ISPRS Journal of Photogrammetry and Remote Sensing 202 (2023) 637–651

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

A fast LiDAR place recognition and localization method by fusing local and global search

Pengcheng Shi [a], Jiayuan Li [b,*], Yongjun Zhang [b,*]

[a] School of Computer Science, Wuhan University, Wuhan 430072, China
[b] School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China

ARTICLE INFO

Keywords:
SLAM
Place recognition
Loop closure detection
Occupied Place Description
LiDAR

ABSTRACT

Place recognition is an important branch of Simultaneous Localization and Mapping (SLAM), which finds revisited places, thereby reducing error accumulations. Most mainstream methods only concentrate on the similarity of two LiDAR scans and cannot obtain the pose information. In this paper, we design a descriptor named Occupied Place Description (OPD) and propose an efficient map-aided place recognition and localization method. Our method can effectively detect loop events, obtain the vehicle's poses, and overcome the overlap dependence of pairwise point cloud registration. It is composed of four modules. First, the map data generation is designed to store the map database as binary files. Then, a trajectory initialization consisting of global search, local search, and trajectory score computation is proposed to determine the vehicle's initial poses. Next, an online localization method is developed by combining descriptor similarity and Kalman Filtering, which only takes less than 5ms per localization. Finally, loop events are detected based on pose information. Exhaustive evaluations on KITTI and MulRan datasets demonstrate the superiority of our method. The average absolute trajectory error (ATE) is 0.5 m and the average relative rotation error is 2.7 degrees. F_1 score measures the success rate of place recognition and our average maximum F_1 score achieves nearly 0.987. The source code will be available in https://github.com/ShiPC-AI/Occupied-Place-Description.

1. Introduction

Place recognition (Shi et al., 2023b) aims to recognize the revisited places under changing viewpoints and environmental conditions, which is an important module in SLAM (Karam et al., 2021). Place recognition can relocate a kidnapped robot (Jiang et al., 2021), detect the online loop closure to mitigate drifts (Fan et al., 2020a), and provide metric localization (Li et al., 2023a; Shi et al., 2023a) when the GPS signal is unavailable. General methods aim to improve one performance such as success rate, efficiency, or localization accuracy. It is still challenging to simultaneously improve the above three metrics.

Benefiting from rich color information and lightweight memory of images, some successful visual-based place recognition approaches (Fan et al., 2019) have been proposed. A popular strategy is to extract some local features (Yang et al., 2017a; Li et al., 2022a, 2023b), then perform feature matching based on a bag-of-words (BoW) model (Cummins and Newman, 2008). However, these methods are sensitive to weather and illumination change and may produce some false matching. Besides, the limited measurement distance and field of view (FOV) of the camera also increase the probability of wrong recognition.

Compared to cameras, Light Detection and Ranging (LiDAR) has a larger measurement distance and FOV. It can directly collect 3D information, which makes place recognition more reliable. However, there are four challenging issues: (1) Lateral and rotational movements: vehicles usually revisit a place in a reverse direction or with lane-level translation offsets in urban environments. (2) View occlusions: sensor signals may be incomplete or damaged due to moving objects (e.g., cars and pedestrians) or special reflective surfaces (e.g., glass and water). (3) Generalization capability: some methods depend on specific landmarks or environmental layouts, which makes them only applicable to limited scenarios. (4) Method performances: success rate and runtime are two important metrics for place recognition, and balancing them well is still very challenging.

Scan Context (Kim and Kim, 2018) is a representative descriptor-based LiDAR place recognition method with the following advantages: (1) It transforms a point cloud to the bird-eye-view (BEV) representation and encodes the points in the discrete bins instead of calculating the number of points (He et al., 2016), which is invariant to the density and normal of the point cloud. (2) It stores height information to preserve the absolute internal structure of the point cloud, which improves

* Corresponding authors.
E-mail addresses: ljy_whu_2012@whu.edu.cn (J. Li), zhangyj@whu.edu.cn (Y. Zhang).

https://doi.org/10.1016/j.isprsjprs.2023.07.008
Received 12 October 2022; Received in revised form 25 June 2023; Accepted 7 July 2023
Available online 20 July 2023

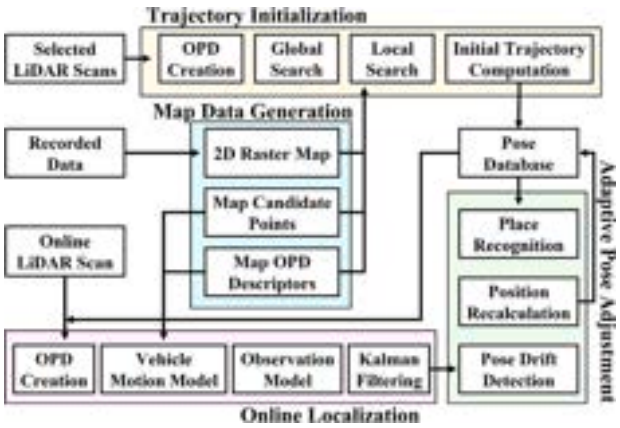


Fig. 1. The workflow of our place recognition and localization. Our system consists of four modules. The map data generation stitches a point cloud map using offline recorded data and builds a hybrid map database including a raster map, candidate points, and map descriptors. The trajectory initialization selects a few scans from the beginning of the sequence to calculate the vehicle's poses and initialize the trajectory. After obtaining the initial trajectory, the online localization calculates the vehicle's poses based on a single online LiDAR scan. The adaptive pose adjustment exploits the displacement error and matching score to correct the false localization and detect revisited places. The map data generation is an offline process, while the other three modules are online.

the discriminative ability and viewpoint invariance of descriptors. (3) It designs a two-phase matching algorithm that uses a ring key to find candidates and then uses brute-force search to determine the best match. This method improves computational efficiency by avoiding searching all databases.

In this paper, we are motivated by Scan Context and propose a map-aided solution to achieve LiDAR place recognition. Fig. 1 presents the workflow of our place recognition and localization. The map data generation module registers offline recorded data together to create a point cloud map, candidate points, and map descriptors, which are used to simulate vehicles driving and collecting data within the map. The trajectory initialization module selects a set of online LiDAR scans and combines local search, global search, and trajectory score computation to determine the vehicle's initial poses. The online localization module designs a descriptor-based Kalman Filtering method including a vehicle motion model and a local search-based observation model, which locates the vehicle using each online LiDAR scan. The adaptive pose adjustment module adjusts poses by the displacement error and the matching score, followed by finding the revisited places.

Although Scan Context achieves good performance in urban scenarios, it is susceptible to lateral offsets (Kim et al., 2021). We combine the point cloud map, Occupied Place Description (OPD) descriptors, and Kalman Filtering to improve place recognition and localization performance. The main differences between the two methods are: (1) We need to establish a hybrid offline map database, but Scan Context does not. (2) Our OPD descriptor stores binary values instead of the float types in Scan Context. (3) Our descriptor similarity is computed based on a binary operation-based loss rather than the traditional cosine distance. (4) We use local search to reduce search space and use motion models to ensure the smoothness of localization results, but Scan Context uses brute-force search to find revisited places. Compared to Scan Context, our descriptor has a smaller memory footprint but faster matching speed. Our method achieves a better place recognition performance. Scan Context is mainly used for place recognition, while our method can be applied to both localization and place recognition.

In summary, our contributions are as follows:

- We propose a novel binary BEV descriptor, which only records whether the region is occupied and improves the matching efficiency using a binary operation-based loss function.
- We design a trajectory initialization method, which uses a set of scans instead of traditional single scans to calculate the initial pose, thus improving the trajectory completeness.
- We develop a descriptor-based Kalman Filtering localization method, which reduces search space by local search and uses a vehicle motion model to ensure the smoothness of the localization results.
- We compare the proposed method to state-of-the-art (SOTA) methods. Experimental results demonstrate that our method achieves better place recognition performance and provides high-precision poses. To benefit the community, the source code will be open-source soon.

2. Related work

In this section, we reviewed recent research on point cloud-based place recognition and classified these methods into handcrafted local descriptor-based, handcrafted global descriptor-based, segment-based, semantics-based, data-driven, and map-based methods. Table 1 summarized their advantages and disadvantages.

2.1. Handcrafted local descriptor-based methods

These methods first extracted key points and then encoded their local geometric properties into descriptor vectors. According to whether using a local reference frame (LRF), we classified these methods into LRF-based and LRF-free methods.

LRF-based Methods. They generally created three orthogonal unit vectors to normalize the local surface around the key points. This feature encoding was invariant to rotation and translation, which could also associate geometric attributes with spatial information. These methods had good descriptive power but relied too much on LRF and were highly susceptible to resolution changes. For instance, SHOT (Salti et al., 2014) created an LRF based on a weighted covariance matrix and divided the spherical support region into 32 small volumes, which formed a histogram description based on the angles between the neighbor point's normal and the z-axis. To overcome the problem of noise and occlusion, ToLDI (Yang et al., 2017b) calculated the z-axis by the normals of the key points, and the x-axis by aggregating the weighted projection vectors of adjacent points. It concatenated three local depth images from three orthogonal view planes into feature vectors. To improve the performance in natural environments, ISHOT (Guo et al., 2019) coupled the geometry information with its calibrated intensity values. This method proposed an adapted key point voting method based on empirical modeling of voting precision.

LRF-free Methods. They directly calculated projection distances, normal angles, or shape indexes to construct histograms or feature matrices to encode local surfaces. They achieved robustness to noise and point cloud density but usually required relatively high computational complexity. For instance, Spin Image (SI) (Johnson and Hebert, 1999) calculated the distance between the neighbor points' normals to key points and their tangent planes to generate a 2D histogram. However, this method was sensitive to resolution changes. Persistent Feature Histograms (PFH) (Rusu et al., 2008) and Fast Point Feature Histogram (FPFH) (Rusu et al., 2009) simultaneously calculated distances and angles to describe local surfaces.

2.2. Handcrafted global descriptor-based methods

These methods only used a single descriptor to describe the entire point cloud. According to whether using a histogram representation, we classified these methods into histogram-based and histogram-free methods.

Histogram-based Methods. These methods represented point clouds as histogram-based features to achieve place recognition, which could address the problems of rotation invariance and noises. For instance,

Table 1
Summary of related work.

Types of methods		Advantages	Disadvantages
Handcrafted local descriptor	LRF	Rotation and translation invariance	Susceptible to resolution changes
	LRF-free	Robustness to noise and point cloud density	High dimension and expensive computation
Handcrafted global descriptor	Histogram	Rotation invariance and robustness to noises	Losing internal structures
	Histogram-free	Rotation invariance and superior generality	Susceptible to large lateral offsets
Segment-based		Compromise between local and global descriptors	Susceptible to moving objects
Semantics-based		Robustness to occlusions and viewpoint changes	Limited semantic labels and complex calculation
Data-driven		Outstanding efficiency and recognition accuracy	Limited training samples and data cleaning
Map-based		Pose available and superior accuracy	Expensive computation and map update costs

VFH (Rusu et al., 2010) extended the FPFH descriptor to the entire point cloud, which found viewpoint direction and encoded the normal angles formed by connecting points into histograms. ESF (Wohlkinger and Vincze, 2011) designed a global shape descriptor that combined the distribution of distance, angle, and area but ignored the spatial information, which limited its application. Röhling et al. (2015) proposed a point projection function, which encoded the height or distance information of a single scan into a histogram, and used a discrete Wasserstein metric to evaluate the similarity. To improve the robustness of histogram features, Rizzini (2017) established a local map, encoded the relative positions of key points into histograms, and calculated the rotation invariance metric between histograms. To keep the advantages of local descriptors and limit the processing time, DELIGHT (Cop et al., 2018) encoded the intensity value into a histogram, and used a chi-squared test to calculate the descriptor distances. Lin and Zhang (2019) developed a fast, complete loop closure system by evaluating the similarities of two keyframes based on the 2D histograms of the planes and line features. To handle the intensity distortion of the descriptor matching, Luo et al. (2022) encoded the normals and used a consensus set maximization algorithm to compute the transformation.

