Chapter 10
Robust and Fast Registration for Lidar Odometry and Mapping

Wenbo Liu and Wei Sun

Abstract Outliers, such as sensor noise, abnormal measurements, or dynamic objects, can damage the overall accuracy of a Simultaneous Localization and Mapping (SLAM) system. Aiming at to improve the performance of Lidar SLAM systems in urban scenes containing a large number of outliers, we propose a real-time, feature-based, and outliers-rejection Lidar SLAM system. By embedding an outlier elimination method based on 4-points congruent sets into a state-of-the-art SLAM framework and further optimizing the traditional single-step registration to coarse-to-fine registration, we can solve the problem of time-consuming, high motion drift, and wrong mapping caused by the current Lidar SLAM systems which cannot effectively detect and eliminate the outliers in surrounding environment.

10.1 Introduction

Simultaneous Localization and Mapping (SLAM) is the most basic prerequisite of intelligent robots and the necessary ability of driverless vehicles. Although there are many accurate and effective solutions to SLAM problems, such as the methods based on vision [1] or Lidar [2, 3], or integrating their advantages [4], most of the existing methods [5, 6], including them, are based on the assumption of a static world, which greatly limits the scope of application of these excellent methods. The typical method to estimate Lidar ego-motion is to apply the iterative closest point (ICP) method to the point clouds of adjacent frames [7]. However, Lidar-based methods can work even in the dark, and many 3D Lidars can capture the details of the environment over a long distance [6]. Therefore, this work focuses on the use of 3D Lidar to deal with the problem of SLAM in urban environment containing complex outliers.

W. Liu · W. Sun (✉)
Xidian University, Xi'an, China
e-mail: wsun@xidian.edu.cn

T.-Y. Wu et al. (eds.), *Advances in Smart Vehicular Technology, Transportation, Communication and Applications*, Smart Innovation, Systems and Technologies 250,
https://doi.org/10.1007/978-981-16-4039-1_10

The main contributions of this paper are as follows:

- We integrate a robust outliers elimination method based on 4-point congruence set into a state-of-the-art SLAM framework to further improve the accuracy and robustness of odometry and mapping.
- We use a reasonable coarse-to-fine registration method to replace the traditional single-step registration method to further reduce computation of the registration part, so as to reduce total runtime consumption of the system.

The rest of the paper is organized as follows: we introduce the related work in Sect. 10.2, describe the proposed method in Sect. 10.3, compare our method with the most advanced method in Sect. 10.4, and summarize our work and put forward the prospect in Sect. 10.5.

10.2 Related Work

10.2.1 Point Cloud Registration

The overlap rate of two point clouds to be registered is a key parameter, which is usually inversely proportional to the number of outliers. If two point clouds have a large overlap rate or obvious point correspondence, then their registration can be achieved by the iterative closest point [8] and various improved algorithms without initialization. ICP will fall into the local optimal solution. Go-ICP [9] and Gogma [10] use branch and bound in six-dimensional space to achieve global registration of point clouds without corresponding points. Because branch and bound method has exponential complexity, these methods are computationally expensive, and they often diverge due to small overlap [15].

The common rough registration methods are divided into hypothesis, test methods are represented by Random Sample Consensus (RANSAC), and geometric feature-based methods are represented by 4PCS [11]. RANSAC and its improved algorithm have cubic complexity and are not suitable for large-scale point clouds. Mellado et al. [13] reduced the complexity to quadratic and linear successively by matching the congruent four-point sets in two point clouds. Theiler et al. [12] use the set of key points for matching, which reduces the size of the set to be searched. Mohamad et al. [14] generalize the construction of coplanar four-point basis, which further improves the efficiency of point cloud registration. Raposo and Barreto [15] only use the topological relationship and normal vector between two points to construct matching rules, which greatly improves the registration efficiency and is suitable for point clouds with smaller overlap ratio. On the basis of the initial solution of coarse registration, precise registration is used to locally minimize the nonconvex error function to obtain the exact solution. Shan and Englot [6] suggest searching rotation and translation separately to reduce the registration complexity.

