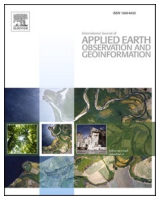
International Journal of Applied Earth Observation and Geoinformation 120 (2023) 103336

Contents lists available at ScienceDirect

International Journal of Applied Earth Observation and Geoinformation

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



LiDAR localization at 100 FPS: A map-aided and template descriptor-based global method

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
Global localization
LiDAR
Template descriptor
KD tree

ABSTRACT

With the development of multi-beam Light Detection and Ranging (LiDAR) sensors, fast and accurate LiDAR-based localization has become a crucial issue in robotics and autonomous driving. However, balancing accuracy and efficiency remains challenging in existing methods. In this paper, we propose a super-fast LiDAR global localization approach that can achieve state-of-the-art (SOTA) accuracy with superior efficiency. Our method leverages template descriptors to capture structural environments and approximates the vehicle's position via map candidate points. Additionally, we create an offline map database to evenly simulate vehicle orientations. We design a loss function to improve localization accuracy. We extensively evaluated the proposed method in public KITTI outdoor sequences and self-collected indoor datasets. The experimental results show that our approach can run at close to 100 frames per second (FPS) on a single-thread CPU, which is much faster than current SOTA methods. Our average absolute translation errors (ATEs) are 0.20m (indoor) and 0.44m (outdoor), and the average localization success rates are 93% (indoor) and 90% (outdoor). The average localization success rates can exceed 97% in large outdoor scenarios with fine-tuned parameters. The source code will be available in https://github.com/ShiPC-AI.

1. Introduction

Localization is a crucial competency for autonomous vehicles to accomplish high-level tasks. Traditional Global Navigation Satellite System (GNSS) (Mendez-Astudillo et al., 2021) suffers from intermittent errors, which limits its localization accuracy in canyons, tunnels, and flyovers. Inertial Navigation System (INS) (Li et al., 2020) inevitably yields drift over a long trajectory. Localization without reliable external infrastructures remains a challenging issue. Simultaneous localization and mapping (SLAM) (Sofonia et al., 2019) has recently gained substantial interest as computer vision and robotics continue to flourish. A SLAM system generally consists of odometry, optimization, loop closure, and mapping sub-systems that jointly provide superior performance.

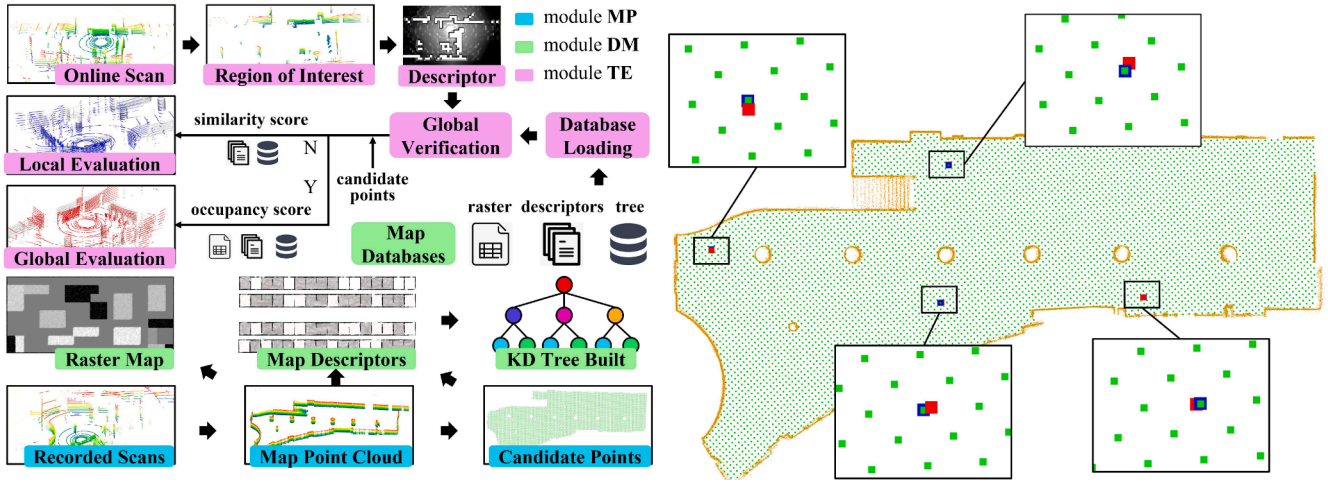
LiDAR is less sensitive to seasonal changes or illumination shifts compared to cameras. Pure LiDAR odometry methods estimate vehicle transformations from consecutive scan associations. However, pose drifts inevitably accumulate in longer trajectories and loop closure detection techniques remain challenging. Fortunately, global localization allows a long-standing high-definition (HD) map to exist. Vehicle transformations within a prebuilt map are directly calculated using scan-to-map associations. Global localization can effectively solve the problem of vehicle re-localization. However, it still has the following issues: (1) Dense point cloud maps require large memory costs. (2) While odometry solutions can provide initial guesses, they are severely affected by sparse LiDAR point densities and fast vehicle movements. (3) Balancing the accuracy and efficiency of localization is still challenging.

Our motivation stems from the need for high-speed autonomous driving when reliable measurements are available. Our research aims to quickly approximate the vehicle's global transformation using a prebuilt map. Fig. 1(b) displays the results of vehicle position estimation in the indoor lobby scenario. Our method produces a large number of candidate points within the prebuilt map. The transformation is refined through efficient template similarity calculations between an online scan and map databases. The major contributions are as follows:

- We propose an efficient template descriptor-based global localization method, which can run at 100 FPS on a single-thread CPU.
- We propose a lightweight template descriptor with a specific loss function to simultaneously guarantee localization efficiency and accuracy.

* 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.jag.2023.103336
Received 19 December 2022; Received in revised form 30 March 2023; Accepted 26 April 2023
Available online 22 May 2023



(a) Overview of the pipeline (b) Vehicle localization in lobby

Fig. 1. Pipeline and localization results in the lobby. In (a): The pipeline has three modules: map processing (MP), database management (DM), and transformation estimation (TE). In (b): Brown denotes preserved map point cloud. Green denotes map candidate points. Red is the real vehicle position. Blue denotes the computed vehicle positions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- We design a map candidate points generation method and use database management to reduce the pressures of map maintenance.
- We design the experiments with both public and self-collected datasets to evaluate our localization performance.

2. Related work

2.1. Map-based localization

We divided map-based localization into three categories based on the map types: OpenStreetMap (OSM)-based, Point Cloud Map (PCM)-based, and Occupancy Grid Map (OGM)-based.

2.1.1. OSM-based methods

OSM (Justiniano et al., 2022) were open-source worldwide maps that covered geographical databases including layers of highways, railways, water systems, and buildings. Ruchti et al. (2015) created a road network from OSM data and classified the environments into road and non-road, then used the classification results to weigh the particles of a Monte Carlo Localization. Suger and Burgard (2017) presented a probabilistic approach to localize a robot in outer-urban environments, which deployed the road and trail information from OSM and combined semantic terrain information with a particle filter framework. Yan et al. (2019) extracted the building and road information from OSM and combined a compact 4-bit semantic descriptor with the particle filter framework to perform global localization. Cho et al. (2022) generated the descriptors by calculating the distance information from the building in OSM and LiDAR point cloud. They determine the vehicle's position by comparing these descriptors. However, these methods worked poorly in long and straight-road environments because they only used vector maps and lacked descriptive 3D features.

2.1.2. OGM-based methods

These methods usually divided the environment into multiple grid cells (Ye et al., 2022) to describe the feature distribution of free or occupied spaces. Guo et al. (2016) introduced model-based feature extraction to classify the laser points and described local characteristics with each extracted point allocated a specified weight. They used a weighted point-based maximum likelihood matching method to match the local map and reference OGM. Millane et al. (2019) designed a signed distance function (SDF) map that defined high-curvature points as keypoints and add SDF values to SIFT descriptors. This method can simultaneously capture the local geometry of both free and occupied space. An and Kim (2022) extracted the statistical signature of geometric and structural features from the 2D Occupancy Grid Maps (OGM) and designed a symmetry score to weigh the submap similarities. To suppress the environmental discretization problem of OGM, Hata et al. (2017) integrated the Monte Carlo Localization (MCL) and Gaussian Process Occupancy Map (GPOM) to enable more accurate localization in urban environments. However, these methods assumed the grid cells were independent, thus ignoring the topology information.

2.1.3. PCM-based methods

These methods usually built a high-definition (HD) point cloud map to provide rich geometric information. Based on the plane-motion assumption, Luo et al. (2022) transformed the localization problem into a bird-eye-view image matching problem and encoded the normal of the point cloud to improve the matching performance. Shi et al. (2021) extracted indoor wall segments from the online LiDAR scan and prior map and then located the robot through wall matching. Yin et al. (2019) proposed a semi-handcrafted feature learning method to transform the place recognition problem into a similarity modeling problem. Then, they achieve global localization by using a particle filter to fuse motion and position information. CT-ICP (Dellenbach et al., 2022) parametrized two poses per scan and elastically distorted the LiDAR scan to compensate for the motion by performing scan-to-map registration. Xu et al. (2022) proposed a Cross-Section Shape Context (CSSC) descriptor to simultaneously encodes the elevation and point density, then designed a two-stage similarity estimation and Nearest Cluster Distance Ratio (NCDR) to improve the place recognition precision. Finally, a Selective Generalized Iterative Closest Point (SGICP) was developed to improve the localization precision. Additionally, some semantic landmarks such as poles (Fan et al., 2021) were exploited in localization applications. However, these methods suffered from costly map construction and maintenance.

2.2. Map-free localization

We mainly divided map-free localization methods into odometry and loop closure detection. Odometry methods (de Miguel-Díez et al., 2022) estimated consecutive vehicle's motion using scan matching. LOAM (Ji and Singh, 2017) first solved pose estimations using key points selection and optimization on the minimization point distance. Recently, deep learning techniques (Lv et al., 2022) were exploited to enhance system

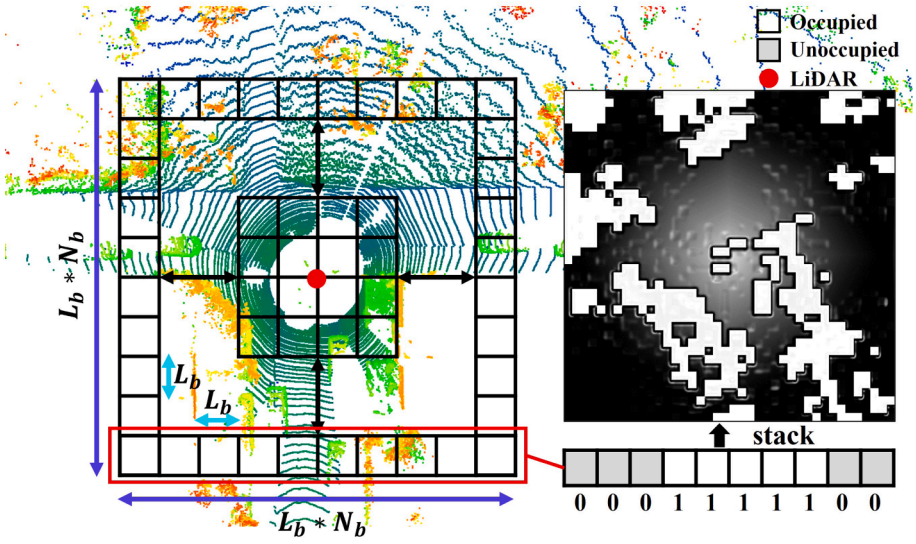


Fig. 2. Space division of the square template descriptor. The LiDAR scan is rendered by height and horizontal spaces are divided into discrete bins.

performance. However, they suffered from sample diversity and model accuracy. Some fusion techniques (Hastaoğlu et al., 2019) exploited Kalman Filter to assist vehicle localization. However, extra calibration work and data association algorithms were required. Loop closure detection (Cattaneo et al., 2022) judged whether the vehicle visited the history positions again. Some global descriptor methods, e.g., M2DP (He et al., 2016) and Scan Context (Kim and Kim, 2018) were proposed to encode the entire point cloud using a single descriptor. FastLCD (Xiang et al., 2021) exploited a comprehensive descriptor and machine learning to achieve reliable and precise results for indoor LiDAR mobile mapping. SegMatch (Dubé et al., 2017) segmented the point cloud into some clusters and employed K-Nearest Neighbors (KNN) retrieval to find revisited places. However, these methods are computationally inefficient.