Histogram-free Methods. These methods generally used point cloud projection, image representation, or spatial relations to describe the point cloud. They achieved superior performance in many scenarios but may fail when there were large lateral offsets between two places. For instance, M2DP (He et al., 2016) projected the point cloud into a set of 2D planes and exploited a 192-dimension vector to describe the density signatures. To overcome the problem of feature loss, Scan Context (Kim and Kim, 2018) created radial and azimuthal bins in the horizontal plane and recorded the maximum height of points in each bin. It relied on the column shifts to achieve rotation invariance but assumed that the vehicle was driving in a flat urban environment. LiDAR Iris (Wang et al., 2020a) created binary signature images by LoG-Gabor filtering and thresholding operations, then computed descriptor similarity by Hamming distance. It applied Fourier Transform in the frequency domain to achieve place recognition. To overcome the seasonal and viewpoint changes, Cieslewski et al. (2016) aggregated sparse triangular landmarks into compact signatures and found revisited places by collecting the matches between query landmarks and databases.

2.3. Segment-based methods

These methods divided the point cloud into sets of segment clusters, and then extracted features from each cluster for place recognition. They provided a good compromise between local and global descriptions but suffered from moving objects and structural changes and required rich 3D geometry structures for segmentation. For instance, SegMatch (Dubé et al., 2017) and Segmap (Dube et al., 2020) used Euclidean clustering to cluster the point cloud into a set of segments, and performed K-Nearest Neighbors (KNN) retrieval in feature space. OneShot (Ratz et al., 2020) followed SegMap but extracted segments from a single LiDAR scan, which integrated visual appearance into a point cloud segment-based descriptor. To avoid global map construction and segment-wise KNN search, Locus (Vidanapathirana et al., 2021) creates a global descriptor by second-order pooling and nonlinear transformation to solve the retrieval problem. To consider the spatial relationship of internal structure, Gong et al. (2021) proposed a two-level framework, which used shape characteristics and spatial relationships of clusters to perform place recognition. Fan et al. (2020b) proposed a segmentation-based egocentric descriptor in a single LiDAR scan, which encoded the topological information of segmented objects to achieve rotation and translation invariance simultaneously.

2.4. Semantics-based methods

These methods usually defined the knowledge database in advance, then performed semantic segmentation to assign specific category information to each point. Semantic features were more similar to human perception and more robust to occlusions and viewpoint changes. However, semantic labels were limited and could not cover all categories. Complex calculations also required more runtime.

Towards humanoid perception, Kong et al. (2020) proposed a semantic graph representation with semantics and topology, and designed a graph similarity network to transform place recognition into a graph-matching problem. GOSMatch (Zhu et al., 2020) represented a LiDAR scan as a graph whose nodes meant position and edges meant distances. This method used graph descriptors to search loop candidates and applied vertex descriptors to calculate the one-to-one correspondence. To improve the descriptor's descriptive ability, Semantic Scan Context (SSC) (Li et al., 2021a) replaced the height information in Scan Context (Kim and Kim, 2018) with semantics to construct descriptors. A two-step global semantic ICP was further proposed to improve the matching accuracy. OverlapNet (Chen et al., 2022) proposed a siamese neural network that used depths, normals, and intensities of semantic classes to predict the overlap and yaw angle between LiDAR pairs.

2.5. Data-driven methods

These methods generally designed learning networks to extract descriptive features for place recognition, but these features were difficult to interpret. They had achieved outstanding efficiency and desired recognition accuracy, but they needed enough training samples and a large amount of data cleaning.

PointNetVLAD (Uy and Lee, 2018) combined PointNet (Charles et al., 2017) and NetVLAD (Arandjelovic et al., 2016) to extract a global descriptor for end-to-end place recognition. Unfortunately, this method suffers from PointNet's weakness to capture high-level features. LPD-Net (Liu et al., 2019) addressed this weakness by adding an adaptive local feature extraction module and a graph-based neighborhood aggregation module. PCAN (Zhang and Xiao, 2019) took the local point features extracted by PointNet and then predicted the significance of each local point feature based on the point context. Minkloc3d (Komorowski, 2021) designed an encoder–decoder structure,

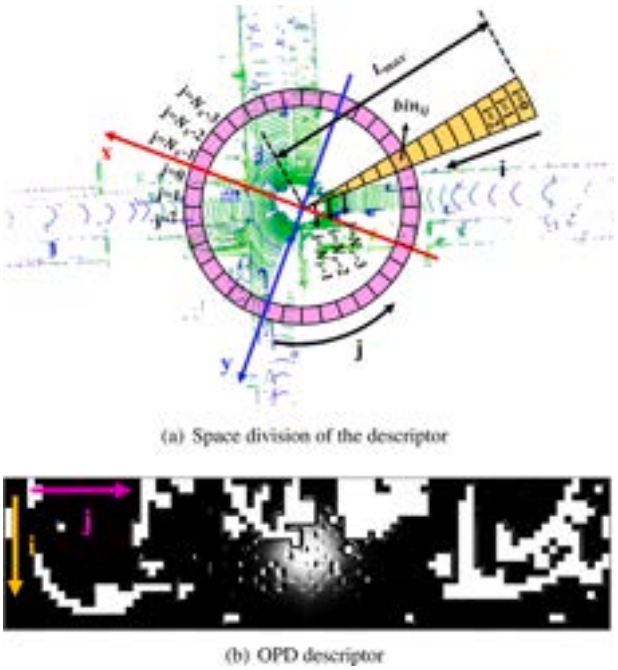


Fig. 2. Occupied place descriptor creation. (a) A raw LiDAR scan is rendered by the intensity. We partition the space around the vehicle into discrete bins in the x–y plane. The yellow region is a sector and the pink region is a ring. Their overlapped area is a bin. (b) An OPD descriptor is transformed into a binary matrix whose row and column indicate the indices of the ring (i) and sector (j), respectively. The white pixels denote occupied regions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

which extracted local features by a simple 3D convolutional neural network, then aggregated the local features into a global descriptor using a simple generalized-mean pooling. However, this method required a large amount of training work. LCDNet (Cattaneo et al., 2022) designed an architecture for loop closure detection and point cloud registration, which consisted of a shared feature extractor, a place recognition head, and a differentiable relative pose head.

2.6. Map-based methods

These methods generally built a prior map and estimated vehicle transformations using the scan-to-map associations. They can relocate a running vehicle, thus were also a solution for pose estimation. However, dense maps made the computation inefficient, and map compression remained a challenging problem.

Millane et al. (2019) designed a distance function map to represent the local geometry of both free and occupied space. Yin et al. (2020) presented a semi-handcrafted feature learning method using artificial statistics and a siamese network, then transformed the place recognition problem into a similarity modeling problem. Shi et al. (2021) extracted indoor wall segments from online LiDAR scan and prior map, respectively, then located the robot based on wall matching. To resolve the issues of low accuracy and limited generality, Xu et al. (2022) transformed the map into virtual scans and simultaneously encodes the elevation and point density, then designed a two-stage similarity estimation and Nearest Cluster Distance Ratio (NCDR) to improve the accuracy of place recognition.

3. OPD descriptor

In this section, we first introduce the Scan Context descriptor and then describe our OPD descriptor accordingly, which aims to highlight the differences between the two descriptors.

3.1. Down-sampling and space division

Down-sampling. Scan Context down-samples each single LiDAR scan with a voxel grid. OPD descriptor uses random sampling to down-sample the input point cloud to retain a fixed number of points.

Space Division. Both Scan Context and our OPD transform the point cloud into a BEV representation. They divide the horizontal space into azimuth rings and radial bins and define the overlapped region as a bin. As shown in Fig. 2, the only difference is that we assign a larger radial index to the points closer to LiDAR, while Scan Context assigns a smaller one.

3.2. Descriptor parameters

Let S be the input point cloud. Scan Context defines the following descriptor parameters:

$$(L_{max}, N_r, N_s) \tag{1}$$

where N_r and N_s are the numbers of rings and sectors, respectively. L_{max} is the horizontal effective range. Our OPD descriptor defines the following parameters:

$$(L_{max}, N_r, N_s, H_{min}, H_{max}) \tag{2}$$

where N_r and N_s are the numbers of rings and sectors, respectively. L_{max} is the horizontal effective range. H_{min} and H_{max} are two height thresholds. $H_{min} = -H_L$, $H_{max} = H_{min} + \triangle h$. H_L is the LiDAR's installation height and $\triangle h$ is a height range, which is usually set to three meters. Because our OPD descriptor focuses on the occupation of vertical structures (buildings and vegetation), we use these two thresholds to eliminate the impact of the ground and ceiling (indoor).

3.3. Encoding function

Let $\mathcal{P}_{ij}$ be the set of points in the bin where ith ring and jth sector overlap. Scan Context assigns a numerical value to this bin by retaining the maximum height of these points, as follows:

$$\phi(\mathcal{P}_{ij}) = \max_{\boldsymbol{p} \in \mathcal{P}_{ij}} z(\boldsymbol{p}) \tag{3}$$

where ϕ is the encoding function. $z(\cdot)$ is the function that returns the z coordinate of the point $\boldsymbol{p}$. We assign a binary value to every bin using the following encoding function:

$$\phi(\mathcal{P}_{ij}) = \begin{cases} 0 & N(\mathcal{P}_{ij}) < N_{min} \\ 1 & N(\mathcal{P}_{ij}) \geq N_{min} \end{cases} \tag{4}$$

where $N(\cdot)$ is the function that counts the points' number within the bin. N_{min} is a numerical threshold. Because the original LiDAR point cloud is down-sampled, we set N_{min} to 1 in our experiments. A LiDAR point cloud is then represented as a $N_r \times N_s$ matrix.

4. Map data generation

In this section, we describe the process of offline map data generation. Firstly, the 2D raster map, then the map candidate points, and finally the map descriptors. We build two offline kd trees, one is $tree_c$ for map candidate points, and the other is $tree_d$ for map descriptors.

4.1. 2D raster map

Our point cloud map is stitched over offline recorded data. We can use the reference data provided by high-precision integrated navigation to calculate the vehicle's poses. If it is unavailable, we can also compute scan-to-scan transformation by LiDAR odometry algorithm such as LOAM (Ji and Singh, 2017). Firstly, we select a new keyframe every time the vehicle travels d_k meters and reserve the points whose heights are between H_{min} and H_{max} in each keyframe. Then, we register all

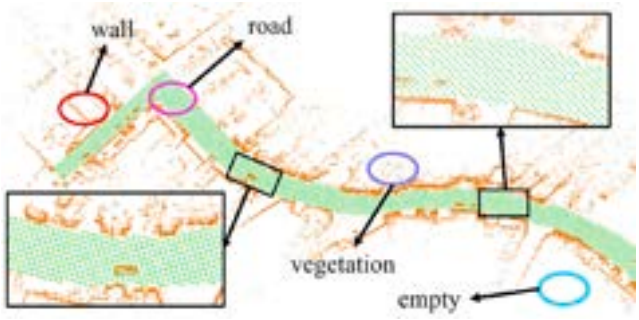


Fig. 3. Illustration of map candidate points. The colored circles denote different map regions and the black denotes map points. The roads are the best regions to generate map candidate points. The orange and green denote map points and candidate points, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

keyframes together based on the pose information to build a 3D point cloud map. Finally, we use random sampling to down-sample the 3D map to retain N_m points and create a 2D raster map with a resolution of C_r in the x–y plane.

4.2. Map candidate points

As illustrated in Fig. 3, we refer to the positions within a map where vehicles may pass by as candidate points. When the vehicle arrives near a certain candidate point, the online descriptor should be most similar to the one generated centered on that candidate point. In other words, the map candidate point with the most similar descriptor will be considered as the vehicle's position at this moment. Generally, vehicles only travel on urban or rural roads, thus the road regions are the best places to generate candidate points. Note that the candidate point generations are not the core of our approach as offline point cloud map processes are allowed before online localization. Our place recognition is generic to other candidate point generation methods. However, we still put forward several suggestions about creating candidate points: (i) Ground points can be used to generate candidate points. (ii) Removal of equidistantly sampled points in the x–y plane is also a good method. In this paper, we use the second method and the distance between two candidate points is r_c. We build a kd tree $tree_c$ for the map candidate points.