There are a lot of related works, but they either cannot fully meet the requirements of real-time, robustness, and high-precision in Lidar SLAM or have not carried out systematic experiments on Lidar point cloud dataset.

10.2.2 Lidar Odometry and Mapping

LOAM [2], rank second in the odometry evaluation project of KITTI vision benchmark [16], has a very low drift in the short-trajectory scenarios, and LOAM has greatly promoted the development of Lidar SLAM field. However, LOAM has no loop closure detection to eliminate the continuous accumulation of errors, so it may produce significant errors in the case of long trajectory. More importantly, LOAM does not consider dynamic objects, resulting in incorrect results in dynamic scenes. LeGo-LOAM [6] optimizes LOAM by adding loop closure detection to reduce motion drift, adding clustering and segmentation modules to filter out clusters with less than 30 points to realize filtering of noise. LOL [17] improves odometry accuracy by detecting the geometric similarity between online 3D point clouds and prior offline maps.

10.3 Proposed System

10.3.1 Task Description and Definitions

The problem addressed in this work is how to use the point cloud observed by 3D Lidar to estimate its ego-motion and build a global map for the traversed environment.

We define the point cloud measured as P, Lidar pose as X, global map as M, use the upper right corner to represent the time stamp and define the K-scan point cloud as $\boldsymbol{P^k}$, where a point is p_i^k, that is, $\boldsymbol{P^k} = \left(p_1^k, p_2^k, \ldots, p_n^k\right)$, $p_i^k = \left(x_i^k, y_i^k, z_i^k\right)$, where n is the size of the point cloud, $k \in \mathbb{N}^+$.

Under the above definition, the problem can be modeled as: Given map M^{t-1}, Lidar pose X^{t-1} at time $t-1$, and point cloud $\boldsymbol{P^t}$ at time t, to update X^t and M^t, as can be shown in (10.1) and (10.2).

$$X^t = \mathbb{F}\left(X^{t-1}, \boldsymbol{P^t}\right) = \mathbb{F}\left(X^{t-1}, \mathbb{W}\left(\boldsymbol{P^{*t}}, W^t\right)\right) \tag{10.1}$$

$$M^t = \mathbb{G}\left(M^{t-1}, \boldsymbol{P^t}\right) = \mathbb{G}\left(M^{t-1}, \mathbb{W}(\boldsymbol{P^{*t}}, W^t)\right) \tag{10.2}$$

where P^{*t} is the description of the real scene in the field of vision at time t. $W^t = (w_1^t, w_2^t, \ldots, w_i^t)$ is the sum of many interference factors such as sensor noise, abnormal measurement values, or dynamic objects. $\mathbb{W}$ is the unknown noise function that transforms the real scene into the measurement point cloud. Next, we define the set of outliers in point cloud caused by W^t that are harmful to registration as $\boldsymbol{P_{out}}$, and the remaining points are $\boldsymbol{P_{in}}$, namely,

$$\boldsymbol{P} = \boldsymbol{P_{out}} \cup \boldsymbol{P_{in}} = \mathbb{W}\big(P^*, W\big) \tag{10.3}$$

The task of this paper is to find the optimal function $\mathbb{F}$ and $\mathbb{G}$, and the core procedure is to find and eliminate $\boldsymbol{P_{out}}$ to optimize them.

10.3.2 Segmentation

The input of the segmentation module is the original 3D LIDAR point cloud $\boldsymbol{P} = \{p_1, p_2, \ldots, p_n\}$, the output is m clusters $\{P_{C_1}, P_{C_2}, \ldots, P_{C_m}\}$ and corresponding cluster label vector $L = (l_1, l_2, \ldots, l_n)$, where $l_i \in [-1, m] \cap \mathbb{Z}$, label -1 represents the point belongs to the ground, and the positive integer represents the serial number of non-ground clusters.

