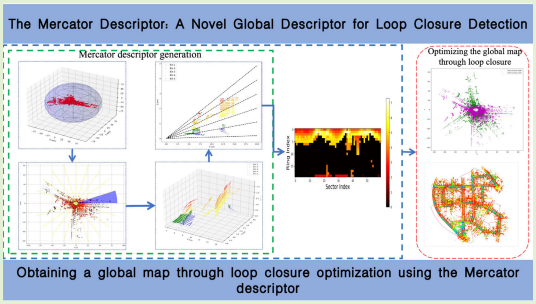
Mercator Descriptor: A Novel Global Descriptor for Loop Closure Detection in LiDAR SLAM

Zhibo Wang, Donghai Xie, Yu Wu, Haiyang Wu, Xingyu Qi, Daoping Huang, Yibo Fu, and Ruofei Zhong

***Abstract*—Loop closure detection is a fundamental technique in simultaneous localization and mapping (SLAM), playing a crucial role in the map-building process. Inspired by the Mercator projection and depth maps, this article presents a novel loop closure detection method for LiDAR SLAM, termed the Mercator descriptor. Our approach applies the Mercator projection principles to the point cloud data captured by the sensor in each frame and extracts depth information to generate descriptors. Additionally, we utilize a two-step identification strategy, from coarse to fine, to improve the accuracy and recall of loop closure detection. Testing on public datasets demonstrates that our algorithm achieves an average maximum $F1$ score of 94.57%, performing well across various datasets. Notably, we have integrated our algorithm into existing SLAM frameworks, and results from multiple datasets show that our algorithm can perform real-time and efficient map optimization. To share our findings and contribute to the community, we have open-sourced our code on GitHub: https://github.com/wangzika/Mercator-Descriptor.git.**



***Index Terms*—Laser radar, location awareness, loop closure detection, sensors, simultaneous localization and mapping, visualization.**

I. Introduction

IN THE field of robotics, simultaneous localization and mapping (SLAM) [1], [2], [3], [4] is a fundamental challenge, requiring the precise estimation of a robot's position and orientation within an unknown environment while simultaneously generating a map of the surroundings. The front-end of SLAM is dedicated to obtaining the relative transformation between two frames to initialize the robot's pose, while the back-end focuses on overall optimization based on the initial values obtained from the front-end. However, relying solely on odometry and considering only adjacent keyframes inevitably leads to the accumulation of errors over time. This cumulative error can result in significant drift within the SLAM system, thereby impacting the creation of globally consistent trajectories and maps.

In the field of SLAM, point cloud place recognition [5], [6] and loop closure detection [7], [8] are particularly crucial, especially during long SLAM operations. This complex challenge revolves around determining whether data captured by sensors at different time instances but from the same spatial coordinates (such as camera images or LiDAR point clouds) can be accurately associated with the same environmental context. Addressing this challenge is paramount for mitigating positional drift and inconsistencies encountered during map construction. Traditional vision-based SLAM systems [9], [10], [11], [12], [13] typically rely on camera images for loop closure detection. While these methods offer rich visual information, they are susceptible to environmental changes such as variations in lighting conditions, object appearance, and viewpoint, which can significantly alter captured images, thus impeding accurate loop closure detection. In contrast, LiDAR sensors directly capture the 3-D structure of the environment, providing rich geometric information that remains consistent even under varying lighting conditions. An effective LiDAR-based place recognition solution should meet several key requirements: first, it should maintain rotation and translation invariance regardless of changes in viewpoint; second,

Received 15 September 2024; accepted 17 September 2024. Date of publication 26 September 2024; date of current version 31 October 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3904101 and Grant 2022YFB3902200, and in part by the National Natural Science Foundation of China under Grant 42071318. The associate editor coordinating the review of this article and approving it for publication was Prof. Ajay G. P. Kottapalli. *(Corresponding authors: Donghai Xie; Yu Wu.)*

Zhibo Wang, Donghai Xie, Haiyang Wu, Xingyu Qi, Daoping Huang, Yibo Fu, and Ruofei Zhong are with the College of Resource Environment and Tourism, Capital Normal University, Beijing 100048, China (e-mail: 2220902181@cnu.edu.cn; xiedonghai@cnu.edu.cn).

Yu Wu is with the Institute of Surface-Earth System Science, School of Earth System Science, Tianjin University, Tianjin 300072, China (e-mail: wu_yu@tju.edu.cn).

Digital Object Identifier 10.1109/JSEN.2024.3464855

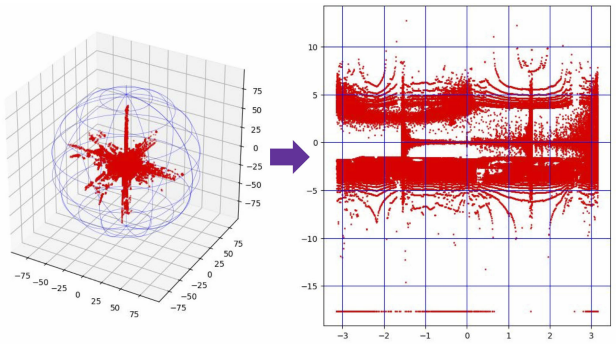


Fig. 1. Application of Mercator projection principles to point clouds.

the method should demonstrate robustness to different LiDAR point cloud densities and diverse environments, considering the sparse nature of LiDAR point clouds that vary with distance, scenes, and LiDAR types.

To address these challenges and enhance the performance of LiDAR in place recognition, we introduce a novel global descriptor inspired by the Mercator projection method, preserving the depth information of point clouds, termed the Mercator descriptor. This descriptor applies a mercator-like projection method to a point cloud frame, as illustrated in Fig. 1, generating a rotation-invariant descriptor and subsequently extracting depth information from each region. This approach effectively captures the fundamental geometric structure of the environment while preserving the corresponding depth information. Subsequently, we construct a database of these descriptors and achieve efficient loop closure detection through matching historical and current frames. Our research primarily contributes.

The main contributions of this article are as follows.

1) We propose a robust and accurate descriptor that applies the Mercator principle to generate grids within point clouds, incorporating depth information extracted from different grid cells. This method not only considers the spatial distribution characteristics of each point cloud frame but also preserves depth information.
2) Leveraging a kd-tree, we establish our database. Each current frame is paired with a latitude key featuring rotational invariance, generated using the Mercator descriptor. Subsequently, a two-stage search is conducted for loop closure candidate frames utilizing these latitude keys, circumventing the need to retrieve all historical frames.
3) Comparative experiments were conducted between our method and several classic methods. Experimental results show that our method achieves commendable results
4) Seamless integration into existing front-end odometry. Experimental evaluation confirms that our method promotes a substantial reduction in cumulative errors and an improvement in mapping accuracy.

II. Related Works

Loop closure detection, as an indispensable component of the SLAM framework, serves the crucial purpose of determining whether previously visited locations are revisited within a given environment. By identifying these loops, loop closure detection aims to correct the accumulated drift in robot pose estimation. Typically, loop closure detection can be categorized into two types: traditional methods and deep learning-based approaches.

A. Traditional Loop Closure Detection Methods

In the early stages of SLAM development, loop closure detection primarily relied on feature extraction techniques such as SIFT [14], SURF [15], BRIEF [16], and ORB [17] for loop matching. These methods facilitated feature matching across disparate images and utilized geometric transformations to verify loop pairs. Subsequent advancements in SLAM methods introduced loop closure detection approaches grounded in the bag-of-words (BoWs) [18] model. This paradigm involves clustering image features into visual words, encoding their occurrences within a BoWs vector, and utilizing similarity metrics derived from these vectors for loop closure detection. However, the effectiveness of visual images in loop closure detection is challenged by vulnerabilities to variations in lighting conditions and perspectives, which can compromise their robustness.

As the cost of LiDAR equipment has decreased and its capability to capture comprehensive spatial information surpasses that of visual images, there has been a proliferation in the development of point cloud-based loop closure detection methods. M2DP [19] employs a technique where 3-D point clouds are projected onto multiple 2-D planes, yielding density signatures for points within each plane. These signatures' left and right singular vectors serve as descriptors for loop closure detection. DELIGHT [20] relies exclusively on LiDAR for global positioning, leveraging reflected intensity data instead of conventional distance metrics for place recognition. The algorithm consists of two stages: initial estimation based on a density descriptor and subsequent geometry-based verification, effectively addressing place recognition challenges. Scan context (SC) [21], [22] enhances discriminability by encoding the maximum height of point clouds within various regions to generate a 2-D global descriptor, albeit at the expense of increased computational time for matching. Intensity SC [23] integrates both geometric and intensity information of point clouds to devise a novel global descriptor for place recognition. LiDAR-iris descriptor [24] employs multiple LoG-Gabor filtering and thresholding steps to generate binary signature images for each point cloud. Loop pairs similarity is assessed by computing the Hamming distance between binary signature images from two point clouds. Stable triangle descriptor (STD) [25] adeptly extracts local keypoints from 3-D point clouds by leveraging the rigidity invariance of triangles, encoding these keypoints into triangle descriptors for place recognition based on matching side lengths among other parameters. FSLAM [26] proposes a loop closure algorithm that fuses visual and scanning information, uses depth features for loop detection, and then combines camera and LiDAR data for loop verification. BoW3D [27], a BoWs algorithm centered on 3-D features, constructs the BoWs using the LinK3D [28] feature. The adoption of a hash table as the database structure facilitates efficient loop retrieval. RLS-LCD [29] proposes a

novel loop closure detection algorithm tailored for rotating LiDAR scans. It utilizes subimage-based recognition and a novel lightweight global descriptor to mitigate the impact of large viewpoint variations inherent in rotating LiDAR scans.

