



S4-SLAM: A real-time 3D LIDAR SLAM system for ground/watersurface multi-scene outdoor applications

Bo Zhou¹ · Yi He¹ · Kun Qian¹ · Xudong Ma¹ · Xiaomao Li²

Received: 23 March 2019 / Accepted: 11 September 2020 / Published online: 9 October 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

For outdoor ground/watersurface multi-scene applications with sparse feature points, high moving speed and high dynamic noises, a real-time 3D LIDAR SLAM system (S4-SLAM) for unmanned vehicles/ships is proposed in this paper, which is composed of the odometry function in front-end and the loop closure function in back-end. Firstly, linear interpolation is used to eliminate the motion distortion caused by robot motions in the data pre-processing step. Two nodes are constructed in the odometry function: the localization node combines the improved Super4PCS with the standard ICP to realize a coarse-to-fine scan matching and outputs the location information of the robot at a high frequency (5 Hz); the correction node introduces a local map with dynamic voxel grid storage structure, which can accelerate the NDT(Normal Distributions Transform) matching process between key-frames and the local map, and then corrects the localization node at a low frequency (1 Hz) to obtain more accurate location information. In the loop closure function, a location-based loop detection approach is introduced and the overlap rate of point clouds is used to verify the loops, so that the global optimization can be carried out to obtain high-precision trajectory and map estimates. The proposed method has been extensively evaluated on the KITTI odometry benchmark and also tested in real-life campus and harbor environments. The results show that our method has low dependence on GPS/INS, high positioning accuracy (with the global drift under 1%) and good environmental robustness.

Keywords Multi-scene mapping · LIDAR odometry · SLAM · Loop detection · Scan matching

1 Introduction

With the development of unmanned driving, real-time localization and navigation research for outdoor mobile robots or unmanned vehicles are receiving more and more attentions. The common solution for navigation tasks of unmanned vehicles at present is heavily relied on high-precision maps and GPS/INS information. However, when GPS data and high-precision maps are not available, SLAM (Simultaneously Localization and Mapping) is thought to be a good substitute to solve the above problems. Visual SLAM and LIDAR

SLAM are two commonly used outdoor SLAM approaches. While visual SLAM algorithms are greatly influenced by the intensity of light, the optical structure and the texture features of environments, the robustness of the system is often challenged, especially for long-term large-scale scenes applications. In contrast, the principle of LIDAR SLAM algorithms is to match point clouds obtained by LIDAR sensors, which mainly depends on the geometric information of the environment. Benefited by wide measurement range, high accuracy and good robustness, LIDAR SLAM has been widely used in unmanned robots or vehicles, and is often considered as the real key to solve autonomous navigation problem such as “unmanned driving”.

Generally speaking, LIDAR SLAM approaches for on-road applications are relatively mature, but there is no satisfactory solution for complicated, unconstructed, multi-scene environments, such as ground/watersurface composite environments (harbors, dams, bridges, islands, etc.) dealt in this paper. Ground/watersurface composite environments are very different from ground-only environments: first, there are fewer laser points or sparser features on the water due to laser

✉ Xiaomao Li
lixiaomaosia@163.com

Bo Zhou
zhoubo@seu.edu.cn

¹ School of Automation, Southeast University, Nanjing 210096, People’s Republic of China

² School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200000, People’s Republic of China

rays' inability to be reflected, so approaches depending on traditional geometric features (lines, corners, edges, and so on) or global/local descriptors (B-shot, FPFH, VFH, and so on) extraction may fail; second, unmanned ships are vulnerable to the influence of wind and waves on the water so that the motion is unstable, which may produce distortions of 3D point cloud images; finally, there will be a lot of dynamic noises on the water in perception process, which may greatly affect the matching results if not processed effectively. Due to above reasons, traditional LIDAR SLAM methods for ground-only environments have poor robustness and performances when applied to complex multi-scene environments with a lot of water. In addition, shortcomings such as the failure of matching in high-speed motion and great dependence on GPS/INS information may also lead to the difficulties in the practical application of traditional SLAM approaches in unmanned navigation systems.

In this paper, the SLAM problem of outdoor robots such as UGV or USV in outdoor multi-scene environments is studied. In view of the above mentioned shortcomings, approaches with traditional geometric features or descriptors extracted from point clouds is not considered here, because under such challenging environments less features can be extracted for most of scans, which may makes the algorithm unstable and time-consuming, especially in low-lapping, high-speed, high dynamic scenes. Instead, the Super4PCS algorithm with only co-planar 4-point extraction is adopted for initial transformation estimation in scan matching for ease of use, better robustness and real-time property even in low-lapping multi-scene applications. Moreover, two other classical iterative point-to-point based scan matching approaches, ICP(Iterative Closest Point) and NDT (Normal Distributions Transform), are also introduced and combined to construct the final comprehensive coarse-to-fine LIDAR SLAM system (S4-SLAM) to achieve a good balance between accuracy (under 1% global drift) and real-time property (5 Hz for localization output, 1 Hz correction and mapping output), which makes it can keep high accuracy and real-time property in both UGV and USV applications.

The main contributions and the innovations of our method are as follows:

1. Traditional SLAM approaches are mainly designed and used in ground-only environments. Their performances will degraded heavily (sometimes the matching may even fail) when used in ground/watersurface composite environments with poor robustness. By introducing the Super4PCS algorithm in which only four co-planar points are needed for scan matching, our method can provide better environmental adaptability to deal with sparse point clouds or feature points, high speed movements, and high noise levels. Furthermore, several modifica-

tions are made on the original Super4PCS algorithm to improve the real-time performance and reduce the mismatch rate, so that it can be applied effectively to complicated multi-scene environments.

2. In this paper the mainstream framework of existing SLAM approaches are adopted and greatly improved. By extending the traditional single localization node into the new two-node localization-correction structure in the front-end, an effective combination or good trade-off among scan matching algorithms such as Super4PCS, ICP, and NDT can be achieved. For example, Super4PCS is used to provide rapidly an initial iteration value for ICP to realize a coarse-to-fine positioning process in the localization node at a high frequency, and NDT is speeded up by introducing a dynamic voxel map to correct the localization node at a lower frequency. Finally, a real-time LIDAR SLAM system with good robustness is constructed;
3. Compared to other SLAM approaches, the proposed S4-SLAM method has lower dependence on GPS/INS data, and can be applied to GPS-loss/GPS-denied environments. Extensive experiments have been carried out on KITTI dataset, campus and harbor environments, and the results show that our method can produce high-frequency and high-precision positioning and mapping outputs when unmanned vehicles/ships moves at a high speed.

2 Related work

LIDAR sensors are becoming more and more popular in mobile robots and unmanned vehicles applications, and up to now a large number of LIDAR SLAM approaches have been proposed accordingly. In this section only 6-Dof LIDAR SLAM approaches with 3D point cloud registration/mapping based on scan matching algorithms are discussed. The earliest and best established algorithm for solving the point cloud matching problem is ICP (Besl and McKay 1992). However, the standard ICP algorithm need a good initial iteration value, otherwise the final iteration result may fall into a local optimum (Pottmann et al. 2006; Bae 2009). Many researchers have proposed modified versions of the original ICP algorithm. Bouaziz et al. (2013) proposed the Sparse ICP algorithm that uses sparse norm to replace the 2-norm and achieves better registration results. But this improved ICP is still a local optimization algorithm, and cannot guarantee the convergence to the global optimal solution. Yang et al. (2016) proposed the GO-ICP algorithm that tries to avoid the local minima problem and ensures global convergence, but it requires a larger overlap of two point clouds. A widely used yet effective global solution is to roughly align two point clouds manually or with some pre-processing

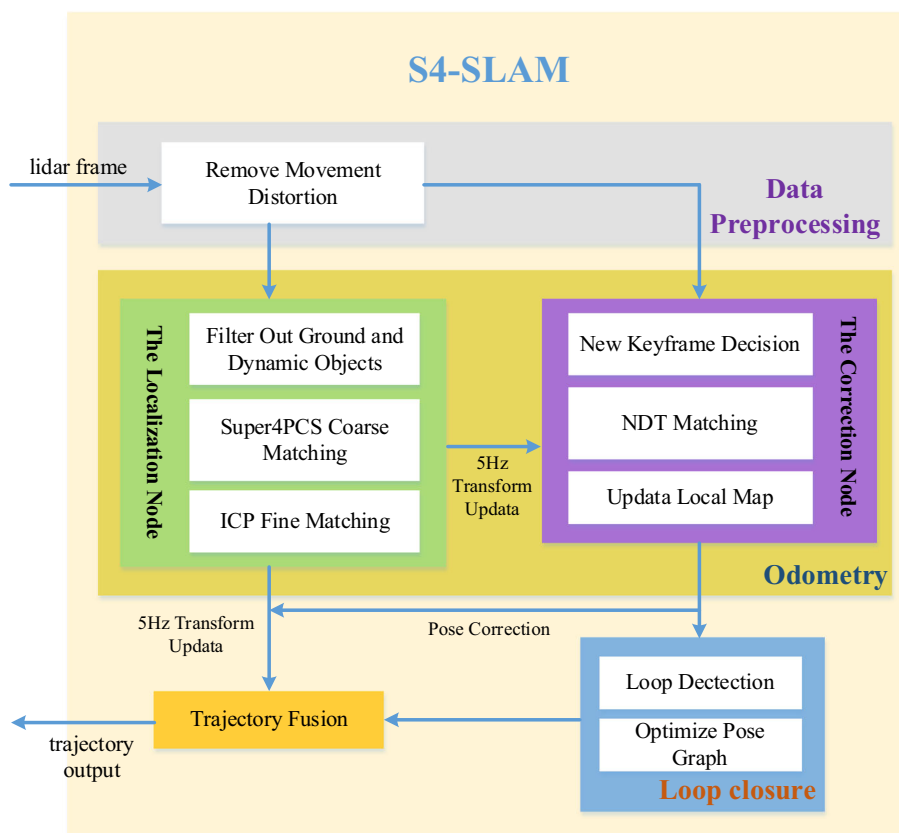
techniques to get a coarse initial pose estimation and then use ICP iteration steps to achieve fine registration, but this coarse-to-fine method is generally more complex and difficult to apply in real-time. Biber and Straßer (2003) proposed the so-called NDT(Normal Distributions Transform) algorithm based on a completely probabilistic model description, and Magnusson (2009) generalized the NDT method to 3D, but the performance is sensitive to the selection of the grid-cell size. Irani and Raghavan (1999) proposed the RANSAC algorithm, which randomly selects three pairs of corresponding points from two point clouds, and calculates the rigid transformation between these two point clouds iteratively. However, this algorithm is still not applicable to large-scale point clouds. Aiger et al. (2008) proposed the 4PCS algorithm as a variant of the RANSAC algorithm. A coplanar 4-point set is extracted from two point clouds to get the rigid transformation with the minimum registration error as the final registration result, which makes 4PCS an excellent global registration algorithm. Furthermore, Mellado et al. (2014) proposed the Super4PCS algorithm with improvements based on the 4PCS algorithm. It can greatly reduce the complexity of the original 4PCS and moreover can handle larger displacement and low overlap between point clouds. Experiment results of this paper also shows the Super4PCS algorithm has good robustness to keep the performance in ground/watersurface multi-scene environments applications that has fewer data and much sparser features. The Super4PCS can be used to find a coarse approximation of the global optimum, and then combined with a local algorithm (e.g., ICP or NDT) to refine the transformation.