3. Proposed approach

Fig. 1 is the overview of our global localization pipeline. It has three major modules: map processing module, database management, and transformation estimation.

3.1. Template descriptor

3.1.1. Descriptor parameters

Fig. 2 shows the space division of the square template descriptor (I) on the bird-eye-view. We assume that the vehicle's motion is locally planar, and use two height thresholds (H_{min} and H_{max}) to discard some points (ground and ceiling) from the raw LiDAR scan. The descriptor parameters only consist of N_b and L_b, where N_b is the number of bins and a square descriptor consists of $N_b{*}N_b$ bins, and L_b is the length of each bin. LiDAR-based vehicle localization using square descriptors has been tested in Section 4. Descriptors of other shapes can be easily applied in our method. Here we only introduce vehicle localization using square descriptors.

3.1.2. Bin encoding

When generating the square template descriptor, we divide a 3D scan into completely separated point clouds on the horizontal plane. Let P_{ij} be the set of points in the ith row and jth column square bin. A binary value is assigned to each bin as:

$$\Phi(P_{ij}) = \begin{cases} 1, N(P_{ij}) > 0 \\ 0, N(P_{ij}) = 0 \end{cases} \quad (1)$$

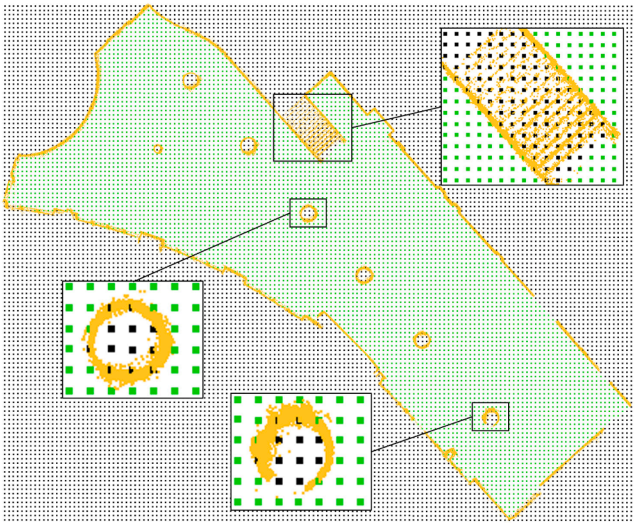


Fig. 3. Generating candidate points in the lobby under the top view. Yellow points mean the map point cloud. The green and black points are equidistant sampling points generated in the horizontal plane according to the maximum and minimum coordinates in the XY direction. Black points are discarded and green points are the candidate points. We use the segmentation module of point cloud software to quickly discard these points by manually drawing closed areas. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where $N(P_{ij})$ is the number of points in P_{ij}. Then we build a binary matrix as the square template descriptor.

3.2. Map processing

Manual or software-assisted offline map construction is very common in autonomous driving applications. We compute the relative transformation of consecutive offline scans and register them into the point cloud map. Section 4.1 describes the calculation details. As shown in Fig. 3, we compute a 3D bounding box of the map and create some sampling points with the equidistant distance $L_{\#}$ in the horizontal plane according to the maximum and minimum coordinates in the XY direction. Then, we discard some points by selecting enclosed areas in the point cloud software. Most discarded points lie outside the scene or

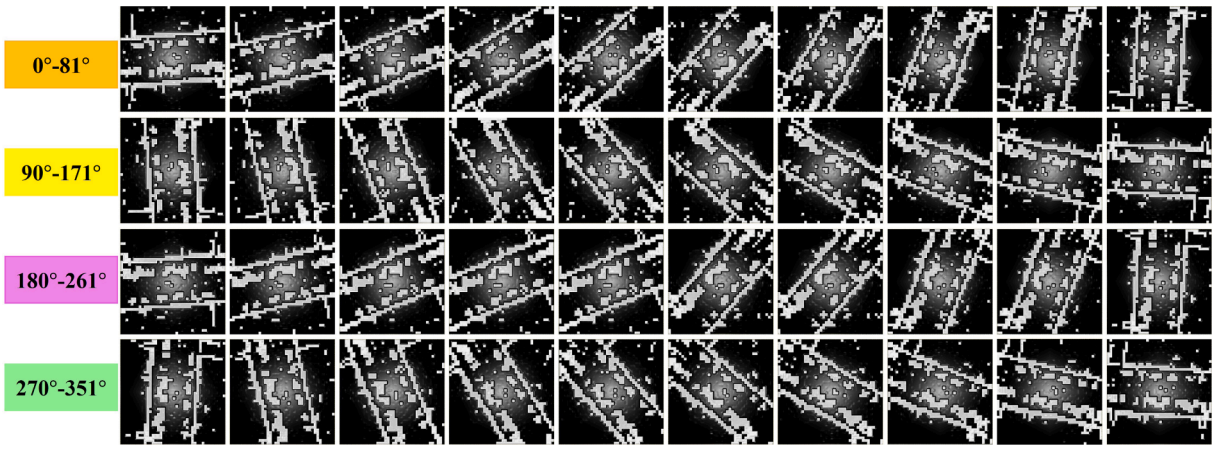


Fig. 4. Map descriptors creation. We rotate the map around each candidate point to create its descriptors. The numbers in the left panel indicate the rotation angles.

inside artificial structures such as pillars, walls, and desks.

3.3. Database management

3.3.1. Raster map

We create a raster map $\mathscr{M}^{\mathscr{R}}$ for the transformation estimation (Section 3.4, global evaluation). After removing the ceiling and ground, the retained map point cloud is projected into a raster map based on the affine transformation:

$$\mathbf{T}_a = \begin{bmatrix} \mathbf{R}_a & \mathbf{t}_a \\ 0 & 0 \end{bmatrix}, \mathbf{R}_a = \begin{bmatrix} \frac{1}{\tau} & 0 \\ 0 & \frac{1}{\tau} \end{bmatrix}, \mathbf{t}_a = \begin{bmatrix} 0 & \frac{x_m}{\tau} \\ 0 & \frac{y_m}{\tau} \end{bmatrix} \tag{2}$$

where $\mathbf{T}_a$ is an affine transformation. x_m and y_m are minimum coordinates of the 2D map and τ is a raster length. Our raster map is binary and the occupied raster index is computed as:

$$\mathscr{M}^{\mathscr{R}}_{u,v} = \begin{cases} u = \mathscr{T}_{\mathscr{R}} \ominus k_{\mathscr{R}} \\ v = \mathscr{T}_{\mathscr{R}} \oslash k_{\mathscr{R}} \end{cases} \tag{3}$$

where u and v mean row and column of the occupied raster, respectively. $\mathscr{T}_{\mathscr{R}}$ is the streaming data and we store it as offline binary files. $k_{\mathscr{R}} = N_{\mathscr{R}}$ is the column capacity of the raster map. $\ominus$ denotes the floor calculation and $\mathscr{T}_{\mathscr{R}} \ominus k_{\mathscr{R}} = floor(\frac{\mathscr{T}_{\mathscr{R}}}{k_{\mathscr{R}}})$. $\oslash$ denotes the mod calculation and $\mathscr{T}_{\mathscr{R}} \oslash k_{\mathscr{R}} = mod(\frac{\mathscr{T}_{\mathscr{R}}}{k_{\mathscr{R}}})$.

3.3.2. Map template descriptors

Vehicle transformation generally consists of rotations and translations. The translation is computed from the positions of the candidate points. In Fig. 4, we create some descriptors under different rotational angles at each candidate point and aggregate them together into the map descriptors:

$$\Gamma^{\mathscr{M}} = \left\{ \Gamma_{i,j}, i \in N_c, j \in N_\beta, N_\beta = \frac{2\pi}{\beta} \right\} \tag{4}$$

where $\Gamma^{\mathscr{M}}$ denotes map template descriptors. $\Gamma^{\mathscr{M}}_{ij}$ denotes the descriptor is created at the ith candidate point under the jth rotational angle. N_c is the number of candidate points. β is the rotational angle resolution and N_β is the number of angles. The online descriptor is theoretically most similar to that created from the adjacent candidate point under the same heading orientation. We use map descriptors to cover the possible vehicle positions and orientations within the map. We also store the map descriptors as offline files. The index of occupied bins in map descriptors is computed by:

$$\Gamma^{\mathscr{M}}_{q,w,i,j} = \begin{cases} q = \mathscr{T}_\Gamma \ominus k_{\mathscr{P}} \\ w = \mathscr{T}_\Gamma \oslash k_{\mathscr{P}} \ominus k_{\mathscr{O}} \\ i = \mathscr{T}_\Gamma \oslash k_{\mathscr{P}} \oslash k_{\mathscr{O}} \ominus k_{\mathscr{I}} \\ j = \mathscr{T}_\Gamma \oslash k_{\mathscr{P}} \oslash k_{\mathscr{O}} \oslash k_{\mathscr{I}} \end{cases}, \begin{cases} k_{\mathscr{P}} = N_\beta k_{\mathscr{O}} \\ k_{\mathscr{O}} = N_b k_{\mathscr{I}} \\ k_{\mathscr{I}} = N_b \end{cases} \tag{5}$$

where $\mathscr{T}_\Gamma$ is the streaming data and we also store it as offline binary files. j, i, w, and q are the indexes of the col, row, angle, and candidate point, respectively.

3.3.3. KD tree building

We decompose each template into a $N_b^2 \times 1$ vector h. We rotate different angles along the z-axis to create descriptors at each candidate point and the map descriptors are reshaped into a huge feature matrix $\mathscr{H}^{\mathscr{M}} = [h_1, h_2, \cdots, h_m]$ with N_b^2 rows and N_c*N_β columns. The consistency checking of the template descriptor is replaced by vector similarity computation. Note that the structure components (occupied bins) deserve more attention for place description and vehicle localization. However, the unoccupied bins cover the majority of the descriptor as we use large bins. This unbalanced proportion leads to traditional vector similarity metrics, e.g., Euclidean, Manhattan, Hellinger, and Hamming Distance seldom work. For this, we present a loss function to refine the accuracy of our descriptor vector similarity computation:

$$loss(h^{src}, h^{dst}) = \sum_i \| 1 - h_i^{src} h_i^{dst} \|_2 \tag{6}$$

where h^{src} and h^{dst} are two template vectors. i is the element index of the vector. Each element of the vector is binary. In tree building, both h^{src} and h^{dst} are map descriptor vectors. In subsequent local evaluation, h^{src} and h^{dst} are online descriptor vectors and map descriptor vectors, respectively. A KD tree with the presented loss function is created in $\mathscr{H}^{\mathscr{M}}$ and the tree structure is stored as offline files as well.