The point cloud $\boldsymbol{P^k}$ is first projected onto the circular range image $P_{U,V} \triangleq \{p_{(u,v)} | u \in [1, 360°/R_{\text{hori}}] \cap \mathbb{Z}, v \in [1, N_{\text{scan}}] \cap \mathbb{Z}\}$ of size $N_{\text{pixel}} = 360°/R_{\text{hori}} * N_{\text{scan}}$. The row index row_i of p_i is shown in (10.4), and the column index col_i is shown in (10.5).

$$\text{row}_i = \left(\tan^{-1}\frac{z_i}{\sqrt{x_i^2 + y_i^2}} + \text{Pit}_{\text{bottom}}\right) / R_{\text{vert}} \tag{10.4}$$

$$\text{col}_i = \tan^{-1}\frac{y_i}{x_i} / R_{\text{hori}} \tag{10.5}$$

The value of the image is the distance from p_i to the Lidar $r_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$, where R_{hori} is the horizontal resolution of the Lidar, R_{vert} is the vertical resolution of the Lidar, $\text{Pit}_{\text{bottom}}$ is the minimum pitch of the Lidar Lidar beam, and N_{scan} is the number of Lidar lines. After the projection, a point $p_{u,v+1}$ and $p_{u,v}$, $v \in \left[1, \frac{1}{2}N_{\text{scan}}\right] \cap \mathbb{Z}$ in P can be uniquely identified by the row index and column index, denoted by:

$$p_i \equiv p_{(u,v)} = \big(x_{(u,v)}, y_{(u,v)}, z_{(u,v)}\big) \tag{10.6}$$

Then the ground is extracted from the range image $P_{U,V}$. For points $p_{u,v+1}$ and $p_{u,v}$, where $v \in \left[1, \frac{1}{2}N_{\text{scan}}\right] \cap \mathbb{Z}$; if the pitch of adjacent rows $Pit_{v,v+1}$ (as shown in (10.7)) are less than 10°, they are denoted as ground points.

$$Pit_{v,v+1} = \tan^{-1}\frac{z_{u,v+1} - z_{u,v+1}}{\sqrt{(x_{u,v+1} - x_{u,v})^2 + (y_{u,v+1} - y_{u,v})^2}} \tag{10.7}$$

Finally, the segmentation clustering method based on range image [18] was applied to cluster the points into several clusters. Points from the same cluster are assigned a unique label. After this process, for $p_i^k \in \boldsymbol{P^k}$, we get l_i, row_i, col_i, and r_i, where $i \in \left[1, N_{\text{pixel}}\right] \cap \mathbb{Z}$. $\boldsymbol{P^k}$ is divided into a ground point set and some non-ground point sets, that is, $\boldsymbol{P^k} = \boldsymbol{P_g} \cup \boldsymbol{P_{ng}}$, and it is worth noting that both types of points may contain outliers. The extraction of features from non-ground points can reduce time-consuming and improve the accuracy of registration through label matching and noise filtering.

10.3.3 Feature Extraction

In order to reduce the computation amount of registration, we extract the representative feature points to reduce the number of the points to be registered. The feature extraction method is similar to that used in LeGo-LOAM. We first used (10.8) to evaluate the curvature c_i of point p_i, defining points whose curvature is greater than threshold c_{thre} as edge features and points whose curvature is less than c_{thre} as plane features.

$$c_i = \frac{1}{|\boldsymbol{S}|||r_i||} \left\| \sum_{j \in S, j \neq i} (r_j - r_i) \right\| \tag{10.8}$$

where $\boldsymbol{S}$ is a set of continuous points on the same row from the image, points in S are uniformly distributed on both sides of p_i, and $|\boldsymbol{S}|$ is the number of points in $\boldsymbol{S}$.

