 3, which demonstrates consistent and accurate planar/non-planar separation across all sequences, validating the generalization ability of the proposed method. The main parameter variations of the system are listed in Table VI. The average processing time t_o of the Onion Ball in all sequences is within 10ms. Even with a 128-line LiDAR, the processing time per frame t is only 21.9 ms, and the Onion Ball takes only 9.2 ms. The onion factor F_k exhibits a positive correlation with the scale of the environment. Hence, the key points of all sequences vary around 1000, which is consistent with our design philosophy that "No matter what the scenario type and LiDAR type, the number of key points is always within a reasonable range."

F. Computational Efficiency

This section presents a computational efficiency analysis of the Onion-LO, focusing on runtime performance and memory

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

usage. The results are shown in Table VII, our method meets real-time requirements at 10 HZ in all datasets. The processing time of the Onion Ball increases with the number of input points, yet consistently remains below 10 ms, highlighting the advantages of its multi-layered structure and parallel processing scheme. In the long-term, large-scale NCLT dataset, Onion-LO is still able to work in real-time with an average processing time of 47.3 ms. As for memory usage, even under high-resolution, long-duration, and large-scale scenarios, our algorithm consistently maintains a memory usage below 1 GB, demonstrating excellent computational efficiency and scalability. Benefiting from its lightweight design, our method achieves real-time performance on embedded platforms such as the NVIDIA Jetson series, whereas the PIN-SLAM operates at approximately 4 Hz on a NVIDIA GeForce RTX 4060Ti GPU with 64 GB RAM.

V. Conclusion

This paper addresses the generalization challenges in LiDAR odometry across varying scenarios and sensor types. We propose Onion-LO, a hierarchical framework inspired by the LiDAR emission model, which adapts automatically to different environments by analyzing the structural properties of point clouds. The experiment results demonstrate that Onion-LO achieves accurate, robust, and consistent performance across various LiDAR types and challenging environments. The Onion-LO supports long-term, large-scale, and high-density applications, and runs in real time on resource-constrained platforms. Compared to learning-based methods, our approach offers better generalization and higher computational efficiency. The code is open-sourced, and we believe Onion-LO has the potential to serve as a new baseline for LiDAR odometry. The "Onion Ball" structure presents numerous unexplored possibilities that merit further investigation. In future work, we plan to extend the Onion Ball framework to tasks such as loop closure detection and dynamic object detection.

References

[1] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)., 2018, pp. 4758–4765.

[2] H. Li, B. Tian, H. Shen, and J. Lu, "An intensity-augmented LiDAR-inertial SLAM for solid-state LiDARs in degenerated environments," IEEE Trans. Instrum. Meas., vol. 71, pp. 1–10, 2022.

[3] J. Lin and F. Zhang, "Loam_Livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2020, pp. 3126–3131.

[4] S. Yi, Y. Lyu, L. Hua, Q. Pan, and C. Zhao, "Light-LOAM: A lightweight LiDAR odometry and mapping based on graph-matching," IEEE Robot. Autom. Lett., vol. 9, no. 4, pp. 3219–3226, 2024.

[5] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP: Real-time elastic LiDAR odometry with loop closure," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA)., 2022, pp. 5580–5586.

[6] Y. Pan, Y. He, Z. Shao, and Z. Li, "MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA)., 2021, pp. 11633–11640.

[7] H. Li et al., "MARS-LVIG dataset: A multi-sensor aerial robots SLAM dataset for LiDAR-visual-inertial-GNSS fusion," Int. J. Robot. Res., vol. 43, no. 8, pp. 1114–1127, 2024.

[8] J. Lin and F. Zhang, "R(3)LIVE++: A robust, real-time, radiance reconstruction package with a tightly-coupled LiDAR-inertial-visual state estimator," IEEE Trans. Pattern Anal. Mach. Intell., vol. 46, no. 12, pp. 11168–11185, Dec. 2024.

[9] M. Zhang et al., "SOAR: Simultaneous exploration and photographing with heterogeneous UAVs for fast autonomous reconstruction," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2024, pp. 10975–10982.

[10] Y. Tian et al., "Resilient and distributed multi-robot visual SLAM: Datasets, experiments, and lessons learned," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2023, pp. 11027–11034.

[11] F. Zhu et al., "Swarm-LIO2: Decentralized efficient LiDAR-inertial odometry for aerial swarm systems," IEEE Trans. Robot., vol. 41, pp. 960–981, 2025.

[12] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," Auton. Robots, vol. 41, no. 2, pp. 401–416, 2016.

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

[14] K. Honda et al., "Generalized LOAM: LiDAR odometry estimation with trainable local geometric features," IEEE Robot. Autom. Lett., vol. 7, no. 4, pp. 12459–12466, 2022.

[15] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in Proc. Robot. Sci. Syst. (RSS), 2018.

[16] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-LiDAR-inertial odometry and mapping," IEEE Robot. Autom. Lett., vol. 6, no. 3, pp. 5167–5174, 2021.

[17] I. Vizzo et al., "KISS-ICP: In defense of point-to-point ICP – Simple, accurate, and robust registration if done the right way," IEEE Robot. Autom. Lett., vol. 8, no. 2, pp. 1029–1036, 2023.

[18] X. Zheng and J. Zhu, "Traj-LO: In defense of LiDAR-only odometry using an effective continuous-time trajectory," IEEE Robot. Autom. Lett., vol. 9, no. 2, pp. 1961–1968, 2024.

[19] W. Xu et al., "FAST-LIO2: Fast direct LiDAR-inertial odometry," IEEE Trans. Robot., vol. 38, no. 4, pp. 2053–2073, 2022.

[20] D. He et al., "Point-LIO: Robust high-bandwidth light detection and ranging inertial odometry," Adv. Intell. Syst., vol. 5, no. 7, Art. no. 2200459, 2023.

[21] G. Wang et al., "Efficient 3D deep LiDAR odometry," IEEE Trans. Pattern Anal. Mach. Intell., pp. 1–17, 2022.

[22] W. Wang et al., "Pwclo-net: Deep LiDAR odometry in 3D point clouds using hierarchical embedding mask optimization," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2021, pp. 15910–15919.

[23] Y. Pan et al., "PIN-SLAM: LiDAR SLAM using a point-based implicit neural representation for achieving global map consistency," IEEE Trans. Robot., vol. 40, pp. 4045–4064, 2024.

[24] X. Cheng et al., "SVM-LO: An accurate, robust, real-time LiDAR odometry with segmentation voxel map for autonomous vehicles," in Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC), 2024, pp. 1870–1877.

[25] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter," IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 3317–3324, 2021.

[26] I. Vizzo et al., "Poisson surface reconstruction for LiDAR odometry and mapping," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2021, pp. 5624–5630.

[27] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2012, pp. 3354–3361.

[28] Y. S. Nguyen et al., "NTU VIRAL: A visual-inertial-ranging-LiDAR dataset from an aerial vehicle viewpoint," Int. J. Robot. Res., vol. 41, pp. 270–280, 2022.

[29] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan North Campus long-term vision and LiDAR dataset," Int. J. Robot. Res., vol. 35, no. 9, pp. 1023–1035, 2015.

[30] M. Helmberger et al., "The Hilti SLAM Challenge Dataset," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7518–7525, 2022.

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

[32] D. Schubert et al., "The TUM VI benchmark for evaluating visual-inertial odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2018, pp. 1680–1687.