ICP based LIDAR SLAM approaches can be used effectively when the vehicle moves slowly or the scanning rate is set to high. In this case, the two adjacent frames of point clouds have a large overlap and can be registered and aligned in a good manner. Almeida and Santos (2013) use the ICP algorithm to match the two adjacent frames and obtain the corresponding position transformation relations, but their method may possibly fall into local optimum when the overlap between frames is small. Besides, this method is difficult to meet the requirements of real-time. Viejo and Cazorla (2014) extract scene vertices and plane features from point clouds to complete matching and positioning. Moosmann and Stiller (2011) proposed a 3D LIDAR SLAM which takes into account the motion distortion caused by the rotation of the LIDAR and used an interpolation strategy to remove the distortion, and then a 3D grid map with small surfaces is constructed. After that the constraint relationship between points and surfaces is used to calculate the transformation matrix. Ceriani et al. (2015) construct a sparse grid map and use a generalized ICP method to accomplish the positioning task. Dubé et al. (2018) proposed a 3D LIDAR SLAM framework based on segment matching, in which the region growing idea is adopted to segment a point cloud and track each seg-

ment after segmentation. Finally, the transformation matrix is obtained by the verification of geometric consistency. Furthermore, Dubé et al. (2017) added a loop detection method based on point cloud to their SLAM algorithm. Their method extracts multiple features of each segment, and uses the random forest to complete segment classification and matching. The geometric consistency of the segments is verified to detect a true loop. Zhang and Singh (2014) put forward a method of real-time LIDAR odometry (LOAM), it estimates the velocity of LIDAR and continuously modifies LIDAR pose by matching edge and planar feature points extracted from the scanned data. Zhang and Singh (2015) put forward a method combining visual odometry and LIDAR odometry (V-LOAM). Their experiments proved that this method is insensitive to the surrounding light condition, and show satisfactory accuracy and robustness even when the robot moves at high speed. Shan and Englot (2018) proposed LeGO-LOAM which improves the LOAM algorithm by extracting flat features on the ground and sharp features off the ground, and then matched the features by two-steps L-M optimization, but this algorithm needs to assume that the ground exists. Koide et al. (2018) proposed a real-time SLAM algorithm that constructs the ground constraints by extracting the ground plane and uses the graph optimization to build the map. However, this algorithm still needs to assume that the ground exists, and cannot operate in other non-ground environments. Deschaud (2018) proposed a SLAM method based on Implicit Moving Least Squares surface representation (IMLS-SLAM). The method extracts sampling points with strongest observability to pose parameters, and then uses the constraint relationship between the sample points and the implicit surfaces to complete the scan-to-model registration. Their algorithm can finally achieve similar precision as LOAM, but cannot be used in real-time. Ji et al. (2018) propose the CPF-G-SLAM algorithm which combines NDT and ICP to perform better than LOAM in open field with fewer feature points, however the computation complexity is high and can not be used in real-time too. Behley and Stachniss (2018) use the Surfel map in the 3D LIDAR SLAM of outdoor large scene which map the point clouds obtained by the rotating laser to 2D planes in order to constructed the Surfel map, but the accuracy and real-time property are still lower than LOAM.

Most above mentioned LIDAR SLAM approaches are designed for ground environments with rich features, and highly depend on ground points or need to treat the ground as an important constraints, which limits their application performances and robustness in more challenging multi-scene environments. By now few related studies on LIDAR SLAM is reported to be used in ground/watersurface composite environments. Most existing algorithms can be find it difficult to achieve good results in this different kind of environments. As far as the authors know, our paper is the first one to carry

Fig. 1 Block diagram of the S4-SLAM system



out LIDAR SLAM research and experimental verification on both unmanned ground vehicles and ships simultaneously. Three existing mainstream matching methods (Super4PCS, ICP, NDT) are fused to improve the standard SLAM framework. Finally a high-precision SLAM system with good robustness and real-time is constructed in order to be applied in complex multi-scene environments.

3 System overview

Figure 1 shows the diagram of the proposed S4-SLAM system. Our SLAM system is composed of two functions: the odometry as the front-end and loop closure optimization as the back-end. In odometry the classic SLAM framework is improved and extended from the traditional single localization node into a new two-node structure including a localization node and a correction node, where the localization node outputs the location information at a high frequency and the correction node corrects the localization node at a lower frequency.

Considering the trade-off between real-time performance and localization accuracy, a coarse-to-fine scan matching method is used by combining the Super4PCS algorithm with the ICP algorithm in the localization node. In view of shortcomings of the Super4PCS algorithm, a filter is proposed to

roughly filter out the ground and the dynamic objects, and a heuristic iterative strategy based on the overlap rate evaluation is introduced to select the four coplanar points. All these improvements can accelerate the matching speed of the original Super4PCS algorithm. After that, the standard ICP matching method is used in fine matching. Since the coarse matching step can provide a good initial value for ICP, an accurate pose estimation between frames could be calculated quickly with less ICP iterations after down-sampling. Moreover, because motion drifts and noises are inevitable in the matching of two adjacent frames, it is necessary to correct the odometry periodically to ensure the localization accuracy. In the correction node a fixed-size dynamic voxel grid local sub-map is built and the odometry is corrected by matching the current key-frame with the sub-map. In the loop closure function our system introduces a location-based loop detection approach and verifies the loops with the overlap rate to reduce the probability of false positives. The trajectory output node integrates the pose message of the odometry function and the loop closure function, and finally outputs the pose message of the vehicle.

4 Data preprocessing

LIDAR devices acquires laser points by rotating laser rays. During scanning, the point cloud acquired from a scan would have motion distortion due to the motion of the vehicle, so a data preprocessing step should be carried out to eliminate it in advance before registration.

In this paper, coordination frames and symbols are defined as follows. The LIDAR coordinate system is expressed as $\{L\}$ with the origin located at the geometric center of the LIDAR, and the x -axis pointing to the forward, the y -axis pointing left, the z -axis is pointing upward. The world coordinate system $\{W\}$ is set to coincide with $\{L\}$ at the initial position for convenience. Let k be the k -th scan of the LIDAR, P_k is the point cloud obtained from the k th scan, \tilde{P}_k is the point cloud obtained after removing the ground and the dynamic objects of P_k , and \hat{P}_k is the point cloud obtained after downsampling of \tilde{P}_k . A point $i, i \in P_k$ is denoted as $X_{(k,i)}^L$ in $\{L\}$, and $X_{(k,i)}^W$ in $\{W\}$, where the right uppercase superscription to indicate the coordinate systems.

Let the start time of the k th scan be t_k and the end time be t_{k+1} , the motion of the vehicle during $[t_k, t_{k+1}]$ is expressed as:

$$\Delta s_{(k,k+1)} = [\Delta\theta_x, \Delta\theta_y, \Delta\theta_z, \Delta p_x, \Delta p_y, \Delta p_z]^T, \tag{1}$$

where $\Delta\theta_x, \Delta\theta_y, \Delta\theta_z$ are rotation angles around the x -, y -, and z -axis, and $\Delta p_x, \Delta p_y, \Delta p_z$ are displacements along the x -, y -, and z -axis respectively. For a laser point $\tilde{X}_{(k,i)}^L$ at $t_i^k \in [t_k, t_{k+1}]$, the linear interpolation strategy can be used to eliminate the influence of motion distortion, that is, the current motion of the vehicle relative to the initial time t_k can be calculated as $c \cdot \Delta s_{(k,k+1)}$, where $c = (t_i^k - t_k)/t_{scan}$ is a defined scalar, and t_{scan} represents the period time required for a scan. However, it is difficult to obtain the time interval $t_i^k - t_k$ accurately in real-life application. Instead the scanning angle of the laser ray is adopted to calculate c as $c = (\alpha_i^k - \alpha_k)/360$, where α_i^k and α_k are the horizontal rotation angle of the laser ray at the current and the initial time respectively. Then $\tilde{X}_{(k,i)}^L$ is projected to the coordinate system of the initial time:

$$X_{(k,i)}^L = R_{(k,i)}^{-1}(\tilde{X}_{(k,i)}^L - T_{(k,i)}), \tag{2}$$

where $T_{(k,i)} = c \cdot [\Delta p_x, \Delta p_y, \Delta p_z]^T$ represents the displacement matrix and $R_{(k,i)} = R_{z(k,i)}R_{y(k,i)}R_{x(k,i)}$ represents the rotation matrix, and $X_{(k,i)}^L$ represents the point after eliminating the motion distortion.

If the vehicle has an inertial navigation system such as an IMU device, $\Delta s_{(k,k+1)}$ can be obtained directly from the device. Otherwise, the constant-velocity motion model can be used to solve the problem approximately, which assumes that the motion of the current frame is equal to that of the

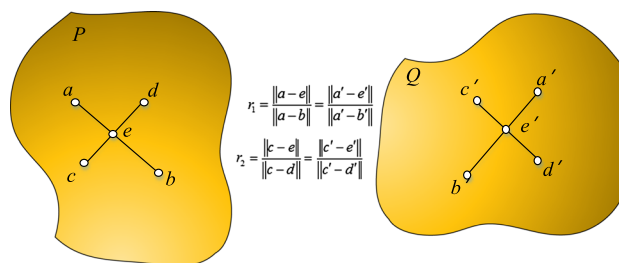


Fig. 2 The cross ratios of line segments remain invariant under affine transformations and rigid motion

previous frame, $\Delta s_{(k,k+1)} = \Delta s_{(k-1,k)}$. Through the above process, the motion distortion is eliminated.

5 LIDAR odometry

The LIDAR odometry is improved and extended from traditional single localization node into a new two-node structure, that is, it consists of a localization node and a correction node. The localization node realizes a coarse-to-fine matching process by combining Super4PCS with ICP to output pose message at a high frequency. The correction node uses the NDT method to match the key-frame with the dynamic local sub-map, and corrects the localization node at a lower frequency.

5.1 The localization node

5.1.1 Review of Super4PCS

The standard 4PCS algorithm solves the global 3D registration problem by using co-planar sets of four points. Figure 2 shows the principle of the algorithm. According to the principle of geometry, the cross ratios of line segments remain invariant under affine transformations and rigid motions. Suppose four points in a set $B = \{a, b, c, d\}$ selected from a point cloud P are co-planar, the line segments ab and cd intersect at point e , and the cross ratios of the two line segments are: $r_1 = \|a - e\|/d_1, r_2 = \|c - e\|/d_2$, where d_1, d_2 are the lengths of two line segments: $d_1 = \|a - b\|, d_2 = \|c - d\|$. According to the invariance of cross ratios r_1, r_2 , all corresponding 4-points sets $U = \{U_1, U_2, \dots, U_v\}$ can be found in the point cloud Q . For each $U_i, i = 1, 2, \dots, v$, compute the optimal rigid transformation matrix between B and U_i . The solution with largest overlap ratio between P and Q is selected as the true corresponding 4-points set $B' = \{a', b', c', d'\}$.

In view of the slow registration speed of 4PCS algorithm, Mellado et al. proposed Super4PCS algorithm, which uses intelligent index to reduce the computational complexity of 4PCS algorithm significantly. The time complexity of the

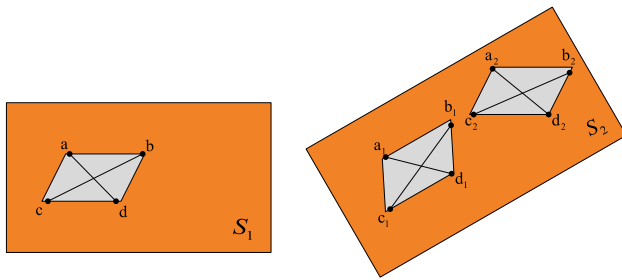


Fig. 3 It is possible to find wrong point sets in the point cloud with large planes, $\{a_1, b_1, c_1, d_1\}$ is the right point set corresponding to $\{a, b, c, d\}$, and $\{a_2, b_2, c_2, d_2\}$ is the wrong candidate point set

4PCS algorithm is $O(n^2 + k)$, where n is the number of points contained in the point cloud Q , k is the number of candidate congruent point sets. The Super4PCS algorithm has made two improvements compared to the 4PCS: (1) The process of searching matching point pairs based on distance is accelerated by rasterizing point clouds; (2) A smart index is proposed to filter out all pairs of redundant points. Through the above two improvements, the time complexity of the algorithm is reduced to linear complexity $O(n + k_1 + k_2)$, where n is the number of points in the target point cloud Q , k_1 is the number of point pairs with distance d , and k_2 is the number of congruent sets obtained after filtering.

However, when using the original Super4PCS algorithm for point cloud matching, there are still following disadvantages existed:

1. Firstly, the 4PCS and the Super4PCS may find wrong corresponding point pairs in the point cloud with large plane, which results in the failures of matching. As shown in Fig. 3, the four points a, b, c, d on the plane S_1 are co-planar, and many corresponding co-planar point sets such as $\{a_1, b_1, c_1, d_1\}, \{a_2, b_2, c_2, d_2\}, \dots$ can be found on the plane S_2 . In fact, the wrong point sets exist in large numbers because all points in this plane is co-planar. To find the correct set and exclude other wrong sets in these sets is very time-consuming. One feasible solution is to filter out these large planes in the point cloud before matching to avoid the unnecessary calculation and improve the efficiency of the algorithm;
2. Secondly, dynamic objects in the environment also affect the registration. It will be difficult to find a 4-points set in point cloud Q that is truly equal with B if one point in the co-planar 4-points set B is on a dynamic object, and it will waste a lot of time to verify these wrong point sets. Therefore, the dynamic objects in point clouds should be removed before registration in order to improve the efficiency and accuracy of registration;
3. Finally, the Super4PCS algorithm may not find the correct corresponding point set if the selected point is not

in the overlapping area. The Super4PCS algorithm first randomly selects three points in the point cloud Q , and then selects the fourth point to make these four points coplanar and the distances between each other are as large as possible. When the overlap rate of two point clouds is high, this strategy can find the best match quickly, however it may not find the best match when the overlap rate is low.