10.3.4 Lidar Odometry

The input of Lidar odometry module is the feature point in the point cloud, which includes estimating the Lidar motion between adjacent scans and eliminating the motion distortion of the point cloud through feature matching. Looking for the Lidar of the adjacent scanning motion—by looking for rotation matrix $\tilde{\boldsymbol{R}} \in \text{SO}(3)$ and translation vector $\tilde{t} \in \mathbb{R}^3$ to minimize the point cloud registration error, such as (10.9).

$$\tilde{\boldsymbol{T}} \triangleq \left(\tilde{\boldsymbol{R}}, \tilde{t}\right) = \arg\min\left(\sum_i ||\tilde{\boldsymbol{R}}^t p_i^t + \tilde{t} - q_i^{t-1}||\right) \tag{10.9}$$

where q_i^{t-1} is the corresponding point at $t - 1$ found by p_i^t.

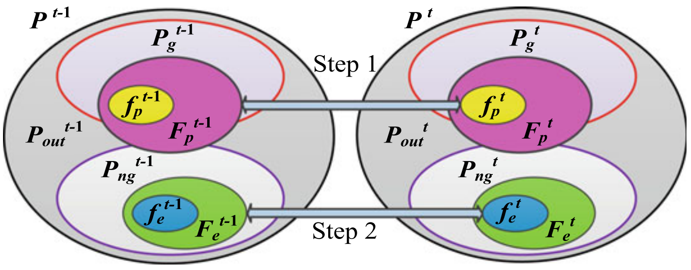

Fig. 10.1 Corresponding relation of feature points at adjacent moments

The LOAM details $\mathbb{F}$, and LeGo-LOAM optimizes it. For brevity, this chapter focuses on further optimization based on these methods, which is achieved by matching only valid feature points in the adjacent frames. The optimization process can be defined as:

$$\min ||\tilde{\boldsymbol{T}}^t - \boldsymbol{T}^{*t}|| \quad \text{s.t. } p_i^t \in \boldsymbol{F}_{in}^t \tag{10.10}$$

There is the optimal transformation matrix from $\boldsymbol{P}^t$ to $\boldsymbol{P}^{t-1}$, $\boldsymbol{R}^*$ and tr* are the corresponding rotation matrix and the translational vectors, and $\boldsymbol{F}_{in}^t$ is the set of effective feature points at time t, namely $\boldsymbol{F}^t = \left(\boldsymbol{F}_{out}^t \cup \boldsymbol{F}_{in}^t\right)$. The process of finding the corresponding relationship is shown in Fig. 10.1, that is, the transformation is found in the plane feature point set and the edge feature point set, respectively, and the computational effort is reduced by reducing the candidate range of corresponding points. We focus on two strategies, outlier elimination and weight compensation of feature point.

(1) Outlier elimination: To solve this problem, we use the two-step coarse-to-fine registration strategy, that is, rough registration is first used to detect outliers, and then the outliers are avoided to participate in the L-M algorithm. We can express the outlier feature point $\boldsymbol{F}_{out}$ as:

$$\boldsymbol{F}_{out}^t = \left\{p_i^t \,|\, ||\boldsymbol{R}^* p_i^t + tr^* - q_i^{t-1}|| > \varepsilon \wedge p_i^t \in \boldsymbol{F}^t\right\} \tag{10.11}$$

where q_i^{t-1} is the feature point corresponding to p_i^t. We first define feature descriptor $\boldsymbol{d}^t$ and then use $\boldsymbol{d}^t$ and $\boldsymbol{d}^{t-1}$ to calculate pose transformation $\tilde{\boldsymbol{T}}_{\text{rough}}$ as follows:

(1) Finding the first point in the four-point basis: We select the feature point $f_{(u_1,v_1)}^t \in \boldsymbol{F}^t$ as the first point a_1^t in the ith range image, namely $a_1^t \triangleq f_{(u_1,v_1)}^t$, where $u_1^t \in \left[1 + \frac{360^\circ*(i-1)}{N_{\text{sub}}*R_{\text{hori}}}, \frac{360^\circ*i}{N_{\text{sub}}*R_{\text{hori}}}\right] \cap \mathbb{Z}$ and $v_1^t \in \left[\frac{N_{\text{scan}}}{4}, \frac{N_{\text{scan}}}{2}\right] \cap [v_1^{t-1} - 1, v_1^{t-1} + 1] \cap \mathbb{Z}$.