We predict the vehicle's positions using map candidate points and rotate the map descriptors to simulate heading, which will generate a huge offline map database. We use the encoding method in Eq. (3) and Eq. (5) to store only the positions of occupied elements in the raster map and map descriptors in binary format, which can effectively reduce the storage space. In terms of real-time running memory, each element of the descriptor only occupies 1 bit, which is very lightweight. The DM module manages three major components including a raster map $\mathscr{M}^{\mathscr{R}}$, a map descriptor matrix $\mathscr{H}^{\mathscr{M}}$, and a KD tree *Tree*. The map databases are merely loaded once at the initialization stage of the localization system.

3.4. Transformation estimation

This section solves vehicle transformations with the online LiDAR scan and offline map databases. Our transformation estimation can be

classified into local and global evaluations based on whether a global verification is used. Local evaluation dedicates to efficiently locating vehicles only using the descriptor similarity computation from the KD tree. The global evaluation further verifies the solved transformation via map consistency. Note that the above two localization solutions can switch alternatively at any time or cooperate at different frequencies.

3.4.1. Scan processing

We first remove ground and ceiling using heights once a new online point cloud $\mathscr{S}$ comes. Then we downsample the remaining point cloud. A random sampling procedure is preferable because the primitive data memory is compressed without corruption.

Algorithm 1 Local Evaluation

Input: KD tree *Tree*, map descriptor matrix $\mathscr{H}^{\mathscr{M}}$, map candidate points $\mathscr{P}_c$, angle resolution β, neighbor capacity N_k, online descriptor vector $h^{\mathscr{S}}$
Output: Transformation $\mathbf{T}$
Initialize: Nearest map descriptors $\mathscr{C}=\{\varnothing\}$, $\mathbf{T}=\mathbf{I}_{4*4}$.
1: $\mathscr{C}\leftarrow$ KNN search (*Tree*, $\mathscr{H}^{\mathscr{M}}$, N_k, $h^{\mathscr{S}}$)
2: Update $\mathbf{T}_0$ in Eq. (7) for $\mathscr{C}_0$
3: Update $score_1$ for $\mathbf{T}_0$ in Eq. (8)
4: **If** $score_1 > \kappa_1$ **then**
5: Return $\mathbf{T}_0$
6: **end if**

3.4.2. Local evaluation

Our local evaluation is described in Algorithm 1. The input includes a KD tree, a map descriptor matrix, map candidate points, an online descriptor vector, and two preset parameters (neighbor capacities of the nearest descriptors and angle resolution). We use the KNN search in map descriptors to find the nearest descriptor set $\mathscr{C}$. Then the transformation of the kth candidate descriptor is computed as:

$$\mathbf{T}_k = \begin{bmatrix} \cos\beta_i & \sin\beta_i & 0 & x_j \\ -\sin\beta_i & \cos\beta_i & 0 & y_j \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where k is the index of the nearest map descriptor. $j = \mathscr{C}_k \ominus N_\beta$ is the index of the map candidate point. $i = \mathscr{C}_k \oslash N_\beta$ is the index of the rotational angle. The computed transformation is then evaluated by:

$$score_1(k) = \frac{n\left(h^{\mathscr{S}}\right) - loss\left(h^{\mathscr{S}}, \mathscr{H}^{\mathscr{M}}_k\right)}{n\left(h^{\mathscr{S}}(1)\right)} \tag{8}$$

where $score_1$ is the similarity score between the online and map descriptor vector. $h^{\mathscr{S}}$ is an online descriptor vector and $\mathscr{H}^{\mathscr{M}}_k$ is a map descriptor vector. $n\left(h^{\mathscr{S}}\right) = N_b{*}N_b$ is the number of descriptor bins. $n(h^{\mathscr{S}}(1))$ is the number of occupied bins in the online descriptor. If a certain similarity score exceeds the threshold κ_1, the local evaluation terminates immediately and outputs the transformation. Since the number of bins and occupied bins are constant for each online descriptor, the transformation derived from the nearest map descriptor always achieves the best similarity score. Consequently, we always take $\mathscr{H}^{\mathscr{M}}_{\mathscr{C}_0}$ as the only candidate descriptor in Algorithm 1. $\mathbf{T}_0$ will be taken as the final transformation if the similarity score of $\mathbf{T}_0$ exceeds κ_1.

3.4.3. Global evaluation

The global evaluation is described in Algorithm 2 to dispose of the low-velocity scenarios and guarantee a success rate of localization in low-frequency situations. The global evaluation leverages the occupancy ratio of the raster map to verify the computed transformation as:

$$score_2(k) = \sum_i \frac{\mathscr{M}^{\mathscr{R}}_{uv}\left[\mathbf{T}_a \mathbf{T}_k \mathscr{S}'_i\right]}{n(\mathscr{S}')} \tag{9}$$

where $score_2$ is the occupancy score between the raster map and the online LiDAR scan. $\mathscr{S}'$ denotes the retained online scan. $n()$ denotes the number of points. $\mathscr{S}'_i$ is the ith point. $\mathscr{M}^{\mathscr{R}}_{uv}[p]$ means p hits the (u, v) raster in the raster map.

Table 1
Details of experimental datasets.

Dataset(/)	Type(/)	Area(m^2)	Frames(/)	$L_{\mathscr{M}}$(m)	N_c(/)
Lobby	1F	56 × 28	548	0.4	5993
Corridor	4F	55 × 15	4529	0.3	4375
Hybrid	1F	60 × 21	1103	0.4	4612
Garage	B1	59 × 40	10,416	0.5	5070
Sequence 00	outdoor	564 × 496	4541	1.0	23,272
Sequence 03	outdoor	471 × 199	801	1.0	6329
Sequence 06	outdoor	23 × 457	1101	1.0	6347
Sequence 07	outdoor	191 × 209	1101	1.0	6897
Sequence 09	outdoor	465 × 568	1591	1.0	19,665
Sequence 10	outdoor	671 × 177	1201	1.0	6833

Algorithm 2 Global Evaluation

Input: Raster map $\mathscr{M}^{\mathscr{R}}$, KD tree *Tree*, map descriptor matrix $\mathscr{H}^{\mathscr{M}}$, map candidate points $\mathscr{P}_c$, angle resolution β, neighbor capacity N_k, online descriptor vector $h^{\mathscr{S}}$
Output: Transformation $\mathbf{T}$
Initialize: Nearest map descriptors $\mathscr{C}=\{\varnothing\}$, $\mathbf{T}=\mathbf{I}_{4*4}$.
1: $\mathscr{C}\leftarrow$ KNN search (*Tree*, $\mathscr{H}^{\mathscr{M}}$, N_k, $h^{\mathscr{S}}$)
2: **for** idx k in $\mathscr{C}$ **do**
3: Update $\mathbf{T}_k$ in Eq. (7)
4: Update $score_1$ in Eq. (8)
5: Update $score_2$ in Eq. (9)
6: **end for**
7: $\mathscr{C}'\leftarrow$ sort $\mathscr{C}$ via $score_2$ in descending order
8: **for** idx k' in $\mathscr{C}'$ **do**
9: **if** $score_1 > \kappa_1$, $score_2 > \kappa_2$ **then**
10: Return $\mathbf{T}_{k'}$
11: **end if**
12: **end for**

Here the occupancy score outperforms the similarity score in evaluating localization results. The transformation with a higher occupancy score is first evaluated. If the similarity score exceeds κ_1 and the occupancy score exceeds κ_2, the procedure terminates immediately and outputs the transformation. The global evaluation aims to compute a transformation that optimally matches the online point cloud with the map.

4. Experiments

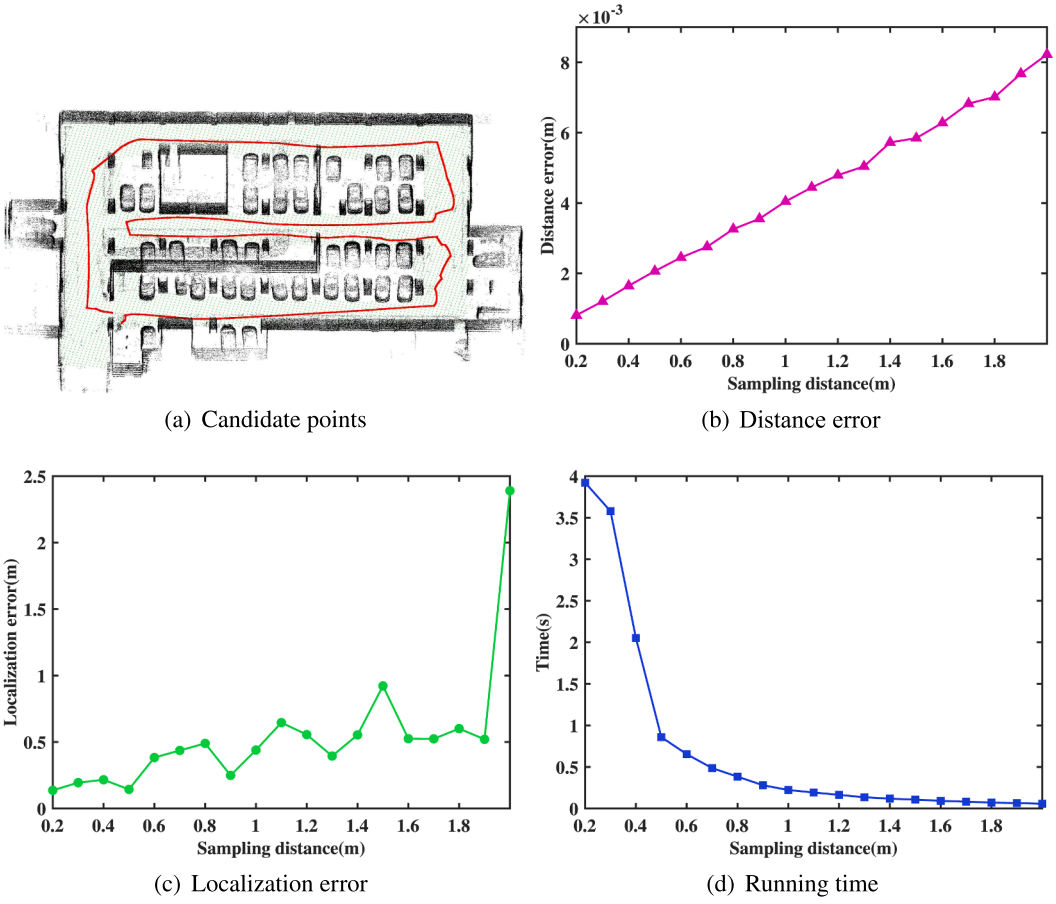
4.1. Datasets and implementation

Table 1 summarized the dataset details. We collected four indoor datasets using a TurtleBot equipped with a RoboSense RS-LiDAR-16. We computed the relative transformation using LOAM whose lateral error was less than 0.55% and whose angular error was about 0.0016 deg/m. We selected a new keyframe every time the robot walked 1 m to build the point cloud map. KITTI odometry provided the LiDAR scans acquired from a Velodyne HDL64 and a GPS/IMU localization unit with RTK correction signals was used to compute the ground truth poses. We selected a keyframe every 10 m to stitch the map. The experimental datasets include different vehicle speeds, scenario types, data densities, and map sampling lengths. Our method was written in C++ language in the Ubuntu 20.04LTS system. Experiments were performed on an Intel core i9-10850K CPU with a single thread.