B. Deep Learning-Based Loop Closure Detection Methods

With the rise of deep learning, an increasing number of SLAM methods have delved into the integration of deep learning models for loop closure detection. These approaches utilize deep learning architectures to acquire more robust feature representations from image or point cloud data, thereby augmenting the accuracy and resilience of loop closure detection mechanisms. SegMatch [30] introduced a place recognition technique predicated on segmenting 3-D point clouds, striking a balance between conventional methods reliant on local and global features for scene identification. This method achieves real-time and dependable loop closure detection. Subsequently, SegMap [31] embraced the segmentation-based paradigm of SegMatch and proposed a 3-D point cloud segmentation descriptor underpinned by deep learning, thereby enhancing localization performance while also extracting semantic insights. PointNetVLAD [32] amalgamates PointNet [33] and NetVLAD [34] in an end-to-end fashion to extract global descriptors conducive to place recognition. FusionVLAD [35] encodes multiview representations of sparse 3-D point clouds into view-independent global descriptors. The system consists of two parallel branches: a spherical view branch for orientation-invariant feature extraction and a top-down view branch for translation-insensitive feature extraction. OverlapNet [36] leverages deep neural networks to discern loop closures utilizing diverse cues derived from LiDAR data. It identifies loops by estimating image overlap and furnishing an approximation of the relative yaw angle between scan pairs through generalized-to-distance images. Their subsequent endeavor introduced a lightweight network, OverlapTransformer [37]. DeLightLCD [38] proposes a very deep and lightweight neural network for loop closure detection. The architecture of this network contains two key modules: a twin feature extraction module and a feature difference module based on dual attention. CVTNet [39] proposes a cross-view transformer-based network to fuse the range image view (RIV) and bird's-eye view (BEV) generated from LiDAR data.

However, due to the requirement of expensive equipment and long training time, deep learning methods are currently not suitable for real-time loop closure detection. Meanwhile, in traditional methods, M2DP lacks rotational robustness; SC, IntensityScan-Context, and LiDAR-iris are restricted by the generation of descriptor methods, resulting in a significant reduction in useful information obtained in certain narrow street scenarios, thereby affecting the accuracy of loop closure detection; the computational complexity of DELIGHT makes it unable to handle real-time processing; while BoW3D and STD rely too much on the extraction of feature points and corners, making them overly sensitive to environmental information in point clouds. Therefore, exploring a more mature loop closure detection descriptor method appears to be particularly important.

III. Methodology

In this section, we will provide a detailed exposition of our work. We begin by introducing the concept of the Mercator projection. Building upon this foundation, we further elaborate on the fundamental principles that underpin our approach to Mercator projection generation. Finally, we describe a carefully designed two-stage method aimed at simplifying the process of identifying candidate keyframes for loop closure detection. The entire process of loop closure detection using Mercator descriptors is illustrated in Fig. 2.

A. Mercator Projection

The Mercator projection, also known as the Mercator map projection or the rectilinear cylindrical equiangular projection, is a conformal cylindrical map projection method.

This projection is named after the geographer Gerardus Mercator from Flanders. In maps produced using this projection, the Earth is unfolded onto a plane, with meridians and parallels intersecting at right angles everywhere, allowing the world map to be drawn within a rectangle. Every point on the map has equal lengths in all directions, facilitating accurate representation of bearings between any two points. In the Mercator projection, the linear scale remains constant around any given point on the map, preserving the angles and shapes of continental outlines postprojection (a characteristic commonly referred to as conformality or equiangularity). However, the Mercator projection distorts areas, with the least distortion occurring near the equator and the greatest distortion occurring at the poles. Yet, because there is minimal distortion between the Tropic of Cancer and the Tropic of Capricorn, which encompasses most shipping routes, it is widely used for map making.

B. Generation Principle of Mercator Descriptor

Inspired by the fundamental principles of the Mercator projection, we propose an innovative global descriptor called Mercator descriptor. Grounded in the principles of the Mercator projection, this descriptor involves the equiangular projection of Earth's latitudinal and longitudinal grid onto a cylindrical framework—akin to enveloping a transparent cylinder around the Earth and projecting the grid onto it while preserving angular relationships. Extending this principle to the domain of point cloud data processing, we introduce Mercator descriptors as a novel approach. The specific process is illustrated in Fig. 3.

In generating the Mercator descriptor, our initial step involves converting the coordinates of the LiDAR point cloud into azimuth and elevation angles. This process aims to enhance our understanding and processing of the point cloud data. Such segmentation not only facilitates the processing of point cloud data but also serves as a foundational stage for subsequent feature extraction and descriptor construction tasks.

Subsequently, we partition the point cloud into discrete, nonoverlapping sectors based on the azimuth and elevation angles of each point. This partitioning is performed within the LiDAR-based coordinate system, predominantly preserving

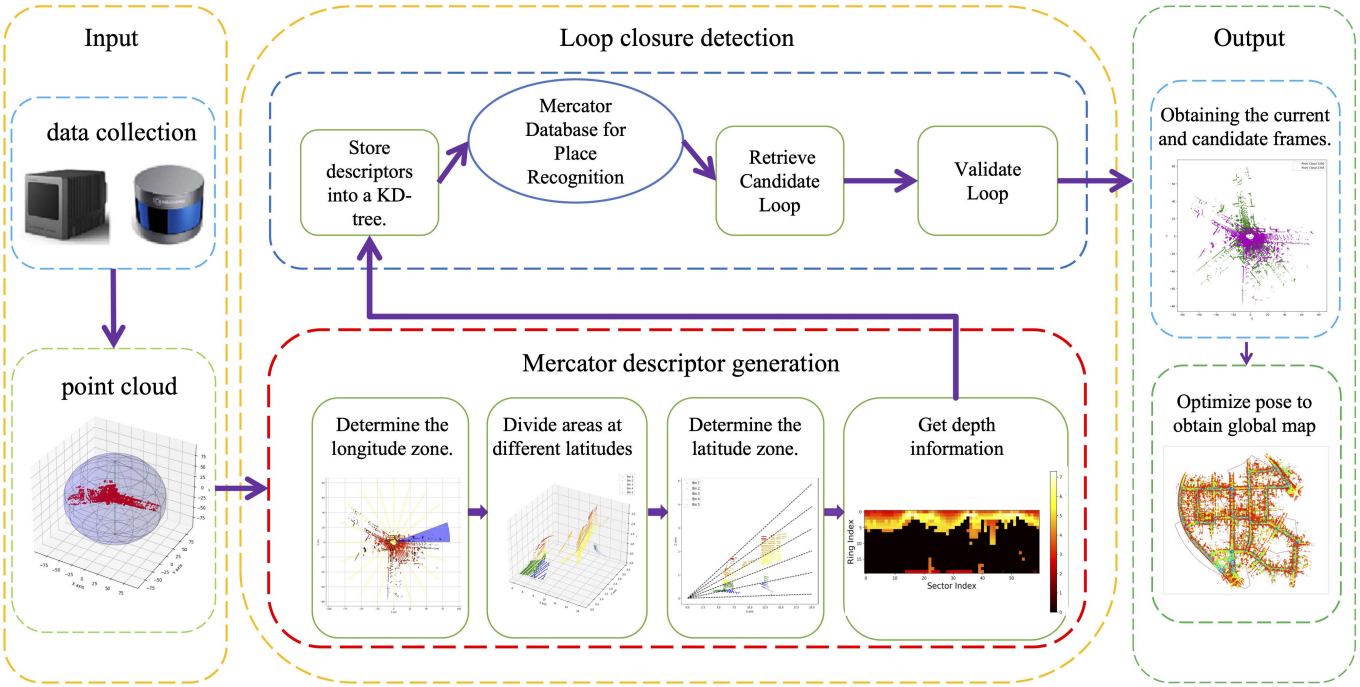


Fig. 2. Workflow of our system mainly consists of four modules: input, Mercator descriptor generation, loop closure detection, output. We embed our algorithm to the loop closing thread, which closes the loops and obtains the globally consistent SLAM results.