4.3. Map descriptors

As shown in Fig. 4, we use the point cloud map, candidate points, and OPD descriptor to create offline map descriptors. Taking a candidate point p_0 as an example, we first create a descriptor F_0 with p_0 as the center and the point cloud map as the input. This process aims to simulate that the vehicle is traveling to point p_0. Then, we perform column shift (Kim and Kim, 2018) on F_0 to produce a descriptor subset $\{F_k\}, k = 0, 1, \ldots, N_{s-1}$ and transform each column-shifted matrix into a row vector. Each column shift represents the vehicle's heading rotates $\frac{2\pi}{N_s}$ degrees around p_0. Next, we stack the descriptor row vectors of all candidate points into a large matrix, which ensures we can calculate the vehicle's position and orientation through the vector's position index. Finally, we build a kd tree $tree_d$ for the matrix. The purpose of creating map descriptors is to offline collect the place and orientation of vehicles that may appear within the map. When we find the best match for the online descriptor in map descriptors, we can quickly calculate the vehicle's poses by descriptor comparisons.

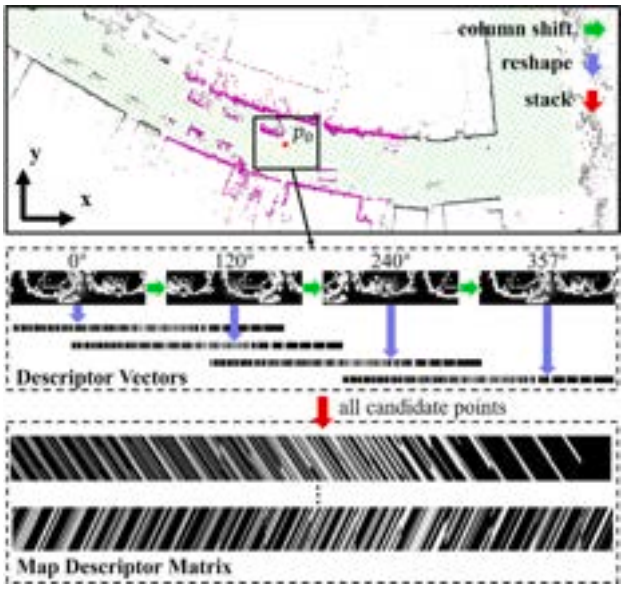


Fig. 4. Illustration of map descriptors creation. Top: the green points denote the map candidate points. p_0 is a query candidate point. The black points are the point cloud map. The pink points represent which are located within the effective horizontal range of p_0. Middle: each descriptor is transformed into a descriptor vector. Bottom: Descriptor vectors of all candidate points are stacked to a large map descriptor matrix. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Trajectory initialization

In this section, We first introduce two pose estimation modules: global search and local search. Then, we elaborate on how they co-operate to compute the vehicle's initial trajectory in a trajectory score computation module.

5.1. Global search

As shown in Fig. 5, global search first calculates the OPD descriptor of a single LiDAR scan, then uses $tree_d$ to find the corresponding best-matched one from map descriptors and calculates the vehicle's pose. Global search can be formalized as:

$$f^* = \arg\min_{i,j} d(f^s, f^m_{i,j}) \tag{5}$$

$$\mathbf{T}^* = \begin{bmatrix} \mathbf{R}_j & \mathbf{t}_i \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \mathbf{R}_j = \begin{bmatrix} \cos\theta_j & -\sin\theta_j \\ \sin\theta_j & \cos\theta_j \end{bmatrix}, \mathbf{t}_i = \begin{bmatrix} p_{i,x} \\ p_{i,y} \end{bmatrix} \tag{6}$$

where $d(\cdot)$ is the presented loss function and $d(f^{src}, f^{dst}) = \sum_{k \in N_r * N_s} \|1 - f^{src}_k f^{dst}_k\|_2$. f^{src} means an online OPD descriptor vector and f^{dst} means a map descriptor vector. k is the index of the bin. f^* is the map descriptor vector with the lowest loss. i and j denote the indices of the map candidate point and column shift, respectively. $\mathbf{T}^*$ is the optimal transformation. θ_j is the rotation angle computing from column shift and $\theta_j = -\frac{2\pi}{N_s} * j$. $\mathbf{t}_i$ is the translation computed from the map candidate point p_i. Global search finds the best match for the online descriptor across the map descriptors each time. The map data used for global search is created offline while the search process is online.

5.2. Local search

The local search uses the vehicle's position of the last scan, candidate maps, and map descriptors to estimate the current poses. Firstly, given the vehicle's position in the latest scan, we search for N_{nc} nearest map candidate points using $tree_c$. Then, the corresponding descriptor subset f^{sub} is retrieved from the map descriptors. Finally, we find

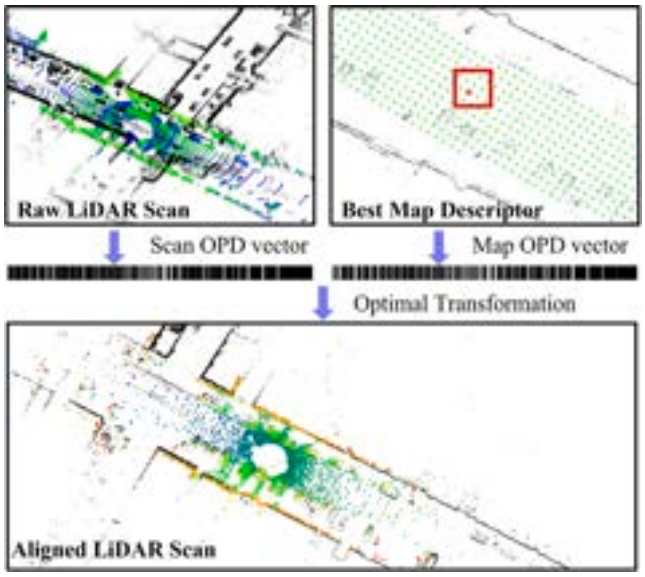


Fig. 5. Illustration of *global search*. The black points denote the point cloud map. Left top: A single LiDAR point cloud is rendered based on its intensity information and it has not been registered into the map coordinate. Right top: The green points denote map candidate points and red is the best-matched one. The loss between a map descriptor at this candidate point and the online descriptor is minimal. Bottom: The registered LiDAR scan is rendered according to its height information to illustrate the localization result. This scan has been successfully registered with the map. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the best-matched descriptor in the descriptor subset and compute the transformation as:

$$f^* = \arg\min_{k_1,k_2} d(f^s, f^{sub}_{k_1,k_2}) \tag{7}$$

where $d(\cdot)$ is the same as that in Eq. (5), which is the proposed loss function. f^{sub} denotes the map descriptor subset. f^* is the best-matched map descriptor. k_1 is the candidate point index and k_2 is the column shift index. Compared to map descriptors, the capacity of the local descriptor subset is significantly reduced, so we can quickly compute the transformation using only brute force search.

5.3. Trajectory score computation

As shown in Fig. 6, we combine global search and local search to compute an initial trajectory. Firstly, we select N_{tc} consecutive scans $S = \{S_i\}$ where $i = 0, 1, \ldots, N_{tc} - 1$, from the beginning of the LiDAR sequence. Then, we respectively perform the global search for each LiDAR scan and calculate the vehicle's global poses. Next, starting with each scan, we sequentially perform the local search to calculate the vehicle's poses in other scans, thus obtaining N_{tc} trajectories. Finally, we calculate the score w_k for all trajectories and select the one with the highest score as the initial trajectory:

$$w_k = \frac{\sum_i s_g(i)}{N_{tc}}, s_g(i) = \frac{N_h(\mathbf{T}_i, S')}{N_{S'}} \tag{8}$$

where k is the trajectory index and i is the scan index. s_g is the global score. $\mathbf{T}_i$ denotes the vehicle pose. S' is the LiDAR scan after downsampling and height filtering using H_{min} and H_{max}. $N_{S'}$ is the number of points of S'. $N_h(\mathbf{T}_i, S')$ is the number of points of hitting the raster map using $\mathbf{T}_i$ and S'.

To obtain a correct initial trajectory, we propose the following two requirements for each scan: (1) The Local score $s_l = 1 - \frac{d(f^s, f^m)}{N_r * N_s}$ should exceed S_1. (2) The global score s_g for should exceed S_2. S_1 and S_2 are two score thresholds. s_l evaluates the distributional similarity of the valid structures in two descriptors and s_g evaluates the overlap rate between the transformed LiDAR scan and the map. The two requirements improve the success rate of the initial trajectory.

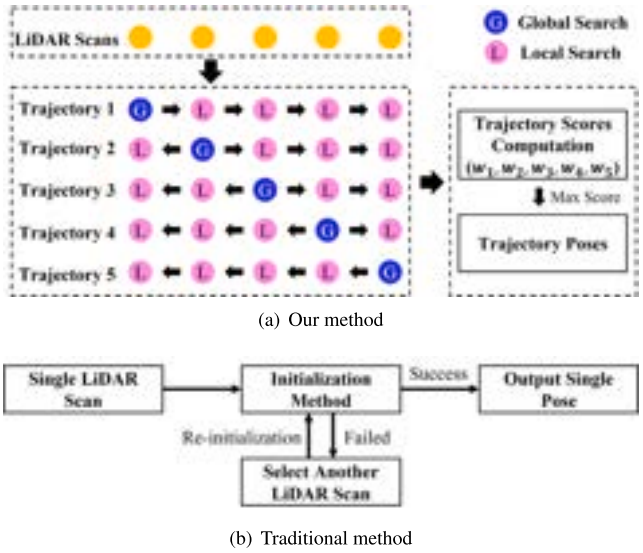


Fig. 6. Comparisons of trajectory initialization. In our method, we plot five LiDAR scans just to illustrate the calculation process.

There are two main differences between our method and traditional initialization methods: (1) Traditional methods typically select a single scan for initialization and output a pose, while our input is a set of scans and the output is a trajectory. (2) Traditional methods select another scan for initialization when localization fails. Our method uses the localization results of other scans and local search to calculate the poses of failed scans, thus providing a complete trajectory.

6. Online localization

In this section, we first describe our vehicle motion model and observation model. Then, we introduce the online localization method based on the proposed Kalman Filtering. The local search can achieve excellent localization performance in short trajectories but may accumulate huge pose drifts if the trajectory becomes longer. Therefore, we propose to join the Kalman Filtering with OPD descriptor matching to improve online localization in Fig. 7.

6.1. Vehicle motion model

Here let $\boldsymbol{x}_{t+1}$ denote the vehicle position variable at time $t+1$. t and $t-1$ are the collecting time of the previous two LiDAR scans. We assume that the vehicle travels at a constant speed during two consecutive scans, and then the motion propagation equations are formalized as:

$$\begin{aligned}\tilde{\mathbf{x}}_{t+1} &= F(\hat{\mathbf{x}}_t, \hat{\mathbf{x}}_{t-1}, \omega_x) \\ &= \mathbf{A}_t(\hat{\mathbf{x}}_t, \hat{\mathbf{x}}_{t-1}) + \omega_x\end{aligned} \tag{9}$$

$$\tilde{\mathbf{V}}_{t+1} = \mathbf{A}_t \hat{\mathbf{V}}_{t,t-1} \mathbf{A}_t^{\mathrm{T}} + \sigma_x^2 \tag{10}$$

where $\tilde{}$ and $\hat{}$ denote prior and posterior position variables, respectively. $F(\cdot)$ means the vehicle motion model. $\mathbf{V}$ is the covariance matrix and $\mathbf{A}$ is the coefficient matrix of the error propagation function. ω_x is the position estimation error and σ_x^2 is the corresponding error variance in motion prediction. We assume that the maximum motion estimation error does not exceed the map sampling distance r_c in each computation. In this paper, we always set ω_x equal to r_c.