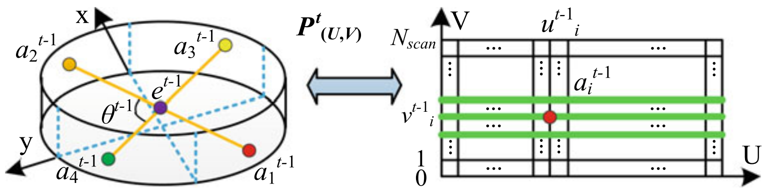

Fig. 10.2 Schematic diagram of four-point base selection

(2) Finding the second point: We select $a_2^t \triangleq f^t_{(u_2,v_2)}$ in the sub-image farthest from a_1^t, where $u_2^t \in \left[1 + \frac{360^\circ*\left(i-1+\frac{N_{\text{div}}}{2}\right)}{N_{\text{sub}}*R_{\text{hori}}}, \frac{360^\circ*\left(i+\frac{N_{\text{div}}}{2}\right)}{N_{\text{sub}}*R_{\text{hori}}}\right] \cap \mathbb{Z}$, $f^t_{(u_2,v_2)} \in P^t_{(U,V)_{i+\frac{1}{2}N_{\text{sub}}}}$, the line segment defined with a_1^t and a_2^t as endpoints is $l_1^t \triangleq a_1^t a_2^t$.

(3) Constructing feature descriptors: We select a_3^t and a_4^t from the sub-images excepting a_1^t and a_2^t to satisfy the conditions that $l_2^t \triangleq a_3^t a_4^t$ is coplanar with l_1^t, $\left|l_2^t\right| > \frac{\left|l_1^t\right|}{4}$, and $\angle\left(l_1^t, l_2^t\right) > \frac{180^\circ}{N_{\text{sub}}}$. The intersection of l_1^t and l_2^t is e^t. Then we record feature descriptor $\boldsymbol{d}^t$, $\boldsymbol{d}^t \triangleq \left(d_1^t, d_2^t\right)$ when the above three conditions are true, otherwise, $\boldsymbol{d}^t \triangleq \left(d_1^t, d_3^t\right)$, where $d_1^t \triangleq \left(a_1^t, a_2^t\right)$, $d_2^t \triangleq \left(a_3^t, a_4^t, e^t, \angle\left(l_1^t, l_2^t\right)\right)$, and $d_3^t \triangleq \left(c^t_{a_1}, c^t_{a_2}\right)$.

(4) Matching feature descriptors: If the condition in (10.3) is true, and we use Super4PCS [13] to match $\boldsymbol{d}^t$ and $\boldsymbol{d}^{t-1}$; otherwise, we use 2PNS [15].

The process of the second L-M step is similar to that in the first step, but the edge feature points is used instead, and the number of rows of feature points in (1) and (2) is no longer limited (Fig. 10.2).

(2) Weight compensation of feature point: LeGo-LOAM gives the feature points a weight s_i which is inversely proportional to the distance between Lidar and points in the iteration process of L-M algorithm. The difference is that we also consider the influence of the point cloud holes in the background caused by the occlusion of dynamic objects on the stability of the system. Therefore, we reduce the weight of the feature points within a certain range of the dynamic cluster.

$$\widetilde{s_i} = s_i*\left(1 - \frac{R_{\text{hori}}}{\left|\theta_{\text{yaw}}\right|}\right) \tag{10.12}$$

where $\widetilde{s_i}$ is the new feature weight and θ_{yaw} is the minimum yaw difference between a feature point and a dynamic object.