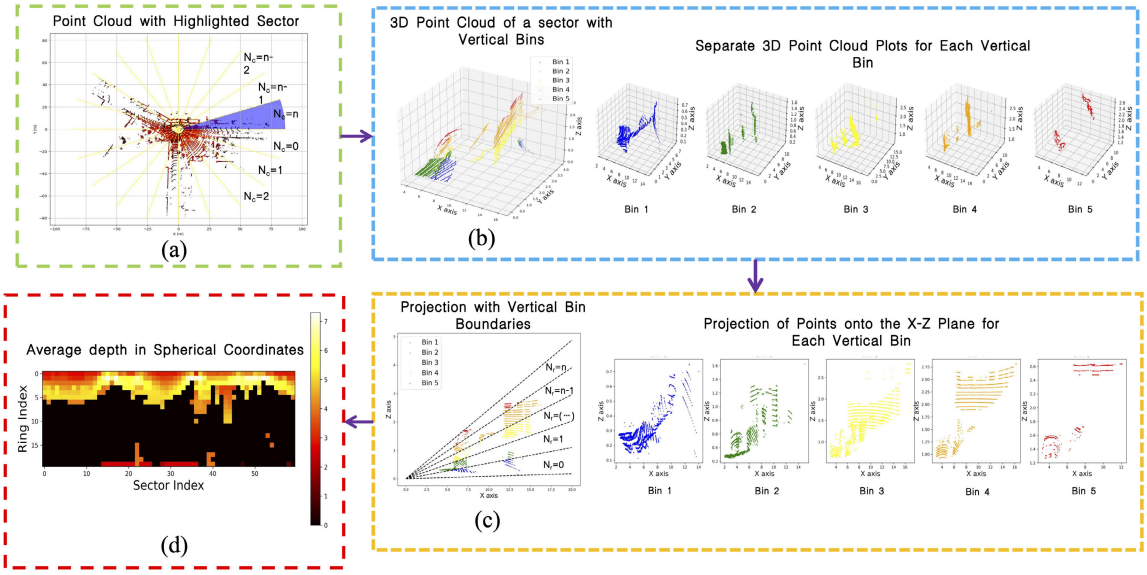


Fig. 3. Entire diagram illustrates the principle behind Mercator descriptor generation. First, (a) represents the perspective of the point cloud from a top-down view, divided into mutually exclusive sectors based on azimuth angles. Subsequently, (b) demonstrates the selection of the blue region from (a), which is further subdivided into different colored bins based on vertical angles. (c) Vertical cross-sectional view of each bin is depicted. Finally, (d) illustrates the extraction of depth information for each region based on the previously obtained bins to generate descriptors.

the inherent equiangular properties of the Mercator projection throughout the entire process.

More specifically, our method comprises three primary stages for each frame: data preprocessing, regional division stage, and depth information extraction.

1) Data Preprocessing: In our method, we avoid indiscriminate processing of all points within each point cloud frame. Instead, we adopt a targeted approach by delineating a specific region of interest for point cloud processing. To achieve this, we first define a spherical area centered around the LiDAR's location with a specific radius R, thereby establishing an effective sensing range. Points that are located outside this sensing range and considered too distant and thus ineffective are systematically excluded.

Upon receiving a point cloud data frame labeled P, we extract the positional information of each point within the cloud relative to the LiDAR's coordinate system, specifically marked as x, y, z, with each point represented as p_k. After determining the spatial coordinates of each point, our main objective is to maintain a spherical region centered around the LiDAR's location, with a predefined radius R, serving as the effective sensing area. Subsequently, points identified as being

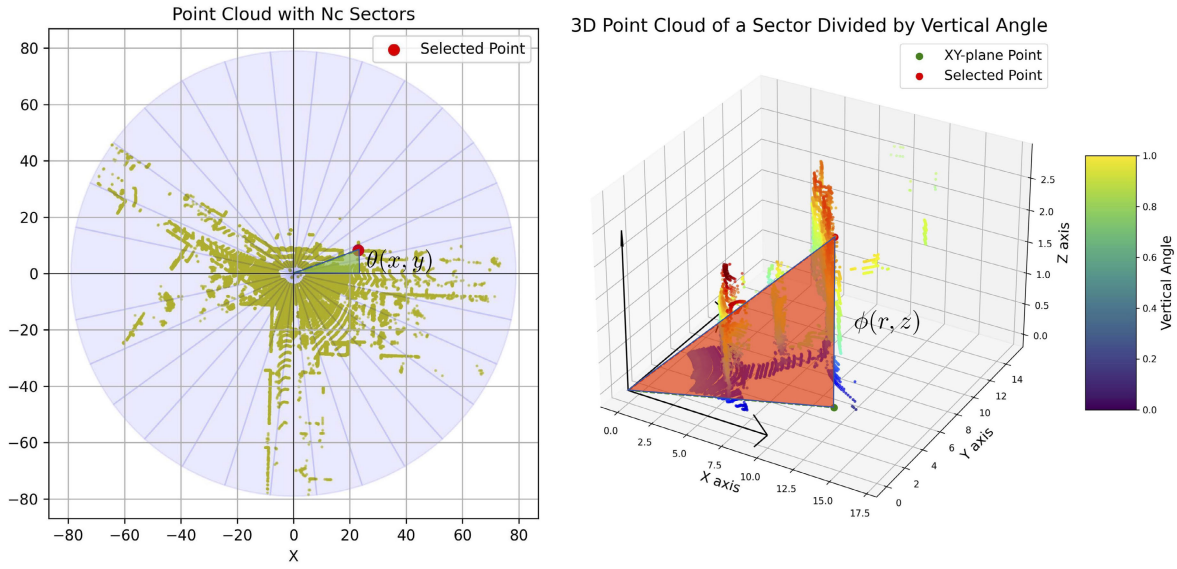


Fig. 4. Figures illustrate the detailed process of region partitioning. Approach to handling longitude regions, where we compute the angle of each point and then assign it to different regions based on the corresponding angle (left). Method for handling latitude regions, where different colors represent different magnitudes of elevation angles (right). Darker colors indicate smaller angles, while warmer colors represent larger angles.

excessively distant from the sensor's center are discarded. The specific formulas guiding this process is outlined as follows:

$$P = \{p_1, p_2, \ldots, p_n\} \tag{1}$$

$$p_k = [x_k, y_k, z_k] \tag{2}$$

$$\|p_k - c\|_2 = \sqrt{(x_k - c_x)^2 + \left(y_k - c_y\right)^2 + (z_k - c_z)^2} \tag{3}$$

$$P_{\text{valid}} = \left\{p_k \in P \mid \|p_k - c\|_2 \leq R\right\} \tag{4}$$

where R is a predetermined hyperparameter, representing the radius of the sphere centered around the sensor. Equation (3) describes the method for calculating the distance of each point from the origin. Equation (4) elucidates the method used to filter effective points from a point cloud frame. And p_k represents the kth point in the point cloud, with x_k, y_k, z_k denoting the respective x, y, and z coordinates of p_k within the LiDAR's coordinate system. Additionally, c_x, c_y, c_z denote the position of the LiDAR within the LiDAR's coordinate system, which coincides with the position of the origin (0, 0, 0).

By applying this criterion, we can calculate the distance for each point to exclude ineffective points from the point cloud.

2) Regional Division Stage: In this section, our segmentation strategy divides the point cloud into latitudinal and longitudinal sectors based on azimuth and elevation angles. This approach not only reflects the grid structure characteristic of the Mercator projection but also facilitates the transformation of the point cloud's spatial data into a cylindrical projection. Effectively, this maintains the inherent angular relationships within the point cloud data.

As illustrated in Fig. 4 (left), illustrates how point clouds are visualized from a BEV perspective. Initially, a frame of point cloud data is divided into N_r distinct sectors. Then, a point is selected, and its azimuth angle is calculated. Based on this angle, the point is assigned to the corresponding sector. Fig. 4 (right) represents the selection of one sector from the (left), where a point from the point cloud is chosen, and its elevation angle is calculated. The different colors of the points indicate varying elevation angles. By analyzing the azimuth and elevation angles of each point within the robot's coordinate system, we subdivide the point cloud into distinct, nonoverlapping regions. This precise regional division strategy preserves the spatial structural integrity of the point cloud and lays a solid foundation for subsequent feature extraction tasks.

Specifically, in determining the azimuth angle $\theta(x_k, y_k)$ and elevation angle $\phi(r, z_k)$ for each point in the point cloud, we proceed based on the coordinates of each point, and the formulas is as follows:

$$\theta(x_k, y_k) = \arctan\left(\frac{y_k}{x_k}\right) \tag{5}$$

$$r(x_k, y_k) = \sqrt{x_k^2 + y_k^2} \tag{6}$$

$$\phi(r, z_k) = \arctan\left(\frac{z_k}{r(x_k, y_k)}\right). \tag{7}$$

The variables $\theta(x_k, y_k)$ and $\phi(r, z_k)$ represent the azimuth and elevation angles of each point within the coordinate system, respectively. These angles for each point p_k are calculated using (5) and (7).