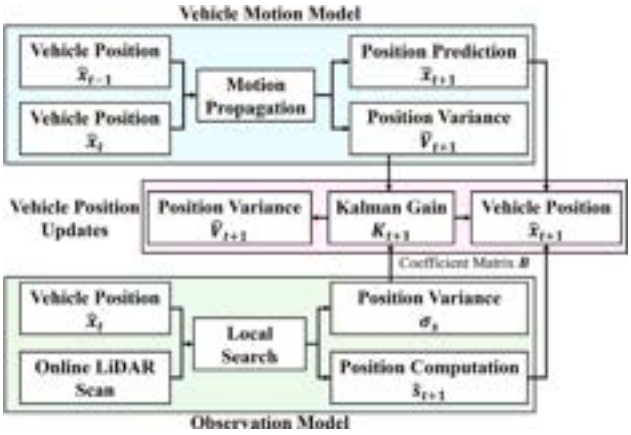


Fig. 7. Illustrations of our online localization. t is the timestamp. $\check{}$ and $\hat{}$ denote prior and posterior variables, respectively.

6.2. Observation model

Traditional Kalman Filtering-based methods often require sensors other than LiDAR to provide observations. Although this improves localization performance, it increases system costs. Our method only needs one LiDAR sensor and uses local search results as observations. The observation model is:

$$\begin{aligned} \boldsymbol{s}_{t+1} &= H(\tilde{\boldsymbol{x}}_{t+1}, \omega_s) \\ &= \mathbf{B}_{t+1}\tilde{\boldsymbol{x}}_{t+1} + \omega_s \end{aligned} \tag{11}$$

$$\sigma_s = \frac{d(f^s, f^*)}{N_r * N_s} * r_c \tag{12}$$

where $H(\cdot)$ is the observation model. $\mathbf{B}$ is the coefficient matrix of the error propagation function and we use local search instead. $s_{t+1} = p_{k_1}$ is the computed vehicle position from local search. ω_s is an observational error. We use the OPD descriptor distance to compute the error variance σ_s in Eq. (12).

6.3. Vehicle position updates

The vehicle motion model and observation model have been illustrated, and the variable update equations are as follows:

$$\mathbf{K}_{t+1} = \widetilde{\mathbf{V}}_{t+1}\mathbf{B}_{t+1}^{\mathrm{T}}(\mathbf{B}_{t+1}\widetilde{\mathbf{V}}_{t+1}\mathbf{B}_{t+1}^{\mathrm{T}} + \sigma_s^2)^{-1} \tag{13}$$

$$\hat{\boldsymbol{x}}_{t+1} = \tilde{\boldsymbol{x}}_{t+1} + \mathbf{K}_{t+1}(s_t - \mathbf{B}_{t+1}\tilde{\boldsymbol{x}}_{t+1}) \tag{14}$$

$$\hat{\mathbf{V}}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{B}_{t+1})\widetilde{\mathbf{V}}_{t+1} \tag{15}$$

where $\mathbf{K}$ is the Kalman gain. $(\cdot)^{-1}$ denotes a inverse matrix. By using the techniques described above, we can quickly calculate the vehicle position. Note that our method provides a new Kalman filtering-based solution for vehicle localization, which does not require additional sensor measurements.

7. Adaptive pose adjustment

The trajectory initialization and online localization can provide robust vehicle poses, which allows us to establish a pose database and quickly calculate revisited places. In this section, we propose an adaptive pose adjustment module to prevent drift accumulation and reduce false place recognition.

7.1. Pose drift detection

We directly compute the horizontal displacement between two consecutive scans. If a displacement exceeds the distance threshold D_t, it means there is a large drift in the current localization result. The distance threshold is mainly related to the vehicle's velocity and sensor frame rate.

7.2. Position recalculation

When there is a large displacement error, we increase the number of neighbor points to $(2N_{nc})$ in the local search and recalculate the vehicle pose in the online localization module. If the global score s_g of the recalculated pose exceeds the threshold S_2, we directly use it as the current vehicle pose. Otherwise, we use the predicted results of the vehicle motion model as the vehicle pose. After position recalculation, we store the pose in the database and find revisited places by comparing it to the database.

8. Datasets and experimental setup

In this section, we first describe experimental datasets. Then, we illustrate the implementation details and baseline methods. Finally, we describe the evaluation metrics.

8.1. Datasets

We chose two representative public autonomous driving datasets including 20 sequences in total. The dataset details are summarized in Table 2. Revisits are the ratio of all revisited scans to total scans. Oppo-ratio is the ratio of reverse revisits to all revisit scans. KA denotes KAIST and RS denotes Riverside. Fig. 8 depicts the sensor setup and vehicle trajectories. Fig. 9 illustrates the revisit distributions. We set d_k to 10 m and N_m to 100,000 for all sequences when creating the 2D raster map.

Table 2
Details of the experimental datasets.

	Seq	Length (km)	Scans	Revisits	Oppo-Ratio
KITTI	00	3.72	4541	19%(882/4541)	4%(31/882)
	01	2.45	1101	–	–
	02	5.07	4661	7%(329/4661)	16%(53/329)
	03	0.56	801	–	–
	04	0.39	271	–	–
	05	2.21	2761	20%(557/2761)	4%(24/557)
	06	1.23	1101	25%(272/1101)	–
	07	0.69	1101	13%(147/1101)	–
	08	3.22	4071	10%(406/4071)	90%(366/406)
	09	1.71	1591	1%(22/1591)	–
	10	0.92	1201	3%(38/1201)	–
MulRan	DCC 01	4.91	3574	53%(1911/3574)	45%(854/1911)
	DCC 02	4.27	3151	47%(1468/3151)	66%(967/1468)
	DCC 03	5.42	3742	53%(1997/3742)	70%(1399/1997)
	KA 01	6.13	4192	41%(1715/4192)	24%(420/1715)
	KA 02	5.97	4190	45%(1876/4190)	24%(444/1876)
	KA 03	6.25	4224	49%(2055/4224)	–
	RS 01	6.43	4533	47%(2151/4533)	–
	RS 02	6.61	4871	45%(2172/4871)	–
	RS 03	7.25	5264	49%(2593/5264)	–

KITTI. The KITTI dataset (Geiger et al., 2012) has been widely used in autonomous driving tasks such as stereo matching, Odometry, object tracking, and semantic segmentation. The recording platform consists of a car, a stereo camera system, a Velodyne HDL-64E LiDAR, and an OXTS RT 3003 localization system. The LiDAR has a 360° horizontal FOV and can produce over one million 3D points per second. It publicly provides 11 LiDAR sequences that cover mid-size cities, rural areas, and highways. The reference poses are available from the GPS/IMU measurements. We select all 11 sequences in our experiments. There are no loop events in sequences 01, 03, and 04. In sequences 09 and 10, the vehicle eventually returns to the starting point, closing the trajectory and generating a few loops. Sequences 06 and 07 only consist of the same-direction loops. Most of the loops in sequence 08 are reversed.

(a) KITTI (b) MulRan

Fig. 8. Experimental datasets. (a) shows the platform setup of the KITTI dataset and color images of 11 sequences. (b) shows the platform setup of the MulRan dataset and the trajectories overlaid on the aerial map. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

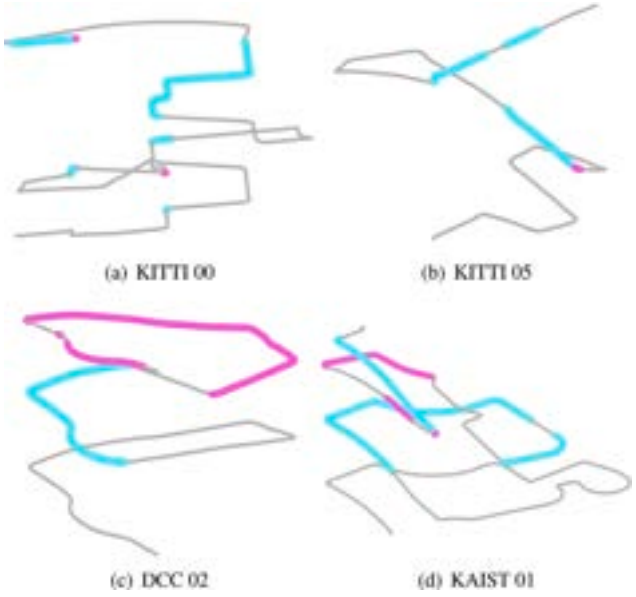


Fig. 9. Revisit distributions. The gray lines represent the vehicle trajectories. The blue points are same-direction revisits and the pink points are oppo-direction revisits. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

MulRan. The Multimodel Range Dataset (MulRan) (Kim et al., 2020) dataset is mainly designed to boost the range sensor-based place recognition research. A 64-beam LiDAR (Ouster OS1-64) and a radar (Navtech CIR204-H) are used to collect the dataset. It records three different sequences for each scenario. The LiDAR FOV loses approximately 70° due to the occlusion of the radar sensor. This dataset covers the city, campus, riverside, and still-developing city. The reference pose is computed based on SLAM using fiber optic gyro (FOG), Virtual Reference Station GPS (VRS-GPS), and Iterative Closest Point (ICP) (Besl and McKay, 1992). The DCC and KAIST contain many reverse loops. Riverside is a challenging scenario where the vehicle travel along the river and repetitive structures.

8.2. Implementation

We implement our method in C++ language on the Ubuntu 20.04LTS system. All experiments run on an Intel Core i9-10850K CPU and 64 GB of RAM. We use the SemanticKITTI (Behley et al., 2019) development kit to manage the KITTI dataset and *Extrinsic Calibration Files* in the website to manage the MulRan dataset. Two places are considered as a pair of real revisited places if the distance is less than eight meters.

8.3. Baseline methods

Descriptor-based Baseline. We compare the proposed method with eight SOTA methods, e.g., ESF (Wohlkinger and Vincze, 2011), SHOT (Salti et al., 2014), Fast Histogram (Röhling et al., 2015), M2DP (He et al., 2016), Scan Context (Kim and Kim, 2018), LiDAR Iris (Wang et al., 2020a), ISC (Wang et al., 2020b), and CSSC (Xu et al., 2022).

Localization-based Baseline. Since our method can provide online vehicle poses, we further compare its localization performance with three visual SLAM methods including ORB SLAM2 (Mur-Artal and Tardós, 2017), LDSO (Gao et al., 2018), and CNN-SVO (Loo et al., 2019), and eight LiDAR SLAM methods including LOAM (Ji and Singh, 2017), LeGo-LOAM (Shan and Englot, 2018), SUMA++ (Chen et al., 2019), HDL-Graph-SLAM (Koide et al., 2019), ISC-LOAM (Wang et al., 2020b), SA-LOAM (Li et al., 2021b), PCA-SLAM (Guo et al., 2022), and GICP-LOAM (Li et al., 2022b).

8.4. Evaluation metrics

Precision–Recall Curve. Similar to Scan Context (Kim and Kim, 2018), we also use the precision–recall curves to evaluate the performance of place recognition. A place recognition method aims to find those revisited places over the database, so it is essentially a classifier. For a query scan, its classification result will be classified into True Positive (TP), False Positive (FP), True Negative (TN), or False Negative (FN). Then, we compute the following classification metrics F_1 score:

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{16}$$

where F_1 score (0–1) is the harmonic average of precision and recall. A higher value represents better place recognition performance.

Quantitative Metrics. We select quantitative metrics to measure the performance. (i) Success rate: the higher success rate indicates a stronger generality of the algorithm. (ii) Localization accuracy: the lateral and heading errors describe the accuracy and stability of the localization system. (iii) Runtime: The execution time decides the efficiency of the algorithm and superior efficiency enables the system to handle more complex tasks.