4.2. Parameters analysis

4.2.1. Candidate point parameters

As shown in Fig. 5, we conduct the parameter experiments for the sampling distance $L_{\mathscr{M}}$ of candidate points in the parking garage. The localization error is calculated in Eq. (13) and the distance error d_c is computed as:



(a) Candidate points (b) Distance error

(c) Localization error (d) Running time

Fig. 5. Parameter experiments of candidate points in the parking garage. In (a): the red line is the vehicle trajectory. Green and black denote the candidate points and retained map, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Localization comparisons under different descriptor parameters in corridor. The bold indicates the best while the italic is the second best.

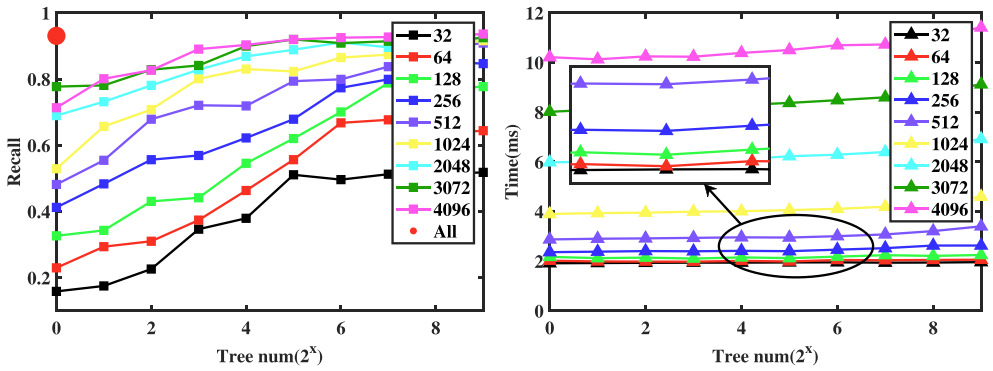
L_b	β															
	$N_b = 10$				$N_b = 20$				$N_b = 30$				$N_b = 40$			
	3	4	5	6	3	4	5	6	3	4	5	6	3	4	5	6
0.42	0.17	0.15	0.16	0.16	0.65	0.60	0.62	0.62	0.79	0.76	0.78	0.77	0.85	0.81	0.84	0.83
0.5	0.29	0.29	0.28	0.28	0.67	0.66	0.65	0.65	0.78	0.77	0.77	0.75	0.86	0.85	0.88	0.87
0.67	0.37	0.37	0.36	0.36	0.72	0.69	0.71	0.70	0.84	0.83	0.83	*0.83*	**0.95**	**0.94**	**0.94**	**0.91**
0.95	*0.39*	**0.41**	*0.38*	*0.39*	0.74	*0.72*	0.71	0.71	*0.89*	**0.86**	*0.86*	0.75	*0.93*	*0.89*	*0.92*	*0.89*
1.0	**0.41**	*0.38*	**0.43**	**0.42**	**0.77**	**0.72**	**0.75**	**0.74**	**0.90**	*0.85*	**0.91**	**0.88**	0.91	0.88	0.90	0.87
1.24	0.34	0.34	0.33	0.34	*0.75*	0.70	*0.73*	*0.74*	0.81	0.78	0.80	0.78	0.85	0.79	0.81	0.79
1.5	0.32	0.31	0.32	0.32	0.67	0.62	0.65	0.64	0.71	0.65	0.68	0.64	0.74	0.71	0.70	0.66
2.0	0.12	0.09	0.11	0.11	0.35	0.33	0.34	0.32	0.45	0.41	0.41	0.38	0.49	0.45	0.44	0.41

$$d_c = \sum_i \| \frac{p_i - q_{i,p}}{n_{gt}} \|_2 \tag{10}$$

where i is the scan index. p is the vehicle position and q is the candidate point closest to the vehicle position. n_{gt} is the number of scans. The distance error quantifies the average minimum distance between the candidate points and the vehicle's position, and the localization error represents the deviation between the vehicle's position and the

Table 3
Map descriptor comparisons under different descriptor parameters on KITTI 06.

Bin Number (/)	10 × 10	20 × 20	30 × 30	40 × 40	50 × 50	60 × 60	70 × 70	80 × 70	90 × 90
Success Rate (/)	0.04	0.53	0.97	1	1	1	1	1	1
Memory (GB)	0.05	0.22	0.51	0.93	1.5	2.3	3.0	3.8	4.7
Load Time (s)	0.74	5.18	7.50	10.93	14.24	25.60	37.59	46.68	55.85
Bin Number (/)	100 × 100	110 × 110	120 × 120	130 × 130	140 × 140	150 × 150	160 × 160	170 × 170	180 × 180
Success Rate (/)	1	1	1	1	1	1	1	1	1
Memory (GB)	5.5	6.3	7.0	7.8	8.5	9.2	9.8	10.5	11.1
Load Time (s)	63.84	71.02	76.39	82.01	89.00	94.08	98.91	102.83	108.93



(a) Recall (b) Time

Fig. 6. Localization comparisons with different tree parameters in the lobby. (a) Red points mean we use a single tree and search all nodes. (b) The running time (1.32 s) of the localization using a single tree and all nodes is far larger than other setups, it is not drawn in the figure. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

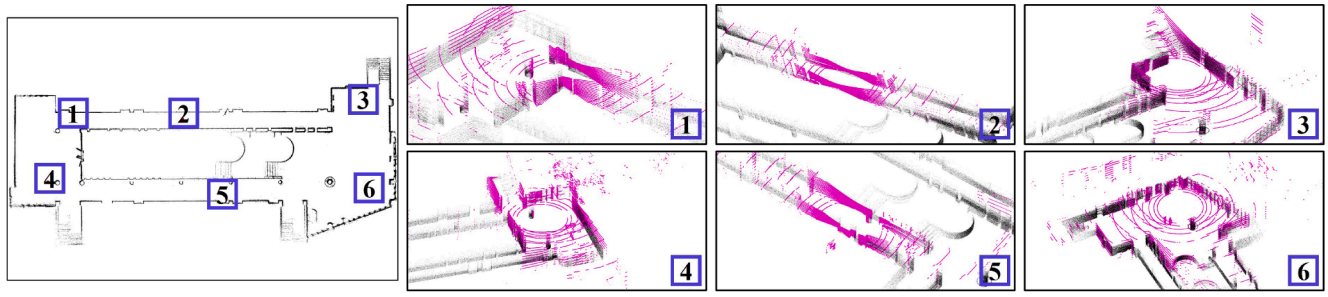
estimated one. We set N_b, β, L_b, and n_{tree} to (40, 6, 1, 1, n_{all}) in the experiment, where n_{all} denote we compare all nodes in the KNN search. The number of candidate points decreases with the increase of sampling distance, which leads to the increase of distance error and localization error (0.1 m-2.4 m) but improves the running efficiency. Based on the localization error and efficiency, we set $L_{\mathscr{M}}$ to 0.5 m in the parking garage and the corresponding average distance error is 0.002 m. Our method is independent of the number of LiDAR scans. Even if the LiDAR sequence recorded offline is short, it can still yield rich candidate points.

4.2.2. Template descriptor parameters

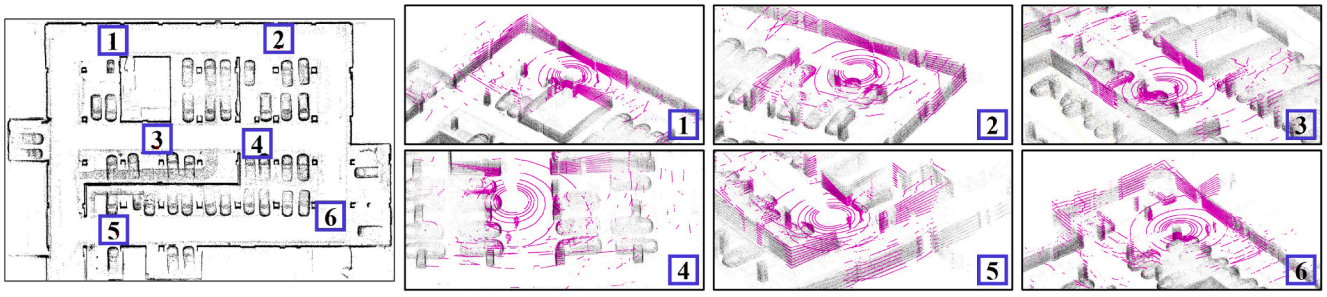
Table 2 summarizes the localization success rate under various template parameter setups in the corridor. The success rate is computed in Eq. (12). The bin length L_b includes eight values from 0.42 m to 2.0 m. The bin number N_b is set to 10, 20, 30, and 40, respectively. The angle resolution is set to 3°, 4°, 5°, and 6°, respectively. We find that the localization success rate can reach more than 90% when the number of bins reaches 40 * 40 and the size is about 1 m.

4.2.3. Map descriptor comparisons

As summarized in Table 3, we compare the localization success rate, map descriptor memory, and loading time under different numbers of bins in KITTI 06. For a fair comparison, all bin sizes are 1 m in the experiment. The localization success rate increases monotonically with the number of bins, but loading descriptors takes more time. If we use a 180 * 180 descriptor, the memory of map descriptors exceeds 11 GB, and the map loading time takes more than 100 s in our experimental platform. To balance the map descriptor loading time and success rate, we create 40 × 40 bins in each square descriptor in later experiments. This bin number may cause the success rate to degrade in some scenarios. In low-speed, closed, and symmetric environments, we suggest using high-resolution descriptors to improve the success rate and localization accuracy. In high-speed, open, structured environments, we suggest using moderated-resolution descriptors to achieve fast localization.



(a) Hybrid scenario



(b) Parking garage

Fig. 7. Global localization in hybrid scenario and garage. The pink is the 16-beam LiDAR point cloud. The black is the preserved map. The black numbers represent vehicle's positions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 4
Localization comparisons of indoor datasets. The bold indicates the best while the italic is the second best.

	Dataset	Submap-to-Map			Scan-to-Map				
		FPFH	K4PCS	TrICP	PWL	SWL	VWL	HDL	**Ours**
PR(/)	Lobby	0.59	0.11	0.04	0.73	**0.96**	0.92	0.16	*0.92*
	Corridor	0.28	0.08	0.02	0.2	*0.75*	0.59	0.14	**0.92**
	Hybrid	0.51	0.16	0.05	0.62	0.79	0.72	*0.80*	**0.94**
	Garage	0.16	0.66	*0.87*	0.35	0.66	0.62	0.83	**0.93**
	Mean	0.31	0.25	0.25	0.48	*0.79*	0.71	0.48	**0.93**
SR(/)	Lobby	*0.59*	0.11	0.04	0.11	0.29	0.42	0.16	**0.92**
	Corridor	0.28	0.08	0.02	0.12	*0.46*	0.42	0.14	**0.92**
	Hybrid	0.51	0.16	0.05	0.11	0.37	0.52	*0.80*	**0.94**
	Garage	0.16	0.66	*0.87*	0.13	0.4	0.55	0.83	**0.93**
	Mean	0.31	0.25	0.25	0.12	0.38	0.47	*0.48*	**0.93**
Time(s)	Lobby	74.294	33.556	2.193	0.224	*0.072*	2.700	0.078	**0.006**
	Corridor	59.978	28.499	1.212	0.676	0.232	1.063	*0.066*	**0.007**
	Hybrid	88.788	51.102	4.881	0.633	0.222	1.371	*0.082*	**0.008**
	Garage	80.563	85.445	1.617	0.707	*0.040*	1.196	0.046	**0.007**
	Mean	75.906	49.651	2.476	0.560	0.142	1.583	*0.068*	**0.007**
d_l(m)	Lobby	9.27	9.06	9.54	**0.11**	*0.14*	0.15	1.81	0.21
	Corridor	15.57	15.25	7.81	0.14	*0.11*	**0.06**	2.38	0.19
	Hybrid	18.35	9.31	7.65	0.11	**0.07**	*0.11*	3.27	0.20
	Garage	15.70	8.02	1.96	0.22	**0.10**	*0.11*	5.34	0.19
	Mean	14.72	10.41	6.74	0.15	**0.11**	*0.11*	3.20	0.20

4.2.4. KD tree parameters

Fig. 6 displays the localization comparisons under various KD tree parameters. The numerical growth of trees and nodes both dramatically promotes the success rate. The running time is independent of the number of trees but increases with the number of nodes. The fastest parameter setting takes less than 2 ms, the slowest is about 10 ms and recall achieves 0.92. This efficiency outperforms most SOTA methods. Because the number of map candidates is limited, we balance the localization accuracy and efficiency by adjusting the two KD tree parameters. If there are few candidate points and the sampling distance is large, we use more search nodes and trees to improve the accuracy. If there are many candidate points and their sampling distance is very small, we will use relatively few search nodes to improve efficiency. Consequently, we use 512 trees and 1024 nodes in the following experiments. Note that only local evaluation is executed here.