5.1.2 Coarse matching based on improved Super4PCS

In view of the above mentioned disadvantages of the original Super4PCS algorithm, several improvements are made in this paper to enhance the performance of this registration approach. For disadvantage (1), a ground filter based on elevation differences is proposed to filter out the ground points in order to avoid a large number of wrong matchings and meanwhile improve the real-time property by reducing the amount of potential matching pairs. For disadvantage (2), small dynamic objects are clustered and filtered out with a 2D binary image projection of the grid map to speed up the cluster process. For disadvantage (3), a heuristic iterative strategy based on the calculation of the overlap rate between two point sets is proposed to optimize the selection of co-planar four points, in order to reduce the amount of candidate points and the noise interference, which is especially useful when the overlap rate is low. All above strategies are integrated to improve the performance of the original Super4PCS by carefully balancing the registration accuracy and real-time property. The details of these improvements are given as follows.

As seen in real-life experiments, points from flat ground terrain usually form the largest plane in the point cloud, which may leads to wrong matchings. So the ground points should be filtered out firstly to improve the efficiency of matching. Considering real-time performance, the concept of elevation differences is used to roughly filter out the ground points without accurately extracting the ground and obstacles (Kammel et al. 2008). As shown in Fig. 4, a local grid map is built with the origin located at the geometric center of the LIDAR. The size of each voxel is set to 30 cm*30 cm, and the actual size of the map is consistent to the scanning range of our LIDAR. Each laser point $X_{(k,i)}^L$ will be mapped onto a single corresponding grid cell $c_{(m,n)}$, and all points in $c_{(m,n)}$ constitute the point set P_{mn} . For P_{mn} , find the maximum elevation different Δh_{mn} . If Δh_{mn} is less than a pre-determined threshold h_{thr} , the cell $c_{(m,n)}$ will be considered a ground cell. After traversing all cells by above steps and removing detected ground cells, the point cloud \tilde{P}_k without ground points can be obtained. Figure 5a shows the point cloud before ground filtering, Fig. 5b shows the point cloud after ground filtering.

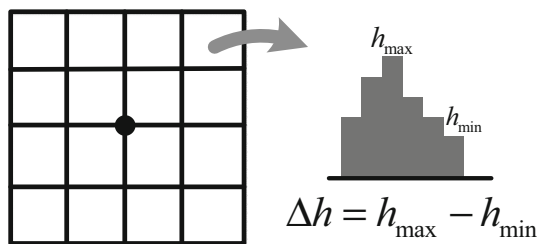
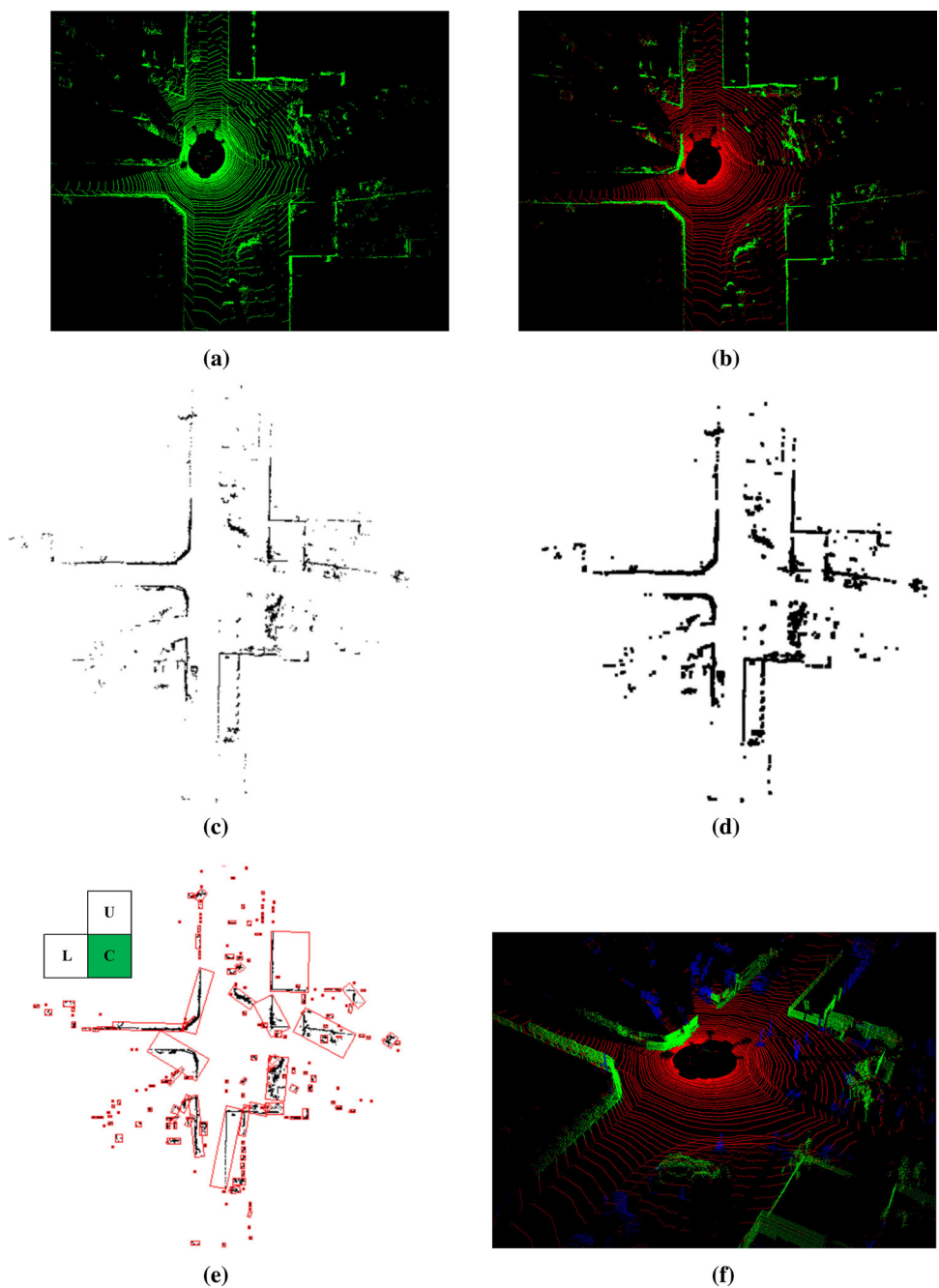


Fig. 4 Set up a local grid map and detect ground points according to the maximum elevation differences

As for the removal of dynamic objects, in our paper small dynamic objects instead of dynamic objects are considered and filtered out, that is, dynamic objects are replaced with small objects based on the observation that dynamic objects in real world are usually small in size. Moreover, the grid map is mapped to 2D image in order to speed up the clustering process greatly. The 2D image is represented by a binary image, as shown in Fig. 5c, in which black points represents objects above the ground.

Because objects in real world may be occluded, they may become discontinuous if clustered directly on the orig-

Fig. 5 The process of filtering out ground and dynamic objects: **a** shows the point cloud before ground filtering, **b** shows the ground detection results, where red points are detected as from ground and should be filtered out later, and green points are from buildings and other objects that should be remained, **c** shows the binary image which is mapped from the grid map, **d** is the image that after be eroded, **e** shows the result after the clustering process, **f** shows the result after filtering out the dynamic objects, where the blue point cloud is the possible dynamic object, the red point cloud is the ground we detected



inal image. In view of that the image erosion operator is used before clustering, as shown in Fig. 5d. After that, the region labeling algorithm is introduced to cluster the point cloud as following steps: (1) traverse the image from rows and columns respectively; (2) if the current pixel value is 0 (black), check whether the two pixel values on the left and on the top are 0; (3) if not, merge them and mark them with a label as a new cluster. Through above labeling steps all pixels with a same label are considered to be in the same cluster, and their rectangular bounding box can be calculated, as shown in Fig. 5e. Those clusters with areas less than a pre-determined threshold s_{thr} will be filtered out. Finally, the point cloud after filtering ground and dynamic objects are recovered from the grid map, are shown in Fig. 5f, where the green point cloud is the original point cloud, the blue is the possible dynamic object, and the red is classified as the ground.

As for the selection of co-planar four points, a heuristic iterative strategy is proposed in this paper according to the concept of LCP(Largest Common Pointset) based on the calculation of the overlap rate, and determines the maximum range in which the selected points should lie. Suppose there are two point clouds P and Q , find a subset P_{max} of P , $\forall p_i \in P_{max}, \exists q_i \in Q$ such that $\|T(p_i) - q_i\| \leq \delta, \delta > 0$ with the maximum overlap ratio calculated by the following formula (da Silva et al. 2011):

$$\lambda_{max} = \frac{|P_{max}|}{|P|}. \tag{3}$$

which means the maximum overlap under the admissible error δ ; the operator $|\cdot|$ stands for the number of all points in a point set. The biggest overlap ratio λ_{max} will correspond to the maximum range for selecting points.

The initial overlap rate can be roughly estimated based on the motion of two adjacent frames. As shown in Fig. 6, since point clouds are acquired by LIDAR scanning, two adjacent frames of point clouds can be approximated to two circles, and the overlap rate of the two frames can be estimated approximately according to the overlap area of the two circles, the area s_c of the shadow in Fig. 6 is:

$$s_c = 2r^2 \arccos\left(\frac{l_{12}}{2r}\right) - l_{12}\sqrt{r^2 - \frac{1}{4}l_{12}^2}, \tag{4}$$

where r is the scanning range of the LIDAR; l_{12} is the distance between centers of these two circles, which can be obtained from the motion of the vehicle. So the estimated initial overlap ratio is:

$$\lambda_{init} = \frac{s_c}{\pi r^2}, \tag{5}$$

The specific algorithm flow for computing the overlap ratio λ iteratively is as Algorithm 1.

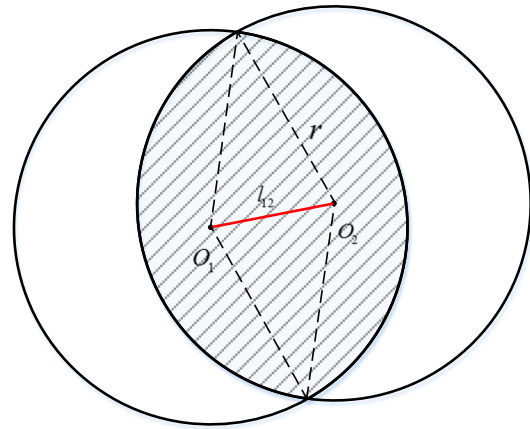


Fig. 6 The overlap rate can be roughly estimated by using the intersecting parts of two circles: overlap rate = shaded area/circle area

Algorithm 1: The registration optimization with overlap ratio

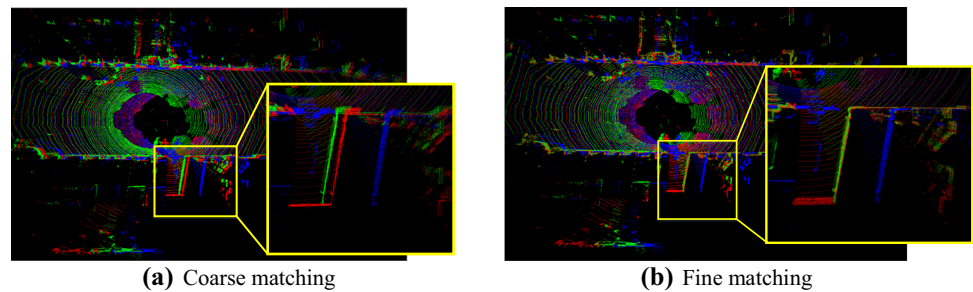
Input : Point cloud P, Q , Parameter α ($0.5 \leq \alpha \leq 1$)

Output : Best transformation matrix T_{acq}

- 1 Estimate the overlap rate of the two frames according to the overlap area of the two circles (see Equation (4)(5));
 - 2 Set initial parameters $\lambda_{min} = \lambda_{max} = \lambda_{init}, \mu_Q = |Q|/4$;
 - 3 **for** $k = 1$ to N **do**
 - 4 $\lambda_k = (\lambda_{min} + \lambda_{max}) / 2$;
 - 5 Select co-planar four-points set B from P by using λ_k ;
 - 6 Search for $U = \{U_1, U_2, \dots, U_v\}$ corresponding to B from Q ;
 - 7 **for** all candidate $U_i \in U$ **do**
 - 8 Calculate T_i and the overlap ratio λ_i of B and U_i ;
 - 9 **if** $\lambda_i > \lambda_{acq}$ **then**
 - 10 $\lambda_{acq} = \lambda_i, T_{acq} = T_i$;
 - 11 **end if**
 - 12 **end for**
 - 13 **if** $\lambda_{acq} > \lambda_k$ **then**
 - 14 $\lambda_{min} = \lambda_{acq}$;
 - 15 $\lambda_{max} = \min(1, 2\lambda_{acq})$;
 - 16 **else if** $v > \mu_Q$ **then**
 - 17 $\mu_Q = \alpha \cdot v$;
 - 18 $\lambda_{min} = \alpha\lambda_{min} + (1-\alpha)\lambda_{max}$;
 - 19 **else if** $v < \mu$ **then**
 - 20 $\mu_Q = (2-\alpha) \cdot v$;
 - 21 $\lambda_{max} = (1-\alpha)\lambda_{min} + \alpha\lambda_{max}$;
 - 22 **end if**
 - 23 **end for**
 - 24 **Return** T_{acq} ;
-

In order to reduce the computing time, a threshold λ_{thr} is set in advance so that the iteration will be terminated when

Fig. 7 The matching result of the localization algorithm. The blue point cloud is the source point cloud, the red point cloud is the target point cloud, and the green point cloud is the resulting point cloud. Coarse matching roughly aligns two frame point clouds, and fine matching aligns the point clouds precisely (Color figure online)



$\lambda \geq \lambda_{\text{thr}}$. The maximum translation distance d_{max} and the maximum rotation angle a_{max} of the vehicle should also be set previously so that movements larger than these two threshold will be excluded to accelerate the iteration.