9. Experimental evaluations

9.1. Steep scenario

As shown in Fig. 10, we select two steep sequences (KITTI 03 and 10) to verify the effectiveness. Their map height differences exceed 50 m and 35 m, respectively. Here, we set $L_{max} = 20, N_r = 20, N_s = 80, H_{min} = -0.8, H_{max} = 2.2$. To be consistent with subsequent

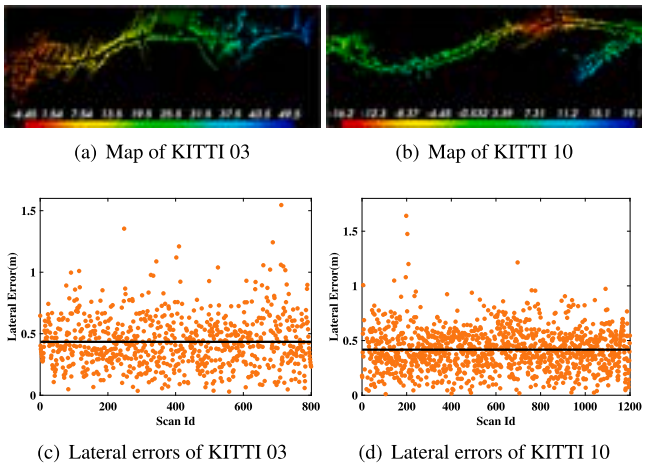


Fig. 10. Steep scenario experiments on KITTI 03 and 10. The map points are rendered based on z-values. The average lateral errors (black line) are 0.434 m and 0.432 m, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

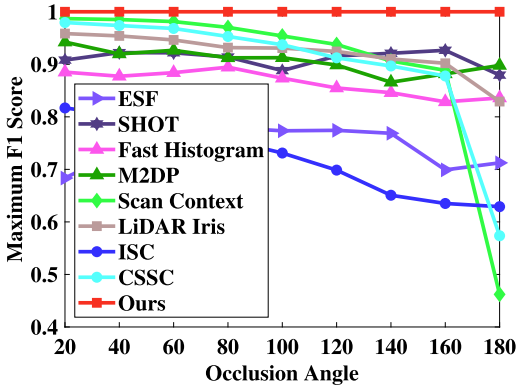


Fig. 11. Maximum F_1 scores under different occlusion angles on KITTI 06.

place recognition experiments, we consider the localization error of fewer than eight meters to be successful. The localization success rates achieve 97.3% and 97.5%, respectively. This proves that our method can successfully localize the vehicle in steep scenarios.

9.2. View occlusions

View occlusion is another serious challenge for localization and place recognition. To simulate the view occlusions, we select KITTI 06 and remove some points from each point cloud based on the yaw angle. As shown in Fig. 11, we calculate the maximum F_1 score under different occlusion angles. Our method performs better than other methods. Scan Context also achieves good results in small occlusions. However, when the occlusion range reaches 180 degrees, its F_1 score drops sharply to 0.46 but our method can still achieve the best results.

9.3. Parameter experiments

To verify the proposed descriptor more fairly, we do not use trajectory information but only use *global search* to calculate vehicle poses. It is easy to understand that if *global search* achieves satisfactory results, our method will further improve the performance when there is trajectory and motion information. As summarized in Table 3, we choose KITTI 04 for ablation experiments, which is a short trajectory without loop events. The parameters in the experiment mainly involve two parts: descriptor parameter (L_{max}, N_r, N_s) and sampling distance

Table 3
Parameter ablation experiments (Localization Success Rate) using *Global Search* on KITTI 04. L_{max} Is the effective range. N_r And N_s are the number of ring and sector, respectively. r_c Is the sampling distance of map candidate points.

L_{max}, N_r	r_c, N_s							
	r_c=1.0				r_c=2.0			
	60	72	90	120	60	72	90	120
(10, 10)	0.09	0.12	0.10	0.14	0.06	0.04	0.05	0.05
(20, 20)	0.81	0.84	0.90	0.92	0.36	0.33	0.34	0.45
(30, 30)	1	1	1	1	0.98	0.95	0.92	0.95
(40, 40)	1	1	1	1	0.99	0.97	0.98	0.98
(20, 10)	0.51	0.57	0.65	0.75	0.23	0.17	0.18	0.21
(40, 20)	1	1	1	1	0.91	0.90	0.94	0.95
(60, 30)	1	1	1	1	0.98	0.99	0.99	1
(80, 40)	1	1	1	1	0.99	1	1	1

of candidate points r_c. We find that larger descriptor parameters yield better performances. Also, if the length of bin $\frac{L_{max}}{N_r}$ is smaller than the sampling distance of candidate points r_c, it may produce more wrong results.

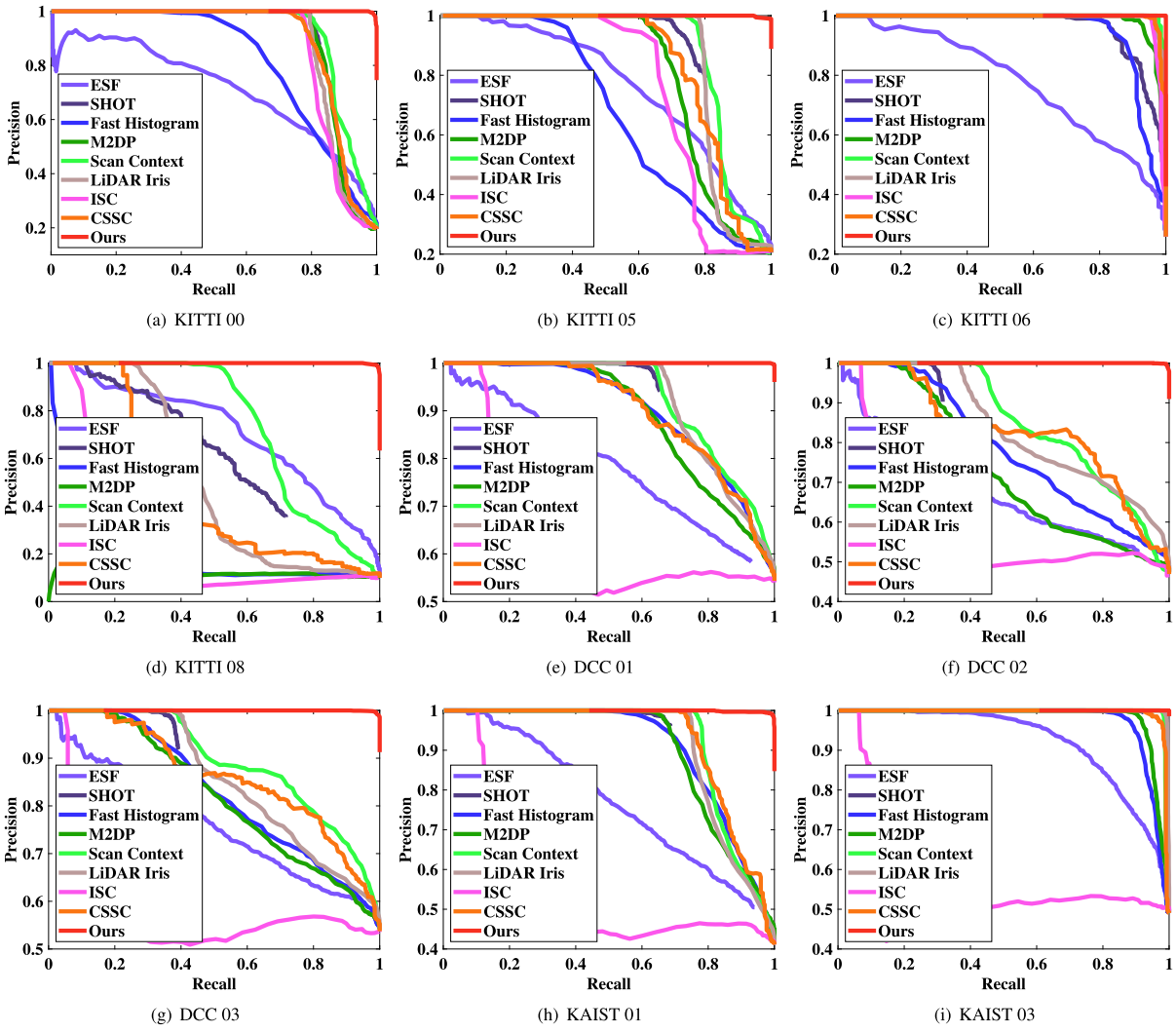
9.4. Place recognition performance

PR Curves and F_1 Score. As shown in Fig. 12, we plot the PR curves of nine sequences. We can see that our method obtains stable PR curves. Although the recall is low at the beginning of the curve, it still achieves high precision, which surpasses other baseline methods. Table 4 summarizes the maximum F_1 score of 15 sequences. Our method achieves average maximum F_1 scores of 0.987 and 0.880 on KITTI and MulRan datasets, respectively. Our method degenerates slightly in Riverside sequences but performs best in other sequences. The Riverside sequences contain two bridges and riverside roads, with highly repetitive structural features. Our local search is stuck in certain map areas, which caused our online localization to deviate from the real trajectory. Our method is prone to degradation for long-time, long-distance driving in narrow, repetitive environments.

Same-direction Revisit. As shown in Fig. 13, we compare the proposed method with M2DP, Scan Context, and LiDAR Iris on KAIST 02, which covers 45% revisits. At maximum precision: LiDAR Iris only yields one correct match but Scan Context, our method, and M2DP obtain 1434, 1137, and 982 correct matches, respectively. At maximum F_1 score: our method obtains more correct results (1861) than Scan Context (1660), LiDAR Iris (1492), and M2DP (1201). At recall of 50%: LiDAR IRIS detects a large number of false loops (2376) while the quantity of our correct matches (1861) is the largest.

Oppo-direction Revisit. As shown in Fig. 14, we choose DCC 02 to compare the performance of detecting reverse loops. In this sequence, the vehicle traverses diverse structural environments including square, narrow roads between high-rise buildings, a mountain, and a cross-roads. It contains 66% of reverse revisits. At maximum precision: Scan Context detects more right loops (591) while we get 351 right results with no false matches. We achieve the best performance, i.e., detecting the most correct loops and the fewest false loops in the other two conditions. M2DP detects the most false matches (2475) at max F_1 score, and LiDAR Iris detects the most false matches (2225) at the recall of 50%. Note that our method also produces a few false matches, however, they are indeed close due to the revisit distance threshold (eight meters). If we moderately increase the distance threshold, our method will produce fewer false results.

Success Rate Under Different Candidates. As summarized in Table 5, we compare the success rate under different candidate capacities on 5 KITTI sequences. Our success rate exceeds 99% in sequence 00, 05, 06, and 07 when only using one candidate, and achieves 100% when using 10 candidates. There are relatively more reverse loops in sequence 02 and all four methods achieve relatively poor results. LiDAR Iris performs best when using one or five candidates, but our method performs better when the number of candidates reaches 10.



(a) KITTI 00 (b) KITTI 05 (c) KITTI 06

(d) KITTI 08 (e) DCC 01 (f) DCC 02

(g) DCC 03 (h) KAIST 01 (i) KAIST 03

Fig. 12. PR curves of nine sequences on KITTI and MulRan datasets.

Table 4
Maximum F_1 Score On KITTI and mulRan datasets. Bold indicates the best and underlined is the second best.