4.3. Indoor evaluations

4.3.1. Map-based comparisons

We compare with some map-based localization methods and divide them into two categories. (1) Scan-to-map: Shi et al. (2021) uses parallel, vertical, and symmetric walls as landmarks, denoted as PWL, VWL, and SWL, respectively. HDL (Koide, 2019) incorporates NDT-based registration into Unscented Kalman Filter (UKF) framework to locate the vehicle. (2) Submap-to-Map: we create a submap every 20 LiDAR scans and register them to the map using FPFH (Rusu et al., 2009), K4PCS (Theiler et al., 2014), and TrICP (Chetverikov et al., 2002), respectively. In our method, N_b, β, L_b, n_{tree}, and n_{node} are set to (40, 6, 1, 512, 1024) and an error of less than 0.5 m is successful. Fig. 7 depicts our localization results. Table 4 summarizes the running time, precision (*PR*), success rate (*SR*) and translation error (d_l) and they are computed as:

$$PR = \frac{TP}{TP + FP} \tag{11}$$

$$SR = \frac{TP}{TP + TN + FP + FN} \tag{12}$$

$$d_l = \|\boldsymbol{t}_{es} - \boldsymbol{t}_{gt}\|_2 \tag{13}$$

where *TP*, *TN*, *FP*, and *FN* represent the number of true positive, true negative, false positive, and false negative localization results, respectively. $\boldsymbol{t}_{gt}$ is the ground truth translation vector and $\boldsymbol{t}_{es}$ is the estimated one.

Table 5
Registration comparisons of indoor datasets. The bold indicates the best while the italic is the second best.

	Dataset	ICP	GICP	NDT	FGR	Ours-fst	**Ours-fst** + ICP
Dist (m)	Lobby	5.97	3.69	0.64	0.61	*0.29*	**0.25**
	Corridor	0.34	**0.05**	0.90	1.87	*0.27*	0.30
	Hybrid	1.82	0.83	*0.78*	1.25	**0.78**	1.81
	Garage	0.06	**0.02**	*0.02*	0.05	0.04	0.06
	Mean	2.05	1.15	*0.59*	0.95	**0.35**	0.61
Rot (°)	Lobby	101.82	98.15	100.20	79.75	*1.47*	**0.94**
	Corridor	4.05	**0.94**	4.04	9.11	*1.44*	2.19
	Hybrid	47.21	48.69	57.50	98.26	*3.31*	**2.34**
	Garage	0.52	0.13	0.11	0.29	*0.09*	**0.03**
	Mean	38.91	36.98	40.46	46.86	*1.58*	**1.38**

The largest parking garage and the smallest corridor cover around 2,400 m^2 and 800 m^2, respectively. The area variations result in different sampling lengths and candidate point capacities. Table 4 shows that our method achieved an average error of 0.2 m which is inferior to the baseline method (PWL, VWL, and SWL). Four datasets contain a few long and salient walls which can provide strong geometric constraints for them, however, they can only be applied to indoor scenarios. Although our localization accuracy is inferior to theirs, it can meet the requirements of autonomous driving. Our localization error is related to the sampling distance of the map candidate points. We set the indoor sampling distance between 0.3 m and 0.5 m, so the error is relatively large. The localization accuracy can be improved by increasing the sampling density. We gain stable performance in the other three metrics. A relatively low κ_1 (0.4) leads to the same precision and recall in our method. It significantly outperforms the baseline methods with average precision and recall of 0.93, and a running time of 7 ms. Fig. 7 shows that our method can locate the vehicle successfully in challenging closed corridors, corners, stairs, and multi-vehicles scenarios.

4.3.2. Registration-based comparisons

We further evaluate the scan-to-scan registration performance and compare the proposed method with some traditional point cloud registration methods, namely, ICP (Besl and McKay, 1992), GICP (Segal et al., 2009), NDT (Magnusson et al., 2007), and FGR (Zhou et al., 2016). In Table 5, we randomly select 200 scan pairs with a distance of 2 m and compute their registration errors. Fig. 8 depicts the pairwise registration

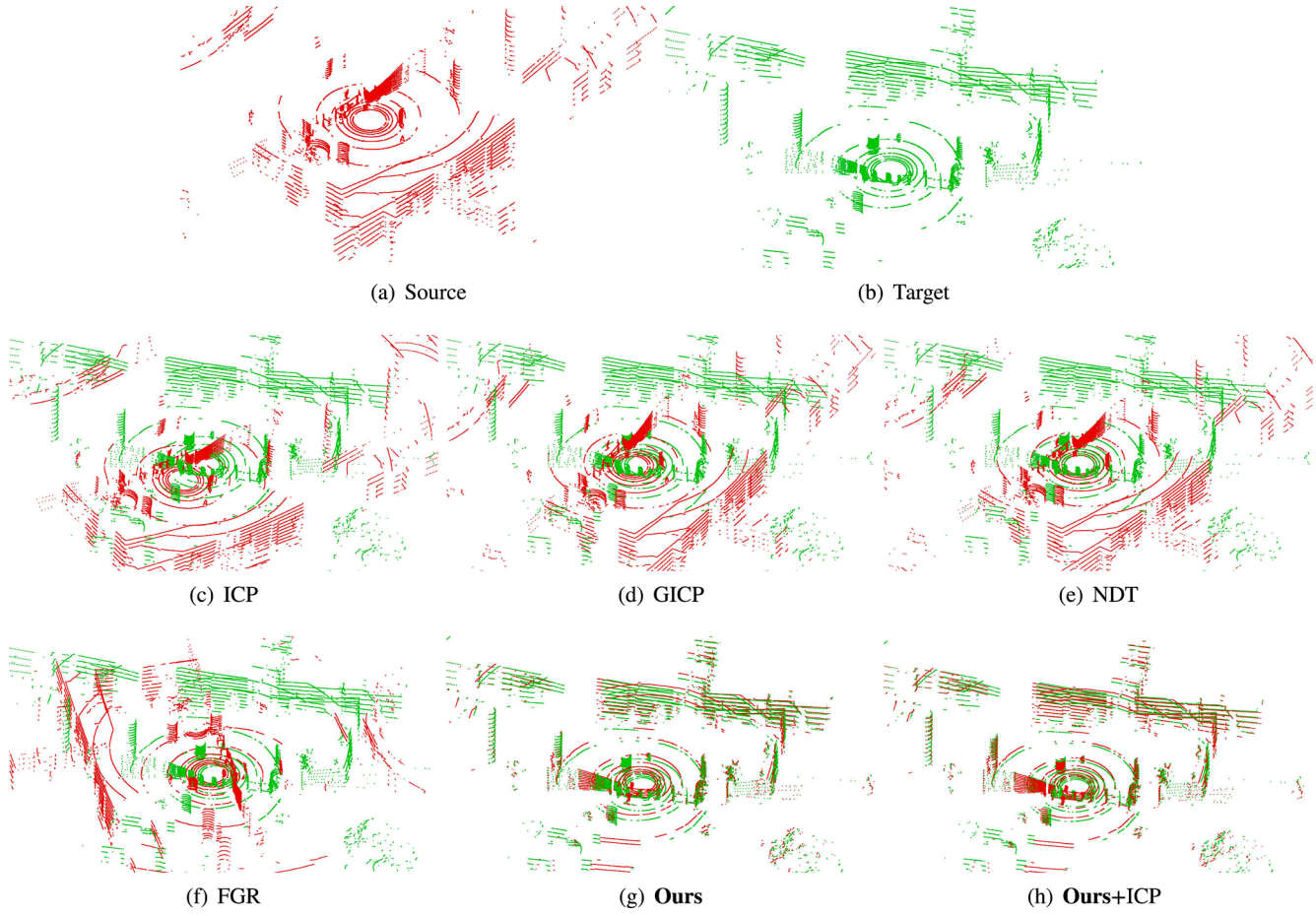


Fig. 8. Pairwise scan registrations in the lobby. ICP, GICP, NDT, and FGR fail to register the pair of LiDAR scans because of the large viewpoint change. Our method and ICP with our method as an initial guess successfully register the pair of LiDAR scans.

Table 6
Map-based localization comparisons on KITTI datasets (%). The bold indicates the best while the italic is the second best.

		00	06	07	09	10	mean
OSM-2m	Top 1	45.72	59.67	37.06	19.80	11.49	43.89
	Top 5	54.22	65.40	51.04	24.07	12.66	41.48
	Top 10	57.74	68.12	54.22	25.83	13.66	43.91
OSM-5m	Top 1	48.34	62.13	37.33	22.25	11.66	36.34
	Top 5	56.86	67.03	50.77	26.21	12.99	42.78
	Top 10	61.15	70.75	56.49	29.23	13.57	46.24
OSM-10m	Top 1	44.13	57.77	36.60	21.68	11.07	34.25
	Top 5	52.87	61.58	50.40	26.15	12.32	40.67
	Top 10	56.20	64.49	54.59	28.41	12.99	43.34
Ours-*fst*	Top 1	*68.20*	*99.00*	*98.09*	*80.14*	*96.75*	*88.44*
	Top 5	*78.24*	*99.36*	*99.46*	*86.49*	*97.92*	*92.29*
	Top 10	*82.40*	*99.91*	*99.91*	*88.75*	*98.50*	*93.89*
Ours-*opt*	Top 1	**79.51**	**100.00**	**99.91**	**86.36**	**99.17**	**92.99**
	Top 5	**82.60**	**100.00**	**100.00**	**92.33**	**99.67**	**94.92**
	Top 10	**83.60**	**100.00**	**100.00**	**93.66**	**99.83**	**95.42**

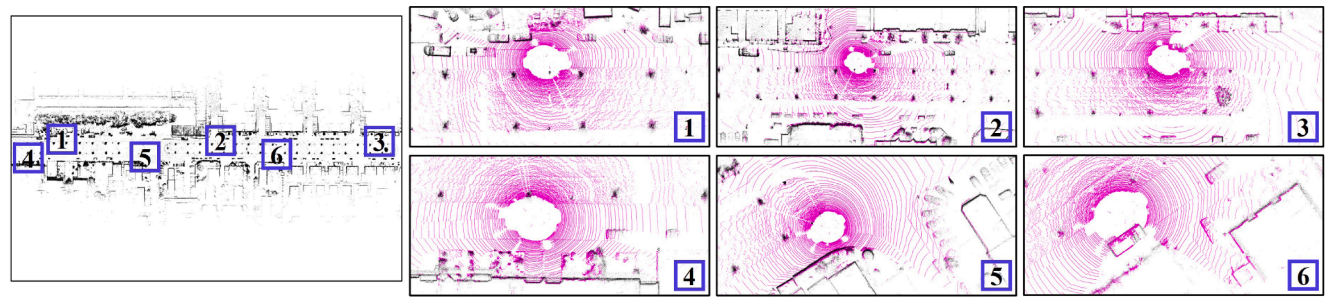
results. The rotation error is computed as Eq. (14). We also pass our results as initial guesses to ICP to solve the registration parameters. Our method achieves a mean distance error of 0.35 m and a rotation error of 1.58°, which significantly outperform other methods. It achieves stable performance in all four scenarios, while the rotation accuracy of ICP, GICP, NDT, and FGR severely degrade in the lobby and hybrid scenario. It shows that our method can solve the scan-to-scan registration problem with a low overlap rate.