After deriving the azimuth and elevation angles, we continue to segment the point cloud into discrete, nonoverlapping latitudinal and longitudinal grid regions, denoted by $N_r \times N_c$. Specifically, this segmentation process follows these guidelines.

1) The range of azimuth angles $[0°, 360°]$ is divided into N_c sectors, each spanning $(360°/N_c)$.
2) The range of elevation angles $[0°, 45°]$ is segmented into N_r sectors, each spanning $(45°/N_r)$

$$N_c^k(N_c, \theta(x, y)) = \max\left(\min\left(N_c, \left\lceil \frac{\theta(x, y)}{360°} \cdot N_c \right\rceil\right), 1\right) \tag{8}$$

$$N_r^k(N_r, \phi(r, z)) = \max\left(\min\left(N_r, \left\lceil \frac{\phi(r, z)}{45°} \cdot N_r \right\rceil\right), 1\right). \tag{9}$$

These formulas define how to map the azimuth and elevation angles of points to indices of longitude sectors and latitude sectors in a point cloud. Formula (8) calculates the longitude sector index N_c^k by scaling the azimuth angle $\theta(x, y)$ of point (x, y) to the total number of longitude sectors N_c, ensuring the result stays within bounds $(1, N_c)$. Similarly, formula (9) calculates the latitude sector index N_r^k by scaling the elevation angle $\phi(x, y)$ of point (r, z) to the total number of latitude sectors N_r, with safeguards to maintain valid index ranges $(1, N_r)$. Both formulas ensure that indices start from 1 and do not exceed their respective total counts, effectively organizing the point cloud data into structured longitude and latitude sectors based on angular measurements. This structured approach of segmenting the point cloud into a grid of latitudinal and longitudinal sectors facilitates systematic and orderly analysis of spatial data. Furthermore, it allows the application of Mercator projection principles to point clouds, ensuring that spatial and angular relationships are preserved during subsequent processing and feature extraction stages.

3) Depth Information Extraction: In constructing the Mercator descriptor, our approach extends beyond merely allocating point cloud data to specific longitude and latitude sectors based on azimuth and elevation angles. We draw inspiration from methodologies in computer vision and image processing [40], [41], particularly in handling depth maps, to delve into the effective extraction of salient features from each sector. Depth information, a fundamental tool for describing the 3-D structure of a scene, underpins this approach. On this basis, we argue that average depth features are a crucial dimension of representation. These features not only illustrate the density of point clouds within each area and their distribution relative to the sensor center but also provide insights into the location and configuration of objects within the scene.

To achieve this goal, we transform latitude and longitude sectors into a 2-D image matrix I. The specific process is described by the following equations:

$$\psi(x_k, y_k, z_k) = \frac{1}{n}\sum_{i=1}^{n} d(x_k, y_k, z_k) \tag{10}$$

$$I = (a_{ij}) \in \mathbb{R}^{N_r \times N_c}, \quad a_{ij} = \psi(x_k, y_k, z_k). \tag{11}$$

First, we introduce an encoding function $\psi(x_k, y_k, z_k)$, which maps the average depth of the point cloud in each longitude sector to the corresponding matrix pixel value. Where $d(x_k, y_k, z_k)$ represents the distance from point p to the origin in the robot coordinate system. Through this approach, we can compute and form a matrix I of size $N_r \times N_c$. Fig. 3(d) provides a clear visual representation of this process, where the red hues intuitively convey the depth information of the point clouds within each sector.

C. Loop Closure Detection

In the field of loop closure detection for point clouds, we employ kd-tree construction to efficiently manage and rapidly compare descriptors. For each current frame, we introduce latitude keys $\mathbf{k}$ with rotational invariance. These latitude keys are derived from a latitude grid composed of Mercator descriptors, and both the latitude keys and associated descriptors for each frame are meticulously organized within a kd-tree structure. The calculation of the latitude key follows the following formulas:

$$\mathbf{k} = (\chi(I_j), \ldots, \chi(I_{N_r})), \quad \text{where } I_j \in I,\ 1 \le j \le N_r \tag{12}$$

$$\chi(I_{ij}) = \frac{1}{N_c}\sum_{j=1}^{N_c}(I_{ij}), \quad \text{where } I_{ij} \in I_j \tag{13}$$

where the latitude key $\mathbf{k}$ is an N_r dimensional vector, while the encoding function $\chi(I_{ij})$ represents the function generating each latitude key value. It is defined as the sum of the means of each latitude vector. Where the variable I_j represents each latitude vector of the $N_r \times N_c$ matrix I, and the variable I_{ij} denotes the value at the ith row and jth column of the 2-D matrix.

Subsequently, we perform the search loop in two steps: first, we perform a nearest neighbor search within the kd-tree using the latitude key to obtain the candidate boxes, and then we use an appropriate distance metric to evaluate the similarity between the two descriptors. The formula is as follows:

$$D(I^q, I^c) = \frac{1}{N_c}\sum_{j=1}^{N_c}\left(1 - \frac{c_j^q \cdot c_j^c}{\left\|c_j^q\right\|\left\|c_j^c\right\|}\right) \tag{14}$$

where I^q and I^c, respectively, represent the Mercator descriptors extracted from the query point cloud and the candidate point cloud. Then, we apply cosine similarity measure to evaluate the similarity between two identical indexed column vectors c_j^q and c_j^c. Finally, we average the cosine similarities of all column vectors between each pair of current frame and candidate frame to obtain the distance $D(I^c, I^q)$.

Based on this distance calculation, we verify whether the current frame and the candidate frame form a loop closure detection. It should be emphasized that we continuously reconstruct the kd-tree, which improves the accuracy of the subsequent search process and optimizes the efficiency of loop closure detection. The overall retrieval algorithm is shown in Algorithm 1.

In addition, considering that in tasks such as loop detection or location recognition, the descriptor needs to be robust to rotation and viewpoint changes. As shown in Fig. 5, this is the Mercator descriptor mapped by scanning point cloud data at the same place at different time points: the change of the sensor viewpoint when revisiting will cause the columns of the Mercator descriptor to shift, but the two matrices contain similar shapes and show the same row order. For this problem, this study adopts a data augmentation strategy to generate multiple variants of the descriptor from different angles and viewpoints by translating the rows and columns of the descriptor, thereby enhancing its robustness. This strategy ensures that the matching frame of the current point cloud frame can be accurately identified even in the face of fluctuations in shooting angles or rotation of objects.

IV. Experimental Results

In this section, we aim to provide a comprehensive evaluation of our algorithm's effectiveness. To achieve this goal,

Algorithm 1 Loop Closure Detection With Latitude Keys and KD-Tree

Input: Des (Mercator Descriptor of a frame)
Output: Candidate Loop Frame ID

Define:
$Latitude\_k$:the latitude keys
Th_r:the distance threshold of the current descriptor and the candidate
Main Loop:
Initialize KD-Tree: kd_tree ← Des, latitude_k
if KD tree needs to be reconstructed **then**
  Clear the list of latitude_k for search
  Copy recent historical latitude_k to the search list
  Reconstruct the KD tree using latitude_k as indices
end if
Coarse detection:
Initialize minimum distance and nearest neighbor index
Find the k nearest neighbors of the candidate_des using a KD tree by latitude_k
Fine detection:
for a candidate_des ∈ candidate_des **do**
  Compute the distance between the current_des and the candidate
  if distance < the minimum distance **then**
    Update minimum distance, and nearest neighbor index
  end if
end for
Loop Closure Check:
if distance < Th_r **then**
  return Candidate Loop Frame ID
end if
return −1

TABLE I
PARAMETER LIST

Parameter	Description	Value
R	Maximum radius	80m / 20m (Out/In)
N_c	Longitude sectors	60
N_r	Latitude sectors	20
K_n	Keyframe interval count	5
T_h	Loop pair distance threshold	5 m
Th_r	Cosine similarity threshold	0.1-0.3

we conducted a series of experimental assessments across four distinct dimensions: 1) precision recall evaluation; 2) performance on LiDAR-based SLAM; 3) impact of hyper-parameters on robustness; and 4) computational complexity. It is noteworthy that all experiments were carried out on a laptop equipped with an Intel Core i5-1135G7 @ 2.4 GHz processor and 16 GB RAM. The parameters and values used in our experiments are listed in Table I

A. Precision Recall Evaluation

To verify our experimental results, we use the KITTI dataset [42] for analysis. This dataset is renowned for its comprehensive coverage of diverse street scenes ranging from urban to suburban environments, and it comprises point cloud data captured at a frequency of 10 Hz using the Velodyne HDL-64E S2 sensor. The dataset provides 11 real pose sequences labeled from 00 to 10, with sequences 00, 02, 05, 06, 07, and 08 exhibiting higher frequencies of loop closures. Notably, sequence 08 features reverse loop closures, while others only involve forward loop closure events.