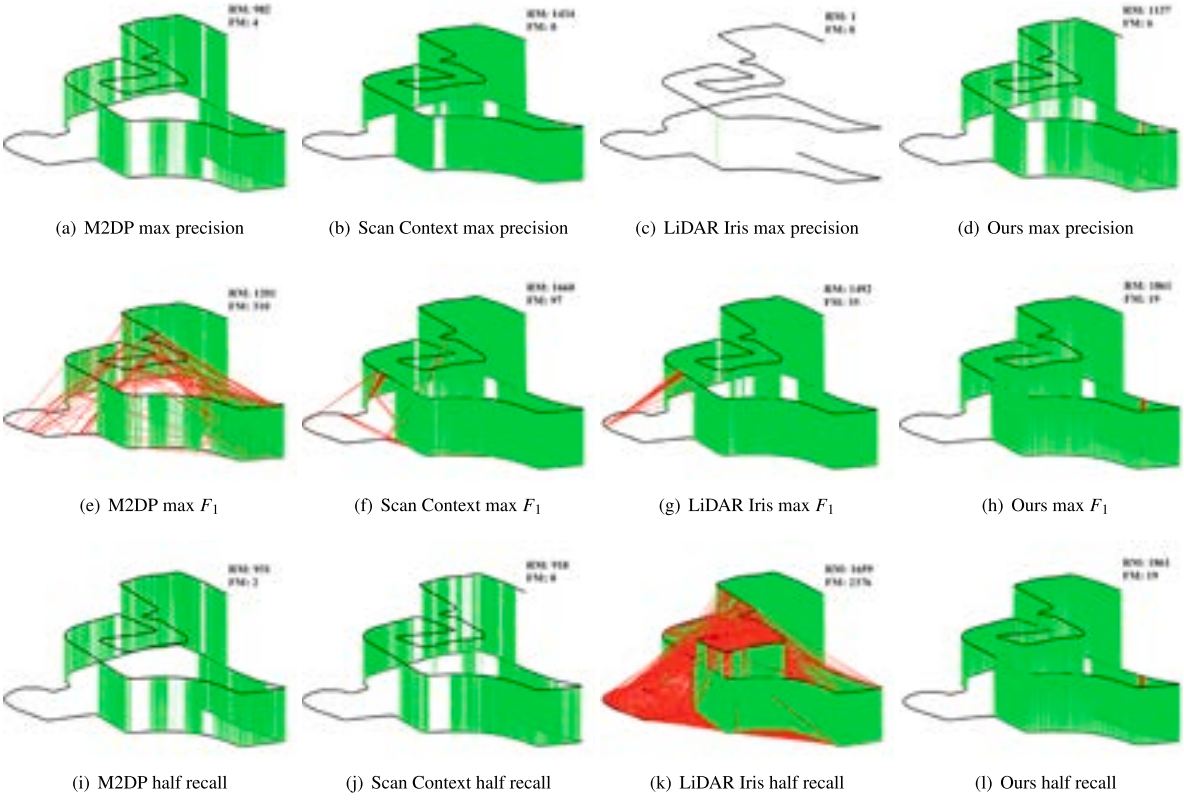
Method	00	02	05	06	07	08	Mean	DC1	DC2	DC3	KA1	KA2	KA3	RI1	RI2	RI3	Mean
ESF (2011)	0.661	0.610	0.683	0.686	0.659	0.650	0.658	0.720	0.670	0.733	0.687	0.702	0.825	0.710	0.645	0.713	0.712
SHOT (2014)	0.881	0.827	0.824	0.895	0.541	0.572	0.757	0.774	0.469	0.550	0.801	0.737	0.958	0.840	0.780	0.833	0.749
Fast Histogram (2015)	0.735	0.581	0.596	0.895	0.314	0.208	0.555	0.800	0.696	0.744	0.805	0.743	0.926	0.770	0.720	0.730	0.758
M2DP (2016)	0.877	0.759	0.775	0.950	0.467	0.205	0.672	0.766	0.660	0.738	0.798	0.767	0.946	0.779	0.745	0.765	0.774
Scan Context (2018)	0.881	0.789	0.863	0.987	0.491	0.697	0.785	0.813	0.750	0.803	0.864	0.919	0.989	0.831	0.731	0.826	0.836
LiDAR Iris (2020)	0.882	0.827	0.875	0.980	0.468	0.506	0.756	0.801	0.752	0.751	0.851	0.879	0.992	**0.916**	**0.785**	**0.880**	0.845
ISC (2020)	0.868	0.748	0.760	0.974	0.478	0.250	0.680	0.705	0.662	0.698	0.616	0.635	0.671	0.752	0.725	0.709	0.663
CSSC (2022)	0.859	0.724	0.790	0.978	0.468	0.424	0.707	0.805	0.773	0.798	0.845	0.906	0.974	0.744	0.667	0.737	0.825
Ours	**0.991**	**0.955**	**0.995**	**1.000**	**0.990**	**0.991**	**0.987**	**0.998**	**0.994**	**0.993**	**0.992**	**0.999**	**1.000**	0.675	0.614	0.656	**0.880**

Table 5
Success rate comparison under different candidates on KITTI sequences. Bold indicates the best and underlined is the second best.

Method	00				02				05				06				07			
	1	5	10	20	1	5	10	20	1	5	10	20	1	5	10	20	1	5	10	20
M2DP (2016)	0.857	0.864	0.869	0.875	0.660	0.691	0.721	0.744	0.731	0.764	0.780	0.802	0.985	0.993	0.993	0.996	0.411	0.489	0.512	0.528
Scan Context (2018)	0.884	0.901	0.911	0.923	0.717	0.776	0.799	0.843	0.819	0.891	0.900	0.935	0.993	0.996	1.000	1.000	0.582	0.657	0.657	0.657
LiDAR Iris (2020)	0.876	0.897	0.905	0.910	**0.778**	**0.784**	0.796	0.799	0.867	0.874	0.878	0.880	0.985	0.989	0.993	0.993	0.551	0.592	0.701	0.728
Ours	**0.993**	**0.999**	**1.000**	**1.000**	0.732	0.783	**0.841**	**0.932**	**0.996**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.986**	**0.993**	**1.000**	**1.000**

9.5. Localization performance

Relative Pose Estimation. Since Scan Context and LiDAR Iris only provide the heading orientation estimation, and M2DP cannot compute any information related to the pose, we compare our method to ICP. Our method separately calculates the global pose T_s and T_d of two scans, then further obtains the relative transformation $T_d * T_s^-$. The visualization of registration is shown in Fig. 15. We randomly select

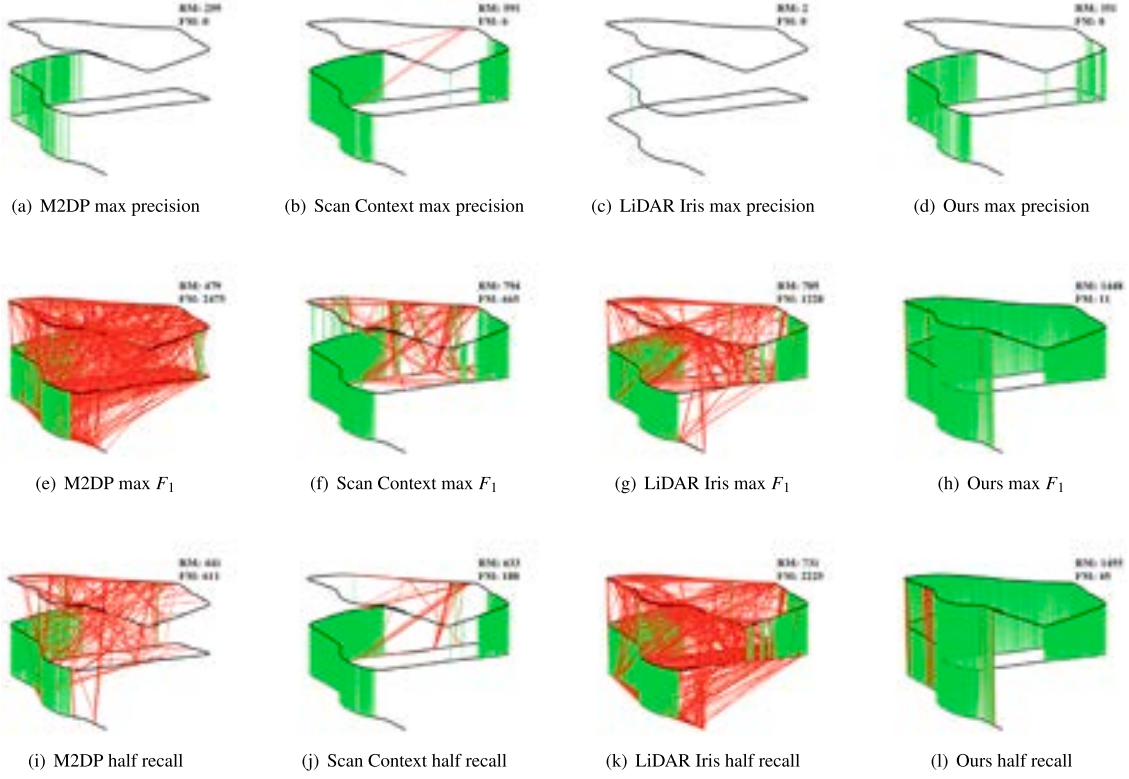


(a) M2DP max precision (b) Scan Context max precision (c) LiDAR Iris max precision (d) Ours max precision

(e) M2DP max F_1 (f) Scan Context max F_1 (g) LiDAR Iris max F_1 (h) Ours max F_1

(i) M2DP half recall (j) Scan Context half recall (k) LiDAR Iris half recall (l) Ours half recall

Fig. 13. Matching of the same-direction revisits on KAIST 02. The black lines denote the vehicle trajectories. The green lines are correct matches. The red lines are wrong matches. RM means the right match and FM means the false one. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

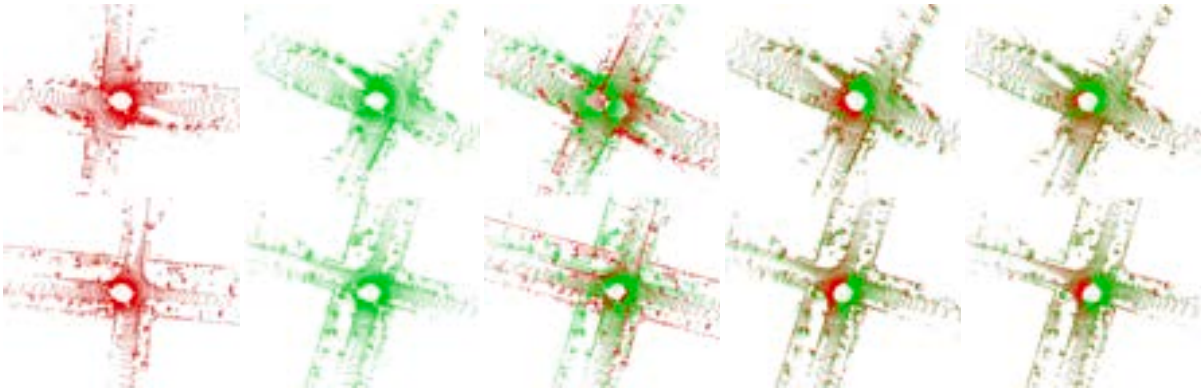


(a) M2DP max precision (b) Scan Context max precision (c) LiDAR Iris max precision (d) Ours max precision

(e) M2DP max F_1 (f) Scan Context max F_1 (g) LiDAR Iris max F_1 (h) Ours max F_1

(i) M2DP half recall (j) Scan Context half recall (k) LiDAR Iris half recall (l) Ours half recall

Fig. 14. Matching of the oppo-direction revisits on DCC 02. The black lines denote the vehicle trajectories. The green lines are correct matches. The red lines are wrong matches. RM means the right match and FM means the false one. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Source (b) Target (c) ICP (d) Ours (e) Ours+ICP

Fig. 15. Pairwise registrations on KITTI 08. We select two pairs of revisited scans, 1422–791 (top) and 3866–2518 (bottom).

Table 6
Relative pose errors (Rotation and Translation) on KITTI sequences. T2d (m) means the 2d translation error and T3d (m) Is The 3d One. Rot (deg) means The 3d rotational error.

Method	00			02			05			06			07			08		
	T3d	T2d	Rot	T3d	T2d	Rot	T3d	T2d	Rot	T3d	T2d	Rot	T3d	T2d	Rot	T3d	T2d	Rot
ICP	6.23	6.20	88.08	5.41	3.15	23.60	7.73	7.72	65.19	2.73	2.72	1.24	8.40	8.39	5.09	7.17	7.15	158.47
Ours	1.21	0.98	2.37	0.89	0.86	2.91	0.64	0.63	3.37	0.44	0.42	0.51	0.44	0.43	1.98	0.63	0.61	2.69
Ours+ICP	**1.21**	**0.98**	**1.11**	**0.54**	**0.52**	**1.01**	**0.64**	**0.63**	**0.80**	**0.30**	**0.28**	**0.40**	**0.42**	**0.41**	**1.58**	**0.61**	**0.44**	**0.37**

Table 7
Comparisons Of mean absolute trajectory error (ATE) in meters on KITTI datasets. N/A: Not available, -: Value is larger than 30. Bold indicates the best and underlined is the second best. We obtain these numbers from the published papers.