5.1.3 Fine matching based on ICP

After coarse matching, ICP is used to accomplish fine matching between two point clouds. The standard ICP matching algorithm was firstly proposed by Besl and McKay (1992) with complexity $O(N_P, N_Q)$, where N_P and N_Q represent numbers of points contained in these two point clouds respectively. In our paper, the ICP algorithm with a 3D-Kdtree structure is used to reduce the time complexity to $O(N_P \log N_Q)$. However, when there are a large number of points in the point cloud, the algorithm can be still time-consuming. Therefore, downsampling is carried out before matching by using voxel grids to compress the points, while maintaining the shape characteristics of the point cloud. The number of points can be greatly reduced. As well as a good initial transformation matrix obtained by the coarse matching step, the matching speed of ICP is greatly accelerated. One of the main contributions of this paper is the use of coarse-to-fine matching methods to form the front-end node of our SLAM system, by combining several mainstream registration technologies to achieve good trade-off between the accuracy and the real-time performance.

5.1.4 Summary of the localization algorithm

The task of the localization node is to obtain the motion transformation matrix of two adjacent frames of point clouds. The whole iterative steps are summarized in Algorithm 2. It takes P_{k-1} (the last scan), P_k (the current scan), and $\tilde{P}_{k-1}, \hat{P}_{k-1}, \tilde{T}_{k-1}^W$ (the outputs from the last recursion) as the inputs. Finally, the algorithm outputs $\tilde{P}_k, \hat{P}_k, \tilde{T}_k^W$ for the next recursion.

Figure 7 shows final matching results of the localization algorithm. Figure 7a shows the matching result of the Super4PCS coarse matching method, and Fig. 7b shows the matching result of the ICP fine matching method.

Algorithm 2: The Coarse-to-Fine Localization

Input : $P_k, P_{k-1}, \tilde{P}_{k-1}, \hat{P}_{k-1}, \tilde{T}_{k-1}^W$
Output : $\tilde{P}_k, \hat{P}_k, \tilde{T}_k^W$

- 1 $\tilde{T}_{(k,k-1)}^L \leftarrow I$;
- 2 Obtain the point cloud \tilde{P}_k after removing the ground and dynamic objects of P_k ;
- 3 Calculate λ_{init} of \tilde{P}_k and \tilde{P}_{k-1} ;
- 4 **if** $\lambda_{\text{init}} < \lambda_{\text{thr}}$
- 5 Match \tilde{P}_k and \tilde{P}_{k-1} by Super4PCS coarse matching to get the transformation matrix $\tilde{T}_{(k,k-1)}^L$;
- 6 **end if**
- 7 Downsample \tilde{P}_k to get \hat{P}_k ;
- 8 Match \hat{P}_k with \hat{P}_{k-1} by ICP fine matching to obtain $\hat{T}_{(k,k-1)}^L$ with initial transformation matrix $\tilde{T}_{(k,k-1)}^L$;
- 9 $\tilde{T}_k^W \leftarrow \tilde{T}_{k-1}^W \cdot \hat{T}_{(k,k-1)}^L$;
- 10 **Return** $\tilde{P}_k, \hat{P}_k, \tilde{T}_k^W$;

5.2 The correction node

The localization algorithm by matching two adjacent frames may inevitably produce cumulative errors, which will grow over gradually with time. In view of that, to make the algorithm effective for long-term navigation tasks, the so-called correction node is introduced in this paper in order to correct the location information periodically with selected key-frames using NDT matching. By doing so, the traditional single localization-node in odometry part of SLAM system is extended to a new two-node structure consist of both localization and correction node, which can further improve the accuracy of the odometry. Furthermore, a dynamic voxel grid map is proposed to speed up the matching process of the NDT algorithm and make it more suitable for real-time localization applications.

In the correction node, key frames are decided in a very simple but yet effective way. That is, a key frame is selected every five frames. No other sophisticated approach such as calculating displacements or over-lapping ratio between frames is used. The reason is given as below. Because the localization node is updated at 5 Hz, the key frame in the cor-

rection node is selected nearly at 1 Hz. Noted that a LIDAR sensor mounted on an unmanned vehicle is often omnidirectional with a wide range of over 100 meters, the displacement between two adjacent key frames is less than 30 meters even if the vehicle travels at a very high speed 100 km/h. That means key frames are not far from each other so that the registration accuracy can be guaranteed even in real-time applications. For other scenes with different LIDARs or vehicle speeds, similar strategy on how to select key-frames can also be made.

When a key-frame is selected, the correction node uses NDT to match the key-frame with the dynamic local sub-map, and the output of the localization node is thus corrected by directly replaced by the resulting transformation. Magnusson et al. (2009) proved that NDT is better than ICP on both robustness and accuracy with a large number of experiments. In addition, the NDT algorithm has a good performance of real-time and stability, and has a good adaptability to different environments. Therefore, NDT is adopted in the correction node to match the current key-frame with the local sub-map.

The NDT algorithm divides the space occupied by reference point cloud into voxel grid. Then the probability density function (PDF) describing the voxel is calculated according to the distribution of scan data dropped in each voxel. In this paper a dynamic voxel grid map is introduced, which can be updated dynamically with newly upcoming points. The map supports dynamic insertion and deletion of points with a HashMap storage structure of voxels. Each voxel contains several attributions including index, centroid (mean), covariance matrix, points and the number of points. Only voxels contain more than a certain number of points are considered as active and used in our correction algorithm. In order to prevent collision, the voxel index $I(q)$ in which a point q lies in is calculated as:

$$I(q) = \lfloor t.x \rfloor + \lfloor t.y \rfloor \ll h_{bits} + \lfloor t.z \rfloor \ll 2h_{bits}, \quad (6)$$

where \ll represents the left shift operator; $t = T_{mg}^{-1} \cdot q \cdot r_{es}^{-1}$, T_{mg} represents the transformation from the current coordinate system to the map coordinate system; r_{es} represents the grid resolution. The index $I(q)$ is used as the key-value of the HashMap and all points are stored in their corresponding HashMap. So that only the mean and covariance matrix of the voxels with newly inserted points need to be recalculated in each iteration. When n points $q_i (i = 1, 2, \dots, n)$ are inserted into a voxel already having m points, the mean value μ of the voxel will be updated as:

$$\mu \leftarrow \left(m \cdot \mu + \sum_{i=1}^n q_i \right) \cdot \frac{1}{n+m}, \quad m \leftarrow n+m, \quad (7)$$

and the covariance matrix is recalculated as:

$$\Sigma = \frac{1}{m-1} \sum_i (q_i - \mu)(q_i - \mu)^T, \quad (8)$$

As the vehicle moves, the dynamic grid map will delete voxels which are outside the radius of the map to reduce the storage memory. Moreover, the eliminated points of ground planes as above mentioned are restored to compensate the accumulated error of the odometer in height. The correction algorithm is concluded as Algorithm 3.

Algorithm 3: The Mapping Algorithm

input : $P_k, M_{j-1}, T_{j-1}^W, \tilde{T}_k^W$

output : M_j, T_k^W, T_j^W

- 1 $P_j \leftarrow P_k$;
 - 2 Downsample P_j to get \hat{P}_j ;
 - 3 Match \hat{P}_j with M_{j-1} by NDT matching method to obtain T_j^W , and \tilde{T}_k^W is the initial transformation matrix;
 - 4 Use \hat{P}_j to update the dynamic local map M_j ;
 - 5 **Return** M_j, T_k^W, T_j^W ;
-

In the above table, the letter j is used to represent the j -th update of the keyframes, M_j is the local sub-map after the j -th update. The inputs of the algorithm are the current frame of point cloud P_k , the current local sub-map M_{j-1} , the pose transformation of the last key-frame T_{j-1}^W , and the current output of the localization node \tilde{T}_k^W . The algorithm collects key-frames at the frequency of 1 Hz, and \hat{P}_j is obtained by down-sampling the current key-frame P_j . Then the current key-frame is matched with the local sub-map using NDT algorithm to obtain T_j^W with initial transformation \tilde{T}_k^W from the localization node. After that, the algorithm updates the local sub-map with \hat{P}_j , and maintains a fixed-size local dynamic voxel grid map. Finally, the algorithm outputs M_j , T_k^W and T_j^W for the next iteration.

Figure 8 shows the process of the correction. Suppose in the k -th scan we got the point cloud P_k (colored in red), the local map M_j (colored in black). Our goal is to find the transformation T_k^W , and to complete the matching of P_k and M_j . The matching result is the green part in Fig. 8.

It should be noted that when a key-frame M_j is selected and corrected by NDT matching with the previous key-frame M_{j-1} , the normal frames between these two key-frames are no need to be corrected with linear interpolation techniques, because our algorithm is used for real-time applications so that only the key-frame should be corrected and updated.

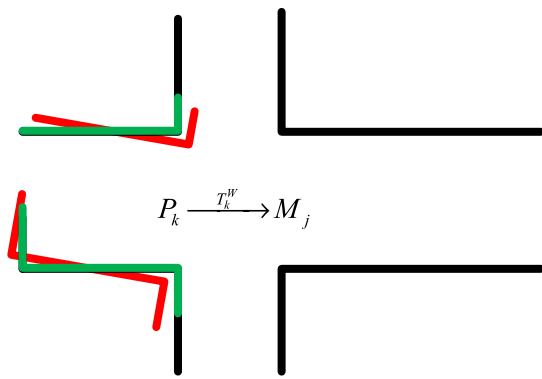


Fig. 8 The correction node uses NDT to match the current key-frame with the local sub-map. The black is the built local map, the red is the source point cloud, and the green is the resulting point cloud (Color figure online)

6 Loop closure

For loop closure in back end, a heuristic location-based detection method is introduced in this paper, that is, using a 3D-Kdtree to save historical locations (namely poses here), and then searching for neighbor historical locations in the Kdtree for currently estimated localization, so that a loop candidate is thought to be detected if the distance between these two locations is less than a threshold r_{th} . It should be noted that though the measures used here is simple (yet effective) and mainly constructed to keep the loop closure node consistent with the proposed two-node localization-correction odometry module, they are more like realization (or tricks) rather than theoretical contributions. However, the implementation of the loop closure is of great significance to the integrity and practical application of the proposed S4-SLAM algorithm, and given as follows in details.

The process is shown in Fig. 9. Considering the real-time performance, the difference of frame indexes between the selected historical location and the currently estimated localization should be greater than a threshold n_{th} , which can avoid a large number of unnecessary verification processes. The historical location satisfying the above conditions will be marked as a candidate loop.

Moreover, the overlap rate (namely LCP) is used to verify candidate loops, which can improve the accuracy and recall rate of the algorithm, and reduces the probability of false positives. The Super4PCS algorithm is used to match two frames under a candidate loop in turn. When the overlap rate is greater than a threshold λ_{loop_th} , the candidate loop will be determined to be a real loop. Finally the ICP algorithm is used to match the loops and terminate the iteration.