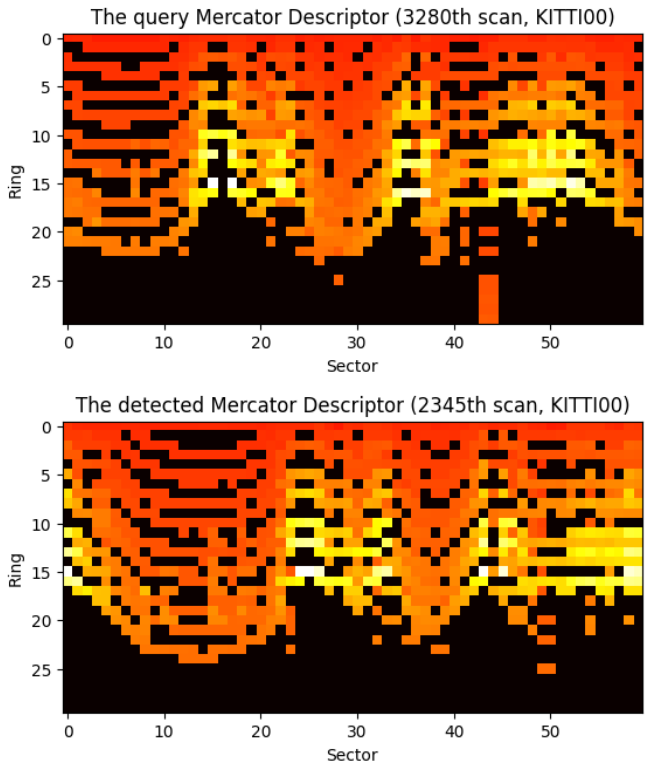


Fig. 5. Example of Mercator descriptor from the same place with time interval.

Our study involves several existing LiDAR loop closure detection and place recognition methods, including M2DP, SC, BoW3D, and STD. To facilitate a comprehensive comparative analysis, we selected six sequences (00, 02, 05, 06, 07, and 08) from the KITTI dataset and rigorously evaluated our proposed model using well-established metrics described in the literature. Key performance metrics evaluated include accuracy and recall rates

$$\text{Precision (P)} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{15}$$

$$\text{Recall (R)} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{16}$$

where true positives (TPs) denote instances accurately identified as loop closures, while false positives (FPs) represent erroneously labeled negative samples. Conversely, false negatives (FNs) indicate positive samples incorrectly classified as nonloop closures.

When assessing the classifier's performance, we not only rely on the precision-recall (PR) curve but also leverage the $F1$ score as a holistic metric. The $F1$ score, being the harmonic mean of precision (P) and recall (R), provides a nuanced evaluation of the classifier's overall performance

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{17}$$

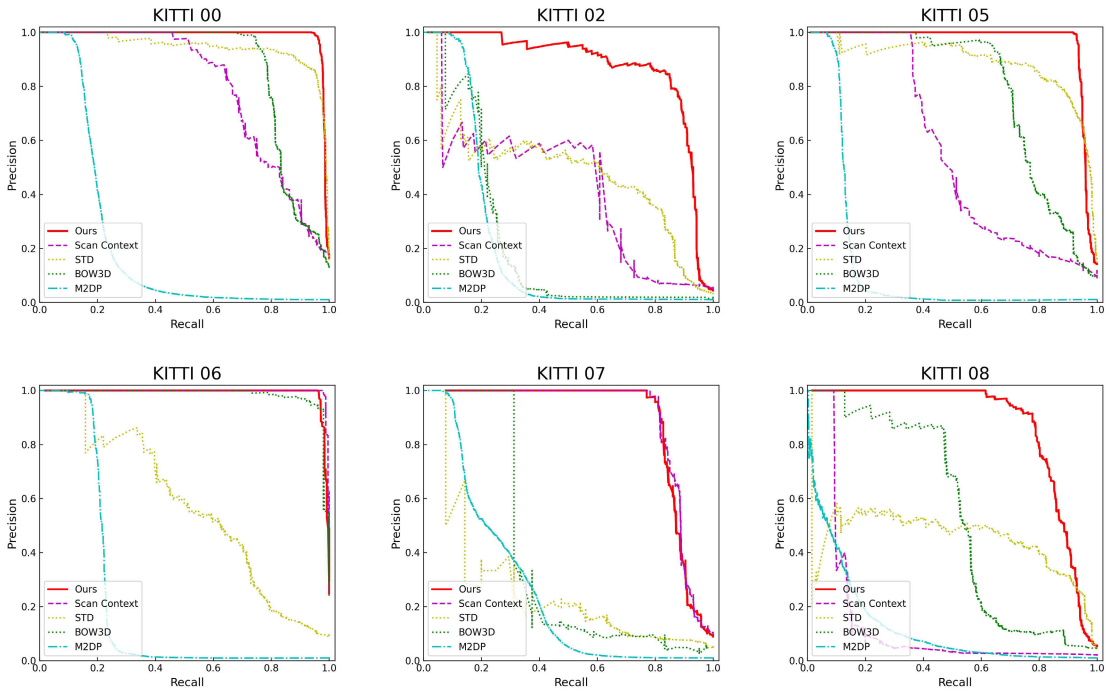


Fig. 6. Comparison of accuracy and recall among different algorithms (M2DP, SC, BoW3D, STD, and Ours) on the KITTI dataset.

TABLE II
COMPARISON OF MAX $F1$ SCORES BY DIFFERENT ALGORITHMS (M2DP, SC, BOW3D, STD AND OURS) AND KITTI SEQUENCE

Method\Max F1 Score	KITTI00	KITTI02	KITTI05	KITTI06	KITTI07	KITTI08
M2DP	0.276	0.282	0.341	0.316	0.204	0.201
Scan Context	0.739	0.583	0.525	0.996	0.935	0.195
BoW3D	0.911	0.462	0.860	0.967	0.667	0.729
STD	0.925	0.609	0.879	0.660	0.417	0.598
Ours	0.985	0.884	0.980	0.993	0.937	0.896

As shown in Fig. 6, our method demonstrates outstanding performance across multiple sequences of the KITTI dataset. In sequences 00, 02, 05, 07, and 08, our algorithm maintains higher precision at high recall levels compared to other methods, indicating its ability to avoid false positives while sustaining a high detection rate.

In Table II, our method achieved the highest $F1$ scores in the 00, 02, 05, 07, and 08 sequences, with a notably superior performance in the loop-back sequence of 08, where our score reached 0.896. This further confirms the superiority of our method in these sequences, especially in handling various complex scenarios with stability and robustness.

In contrast, other methods like SC, BoW3D, and STD exhibited significant variability in performance across certain sequences. For instance, SC performed excellently in the 06 and 07 sequences but saw a sharp decline in 08. BoW3D showed good performance in sequences 00, 05, and 06, but lower $F1$ scores in 07 and 02.

The ability of our method to outperform other algorithms in most sequences is primarily attributed to its optimized feature extraction and matching algorithms. By more effectively capturing and utilizing the 3-D structural information of the environment, our algorithm can more accurately identify loop closures. Additionally, our method demonstrates stronger adaptability to different environmental conditions, reflected in the consistently high precision across various sequences. However, we also noted that in some sequences, such as 06, our $F1$ score was slightly lower than SC, which we attribute to the simpler path configuration of sequence 06 that allowed all methods to perform well.

B. Performance on LiDAR-Based SLAM

In this section, we explore the integration of Mercator descriptors into SLAM for loop closure correction, aiming to comprehensively assess the applicability and robustness of our algorithm. We utilize four sequences from the KITTI dataset (00, 05, 07, 08), two sequences from the MulRan dataset [43], and our own campus dataset to test the performance of our algorithm on different types of LiDAR.