Method	00	01	02	03	04	05	06	07	08	09	10
ORB SLAM2[a] (2017, Visual)	1.3	10.4	5.7	0.6	**0.2**	0.8	0.8	0.5	3.6	3.2	1.0
LDSO[a] (2018, Visual)	9.32	11.68	–	2.85	1.22	5.10	13.55	2.96	–	21.64	17.36
CNN-SVO[a] (2019, Visual)	17.53	N/A	–	3.46	2.44	6.51	11.51	6.51	10.97	10.69	4.84
LOAM[b] (2017, LiDAR)	4.51	18.78	–	0.85	0.4	2.1	1.1	0.7	4.32	1.53	1.49
LeGo-LOAM[b] (2018, LiDAR)	4.12	–	–	1.16	0.75	2.44	1.02	0.84	4.73	2.79	2.61
SUMA++[c] (2019, LiDAR)	1.17	N/A	12.99	N/A	N/A	0.64	0.56	0.37	2.44	1.19	N/A
HDL-Graph-SLAM[b] (2019, LiDAR)	5.00	–	20.09	1.11	27.31	2.78	1.25	0.62	N/A	N/A	N/A
ISC-LOAM[c] (2020, LiDAR)	1.60	N/A	4.77	N/A	N/A	2.49	1.03	0.56	4.88	2.31	N/A
SA-LOAM[c] (2021, LiDAR)	0.99	N/A	9.24	N/A	N/A	0.75	0.64	**0.36**	3.24	1.20	N/A
PCA-SLAM[a] (2022, LiDAR)	2.90	10.01	8.92	0.80	0.42	1.96	0.89	0.68	4.11	1.9	1.47
GICP-LOAM[b] (2022, LiDAR)	2.78	17.45	12.22	1.31	0.92	2.45	1.34	0.82	N/A	N/A	N/A
Ours (LiDAR)	**0.49**	**0.65**	**0.96**	**0.48**	0.41	**0.42**	**0.43**	0.45	**0.53**	**0.52**	**0.47**

[a]Indicates results are reported by corresponding original papers.
[b]Indicates results are from GICP-LOAM (Li et al., 2022b).
[c]Indicates results are from SA-LOAM (Li et al., 2021b).

50 pairs of loop scans and summarize the average registration errors in Table 6. Since the distance between two scans exceeds one meter, traditional ICP falls in the local minima, but our method can overcome the problem of overlap and achieve an average 2D lateral error of 0.6 m and an average angle error of 2.3 degrees in six sequences. We use our localization results as an initial guess, which helps ICP to converge better and the average rotation error down to 0.88 degrees.

Absolute Trajectory Error. As summarized in Table 7, we compute the Absolute Trajectory Error (ATE) on KITTI sequences to evaluate the ability to reduce cumulative errors. ORB SLAM2 achieves the best accuracy (0.2 m) on sequence 04, while SA-LOAM and SUMA++ achieve better accuracy (0.36 m and 0.37 m) on sequence 07. Our method obtains the best performance in the other nine sequences, with an average error of 0.528 m in all 11 sequences. Our localization errors do not increase as the trajectory gets longer, which is important for vehicle localization. To more intuitively analyze our localization performance, we plot the heading errors for all scans of sequence 00 in Fig. 16 and plot the trajectory errors of sequence 09 using a color map in Fig. 17.

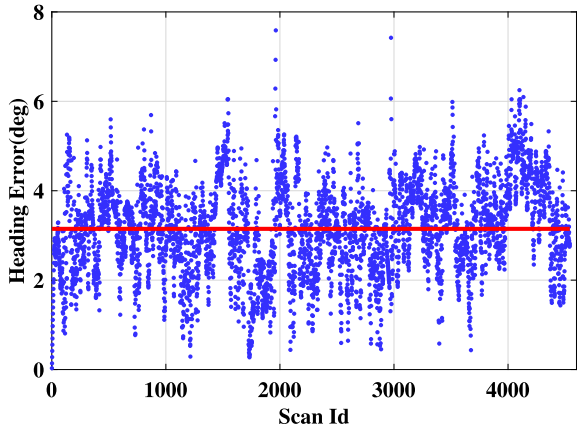


Fig. 16. Heading errors on KITTI 00. The average error (red line) is 3.14°. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

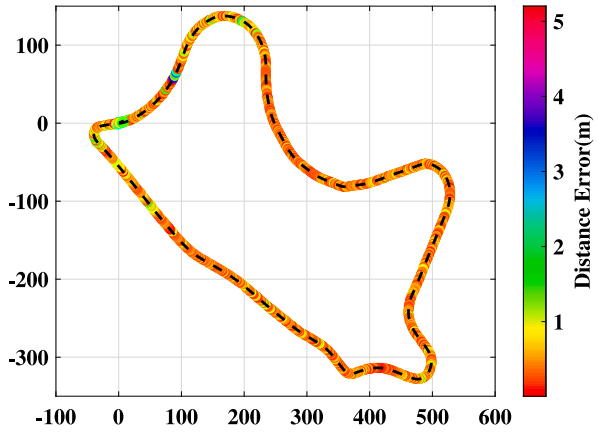


Fig. 17. The trajectory errors of KITTI 09. The black line denotes the vehicle trajectory. The point's colors are rendered based on trajectory errors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 8
Runtime comparisons (Millisecond) of place recognition on DCC 01. Bold indicates the best, and underlined is the second best.

Method	Descriptor extraction	Pairwise comparison	Map querying
M2DP (2016)	127.91	0.01	2.59
Scan Context (2018)	38.56	0.54	37.43
LiDAR Iris (2020)	**4.59**	6.76	11918.70
Ours	5.16	**0.0002**	**0.51**

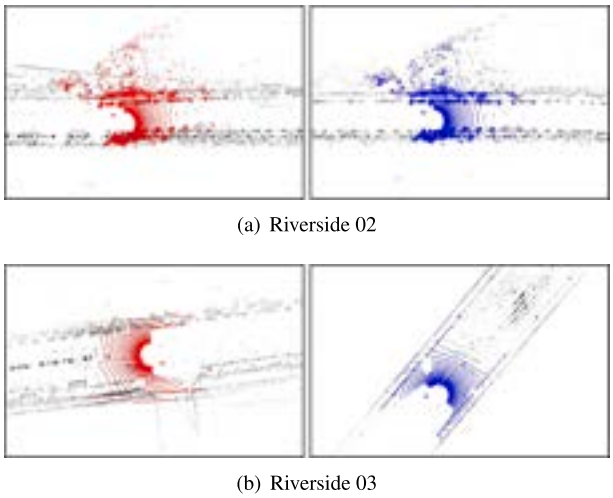
Table 9
Runtime (Seconds) of our each module.

Seq	Tree built	Map data loading	Trajectory initialization	Online localization
KITTI 05	53.123	21.920	0.184	0.005
KITTI 06	16.441	7.473	0.122	0.005
KITTI 07	4.850	1.745	0.083	0.005
DCC 03	20.612	6.932	0.080	0.004
KAIST 02	58.612	25.391	0.192	0.005
Riverside 01	54.211	21.312	0.163	0.004

9.6. Runtime analysis

Runtime Comparison. Table 8 records the runtime of place recognition. The map querying means the time of comparing the current descriptor to all previous scans. We use random sampling to downsample the online LiDAR scan to retain 8000 points. Our descriptor extraction includes point cloud sampling and descriptor generation, which takes about 5 ms and is slightly slower than LiDAR Iris, but faster than M2DP and Scan Context. Our pairwise comparison is fast (0.0002 ms) because we use the computed pose to detect loop closures. M2DP directly calculates the distance between two 192-dimensional vectors. Scan Context first extracts the ring key and then performs a column shift, which takes more comparison time. LiDAR Iris requires complex filtering and thresholding operations, which results in the longest comparison time.

Runtime of Every Module. Table 9 records the runtime of every module. The tree built is a vital component of the offline map data generation module (Section 4). Map data loading is a preliminary step in the system, preceding the trajectory initialization (Section 5) and online localization module (Section 6). The online process encompassing trajectory initialization and online localization does not necessitate extra time for processing map data. We use 10 scans in the trajectory initialization and the online localization uses the sampled point cloud. The superior localization efficiency enables our method to aid vehicles to better accomplish other tasks.



(a) Riverside 02

(b) Riverside 03

Fig. 18. Failure cases. The black points are map points. The red points are wrong localization results. The blue points are real vehicle positions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

9.7. Failure cases

As shown in Fig. 18, we illustrate two failure cases in Riverside 02 and 03. Riverside sequences contain straight riverside roads and two bridges and the environmental features are highly repetitive. Both the trees on the roadside and the buildings on the bridge are highly symmetrical, and the road's width is also very close to that of the bridge, which leads to their OPD descriptors being highly similar. Our online localization depends on descriptor-based local search and trajectory information, whose localization performance will degrade as the discriminant ability of descriptors decreases. These failure cases rarely happen because there are many structural features (e.g., roads and buildings) in most urban, rural, and highway scenarios, where our OPD descriptor has robust descriptive capabilities. Our experimental datasets include many types of scenarios and the results prove that our method has achieved outstanding performance in the majority of sequences.

10. Conclusions

In this paper, we proposed an efficient map-aided place recognition and localization approach. Different from mainstream place recognition methods, it can simultaneously find revisited places and provide high-accuracy poses. Our binary descriptor consumes a small footprint while can improve feature computational efficiency. Although our trajectory initialization module is a bit complicated, it improves the success rate of the initialization. We innovatively combine Kalman filtering with descriptor similarities, which overcomes the traditional Kalman-based method's reliance on multi-source measurements and greatly improves efficiency and accuracy. Our method overcomes the rotation and translation changes, view occlusions, and scenario limitations. It also solves the problem that pairwise point cloud registration is too dependent on initial poses. The exhaustive evaluations of public datasets demonstrate our method outperforms many state-of-the-art (SOTA) place recognition and localization methods. Although the proposed method achieves satisfactory performance, it cannot immediately respond to achieve the right re-localization when a localization failure occurs. In the future, we intend to develop a multi-sensor system that adds external sensor measurements to improve the generalization of our method.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 42030102 and 42271444, and the Science and Technology Major Project of Hubei Province, China under Grant 2021AAA010.

References

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J., 2016. NetVLAD: CNN architecture for weakly supervised place recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 5297–5307. http://dx.doi.org/10.1109/CVPR.2016.572.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J., 2019. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In: 2019 IEEE/CVF International Conference on Computer Vision. ICCV, pp. 9296–9306. http://dx.doi.org/10.1109/ICCV.2019.00939.

Besl, P., McKay, N.D., 1992. A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. 14 (2), 239–256. http://dx.doi.org/10.1109/34.121791.

Cattaneo, D., Vaghi, M., Valada, A., 2022. LCDNet: Deep loop closure detection and point cloud registration for LiDAR SLAM. IEEE Trans. Robot. 1–20. http://dx.doi.org/10.1109/TRO.2022.3150683.

Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J., 2017. PointNet: Deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 77–85. http://dx.doi.org/10.1109/CVPR.2017.16.

Chen, X., Läbe, T., Milioto, A., Röhling, T., Behley, J., Stachniss, C., 2022. OverlapNet: a siamese network for computing LiDAR scan similarity with applications to loop closing and localization. Auton. Robots 46 (1), 61–81.

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

Cieslewski, T., Stumm, E., Gawel, A., Bosse, M., Lynen, S., Siegwart, R., 2016. Point cloud descriptors for place recognition using sparse visual information. In: 2016 IEEE International Conference on Robotics and Automation. ICRA, pp. 4830–4836. http://dx.doi.org/10.1109/ICRA.2016.7487687.

Cop, K.P., Borges, P.V.K., Dubé, R., 2018. Delight: An efficient descriptor for global localisation using LiDAR intensities. In: 2018 IEEE International Conference on Robotics and Automation. ICRA, pp. 3653–3660. http://dx.doi.org/10.1109/ICRA.2018.8460940.

Cummins, M., Newman, P., 2008. FAB-MAP: Probabilistic localization and mapping in the space of appearance. Int. J. Robot. Res. 27 (6), 647–665.

Dube, R., Cramariuc, A., Dugas, D., Sommer, H., Dymczyk, M., Nieto, J., Siegwart, R., Cadena, C., 2020. SegMap: Segment-based mapping and localization using data-driven descriptors. Int. J. Robot. Res. 39 (2–3), 339–355.