Table 7
Comparisons of localization error on KITTI datasets. The bold indicates the best while the italic is the second best.

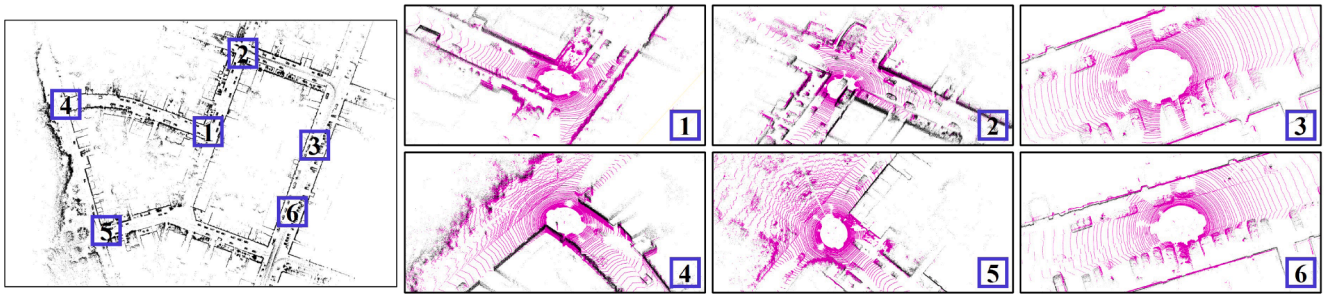
	(Yan et al., 2017)	(Gao et al., 2018)	(Zhan et al., 2020)	(Loo et al., 2019)	(Zhao et al., 2021)	(Engel et al., 2018)	Ours
03	–	2.85	2.04	3.46	–	*1.22*	**0.46 ± 0.22**
06	*0.83*	13.55	2.53	11.51	–	48.64	**0.43 ± 0.18**
07	*1.0*	2.96	1.72	6.51	7.92	15.65	**0.44 ± 0.18**
10	*1.89*	17.36	3.72	4.84	16.46	7.43	**0.44 ± 0.20**

4.4. Outdoor evaluations

Since outdoor scenarios produce larger maps, we use 1 m as the maximum localization error and set $L_{/\!/}$ to 1 m. We present two parameter setups named *fst* and *opt* in the local evaluation to evaluate our localization performance. N_b, β, L_b, n_{tree}, and n_{node} are set to (40, 6, 1, 512, 1024) and (40, 3, 1, 1, n_{all}) in *fst* and *opt*, respectively.

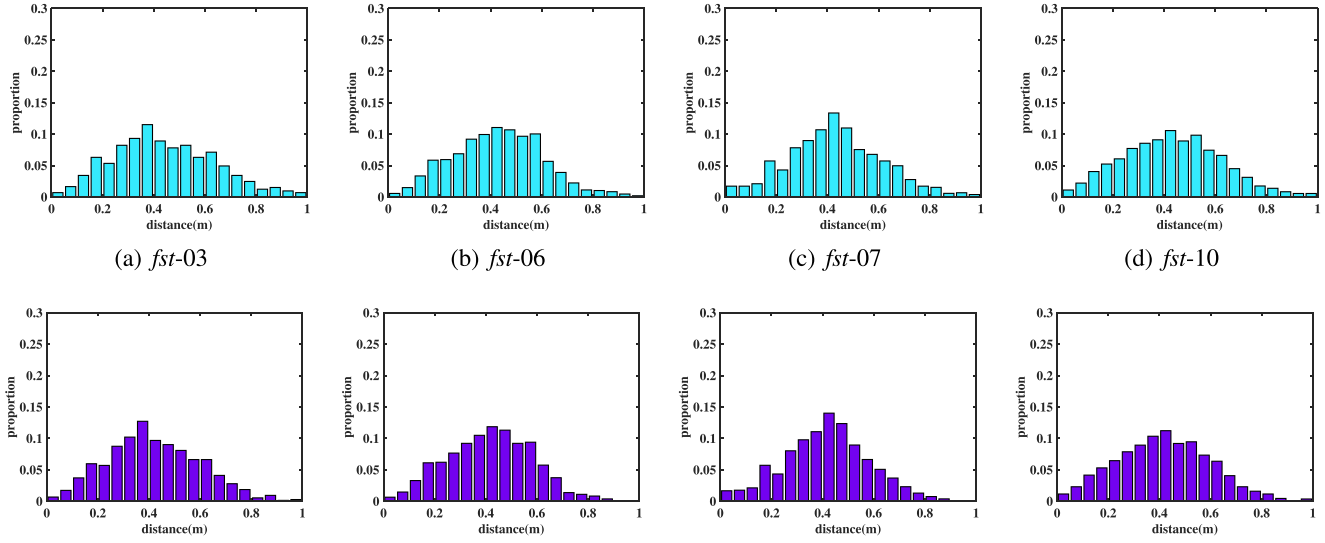


(a) Sequence 06



(b) Sequence 07

Fig. 9. Global localization in KITTI 06 and 07. The pink is the 64-beam LiDAR point cloud. The black is the preserved map. The black numbers represent vehicle's positions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) *fst*-03 (b) *fst*-06 (c) *fst*-07 (d) *fst*-10

(e) *opt*-03 (f) *opt*-06 (g) *opt*-07 (h) *opt*-10

Fig. 10. Lateral error histograms of *fst* and *opt* in KITTI datasets.

4.4.1. Map-based comparisons

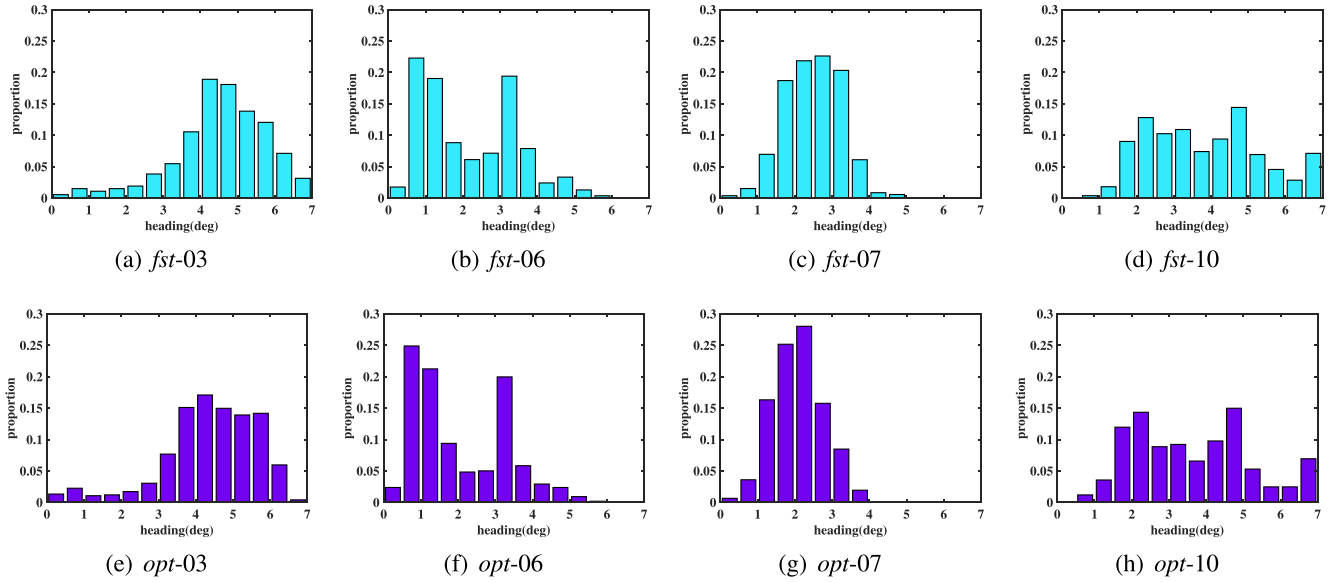
Table 6 shows the comparison of our method with a SOTA map-based localization method (Cho et al., 2022). It uses an OSM as a prior map and a single LiDAR scan as input. We compare the performance of 1, 5, and 10 top candidates, respectively. The baseline method has three bin resolutions (2 m, 5 m, and 10 m). Our method achieve better performance on all five sequences at all candidate capacities (1, 5, and 10). The success rate can reach 100% in sequence 06 and 07.

4.4.2. Localization errors

Table 7 summarizes the ATE error comparisons of our method (*opt*) to some SOTA methods. We achieve an average ATE of 0.44 m that outperforms the other methods. As the trajectory becomes longer, the localization errors of other methods dramatically increase while our method does not degrade. As shown in Fig. 9, the online LiDAR scans are consistently registered into the prior map. Fig. 10 depicts the quantitative lateral errors of *fst* and *opt*. The lateral errors of the two solutions are mostly between 0 and 0.6 m, and errors less than 0.4 m account for a larger proportion in *opt*.

4.4.3. Heading orientation errors

Besides ATE, heading errors are also crucial in loop closure detection and vehicle localization. The heading error α_h is calculated as:



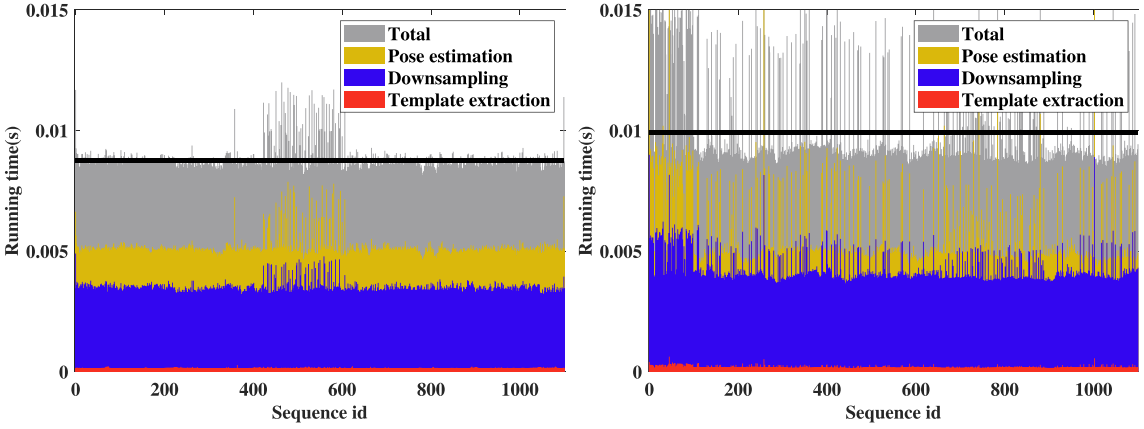
(a) *fst*-03 (b) *fst*-06 (c) *fst*-07 (d) *fst*-10

(e) *opt*-03 (f) *opt*-06 (g) *opt*-07 (h) *opt*-10

Fig. 11. Heading error histograms of *fst* and *opt* in KITTI datasets.