Our method can still achieve good results in the case of large displacements and low-overlap of point clouds because the Super4PCS algorithm is a global matching algorithm, and the verification process can reduce the probability of

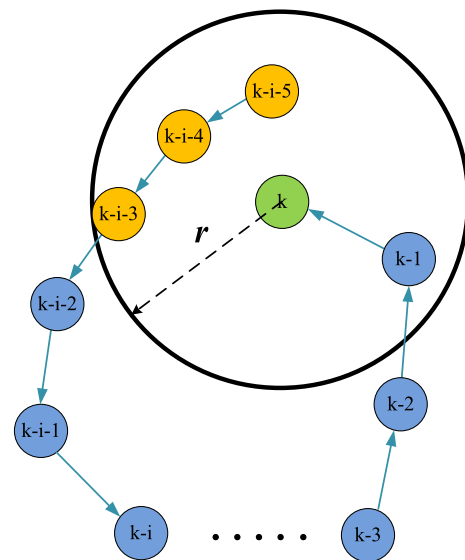


Fig. 9 Loop detection on the pose graph via a location-based strategy. For example, the current frame index is k , and r is the radius to search for the historical locations. Three frames with index $k-i-5$, $k-i-4$, $k-i-3$ are candidate loops, while the frame with index $k-1$ should be excluded

false positives. It could work well when the accumulative drift is less than r_{th} , but the performance depends heavily on the selection of r_{th} . If r_{th} is too large, many unnecessary verifications may be carried out, else if r_{th} is too small, it may not be able to detect a real loop. After a lot of experiments, our paper takes $r_{th} = 20$ m. If GPS data is available, it could be useful to assist the above process in a more effective way.

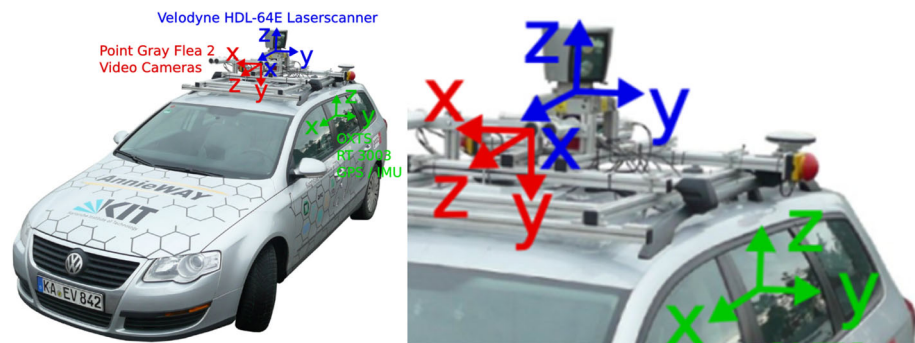
Based on the pose graph, an optimization step is carried out to close the detected loop in a similar way to Latif et al. (2013). Nodes are set to poses of key-frames, and an edge is the motion estimation between two nodes. The optimization problem is transformed into a least squares problem as follows:

$$\min_{\xi} \frac{1}{2} \sum_{i,j \in \varepsilon} e_{ij}^T \sum_{ij}^{-1} e_{ij}, \tag{9}$$

where ε is the set of all edges; ξ is the optimization variable (the poses of all nodes); e_{ij} is the error function of poses; and \sum_{ij}^{-1} is the information matrix.

It should also be noted that the loop closure algorithm only optimizes the key-frames, and normal frames between key-frames do not participate in the optimization process. In this paper, the linear interpolation technique is used to obtain the pose information of the normal frames between two adjacent key-frames. The interpolation scaling can be calculated as the scaling of current pose estimations between these frames with the output of the localization node and the correction node, or simply set to the equal values.

Fig. 10 The KITTI platform is equipped with a Velodyne HDL-64E LIDAR, two stereo HDL-64E LIDAR, two stereo cameras, and a high-precision GPS/INS for ground truth acquisition



7 Experimental results

In this section, our proposed S4-SLAM method is tested extensively on KITTI dataset, and real-life outdoor experiments like campus and harbor with a ground mobile robot and an unmanned surface vehicle respectively. All algorithms are realized in the ROS(Robot Operation System) environment on a PC platform with Intel (R) Core (TM) i5-3210 3.0 GHz dual-core CPU and 16GDDR3 RAM. In experiments, we use the same criterion namely the drift ratio as the KITTI server, which is calculated as following steps: (1) Calculate drifts every 10 frames, and find frames that are 100 m, 200 m, ..., 800 m away from the current frame according to ground truth; (2) Find the same numbered frames in the test dataset, and calculate pose transformations between these frames and the current frame; (3) Compare these transformation to true pose transformations to obtain the translational error $t_e(x_e, y_e, z_e)$. The drift ratio can be calculated as:

$$\text{Drift} = \frac{\|t_e\|}{L} \times 100\%, \quad (10)$$

where L is the displacement distance of the vehicle.

7.1 Tests on the KITTI datasets

The KITTI dataset was established by the German Karlsruhe Institute of Technology and the United States Institute of Technical Research (Geiger et al. 2012; Geiger et al. 2013). It is currently the largest computer vision algorithm evaluation dataset for automatic driving in the world. The data collection platform for the KITTI dataset is assembled with 2 grayscale cameras, 2 color cameras, a Velodyne HDL-64E LIDAR, 4 optical lenses, and a GPS/INS navigation system (See Fig. 10). Our method uses data from the Velodyne LIDAR only. The LIDAR runs at 10 Hz frequency and has 360° horizontal view field, 26.8° vertical view field and 120 m measuring range. The ground truth used in the experiment is obtained by a high-precision GPS/INS navigation system with drift less than 10 cm. The odometry benchmark

Table 1 Effects of ground and dynamic objects removal

Ground and dynamic objects removal	Drift (%)
With	0.93
Without	1.56

consists of 22 sequences of data which contain grayscale images, color images, and laser point cloud data. For the sake of fairness, only Sequence 00-10 were provided ground truth, and Sequence 11-21 without ground truth were used for the website to evaluate algorithms. From all test sequences, the evaluation of the website computes translational and rotational errors for all possible subsequences of length (100, ..., 800) meters, and the server ranks algorithms according to the results of the evaluation. The average drift of our algorithm is under 1%.

7.1.1 Verification for our improved Super4PCS

In experiments, Sequences 00-10 of KITTI dataset are used to test our improved Super4PCS algorithm on both accuracy and real-time performances.

Table 1 shows the accuracy improvement by filtering out ground and dynamic objects. It can be seen that the new algorithm can reduce the drift from 1.56 to 0.93%, that means over 40% promotion on accuracy.

As shown in Table 2, the matching speeds, translation errors and rotation errors between the improved Super4PCS algorithm and the original Super4PCS algorithm are compared with Sequences 01, 03, 05, 07 and 09. The average time consumed by the improved Super4PCS is about 0.1 s, which is over 60% faster than the original Super4PCS algorithm. For accuracy comparison, according to KITTI's suggestion, the translation error and the rotation error are calculated every two frames (the actual driving distance is about 3 m) on each sequence, and the average is finally taken. For each two frames $\text{Frame}(i)$ and $\text{Frame}(i+3)$, calculate the translation matrix between two algorithm results as T , that is, $\text{Frame}(i+$

Table 2 Comparison of time costs, translation and rotation errors between the improved Super4pcs and the original Super4pcs

Sequence	Number of Frames	Time Cost (ms)		Translation Error (m)		Rotation Error (rad)	
		Original	Improved	Original	Improved	Original	Improved
01	1101	390	92	0.83	0.55	0.029	0.023
03	801	302	105	0.65	0.43	0.051	0.036
05	2761	346	101	0.76	0.36	0.021	0.016
07	1101	384	95	0.74	0.42	0.029	0.021
09	1591	376	98	0.81	0.46	0.067	0.047

Better results among the comparison of different methods are given in bold

3) = $T * \text{Frame}(i)$. Set the true transformation as T_g , we can have

$$T^{-1}T_g = \begin{pmatrix} a & * & * & dx \\ * & b & * & dy \\ * & * & c & dz \\ * & * & * & 1 \end{pmatrix} \tag{11}$$

The translation error and the rotation error between two frames are then calculated as

$$\varepsilon_{\text{translation}} = \sqrt{dx^2 + dy^2 + dz^2} \tag{12}$$

$$\varepsilon_{\text{rotation}} = \cos^{-1} \left(\max \left(\min \left(\frac{a+b+c-1}{2}, 1 \right), -1 \right) \right) \tag{13}$$

In Table 2 the translation errors of the improved Super4PCS are over 30% smaller than the original error, while the rotation errors are over 15% smaller. It can be concluded that the improvements of Super4PCS not only speed up the matching process, but also improve the precision, which enable the algorithm to run in scenes with high real-time and accuracy requirements.

7.1.2 Positioning and mapping experiments on the KITTI dataset

Table 3 shows the average time-consuming of each algorithm (including ground and dynamic objects filtering, Super4PCS, ICP, NDT) used in the S4-SLAM system. The point cloud data in KITTI dataset was de-skewed with an external odometry, so we don't need to consider the effects of motion distortion. Finally, the system outputs positioning information at a frequency of about 5 Hz, and outputs mapping information at a frequency of about 1 Hz. Since the operating frequency of the LIDAR in the KITTI dataset is 10 Hz, therefore, our system runs at half the original frequency.

The KITTI datasets mainly contain three types of environments: “urban” with buildings around, “country” on small roads with vegetations in the scene, and “highway” where roads are wide and the surrounding environment is relatively

Table 3 Time-consuming results of the algorithms in S4-SLAM system

Algorithm	Time Cost (ms)
Ground and dynamic objects filtering	26
Super4PCS	102
ICP	78
NDT	155

clean. Among them, the KITTI data acquisition platform has a slower speed in urban and country environments and a faster speed on in highway (the highest moving speed is 85 km/h) environments. In order to test the performance of S4-SLAM under different environments and conditions, especially with high speed, large displacement and low overlapping of point clouds, the above datasets are sampled for test at intervals of 2, 3 and 4 frames respectively. Table 4 shows the drift and the time cost of the algorithm, and the average drifts are separately 1.03%, 1.31%, and 1.43%, which is a little larger than the original result 0.93% in Table 2. Moreover, time costs for each frame are separately 233 ms, 245 ms, 269 ms, which can basically meet the requirements of real-time applications. It can be concluded that our S4-SLAM can maintain good robustness in cases of high speeds, large displacements and low overlapping of point clouds.

Figure 11 shows the trajectory comparison between our algorithm and the ground truth in sequence 00, 05, 06, 07, 08, 09, it can be seen that all trajectories can basically coincides with the ground truth, which proves that the S4-SLAM system can meet the crucial requirement of long-distance accurate localization for outdoor vehicles.

Figure 12 shows mapping results of S4-SLAM with datasets collected from different on-road ground environments. Among them, Fig. 12a shows the map built in a city block, Fig. 12b shows the map that is built on a country road, and Fig. 12c shows the map that is built on a field highway road. From the above results, it can be seen that the established 3D point cloud map is consistent with the actual environment.

The localization and mapping experiments from Figs. 11 to 12 show that the proposed S4-SLAM algorithm has good environmental adaptability and can basically meets the

Table 4 Time cost and drift of our method

Sequence	2 Frames interval		3 Frames interval		4 Frames interval	
	Drift (%)	Time cost (ms)	Drift (%)	Time cost (ms)	Drift (%)	Time Cost (ms)
00–10	1.03	233	1.31	245	1.43	269