For the KITTI dataset, we integrated our method with the A-LOAM algorithm (an advanced variant of LOAM). The experimental results are shown in Fig. 7. In the first column, we demonstrate the comparison between pose graph optimization using loop closure detection techniques and the ground truth trajectory. The error distribution shown in the color maps indicates that in certain specific areas, our algorithm exhibits relatively lower errors (colors tending toward blue), suggesting higher localization accuracy in these regions. Conversely, higher errors may be observed in other areas. Moving to the second column, we observe a comparison between three different trajectories and ground truth data, indicating a significant enhancement of our method over traditional frame-to-frame odometry. Moreover, compared to frame-to-map odometry, our

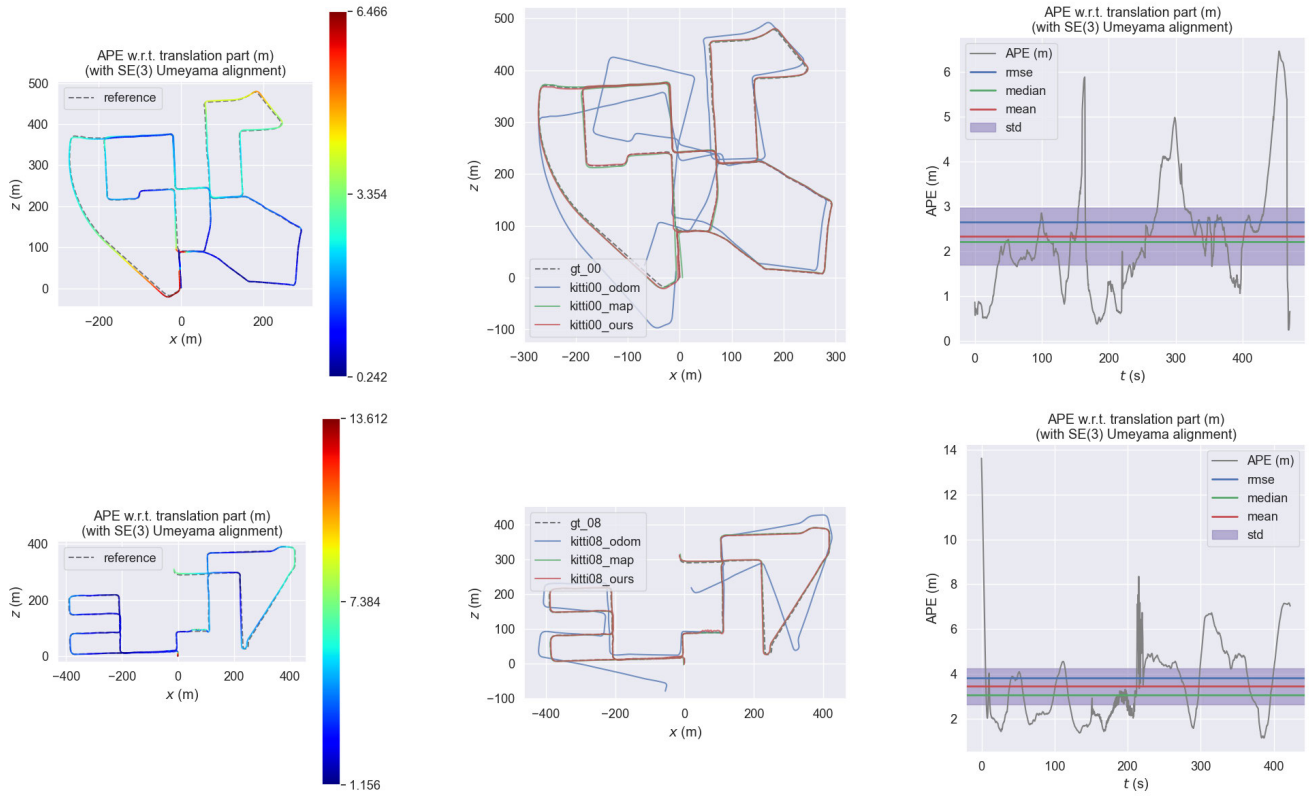


Fig. 7. Evaluation of A-LOAM and our integration of loop closure detection for pose graph optimization on the KITTI dataset: The first column depicts the absolute pose error (APE) after pose graph optimization with our proposed loop closure detection seamlessly integrated into A-LOAM. Within the color bar, a prevalence of blue hues signifies diminished relative errors, while a dominance of red hues indicates escalated relative errors; The second column illustrates the comparison among three distinct trajectories and the corresponding ground truth data; whereas the final column showcases diverse error curves of our method in contrast to the ground truth across sequences 00 and 08, sequentially.

TABLE III
ERROR METRICS OF OUR METHOD ON DIFFERENT KITTI SEQUENCES

Sequence \Metric (m)	RMSE	Mean	Median	Std
KITTI00	2.6549	2.3341	2.2178	1.2651
KITTI05	2.6613	2.2768	1.8207	1.3780
KITTI07	0.5557	0.5035	0.4638	0.2350
KITTI08	3.8164	3.4619	3.0817	1.6064

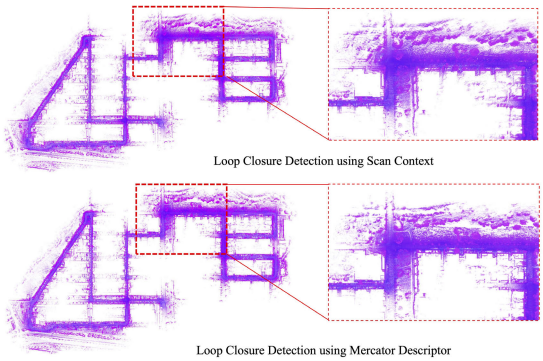


Fig. 8. Comparison of loop detection and mapping optimization effects of different algorithms (SC, our algorithm) on KITTI sequence 08.

method strategically optimizes loop traversal positions, thereby enhancing global map consistency and showing considerable advantages in overall performance.

Finally, the third column displays the error curves comparing our algorithm with the actual conditions of different sequences. Table III shows that in the 00 and 05 sequences, the RMSE values are close to 2.65 m, indicating some fluctuation in positional estimation in these sequences, but overall the errors remain within a reasonable range. The 07 sequence performs excellently, with an RMSE of only 0.56 m, and both the average and median errors are very low, indicating extreme precision and stability of the method in this sequence. In contrast, the 08 sequence shows the highest errors, with an RMSE reaching 3.82 m, likely due to more complex scenarios and reverse loops present in the sequence.

In addition, we compared our algorithm with SC on the KITTI dataset sequence 08. The KITTI08 sequence is a large-scale outdoor scene containing more reverse loops than other sequences. As shown in Table IV, our algorithm achieved a lower RMSE of 3.82 compared to SC's RMSE of 4.11, indicating that our method detects loop closures more accurately. The standard deviation of our algorithm was also lower, meaning the errors are more consistent, which leads to more stable map reconstruction. SC failed to detect loops in some scenes, resulting in ghosting, while our algorithm more effectively detected these scenes, significantly reducing ghosting and optimizing the map, as shown in Fig. 8.

For solid-state LiDAR, we utilized the KAIST and Riverside sequences from the MulRan dataset. KAIST represents a complex outdoor environment, while Riverside is a reverse sequence designed to test rotation (viewpoint) invariant loop closure detection. Experimental results demonstrate that integrating our loop closure algorithm significantly improves map

TABLE IV
PERFORMANCE COMPARISON ON DIFFERENT DATASETS

Dataset \Metric (m)	RMSE	Mean	Median	Std
KITTI08				
Our	3.8164	3.4619	3.0817	1.6064
SconContext	4.1095	3.3922	2.8490	2.3196
Reverse				
Our	9.0996	6.1343	5.2414	6.3896
SconContext	9.3210	6.7243	4.5769	6.4548
KAIST				
Our	1.9834	1.8376	1.7935	0.7962
SconContext	1.9548	1.7975	1.6933	0.7684

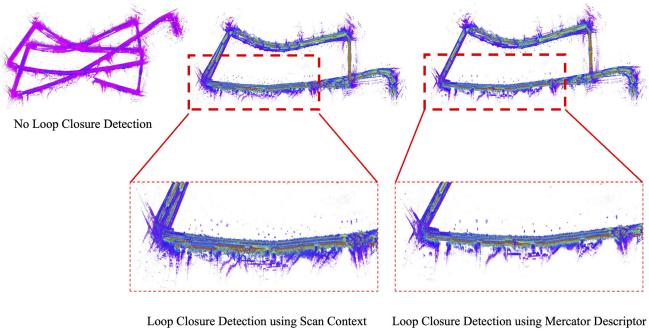


Fig. 9. Results of different loop closure detection algorithms on reverse sequence map construction.

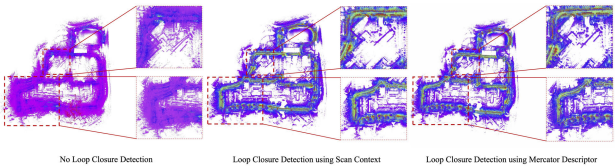


Fig. 10. Results of different loop closure detection algorithms on KAIST sequence map construction.

accuracy. Specifically, in the riverside sequence Table IV, our algorithm achieved a lower RMSE of 9.10 compared to SC's RMSE of 9.32, with a more consistent mean error and a lower standard deviation of 6.39, which helps reduce accumulated drift over time. Without loop closure, significant drift was observed due to accumulated errors. However, after integrating both SC and our loop closure algorithm, this drift was significantly reduced. In contrast, our algorithm effectively detected these scenes and optimized the errors, as shown in Fig. 9.

In the KAIST sequence Table IV, without loop closure, the alignment of local areas in the map was poor, leading to more deviations and overlaps, and consequently, a lower quality map. Our algorithm achieved a slightly higher RMSE of 1.98 compared to SC's RMSE of 1.95, and the mean error of 1.84 for our method was slightly higher than SC's 1.80. SC thus outperforms us on this dataset, but as shown in Fig. 10, both algorithms significantly reduced alignment errors.