Dubé, R., Dugas, D., Stumm, E., Nieto, J., Siegwart, R., Cadena, C., 2017. SegMatch: Segment based place recognition in 3D point clouds. In: 2017 IEEE International Conference on Robotics and Automation. ICRA, pp. 5266–5272. http://dx.doi.org/10.1109/ICRA.2017.7989618.

Fan, Y., Feng, Z., Shen, C., Khan, T.U., Mannan, A., Gao, X., Chen, P., Saeed, S., 2020a. A trunk-based SLAM backend for smartphones with online SLAM in large-scale forest inventories. ISPRS J. Photogramm. Remote Sens. 162, 41–49.

Fan, Y., Han, H., Tang, Y., Zhi, T., 2019. Dynamic objects elimination in SLAM based on image fusion. Pattern Recognit. Lett. 127, 191–201.

Fan, Y., He, Y., Tan, U.-X., 2020b. Seed: A segmentation-based egocentric 3D point cloud descriptor for loop closure detection. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 5158–5163. http://dx.doi.org/10.1109/IROS45743.2020.9341517.

Gao, X., Wang, R., Demmel, N., Cremers, D., 2018. LDSO: Direct sparse odometry with loop closure. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 2198–2204.

Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, pp. 3354–3361.

Gong, Y., Sun, F., Yuan, J., Zhu, W., Sun, Q., 2021. A two-level framework for place recognition with 3D LiDAR based on spatial relation graph. Pattern Recognit. 120, 108171.

Guo, J., Borges, P.V.K., Park, C., Gawel, A., 2019. Local descriptor for robust place recognition using LiDAR intensity. IEEE Robot. Autom. Lett. 4 (2), 1470–1477. http://dx.doi.org/10.1109/LRA.2019.2893887.

Guo, S., Rong, Z., Wang, S., Wu, Y., 2022. A LiDAR SLAM with PCA-based feature extraction and two-stage matching. IEEE Trans. Instrum. Meas. 71, 1–11.

He, L., Wang, X., Zhang, H., 2016. M2DP: A novel 3D point cloud descriptor and its application in loop closure detection. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 231–237. http://dx.doi.org/10.1109/IROS.2016.7759060.

Ji, Z., Singh, S., 2017. Low-drift and real-time lidar odometry and mapping. Auton. Robots 41 (2), 401–416.

Jiang, F., Chen, J., Ji, S., 2021. Panoramic visual-inertial SLAM tightly coupled with a wheel encoder. ISPRS J. Photogramm. Remote Sens. 182, 96–111.

Johnson, A., Hebert, M., 1999. Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Trans. Pattern Anal. Mach. Intell. 21 (5), 433–449. http://dx.doi.org/10.1109/34.765655.

Karam, S., Lehtola, V., Vosselman, G., 2021. Simple loop closing for continuous 6DOF LIDAR&IMU graph SLAM with planar features for indoor environments. ISPRS J. Photogramm. Remote Sens. 181, 413–426.

Kim, G., Choi, S., Kim, A., 2021. Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments. IEEE Trans. Robot.

Kim, G., Kim, A., 2018. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 4802–4809.

Kim, G., Park, Y.S., Cho, Y., Jeong, J., Kim, A., 2020. Mulran: Multimodal range dataset for urban place recognition. In: 2020 IEEE International Conference on Robotics and Automation. ICRA, IEEE, pp. 6246–6253.

Koide, K., Miura, J., Menegatti, E., 2019. A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement. Int. J. Adv. Robot. Syst. 16 (2), 1729881419841532.

Komorowski, J., 2021. MinkLoc3D: Point cloud based large-scale place recognition. In: 2021 IEEE Winter Conference on Applications of Computer Vision. WACV, pp. 1789–1798. http://dx.doi.org/10.1109/WACV48630.2021.00183.

Kong, X., Yang, X., Zhai, G., Zhao, X., Zeng, X., Wang, M., Liu, Y., Li, W., Wen, F., 2020. Semantic graph based place recognition for 3D point clouds. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 8216–8223. http://dx.doi.org/10.1109/IROS45743.2020.9341060.

Li, L., Kong, X., Zhao, X., Huang, T., Li, W., Wen, F., Zhang, H., Liu, Y., 2021a. SSC: Semantic scan context for large-scale place recognition. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 2092–2099. http://dx.doi.org/10.1109/IROS51168.2021.9635904.

Li, L., Kong, X., Zhao, X., Li, W., Wen, F., Zhang, H., Liu, Y., 2021b. SA-LOAM: Semantic-aided LiDAR SLAM with loop closure. In: 2021 IEEE International Conference on Robotics and Automation. ICRA, pp. 7627–7634. http://dx.doi.org/10.1109/ICRA48506.2021.9560884.

Li, J., Shi, P., Hu, Q., Zhang, Y., 2023a. QGORE: Quadratic-time guaranteed outlier removal for point cloud registration. IEEE Trans. Pattern Anal. Mach. Intell. 1–16. http://dx.doi.org/10.1109/TPAMI.2023.3262780.

Li, J., Shi, P., Hu, Q., Zhang, Y., 2023b. RIFT2: Speeding-up RIFT with a new rotation-invariance technique. arXiv preprint arXiv:2303.00319.

Li, J., Xu, W., Shi, P., Zhang, Y., Hu, Q., 2022a. LNIFT: Locally normalized image for rotation invariant multimodal feature matching. IEEE Trans. Geosci. Remote Sens. 60, 1–14.

Li, X., Zhang, A., Liu, H., Fan, C., Liu, C., Zheng, R., 2022b. GICP-LOAM: Lidar odometry and mapping with voxelized generalized iterative closest point. In: 2022 China Automation Congress. CAC, pp. 2103–2108. http://dx.doi.org/10.1109/CAC57257.2022.10055138.

Lin, J., Zhang, F., 2019. A fast, complete, point cloud based loop closure for lidar odometry and mapping. arXiv preprint arXiv:1909.11811.

Liu, Z., Zhou, S., Suo, C., Yin, P., Chen, W., Wang, H., Li, H., Liu, Y., 2019. LPD-Net: 3D point cloud learning for large-scale place recognition and environment analysis. In: 2019 IEEE/CVF International Conference on Computer Vision. ICCV, pp. 2831–2840. http://dx.doi.org/10.1109/ICCV.2019.00292.

Loo, S.Y., Amiri, A.J., Mashohor, S., Tang, S.H., Zhang, H., 2019. CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction. In: 2019 International Conference on Robotics and Automation. ICRA, IEEE, pp. 5218–5223.

Luo, L., Cao, S.-Y., Sheng, Z., Shen, H.-L., 2022. LiDAR-based global localization using histogram of orientations of principal normals. IEEE Trans. Intell. Veh. 1. http://dx.doi.org/10.1109/TIV.2022.3169153.

Millane, A., Oleynikova, H., Nieto, J., Siegwart, R., Cadena, C., 2019. Free-space features: Global localization in 2D laser SLAM using distance function maps. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 1271–1277. http://dx.doi.org/10.1109/IROS40897.2019.8967683.

Mur-Artal, R., Tardós, J.D., 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Trans. Robot. 33 (5), 1255–1262.

Ratz, S., Dymczyk, M., Siegwart, R., Dubé, R., 2020. OneShot global localization: Instant LiDAR-visual pose estimation. In: 2020 IEEE International Conference on Robotics and Automation. ICRA, pp. 5415–5421. http://dx.doi.org/10.1109/ICRA40945.2020.9197458.

Rizzini, D.L., 2017. Place recognition of 3D landmarks based on geometric relations. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 648–654. http://dx.doi.org/10.1109/IROS.2017.8202220.

Röhling, T., Mack, J., Schulz, D., 2015. A fast histogram-based similarity measure for detecting loop closures in 3-D LIDAR data. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 736–741. http://dx.doi.org/10.1109/IROS.2015.7353454.

Rusu, R.B., Blodow, N., Beetz, M., 2009. Fast point feature histograms (FPFH) for 3D registration. In: 2009 IEEE International Conference on Robotics and Automation. pp. 3212–3217. http://dx.doi.org/10.1109/ROBOT.2009.5152473.

Rusu, R.B., Blodow, N., Marton, Z.C., Beetz, M., 2008. Aligning point cloud views using persistent feature histograms. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3384–3391. http://dx.doi.org/10.1109/IROS.2008.4650967.

Rusu, R.B., Bradski, G., Thibaux, R., Hsu, J., 2010. Fast 3D recognition and pose using the Viewpoint Feature Histogram. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2155–2162. http://dx.doi.org/10.1109/IROS.2010.5651280.

Salti, S., Tombari, F., Di Stefano, L., 2014. SHOT: Unique signatures of histograms for surface and texture description. Comput. Vis. Image Underst. 125, 251–264.

Shan, T., Englot, B., 2018. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 4758–4765.

Shi, P., Li, J., Zhang, Y., 2023a. LiDAR localization at 100 FPS: A map-aided and template descriptor-based global method. Int. J. Appl. Earth Obs. Geoinf. 120, 103336.

Shi, P., Ye, Q., Shaoming, Z., Haifeng, D., 2021. Localization initialization for multi-beam LiDAR considering indoor scene feature. Acta Geod. Cartogr. Sin. 50, 1594–1604. http://dx.doi.org/10.11947/j.AGCS.2021.20210268.

Shi, P., Zhang, Y., Li, J., 2023b. LiDAR-based place recognition for autonomous driving: A survey. arXiv preprint arXiv:2306.10561.

Uy, M.A., Lee, G.H., 2018. Pointnetvlad2018: Deep point cloud based retrieval for large-scale place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4470–4479.

Vidanapathirana, K., Moghadam, P., Harwood, B., Zhao, M., Sridharan, S., Fookes, C., 2021. Locus: LiDAR-based place recognition using spatiotemporal higher-order pooling. In: 2021 IEEE International Conference on Robotics and Automation. ICRA, pp. 5075–5081. http://dx.doi.org/10.1109/ICRA48506.2021.9560915.

Wang, Y., Sun, Z., Xu, C.-Z., Sarma, S.E., Yang, J., Kong, H., 2020a. LiDAR Iris for loop-closure detection. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 5769–5775. http://dx.doi.org/10.1109/IROS45743.2020.9341010.

Wang, H., Wang, C., Xie, L., 2020b. Intensity scan context: Coding intensity and geometry relations for loop closure detection. In: 2020 IEEE International Conference on Robotics and Automation. ICRA, IEEE, pp. 2095–2101.

Wohlkinger, W., Vincze, M., 2011. Ensemble of shape functions for 3D object classification. In: 2011 IEEE International Conference on Robotics and Biomimetics. pp. 2987–2992. http://dx.doi.org/10.1109/ROBIO.2011.6181760.

Xu, D., Liu, J., Liang, Y., Lv, X., Hyyppä, J., 2022. A LiDAR-based single-shot global localization solution using a cross-section shape context descriptor. ISPRS J. Photogramm. Remote Sens. 189, 272–288.

Yang, B., Liu, Y., Dong, Z., Liang, F., Li, B., Peng, X., 2017a. 3D local feature BKD to extract road information from mobile laser scanning point clouds. ISPRS J. Photogramm. Remote Sens. 130, 329–343.

Yang, J., Zhang, Q., Xiao, Y., Cao, Z., 2017b. TOLDI: An effective and robust approach for 3D local shape description. Pattern Recognit. 65, 175–187.

Yin, H., Wang, Y., Ding, X., Tang, L., Huang, S., Xiong, R., 2020. 3D LiDAR-based global localization using siamese neural network. IEEE Trans. Intell. Transp. Syst. 21 (4), 1380–1392. http://dx.doi.org/10.1109/TITS.2019.2905046.

Zhang, W., Xiao, C., 2019. PCAN: 3D attention map learning using contextual information for point cloud based retrieval. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 12428–12437. http://dx.doi.org/10.1109/CVPR.2019.01272.

Zhu, Y., Ma, Y., Chen, L., Liu, C., Ye, M., Li, L., 2020. Gosmatch: Graph-of-semantics matching for detecting loop closures in 3d lidar data. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 5151–5157.