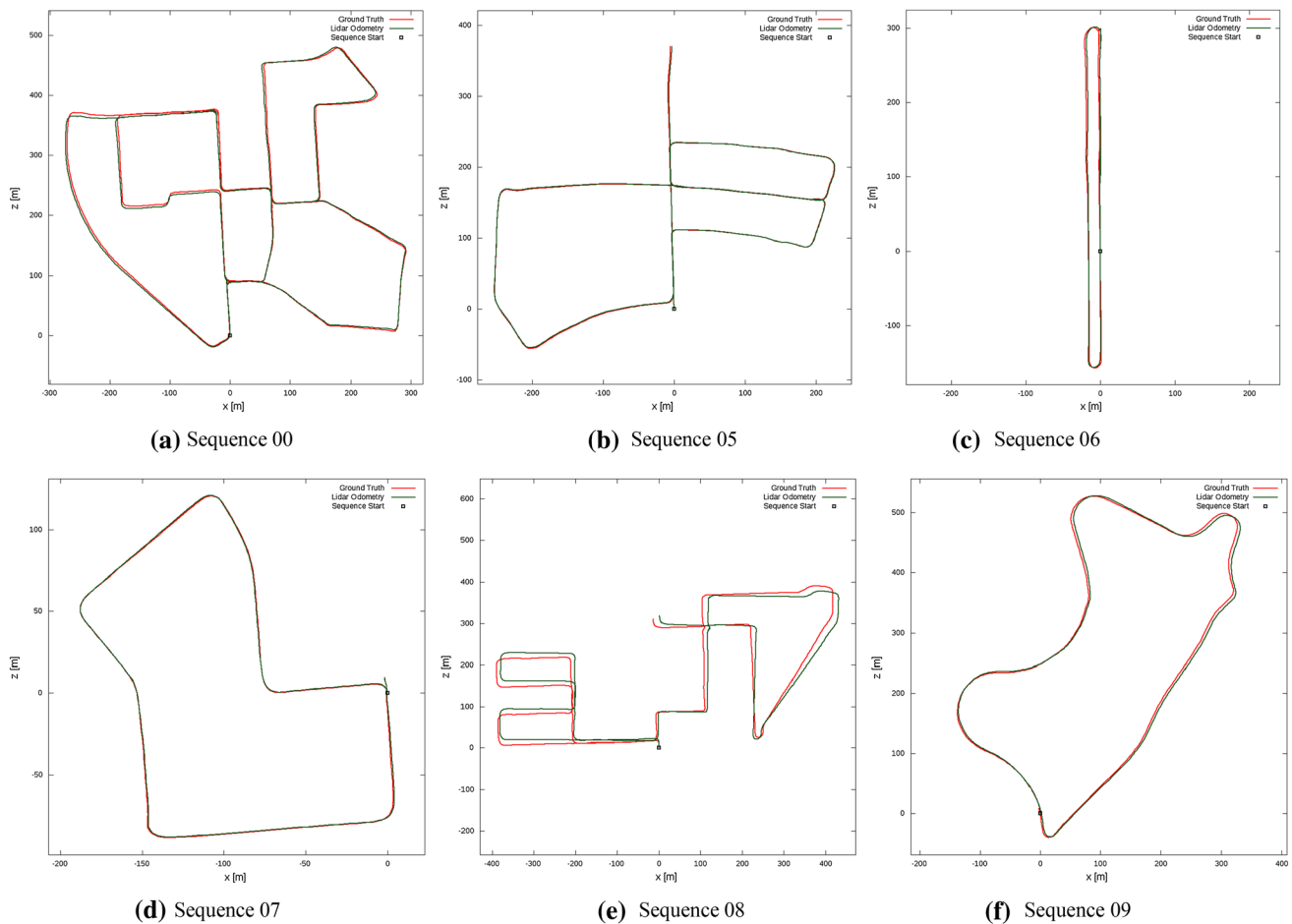


Fig. 11 The trajectory comparison between the results of our algorithm and the groundtruth on the KITTI training dataset. The red trajectory is the groundtruth and the green trajectory is our results, we successfully

detect the real loops in sequence 00, 05, 06, 07, 09, but fail in sequence 08, because the cumulative drift is greater than the detection radius r_{th} in sequence 08 (Color figure online)

requirements of real-time location and mapping for outdoor ground vehicles even with high driving speed.

Furthermore, our method is also compared with the top rank LOAM algorithm on KITTI, as Table 5 shows, in which results of LOAM are directly adopted from (Zhang and Singh 2017). LOAM has been extensively recognized as the state-of-the-art LIDAR-only outdoor SLAM approach in the KITTI benchmark, which makes it suitable and typical for comparison with important reference significance. It can be seen that our algorithm outperform LOAM in some sequences such as 00, 01, 03, 05 and 07. The results show that the performance of S4-SLAM is generally comparable to LOAM, though there are still some differences between

them. In constructed environments like urban and highway, as buildings often produce less noises and outliers, S4-SLAM performs even better than LOAM. On the other hand, as for country environments with many trees and brushes which may produce more noises and outliers, S4-SLAM is affected somehow and performs not as well as LOAM. It also shows that a good pre-processing filter to remove noises and outliers is important for S4-SLAM, because less mismatch can be achieved.

As for real-time property, the updating frequency of LOAM is 10 Hz, while S4-SLAM is 5 Hz. However, though slower than LOAM, S4-SLAM can still keep effective in more challenging environments, such as

Fig. 12 The mapping results for different scenarios in KITTI datasets, **a** is in a city block (Sequence 07), **b** is on a small road with many vegetations (Sequence 09), and **c** is on a highway (Sequence 01)

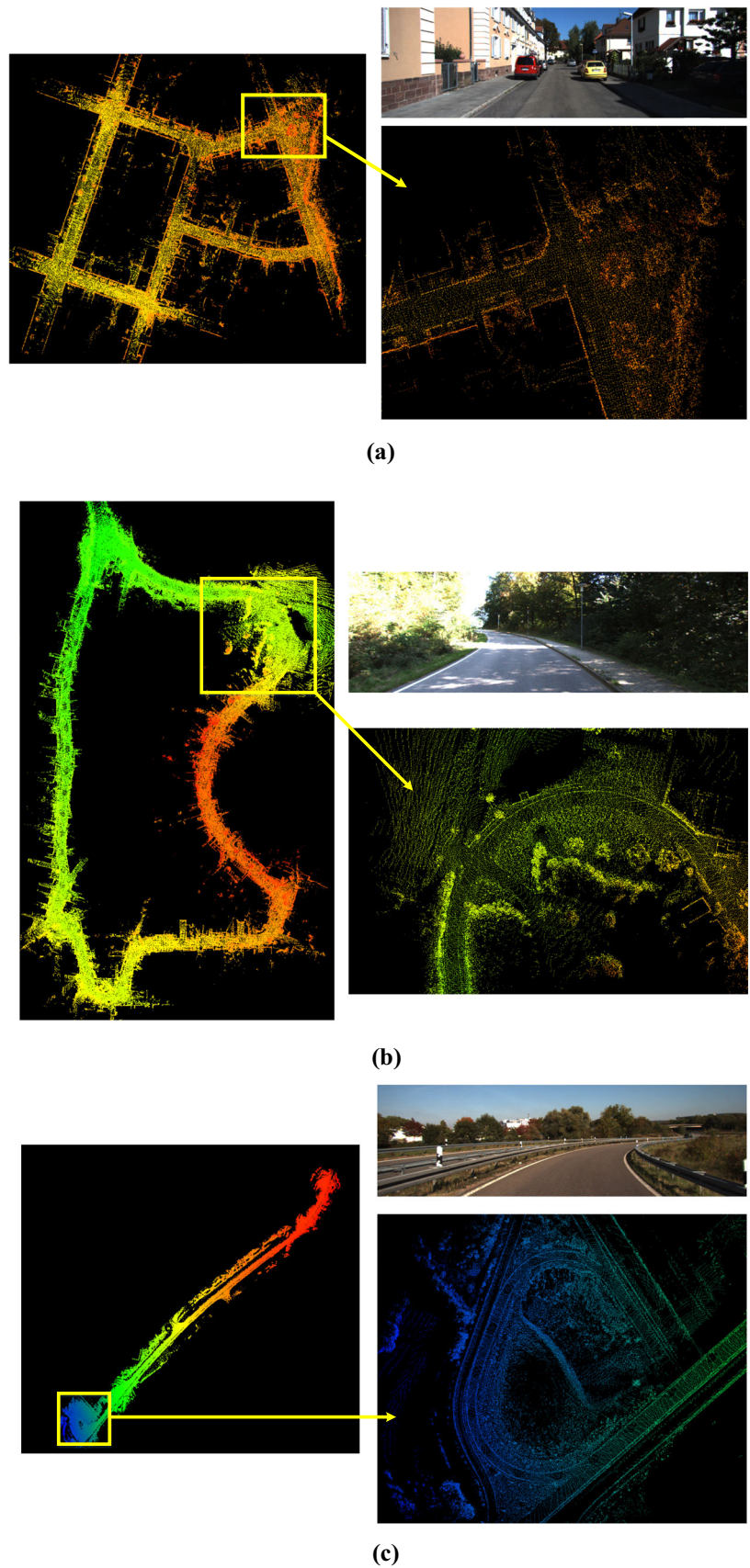


Table 5 Comparison of drift between our SLAM method and LOAM

Sequence	Environment	Drift (%)	
		Our SLAM	LOAM
00	Urban	0.62	0.78
01	Highway	1.11	1.43
02	Urban + Country	1.63	0.92
03	Country	0.82	0.86
04	Country	0.95	0.71
05	Urban	0.50	0.57
06	Urban	0.65	0.65
07	Urban	0.60	0.63
08	Urban + Country	1.33	1.12
09	Urban + Country	1.05	0.77
10	Urban + Country	0.96	0.79

Better results among the comparison of different methods are given in bold

ground/watersurface multi-scene applications, as the latter experimental results in a harbor shows, which indicates S4-SLAM has better robustness compared to LOAM.

7.2 Outdoor tests

In order to test the proposed S4-SLAM method with real-life applications, a ground wheeled mobile robot (namely a UGV) and an unmanned surface vehicle (namely a USV) are used respectively for experiments in a campus off-road ground environment and a ground/watersurface combined harbor environment without GPS or IMU informations. Both the UGV and USV platforms are equipped with a Hesai Pandar40 LIDAR for acquiring point cloud data by perceptions of environments. The Hesai Pandar40 is a hybrid solid-state LIDAR sensor with 40 scanning lines, 10 Hz scanning frequency and 150 m scanning range.

7.2.1 SEU(Southeast University) campus tests

Firstly an outdoor off-road ground experiment in the campus of Southeast University is carried out. The UGV platform used in this experiment is shown in Fig. 13a. It is equipped with a high-precision IMU, a differential GPS and a Hesai Pandar40 LIDAR. However, only LIDAR data is used in S4-SLAM, while the differential GPS and the IMU data can be used to provide the ground truth. Our aim is to test the LIDAR-only S4-SLAM algorithm in GPS-denied environments. Figure 13b shows the traveling trajectory of the robot. The end of the trajectory is set to the start point, so that the final trajectory forms a loop. The average moving speed of the mobile robot is about 3 m/s, and the total trajectory length is approximately 0.6 km. Figure 13c shows the actual scene of the SEU campus. There are a great number of structured

and unstructured features in the environment such as trees, buildings, pedestrians, vehicles and a big fountain filled in waters, which make it really a complicated environment. As shown in Fig. 13d, the drift of the front-end LIDAR odometry without loop closing is 0.89%, while the average drift of S4-SLAM with loop closure can be further reduced to 0.48%, which means both the odometry function and the correction function can output satisfactory localization result. The loop closure is very effective and useful in promoting the localization accuracy, though the real-time performance of the system is slightly reduced. The above results indicate our S4-SLAM algorithm can be used to meet different level of requirements in real-life applications. If high real-time performance is crucial, only the odometry function can work very well. Otherwise the S4-SLAM can provide more accurate outputs. Figure 13e shows the mapping result of S4-SLAM. Buildings, trees, the big fountain and other objects in the scene can be clearly seen in the figure. The established 3D point cloud map is highly consistent with the actual environment. The SEU campus test demonstrates the effectiveness and practicability of S4-SLAM in complicated ground environments with structured/unstructured terrains, as well as dynamic targets like fast moving pedestrians and vehicles.

7.3 Tests in harbor

In order to verify the robustness of the proposed method in scenes with sparse feature points, high moving speed and high dynamic noise, experiments are also carried out in a harbor, which means a ground/watersurface composite environment. In the experiment, the “Jinghai No. 3” unmanned ship platform is used, as shown in Fig. 14a. The platform is equipped with a high-precision IMU, a differential GPS, two cameras and a Hesai Pandar40 LIDAR. Again only LIDAR data is used in S4-SLAM, and the ground truth is obtained by the differential GPS and the high-precision IMU. Figure 14b, c shows the photos of the harbor. The distance between the unmanned ship and the shore is not more than 30 m. The average speed of the unmanned ship is about 10 m/s in the experiment, and the total length of the trajectory is about 3 km. Figure 14d shows the trajectory comparison of S4-SLAM with the ground truth for the first test, where the red line is the result of our algorithm, the blue line is the result of LOAM, and the black line is the ground truth. It can be seen that the two trajectories from S4-SLAM and LOAM are nearly coincide with the drift of 0.83%, while the accuracy of the LOAM degrades rapidly and produce large drift. Figure 14e shows the mapping results. The established 3D point cloud map of S4-SLAM is also highly consistent with the actual environment.