For our own campus dataset, we collected data using a backpack-style LiDAR system. Our backpack LiDAR scanning system includes two Velodyne VLP-16 LiDARs (mounted horizontally and vertically), an embedded IMU operating at 200 Hz, a GNSS receiver, a communication module for receiving GNSS differential data, and a battery. Data collection took place on the campus of Capital Normal University,

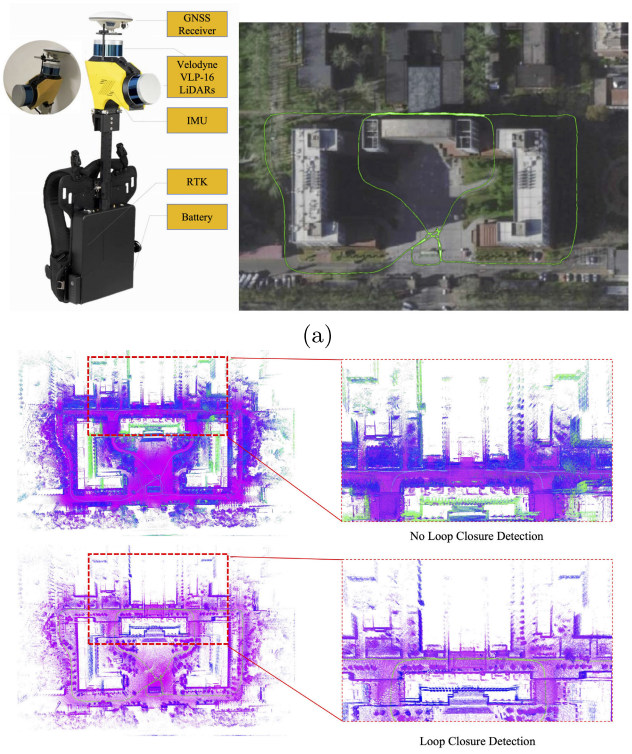


Fig. 11. Campus dataset. (a) Depicts an overview of the backpack laser scanning system and the campus dataset. (b) Illustrates the comparative mapping results with loop closure integration and without loop closure integration.

covering a route that mainly includes three main buildings and a square, as shown in Fig. 11(a).

Fig. 11(b) demonstrates the improvements brought by our loop closure detection algorithm, which reduces cumulative errors and allows for more detailed representation of local maps. The top image shows the mapping results without using loop closure detection. It is evident that there are significant drifts and overlaps in the map, especially in the enlarged local detail images, where the alignment of buildings and streets is poor, resulting in lower map quality. The bottom image shows the mapping results after applying loop closure detection. Compared to the case without loop closure detection, the drift in the map is significantly reduced, the local details are clearer and more accurate, and the overall map accuracy and consistency are greatly improved.

C. Impact of Hyperparameters on Robustness

In this section, we take an in-depth look at the various parameters within the descriptor framework that influence the precision and accuracy of loop closure detection. Through systematic experiments, we fine-tuned the number of latitude and longitude sectors, aiming to reveal the impact of these variables on descriptor performance.

In our experiments, the number of latitude sectors varied from 10 to 28 and the number of longitude sectors varied from 35 to 85. This series of experiments provides us with valuable insights. The results shown in Fig. 12 show some notable trends. The results in Table V reveal subtle dynamics. In particular, when the number of longitude sectors (N_c) is 35,

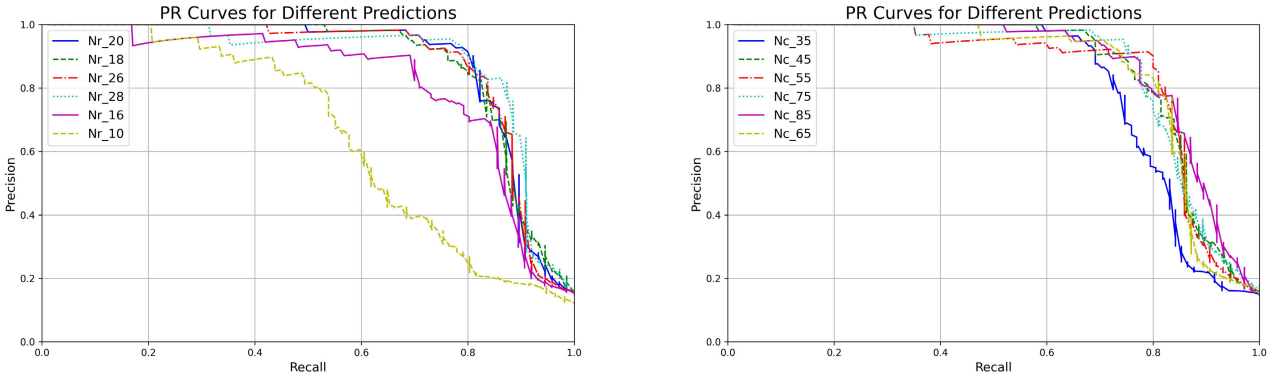


Fig. 12. Effect of different N_r and N_c on accuracy and recall.

TABLE V

$F1$ SCORE FOR DIFFERENT PARAMETER PAIRS, WITH N_r ON THE LEFT AND N_c ON THE RIGHT

N_r Dataset		N_c Dataset	
Value	Max F1 Score	Value	Max F1 Score
10	0.6364	35	0.7887
16	0.7832	45	0.8133
18	0.8299	55	0.8477
20	0.8533	65	0.8163
26	0.8421	75	0.8356
28	0.8533	85	0.8267

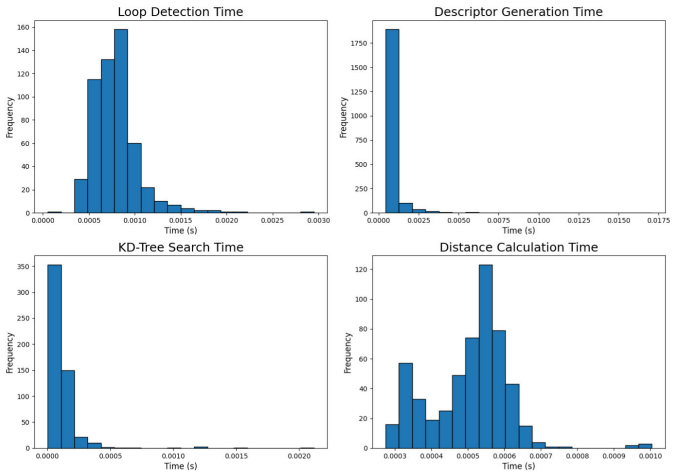


Fig. 13. Runtime of each keyframe in the algorithm.

the descriptor performance drops significantly, peaking at 55, and exhibiting slight performance fluctuations around 75.

In contrast, there is a significant correlation between the number of latitude sectors (N_r) and the accuracy and recall of the descriptor. The effectiveness of loop closure detection increases significantly as the number of latitude sectors increases, highlighting the critical role of this parameter in optimizing descriptor performance.

D. Computational Complexity

In this section, we seamlessly integrated our algorithm into the SLAM system and conducted a detailed analysis of the runtime performance of each module within the system. We chose the widely recognized KITTI sequence 00 as our evaluation benchmark, ensuring the robustness and standardization of our research. Through meticulous design, we set the parameters $N_r = 20$ and $N_c = 60$, ensuring consistency and reliability in our experimental framework.

TABLE VI

PERFORMANCE STATISTICS FOR DIFFERENT STAGES OF THE PROCESS

	Loop Det.	Desc. Gen.	KD-Tree	Dist. Calc.
count	545.000000	2065.000000	544.000000	544.000000
mean	0.000789	0.000950	0.000110	0.000504
std	0.000257	0.000822	0.000163	0.000110
min	0.000052	0.000439	0.000003	0.000274
max	0.002950	0.017074	0.002116	0.001004

Fig. 13 shows the runtime distribution across various stages, while Table VI presents the performance statistics for different processing stages, including "loop detection," "descriptor generation," "KD-tree search," and "distance calculation." From the table, it can be observed that the "descriptor generation" stage has the highest computation load, with the largest maximum value (0.017074) and a relatively large standard deviation (0.000822), indicating significant fluctuation in the computation time. The "KD-tree search" stage has the lowest average time (0.000110), but with a relatively large standard deviation (0.000163), suggesting some variability in the search time. In contrast, the "loop detection" and "distance calculation" stages have lower standard deviations and maximum values, indicating more stable computation times with less variation. Overall, descriptor generation is the most time-consuming step, while KD-tree search, despite its lower average time, ensures the real-time performance of the Mercator descriptor when applied to 3-D LiDAR SLAM.

V. CONCLUSION

In this article, we propose a novel descriptor generation algorithm based on the Mercator projection principle for place recognition. The algorithm consists of three parts: descriptor generation, loop closure detection, and map optimization with loop correction. We adopt a two-stage loop retrieval strategy to achieve efficient retrieval performance. Our Mercator descriptor not only exhibits competitive performance compared to state-of-the-art methods, but also enables real-time correction of accumulated errors. Unlike deep learning-based methods that require pretraining and GPU resources, our method does not require these resources. Furthermore, we integrate this method into a LiDAR odometry system to verify its performance in practical applications. Compared to traditional

odometry methods, our 3-D LiDAR SLAM system with loop closure detection exhibits significantly lower drift in scenes with loops. However, it is undeniable that the position dependence problem does exist. Since we use a projection method centered on the center point, when the LiDAR position changes, the projection result will also change, which to some extent affects the position invariance of the method. In the future, we are considering new explorations, to solve the problem of position invariance and further improve the robustness of our descriptor to different scenes.