Table 8
Comparisons of localization efficiency on KITTI datasets. The bold indicates the best while the italic is the second best.

	Dellenbach et al. (2022)	Ji and Singh (2017)	Koide et al. (2021)	Pan et al. (2021)	Cvišić et al. (2018)	Cvišić et al. (2021)	Zhang and Singh (2015)	Engel et al. (2015)	Pire et al. (2017)	Zhang et al. (2014)	Ours
Time (s)	0.06	0.1	0.1	0.08	0.1	0.1	0.1	0.07	*0.03*	0.1	**0.01**



(a) Sequence 06 (b) Sequence 07

Fig. 12. Running time of the online localization. The average time of sequence 06 and 07 is around 8.7 ms and 9.8 ms, respectively.

$$\alpha_h = \text{acos}\left(\frac{\text{trace}(\mathbf{R}_{gt}^{\mathrm{T}}\mathbf{R}_{es}) - 1}{2}\right)\frac{180}{\pi} \tag{14}$$

where $\mathbf{R}_{gt}$ is the ground truth rotation. $\mathbf{R}_{es}$ is the estimated rotation. Fig. 11 displays the heading error comparisons of our two solutions and most heading errors are within 6°. Although the urban road environment is variable, we still achieve outstanding heading accuracy due to the many static buildings. We achieve the best heading accuracy in sequence 06 as the vehicle rarely steers and there are abundant structural components along the road.

Table 9
Running time (ms) comparisons to scan context on KITTI 07.

Method	Descriptor Creation	Pairwise Comparisons	Map Query
Scan Context	29.92	0.66	0.72
Ours	4.86	0.0002	0.06

4.4.4. Running efficiency

(1) As summarized in Table 8, we compare the localization efficiency to some SOTA methods in KITTI sequences. Our method achieves the best efficiency (0.01 s), followed by S-PTAM (0.03 s), which is faster than the majority of the localization methods based on scan-to-scan matching. (2) As shown in Fig. 12, we compute the running time of the online localization in sequence 06 and 07. Online localization takes an average of 8.7 ms and 9.8 ms, respectively. Indoor localization even takes only 7 ms in Table 4. Our method achieves the localization efficiency of 100FPS on a single-thread CPU. Our efficiency is affected by

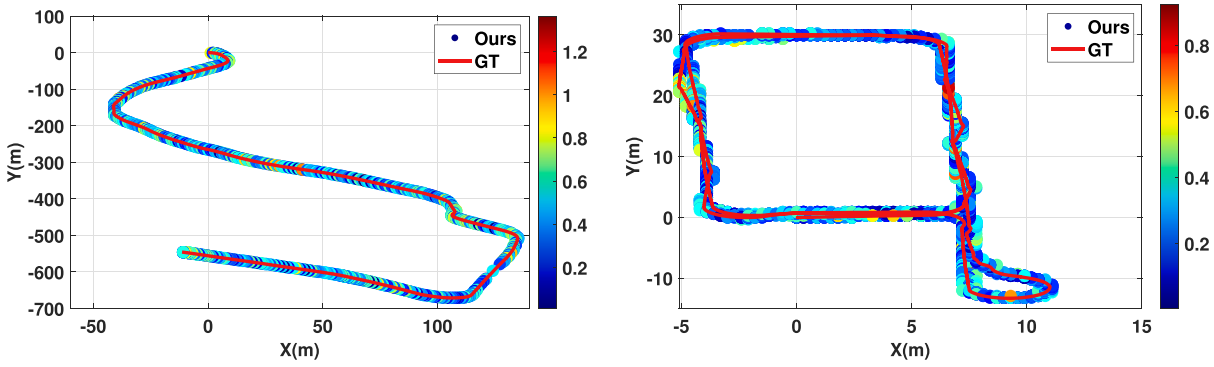
Table 10
Success rate and running time of our two solutions on KITTI datasets. The bold indicates the best while the italic is the second best.

Solution	Metric	03	06	07	10	Mean
fst	Recall	*0.92*	**0.92**	0.89	0.88	0.90
fst	Time(s)	*0.01*	**0.009**	0.01	0.01	0.01
opt	Recall	0.94	**1.0**	*0.99*	0.93	0.97
opt	Time(s)	*2.39*	**2.04**	2.71	2.64	2.44

two factors. One is the map database including candidate points, rotation angle resolution, and the number of bins. The other is the number of KNN search nodes. In our experiments, we just adopt a few parameter setups to balance the localization accuracy and efficiency. (3) We further compare our efficiency to SOTA Scan Context (Kim and Kim, 2018) that also uses KD-Tree to accelerate the descriptor search in Table 9. The results show that our method outperforms Scan Context in descriptor creation, pairwise comparison, and map query. The descriptor creation takes less than 5 ms.

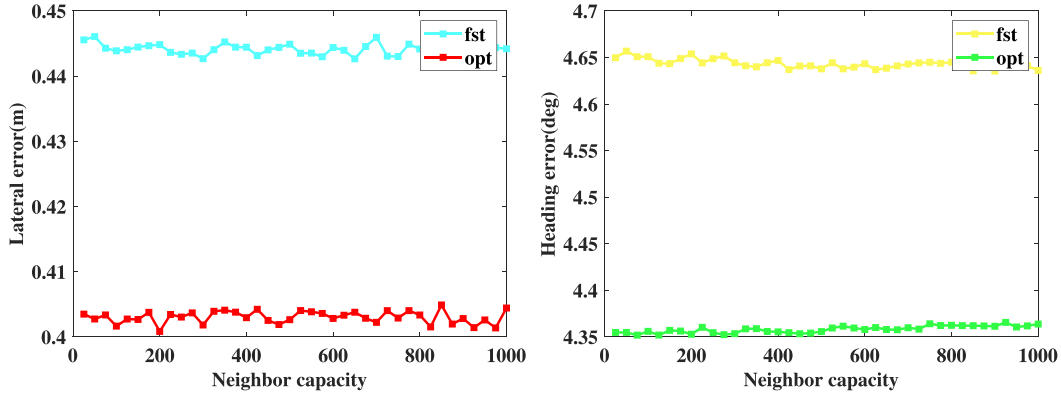
4.4.5. Recall and trajector

Table 10 summarizes the success rate and efficiency using the two

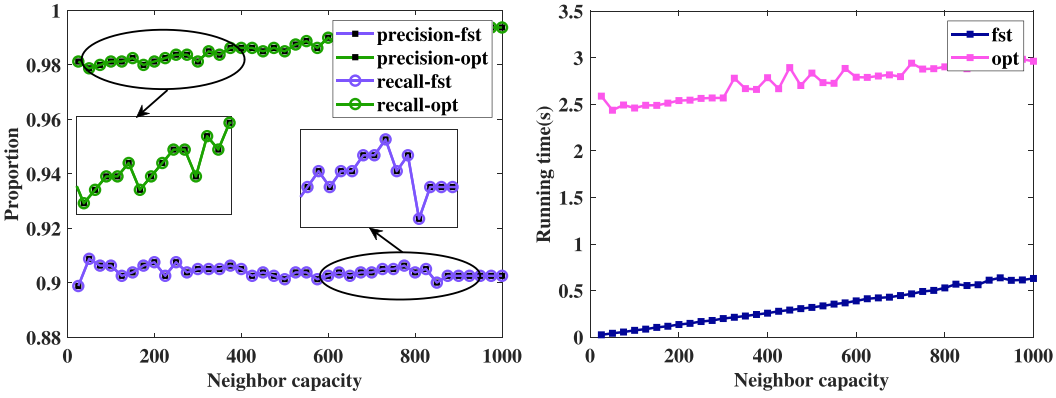


(a) Sequence 10 (b) Corridor

Fig. 13. Trajectory analysis in KITTI 10 and corridor. The red line denotes the ground truth trajectory. The solid points denote the computed vehicle positions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) Distance error (b) Heading error



(c) Precision and recall (d) Running time

Fig. 14. Comparisons of global evaluation with various descriptor neighbor capacities in KITTI 03.

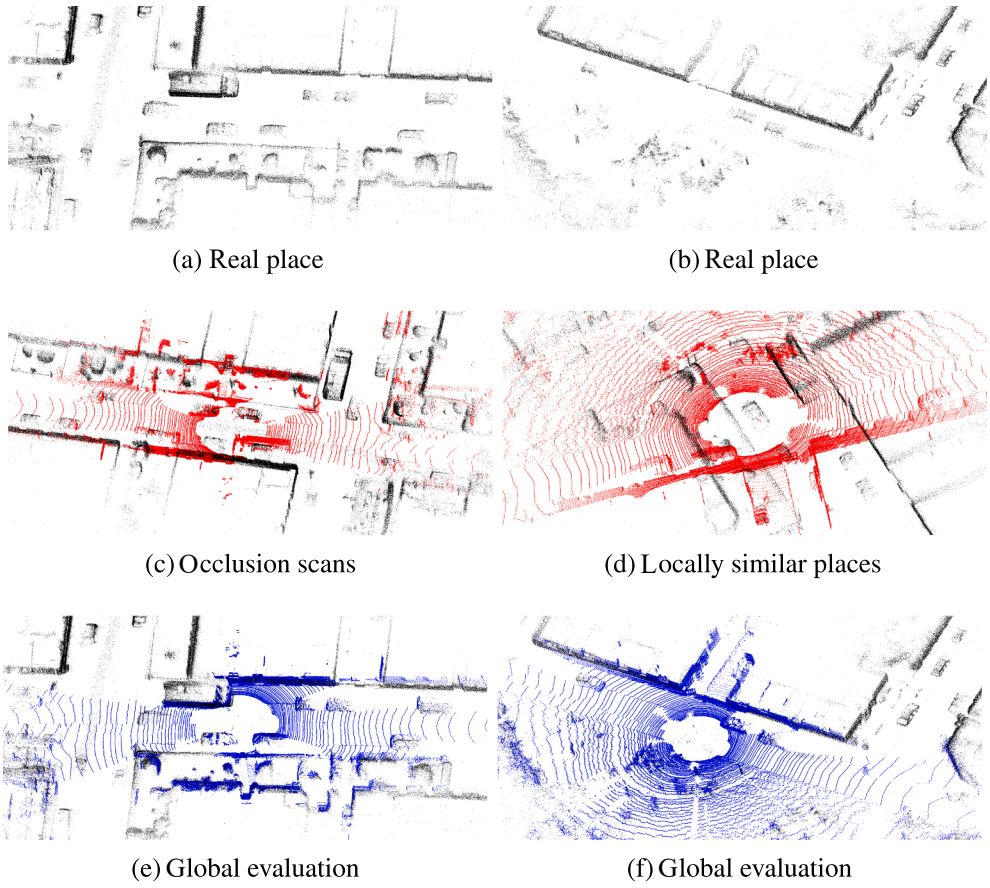


Fig. 15. Failure cases in KITTI 07. (a) and (b) are two real positions. (c) and (d) are two failure cases in local evaluation. (e) and (f) denote the vehicle is located successfully in global localization.

solutions in four KITTI sequences. The average time of *fst* takes less than 10 ms. *fst* achieves a success rate of more than 90% with an average time of less than 10 ms. *opt* takes relative more time (2 s) to achieve a higher success rate (97%). Fig. 13 depicts the global trajectory deviation in sequence 03 and corridor. The small trajectory deviation further verifies the localization effectiveness for both indoor and outdoor scenarios. Our method can successfully compute the vast majority of vehicle poses.