Moreover, other tests are carried out with different times and driving trajectories to show the effectiveness of S4-SLAM. Four of them are selected and illustrated in Fig. 15

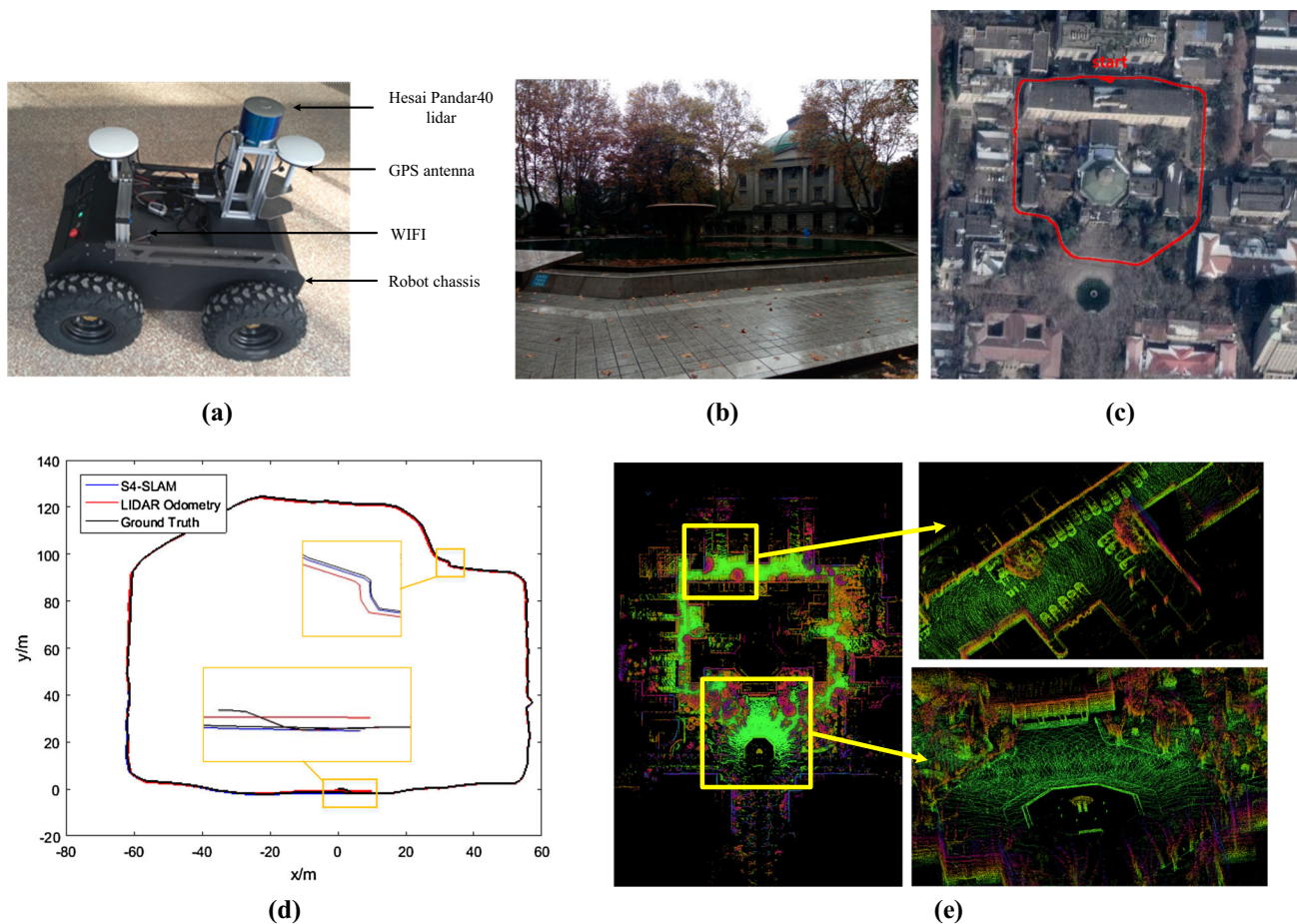


Fig. 13 Positioning and mapping experiments in campus, **a** is a photo of the mobile robot, **b** is a photo of our experimental environment, **c** is the aerial photo of the campus from Google Maps, **d** is the trajectory comparison between S4-SLAM, LIDAR odometry and ground truth, the trajectory of S4-SLAM and the ground truth are basically coinci-

dent, but the error of the LIDAR odometry is larger, and the trajectory of the odometry does not form a loop, **e** is the mapping result of the campus, and the map built by S4-SLAM are basically consistent with the actual environment

and analyzed. The global drifts of S4-SLAM and LOAM for these four sequences are compared as in Table 6. All parameters of LOAM are re-tuned in order to produce as good result as it can. It can be seen clearly that LOAM’s drifts increase with the traveling length, while S4-SLAM can track the ground truth with better accuracy. By checking the frame-to-frame registration results, it can be found out that not only orientation but also translation errors are accumulated gradually as the vehicle travels. This may be because the main idea of LOAM is to extract both edge and plane points to finish scan matching, which makes it works well in ground-only environments. However, in harbor environments, the lack of data on the watersurface makes plane features much less, so that the algorithm may be unable to register two frames very well with only edge points, as can be seen in Fig. 16a. It should also be noted that the high speed movements and the high noise

levels in this challenging environment, the latter mostly risen from the diffuse reflection on the watersurface, can lead to bad features or wrong matchings and further weakened performance of the LOAM algorithm. The registration results of LOAM can be seen in Fig. 16b, which show registration error clearly in top view, while Fig. 16c also shows a large bending error in the front view. Sometimes LOAM may even fail if its parameters are not tuned properly. However, the proposed S4-SLAM algorithm still performs very well with better registration results as well as much less drifts, as can be seen in Fig. (d) and (e). Similar results can also be observed under other frame-to-frame matchings in experiments.

From above experimental results it can be concluded that the performance of LOAM highly depends on the type of environment, and cannot be proved robust in different kinds of applications, especially in multi-scene environments such

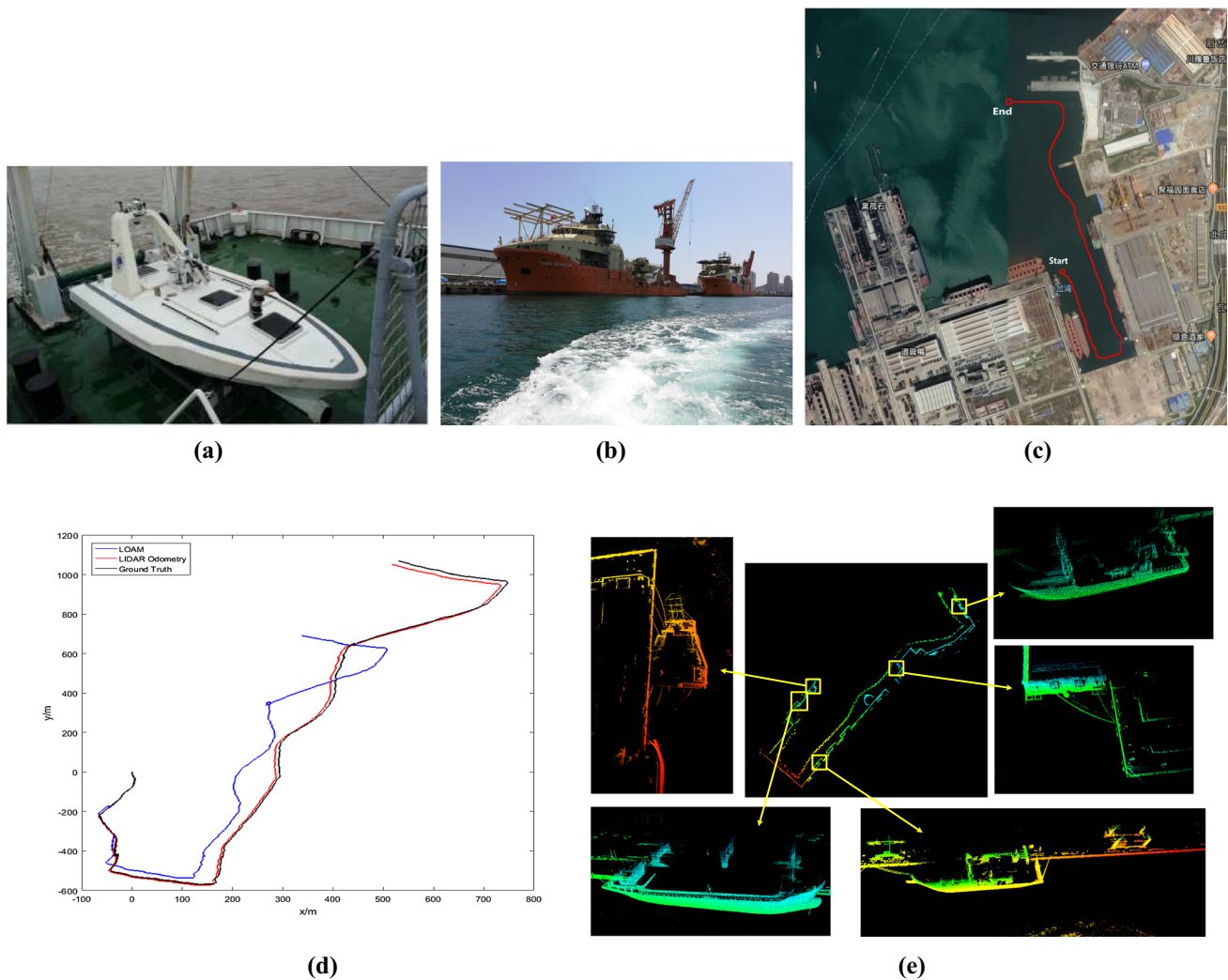


Fig. 14 Positioning and mapping experiments in harbor for test 01, **a** is a photo of the unmanned ship, **b** is a photo of our experimental environment, **c** is the aerial photo of the harbor from Google Maps, **d** is the

trajectory comparison between our algorithm, LOAM and ground truth, **e** is the mapping result of the harbor, the ship in the harbor is clearly visible in the map built by the proposed method

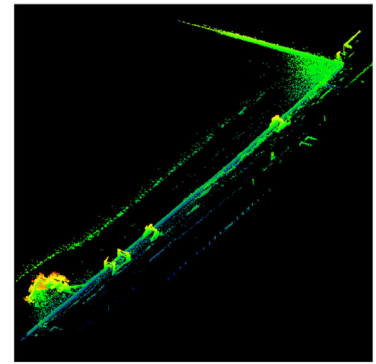
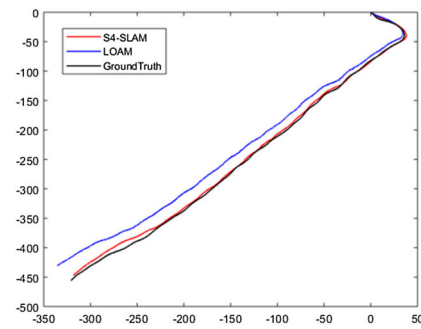
as ground/watersurface composite environments discussed in this paper. It works quite well in scenes with rich feature points and fewer disturbances or noises, but not as satisfying in scenes with sparse feature points, high noise levels that may lead to more registration errors or drifts. On the other hands, in S4-SLAM several algorithms including Super4PCS, ICP and NDT are incorporated so that a coarse-to-fine registration process is created to keep the whole framework stable and robust even in ground/watersurface composite environments with the global drift under 1%. The S4-SLAM method is proved to be an effective SLAM solution to more challenging environments. The core idea of S4-SLAM is to integrate the existing mainstream registration technologies (such as Super4PCS, ICP and NDT) to solve the

problem of registration with sparse feature points and high noise levels in challenging multi-scene environments mentioned above. In addition, improvements are also made to the existing SLAM framework to achieve an effective combination and hierarchical processing of different registration algorithms. The SLAM system constructed in this paper is accurate, robust to noises, and has balanced real-time performance even in complicated multi-scene environments.