ACKNOWLEDGMENT

The authors would like to acknowledge the anonymous reviewers for their valuable comments.

REFERENCES

[1] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot., Sci. Syst. X*, Jul. 2014, pp. 109–111.

[2] 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)*, Oct. 2018, pp. 4758–4765.

[3] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5135–5142.

[4] 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, Apr. 2021.

[5] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term LiDAR localization using scan context image," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1948–1955, Apr. 2019.

[6] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi, "Real-time RGB-D camera relocalization via randomized ferns for keyframe encoding," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 5, pp. 571–583, May 2015.

[7] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Trans. Robot.*, vol. 29, no. 3, pp. 734–745, Jun. 2013.

[8] J. Lin and F. Zhang, "A fast, complete, point cloud based loop closure for LiDAR odometry and mapping," 2019, *arXiv:1909.11811*.

[9] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[10] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[11] Y. Zhang, R. Liu, H. Yu, B. Zhou, and K. Qian, "Visual loop closure detection with instance segmentation and image inpainting in dynamic scenes using wearable camera," *IEEE Sensors J.*, vol. 22, no. 16, pp. 16628–16637, Aug. 15, 2022.

[12] T. Ran, L. Yuan, J. Zhang, D. Tang, and L. He, "RS-SLAM: A robust semantic SLAM in dynamic environments based on RGB-D sensor," *IEEE Sensors J.*, vol. 21, no. 18, pp. 20657–20664, Sep. 2021.

[13] J. He, M. Li, Y. Wang, and H. Wang, "OVD-SLAM: An online visual SLAM for dynamic environments," *IEEE Sensors J.*, vol. 23, no. 12, pp. 13210–13219, Jun. 2023.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[16] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proc. 11th Eur. Conf. Comput. Vis.*, vol. 6314, 2010, pp. 778–792.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[18] S. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, vol. 2, Oct. 2003, pp. 1470–1477.

[19] L. He, X. Wang, and H. Zhang, "M2DP: A novel 3D point cloud descriptor and its application in loop closure detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 231–237.

[20] K. P. Cop, P. V. K. Borges, and R. Dubé, "Delight: An efficient descriptor for global localisation using LiDAR intensities," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3653–3660.

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

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

[23] H. Wang, C. Wang, and L. Xie, "Intensity scan context: Coding intensity and geometry relations for loop closure detection," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 2095–2101.

[24] Y. Wang, Z. Sun, C.-Z. Xu, S. E. Sarma, J. Yang, and H. Kong, "LiDAR iris for loop-closure detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5769–5775.

[25] C. Yuan, J. Lin, Z. Zou, X. Hong, and F. Zhang, "STD: Stable triangle descriptor for 3D place recognition," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 1897–1903.

[26] Z. Zhou, C. Guo, Y. Pan, X. Li, and W. Jiang, "A 2D LiDAR-SLAM algorithm for indoor similar environment with deep visual loop-closure," *IEEE Sensors J.*, vol. 23, no. 13, pp. 14650–14661, 2023, doi: 10.1109/JSEN.2023.3260104.

[27] Y. Cui, X. Chen, Y. Zhang, J. Dong, Q. Wu, and F. Zhu, "BoW3D: Bag of words for real-time loop closing in 3D LiDAR SLAM," *IEEE Robot. Autom. Lett.*, vol. 8, no. 5, pp. 2828–2835, May 2023.

[28] Y. Cui, Y. Zhang, J. Dong, H. Sun, X. Chen, and F. Zhu, "LinK3D: Linear keypoints representation for 3D LiDAR point cloud," *IEEE Robot. Autom. Lett.*, vol. 9, no. 3, pp. 2128–2135, Mar. 2024.

[29] Q. Zhang, S. Zheng, R. Li, X. Wang, Y. He, and X. Wang, "RLS-LCD: An efficient loop closure detection for rotary-LiDAR scans," *IEEE Sensors J.*, vol. 24, no. 4, pp. 4807–4820, Feb. 2024.

[30] R. Dube, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "SegMatch: Segment based place recognition in 3D point clouds," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 5266–5272.

[31] R. Dubé et al., "SegMap: Segment-based mapping and localization using data-driven descriptors," *Int. J. Robot. Res.*, vol. 39, nos. 2–3, pp. 339–355, Mar. 2020.

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

[33] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[34] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5297–5307.

[35] P. Yin, L. Xu, J. Zhang, and H. Choset, "FusionVLAD: A multi-view deep fusion networks for viewpoint-free 3D place recognition," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2304–2310, Apr. 2021.

[36] X. Chen et al., "OverlapNet: Loop closing for LiDAR-based SLAM," 2021, *arXiv:2105.11344*.

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

[38] H. Xiang, X. Zhu, W. Shi, W. Fan, P. Chen, and S. Bao, "DeLightLCD: A deep and lightweight network for loop closure detection in LiDAR SLAM," *IEEE Sensors J.*, vol. 22, no. 21, pp. 20761–20772, Nov. 2022.

[39] J. Ma, G. Xiong, J. Xu, and X. Chen, "CVTNet: A cross-view transformer network for LiDAR-based place recognition in autonomous driving environments," *IEEE Trans. Ind. Informat.*, vol. 20, no. 3, pp. 4039–4048, Mar. 2024.

[40] A. S. Malik, *Depth Map and 3D Imaging Applications: Algorithms and Technologies: Algorithms and Technologies*. Hershey, PA, USA: IGI Global, 2011.

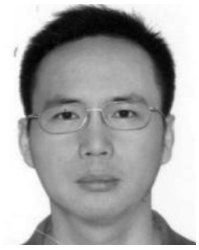
[41] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, "Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2511–2519.

[42] 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.*, Jun. 2012, pp. 3354–3361.

[43] G. Kim, Y. S. Park, Y. Cho, J. Jeong, and A. Kim, "MulRan: Multimodal range dataset for urban place recognition," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 6246–6253.

Zhibo Wang received the bachelor's degree from Henan Polytechnic University, Jiaozuo, China, in 2022. He is currently pursuing the master's degree with the Capital Normal University, Beijing, China.

His research interests include simultaneous localization and mapping (SLAM) and deep learning.



Donghai Xie received the Ph.D. degree from Chinese Academy of Science, Beijing, China, in 2012.

From 2005 to 2009, he was an Algorithm Engineer with Vimicro Company, Beijing. He is an Associate Professor with the Institute of Resource and Environment, Capital Normal University, Beijing. His research focuses on image processing and computer vision, particularly in the domains of deep learning-based object detection, and semantic segmentation.

Yu Wu received the Ph.D. degree in signal and information processing from the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing, China, in 2013.

He is an Associate Professor with the Institute of Surface-Earth System Science, School of Earth System Science, Tianjin University, Tianjin, China. His research interests include image quality improvement of space-borne remote sensing.

Haiyang Wu received the bachelor's degree from Chang'an University, Xian, China, in 2019. He is currently pursuing the master's degree with the Capital Normal University, Beijing, China.

His research interests include simultaneous localization and mapping (SLAM) and deep learning.

Xingyu Qi received the bachelor's degree from Southwest Jiaotong University, Chengdu, China, in 2022. He is currently pursuing the master's degree with the Capital Normal University, Beijing, China.

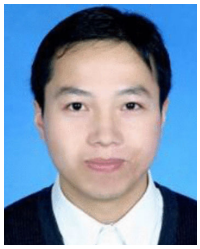
His research interests include simultaneous localization and mapping (SLAM) and deep learning.

Daoping Huang received the bachelor's degree from Shandong University of Science and Technology, Taian, China, in 2022. He is currently pursuing the master's degree with the Capital Normal University, Beijing, China.

His research interests include simultaneous localization and mapping (SLAM) and deep learning.

Yibo Fu received the bachelor's degree from Zhejiang Agriculture and Forestry University, Hangzhou, China, in 2021. He is currently pursuing the master's degree with the Capital Normal University, Beijing, China.

His research interests include remote sensing and deep learning.



Ruofei Zhong received the Ph.D. degree in geo-informatics from Chinese Academy of Science, Institute of Remote Sensing Applications, Beijing, China, in 2005.

He is a Professor with Beijing Advanced Innovation Center for Imaging Technology, Key Laboratory of 3-D Information Acquisition and Application, College of Resource Environment and Tourism, Capital Normal University, Beijing. His research interests include remote-sensing satellite design and data preprocessing, multisensor LiDAR integration and postprocessing, deep learning, and computer vision.