4.4.6. Global evaluation comparisons

As shown in Fig. 14, we compare the global evaluation under different capacities of the nearest map descriptors in sequence 03. Our running time increases proportionally with the neighbor's capacity. *opt* takes 2.7 s when searching 1,000 candidates. The success rate of *opt* initially exceeds 0.98 and achieves 100% as the capacity reaches 1,000. However, the capacity growth does not significantly improve the success rate of *fst*. Similar to indoor evaluation (Section 4.3), a relatively low κ_1 led to the high overlap of precision and recall curves. The lateral and heading errors of *opt* are approximately 0.05 m and 0.3° less than *fst*, respectively. The local evaluation achieves comparable accuracy and desirable efficiency.

4.4.7. Failure cases

Fig. 15 displays two failure cases in local evaluation. In Fig. 15(c), most LiDAR's laser emissions are blocked by a truck. In Fig. 15(d), surrounding similar wall distribution leads to another failed localization. However, global evaluation successfully localizes the vehicle in both cases. The local and global evaluation can alternatively switch to balance the efficiency and success rate or jointly collaborate at different frequencies to build a multi-thread system.

5. Conclusions

In this paper, we proposed a template descriptor-based LiDAR global localization method. It employs a straightforward template representation to describe the structural environments. We estimate the vehicle poses using the associations between the scans and map databases. Using a KD tree with our proposed loss function enables efficient localization (100 FPS) with desirable success rates (93%). Presented local and global evaluation can alternatively switch or cooperate at different frequencies for superior performance. We conducted extensive experiments to confirm the feasibility and validity of the proposed method. Its localization efficiency and accuracy overwhelmingly exceeded SOTA baseline methods. In future work, we intend to incorporate our method into an odometry framework to serve as a real-time SLAM system.

CRediT authorship contribution statement

Pengcheng Shi: Conceptualization, Methodology, Validation, Software, Investigation, Formal analysis, Writing – original draft. **Jiayuan Li:** Supervision, Methodology, Data curation, Funding acquisition, Writing – review & editing. **Yongjun Zhang:** Supervision, Visualization, Writing – review & editing.

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.

Data availability

Data will be made available on request.

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 under Grant 2021AAA010.

References

An, S.Y., Kim, J., 2022. Extracting statistical signatures of geometry and structure in 2d occupancy grid maps for global localization. IEEE Rob. Autom. Lett. 7, 4291–4298.
Besl, P., McKay, N.D., 1992. A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. 14, 239–256. https://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. Rob. 1–20. https://doi.org/10.1109/TRO.2022.3150683.
Chetverikov, D., Svirko, D., Stepanov, D., Krsek, P., 2002. The trimmed iterative closest point algorithm. In: 2002 International Conference on Pattern Recognition, Vol. 3, pp. 545–548. doi: 10.1109/ICPR.2002.1047997.
Cho, Y., Kim, G., Lee, S., Ryu, J.H., 2022. Openstreetmap-based lidar global localization in urban environment without a prior lidar map. IEEE Rob. Autom. Lett. 7, 4999–5006. https://doi.org/10.1109/LRA.2022.3152476.
Cvišić, I., Česić, J., Marković, I., Petrović, I., 2018. Soft-slam: computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles. J. Field Rob. 35, 578–595.
Cvišić, I., Marković, I., Petrović, I., 2021. Recalibrating the kitti dataset camera setup for improved odometry accuracy. In: 2021 European Conference on Mobile Robots (ECMR), IEEE, pp. 1–6.
de Miguel-Díez, F., Reder, S., Wallor, E., Bahr, H., Blasko, L., Mund, J.P., Cremer, T., 2022. Further application of using a personal laser scanner and simultaneous localization and mapping technology to estimate the log's volume and its comparison with traditional methods. Int. J. Appl. Earth Obs. Geoinf. 109, 102779.
Dellenbach, P., Deschaud, J.E., Jacquet, B., Goulette, F., 2022. Ct-icp: Real-time elastic lidar odometry with loop closure. In: 2022 International Conference on Robotics and Automation (ICRA), IEEE. pp. 5580–5586.
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. doi: 10.1109/ICRA.2017.7989618.
Engel, J., Stückler, J., Cremers, D., 2015. Large-scale direct slam with stereo cameras. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1935–1942. doi: 10.1109/IROS.2015.7353631.
Engel, J., Koltun, V., Cremers, D., 2018. Direct sparse odometry. IEEE Trans. Pattern Anal. Mach. Intell. 40, 611–625. https://doi.org/10.1109/TPAMI.2017.2658577.
Fan, W., Yang, B., Dong, Z., Liang, F., Xiao, J., Li, F., 2021. Confidence-guided roadside individual tree extraction for ecological benefit estimation. Int. J. Appl. Earth Obs. Geoinf. 102, 102368.
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), pp. 2198–2204. doi: 10.1109/IROS.2018.8593376.
Guo, L., Yang, M., Wang, B., Wang, C., 2016. Occupancy grid based urban localization using weighted point cloud. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 60–65. doi: 10.1109/ITSC.2016.7795532.
Hastaoğlu, K.Ö., Gül, Y., Poyraz, F., Kara, B.C., 2019. Monitoring 3d areal displacements by a new methodology and software using uav photogrammetry. Int. J. Appl. Earth Obs. Geoinf. 83, 101916.
Hata, A.Y., Ramos, F.T., Wolf, D.F., 2017. Monte Carlo localization on Gaussian process occupancy maps for urban environments. IEEE Trans. Intell. Transp. Syst. 19, 2893–2902.
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. doi: 10.1109/IROS.2016.7759060.
Ji, Z., Singh, S., 2017. Low-drift and real-time lidar odometry and mapping. Auton. Robot. 41, 401–416.
Justiniano, E.F., dos Santos Junior, E.R., de Melo, B.M., Siqueira, J.V.N., Morato, R.G., Fantin, M., Pedrassoli, J.C., Martines, M.R., Kawakubo, F.S., 2022. Proposal for an index of roads and structures for the mapping of non-vegetated urban surfaces using OSM and sentinel-2 data. Int. J. Appl. Earth Obs. Geoinf. 109, 102791.
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.
Koide, K., 2019. A portable 3d lidar-based system for long-term and wide-area people behavior measurement. Int. J. Adv. Rob. Syst. 16.
Koide, K., Yokozuka, M., Oishi, S., Banno, A., 2021. Globally consistent 3d lidar mapping with GPU-accelerated GICP matching cost factors. IEEE Rob. Autom. Lett. 6, 8591–8598. https://doi.org/10.1109/LRA.2021.3113043.
Li, S., Li, G., Wang, L., Qin, Y., 2020. Slam integrated mobile mapping system in complex urban environments. ISPRS J. Photogramm. Remote Sens. 166, 316–332.
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 singleimage depth prediction. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 5218–5223. doi: 10.1109/ICRA.2019. 8794425.
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. 7, 771–782. https://doi.org/10.1109/TIV.2022.3169153.
Lv, L., Chen, T., Dou, J., Plaza, A., 2022. A hybrid ensemble-based deep-learning framework for landslide susceptibility mapping. Int. J. Appl. Earth Obs. Geoinf. 108, 102713.
Magnusson, M., Lilienthal, A., Duckett, T., 2007. Scan registration for autonomous mining vehicles using 3D-NDT. J. Field Rob. 24, 803–827.
Mendez-Astudillo, J., Lau, L., Tang, Y.T., Moore, T., 2021. A new global navigation satellite system (GNSS) based method for urban heat island intensity monitoring. Int. J. Appl. Earth Obs. Geoinf. 94, 102222.
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. doi: 10.1109/IROS40897.2019.8967683.
Pan, Y., Xiao, P., He, Y., Shao, Z., Li, Z., 2021. Mulls: Versatile lidar slam via multi-metric linear least square. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 11633–11640. doi: 10.1109/ICRA48506.2021.9561364.
Pire, T., Fischer, T., Castro, G., De Cristóforis, P., Civera, J., Berlles, J.J., 2017. S-ptam: Stereo parallel tracking and mapping. Rob. Auton. Syst. 93, 27–42.
Ruchti, P., Steder, B., Ruhnke, M., Burgard, W., 2015. Localization on openstreetmap data using a 3d laser scanner. In: 2015 IEEE international conference on robotics and automation (ICRA), IEEE. pp. 5260–5265.
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. doi: 10.1109/ROBOT.2009.5152473.
Segal, A., Haehnel, D., Thrun, S., 2009. Generalized-ICP. In: Robotics: science and systems, Seattle, WA. p. 435.
Shi, P., Ye, Q., Shaoming, Z., Haifeng, D., 2021. Localization initialization for multi-beam lidar considering indoor scene feature. Acta Geodaetica et Cartographica Sinica 50, 1594–1604. doi: 10.11947/j.AGCS.2021.20210268.
Sofonia, J., Shendryk, Y., Phinn, S., Roelfsema, C., Kendoul, F., Skocaj, D., 2019. Monitoring sugarcane growth response to varying nitrogen application rates: a comparison of UAV slam lidar and photogrammetry. Int. J. Appl. Earth Obs. Geoinf. 82, 101878.
Suger, B., Burgard, W., 2017. Global outer-urban navigation with openstreetmap. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 1417–1422.
Theiler, P.W., Wegner, J.D., Schindler, K., 2014. Keypoint-based 4-points congruent sets – automated marker-less registration of laser scans. ISPRS J. Photogramm. Remote Sens. 96, 149–163. https://doi.org/10.1016/j.isprsjprs.2014.06.015.
Xiang, H., Shi, W., Fan, W., Chen, P., Bao, S., Nie, M., 2021. Fastlcd: a fast and compact loop closure detection approach using 3d point cloud for indoor mobile mapping. Int. J. Appl. Earth Obs. Geoinf. 102, 102430.
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.
Yan, M., Wang, J., Li, J., Zhang, C., 2017. Loose coupling visual-lidar odometry by combining viso2 and loam. In: 2017 36th Chinese Control Conference (CCC), pp. 6841–6846.
Yan, F., Vysotska, O., Stachniss, C., 2019. Global localization on openstreetmap using 4-bit semantic descriptors. In: 2019 European Conference on Mobile Robots (ECMR), IEEE. pp. 1–7.
Ye, R., Huang, Z., Li, L., Shan, X., 2022. Geounet: a novel AI model for high-resolution mapping of ecological footprint. Int. J. Appl. Earth Obs. Geoinf. 112, 102803.
Yin, H., Wang, Y., Ding, X., Tang, L., Huang, S., Xiong, R., 2019. 3d lidar-based global localization using Siamese neural network. IEEE Trans. Intell. Transp. Syst. 21, 1380–1392.
Zhan, H., Weerasekera, C.S., Bian, J.W., Reid, I., 2020. Visual odometry revisited: What should be learnt?. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 4203–4210. doi: 10.1109/ICRA40945.2020.9197374.
Zhang, J., Singh, S., 2015. Visual-lidar odometry and mapping: low-drift, robust, and fast. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2174–2181. doi: 10.1109/ICRA.2015.7139486.
Zhang, J., Kaess, M., Singh, S., 2014. Real-time depth enhanced monocular odometry. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4973–4980. doi: 10.1109/IROS.2014.6943269.
Zhao, C., Tang, Y., Sun, Q., Vasilakos, A.V., 2021. Deep direct visual odometry. IEEE Trans. Intell. Transp. Syst. 1–10. https://doi.org/10.1109/TITS.2021.3071886.
Zhou, Q.Y., Park, J., Koltun, V., 2016. Fast global registration. In: European Conference on Computer Vision. Springer. pp. 766–782.