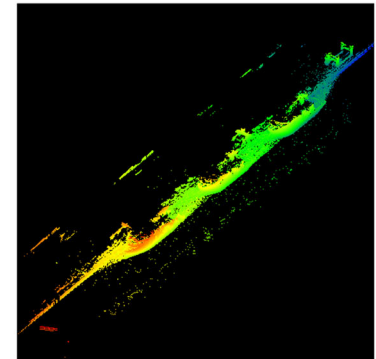
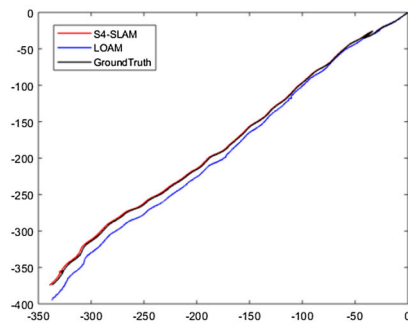
8 Conclusion and future work

In this paper, a novel real-time LIDAR SLAM system namely S4-SLAM is constructed to solve the problem of long-distance accurate localization and mapping of robots such as

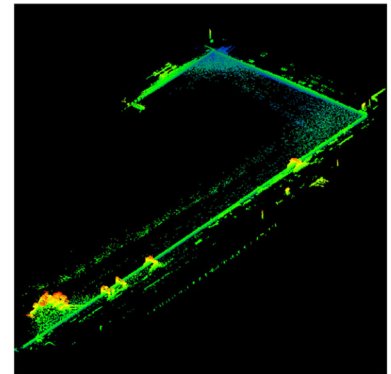
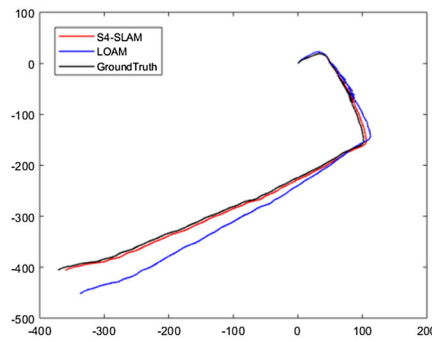
Fig. 15 Other four tests in the harbor. **a, b, c, d** are the results for test 02 to 05. The left figures are the localization results, where the black lines are the ground truth, and the red lines are the resulting trajectories of S4-SLAM. The right figures are the mapping results (Color figure online)



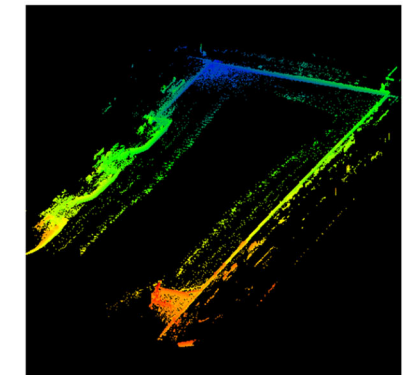
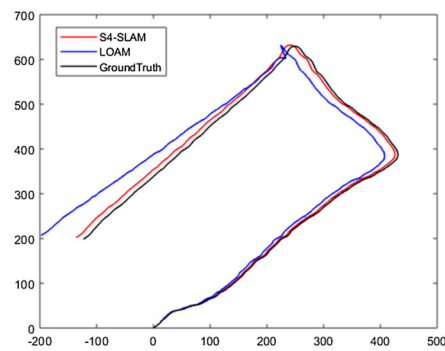
(a)



(b)



(c)



(d)

Table 6 Drifts of S4-SLAM vs LOAM for harbor tests

Test number		02	03	04	05
Drift (%)	S4-SLAM	0.54	0.49	0.76	0.93
	LOAM	5.54	6.14	8.81	11.48

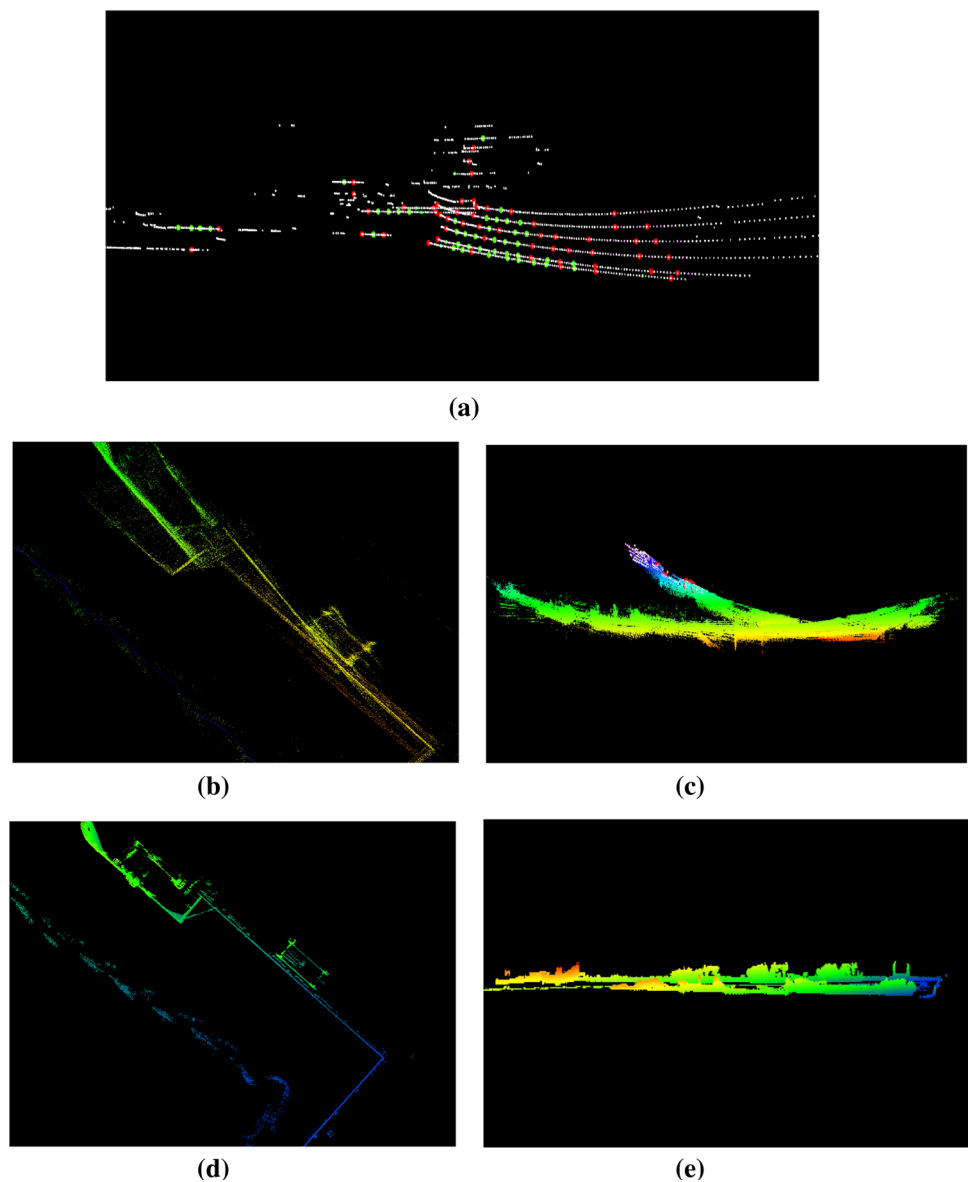
Better results among the comparison of different methods are given in bold

UGVs or USVs in ground/watersurface multi-scene applications. The system is composed of LIDAR odometry function and loop closure function. In the odometry function a new correction node is added to the traditional localization node so that pose outputs with different level of accuracy can be

produced at different frequency, and in the loop closure function loops is detected to optimize the map. Our system works well on both the KITTI datasets and real-life scenes including a campus and a harbor. It is proved to achieve satisfactory trade-off between real-time and accuracy, and keep robust even in cases of few feature points, high moving speed and high dynamic noises.

In our current system, the loop detection is a simple process carried out by a heuristic location-based method. Finding a real-time and more stable loop closure algorithm will be the focus of our future work. Other works include more water areas test including lakes, rivers and seaside, and the fusion of GPS and IMU data to current system in order to build a LIDAR SLAM system with better performances.

Fig. 16 Comparative analysis of LOAM and S4-SLAM. **a** shows features points which are extracted by LOAM from one frame; **b** shows the registration error of LOAM in top view; **c** shows the registration error of LOAM in front view; **d** shows the registration result of S4-SLAM in top view; **e** shows the registration result of S4-SLAM in front view



Acknowledgement This work is partly supported by the National Natural Science Foundation (NNSF) of China under the Grants Nos. 62073075, 61673254, U1613226, and 61573100.

References

- Aiger, D., Mitra, N. J., & Cohen-Or, D. (2008). 4-points congruent sets for robust pairwise surface registration. *ACM Transactions on Graphics (TOG)*, 27(3), 85.
- Almeida, J., & Santos, V. M. (2013). Real time egomotion of a non-holonomic vehicle using LIDAR measurements. *Journal of Field Robotics*, 30(1), 129–141.
- Bae, K. H. (2009). Evaluation of the convergence region of an automated registration method for 3D laser scanner point clouds. *Sensors*, 9, 355–375.
- Behley, J., & Stachniss, C. (2018). *Efficient Surfel-based SLAM using 3D laser range data in urban environments*. In Robotics: Science and Systems.
- Besl, P. J., & McKay, N. D. (1992). Method for registration of 3-D shapes. *Sensor Fusion IV: Control Paradigms and Data Structures*, 1611, 586–607.
- Biber, P., & Straßer, W. (2003). The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, USA.
- Bouaziz, S., Tagliasacchi, A., & Pauly, M. (2013). Sparse iterative closest point. In *Proceedings of the Eleventh Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, Oxford, UK.
- Ceriani, S., Sánchez, C., Taddei, P., Wolfart, E., & Sequeira, V. (2015). Pose interpolation slam for large maps using moving 3d sensors. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 750–757).
- da Silva, J. P., Borges, D. L., & de Barros Vidal, F. (2011). A dynamic approach for approximate pairwise alignment based on 4-points congruence sets of 3D points. In *2011 18th IEEE International Conference on Image Processing* (pp. 889–892).
- Deschaud, J. E. (2018, May). IMLS-SLAM: scan-to-model matching based on 3D data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2480–2485).
- 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).
- Dubé, R., Gollub, M. G., Sommer, H., Gilitschenski, I., Siegwart, R., Cadena, C., et al. (2018). Incremental-segment-based localization in 3-d point clouds. *IEEE Robotics and Automation Letters*, 3(3), 1832–1839.
- Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32, 1229–1235.
- 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* (pp. 3354–3361).
- Irani, S., & Raghavan, P. (1999). Combinatorial and experimental results for randomized point matching algorithms. *Computational Geometry*, 12(1–2), 17–31.
- Ji, K., Chen, H., Di, H., Gong, J., Xiong, G., Qi, J., & Yi, T. (2018). CPFGL-SLAM: a robust simultaneous localization and mapping based on LIDAR in off-road environment. In *2018 IEEE Intelligent Vehicles Symposium (IV)* (pp. 650–655).
- Kammel, S., Ziegler, J., Pitzer, B., Werling, M., Gindele, T., Jagzent, D., et al. (2008). Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9), 615–639.
- Koide, K., Miura, J., & Menegatti, E. (2018). A Portable 3D LIDAR-based system for long-term and wide-area people behavior measurement. *IEEE Transactions on Human-Machine Systems*.
- Latif, Y., Cadena, C., & Neira, J. (2013). Robust loop closing over time for pose graph SLAM. *The International Journal of Robotics Research*, 32(14), 1611–1626.
- Magnusson, M. (2009). The Three-Dimensional Normal-Distributions Transform: an Efficient Representation for Registration, Surface Analysis, and Loop Detection. *Renewable Energy*, 28(4), 655–663.
- Magnusson, M., Nuchter, A., Lorken, C., Lilienthal, A. J., & Hertzberg, J. (2009). Evaluation of 3D registration reliability and speed-A comparison of ICP and NDT. In *2009 IEEE International Conference on Robotics and Automation* (pp. 3907–3912).
- Mellado, N., Aiger, D., & Mitra, N. J. (2014). Super 4pcs fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 33(5), 205–215.
- Moosmann, F., & Stiller, C. (2011). Velodyne slam. In *2011 IEEE Intelligent Vehicles Symposium (IV)* (pp. 393–398).
- Pottmann, H., Huang, Q. X., Yang, Y. L., & Hu, S. M. (2006). Geometry and convergence analysis of algorithms for registration of 3D shapes. *International Journal of Computer Vision*, 67, 277–296.
- 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)* (pp. 4758–4765).
- Viejo, D., & Cazorla, M. (2014). A robust and fast method for 6DoF motion estimation from generalized 3D data. *Autonomous Robots*, 36(4), 295–308.
- Yang, J., Li, H., Campbell, D., & Jia, Y. (2016). Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11), 2241–2254.
- Zhang, J., & Singh, S. (2014). LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems*, 2(9).
- 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).
- Zhang, J., & Singh, S. (2017). Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2), 401–416.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Bo Zhou is an Associate Professor in the School of Automation, Southeast University in China. He has received the B.S. degree in the School of Automation, University of Science and Technology of China, Hefei, China, in 2003, and Ph.D. degree from Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, in 2009. His research interests include autonomous and semi-autonomous systems, with a special emphasis on indoor and field robotics, covering the areas of sensor fusion, robot localization, environment perception and mapping, path planning, motion decision modeling, and large scale robotic applications.



Yi He is a master student in the School of Automation, Southeast University in China. His research interest is focused on robot localization, environment perception and mapping.



Xudong Ma is a Professor at the Institute of Intelligent Robot and Intelligent Control, School of Automation, Southeast University. He received his M.S. degree from Tsinghua University, Beijing, China, in 1988. His research interest covers computer control and management system, intelligent robot control, embedded controller and DSP system application.



Kun Qian is an Associate Professor at the Institute of Intelligent Robot and Intelligent Control, School of Automation, Southeast University. He received his Ph. D. degree from School of Automation, Southeast University, Nanjing, China, in 2010. His research interest covers intelligent robot system and computer vision.



Xiaomao Li received the Ph.D. degree in Pattern Recognition and Intelligent System from Shenyang Institute of Automation Chinese Academy of Sciences, (Shenyang, China), in 2008. He worked at Nanjing Research Institute of Electronic Technology doing research on radar data processing from 2008 until 2014(Nanjing, China). Since 2015, he joined the Research Institute of USV Engineering of Shanghai University(Shanghai, China).