10.3.5 Lidar Mapping

Let us review the mapping model $\mathbb{G}$ (10.2), which is to use M^{t-1} and $\boldsymbol{P}^t$ to build M^t because the global map contains a lot of feature information, and the earlier information contains less error. The Lidar mapping module further optimizes the pose estimation by matching the $\boldsymbol{F}^t$ with the submap M_{sub}^{t-1} around the Lidar. Please refer to the description in [20] for detailed matching and optimization procedures. We use the map storage method in LeGo-LOAM to store only the feature point set $\boldsymbol{F}^t$.

$$M^{t-1} \triangleq \left\{ \underbrace{\{\boldsymbol{F}^1\}, \{\boldsymbol{F}^2\}, \dots, \{\boldsymbol{F}^{t-k2}\}}_{M_{\text{hist}}^{t-1}}, \dots, \underbrace{\{\boldsymbol{F}^{t-k}\}, \dots, \{\boldsymbol{F}^t\}}_{M_{\text{sub}}^{t-1}} \right\} \tag{10.13}$$

In addition, the optional function is to further eliminate the error of the mapping module by detecting the loop closure. In LeGo-LOAM, if $\boldsymbol{F}^t$ can find corresponding feature in earlier map M_{hist}^{t-1} or recent features by ICP, a new constraint is added. The difference is that we use coarse-to-fine registration strategy in odometry module again to replace single-step ICP.

Then, the estimated pose of the sensor is updated by sending the pose map to the optimized system (e.g., [19]).

10.4 Experiments

All experiments were performed on a laptop equipped with a 2.5 GHz Intel i7-4700 processor, 16 GB of RAM.

10.4.1 Feature Extraction

Figure 10.3 shows a frame of point cloud and feature points (in green) in scenario 1, including five moving pedestrians (circled in white box), where Fig. 10.3a shows the wrong feature extraction results of LeGo-LOAM. It can be seen that many feature points are distributed on unstable outliers belong to pedestrians, while our result (Fig. 10.3b) is more reasonable because we detect outliers and avoid extracting feature points from them to avoid the incorrect odometry and mapping results caused by wrong matching of feature points. To be concise, we only show the feature edge points that are sufficient to illustrate the problem.

The results show that our method has the advantage of effective feature extraction in scenes containing a large proportion of outliers (Table 10.1).

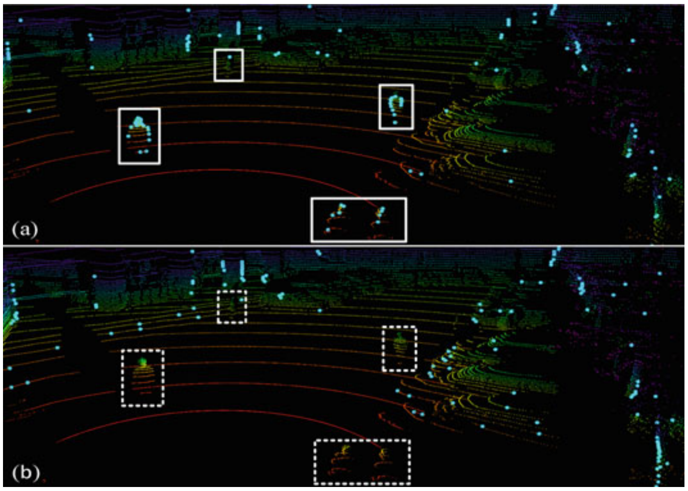

Fig. 10.3 Feature extraction results of LeGo-LOAM and ours

Table 10.1 Comparison of effective features ratio between LeGo-LOAM and ours

Scenario	F_e		f_e		F_p		f_p	
	Ours	LeGO-LOAM	Ours	LeGO-LOAM	Ours	LeGO-LOAM	Ours	LeGO-LOAM
1	0.96	0.57	0.97	0.59	0.99	0.52	0.98	0.83
2	0.98	0.82	0.98	0.83	0.99	0.80	0.99	0.94

10.4.2 Odometry

Figure 10.4 shows the odometry comparison results of scenario 1, where the red curve is the result of LeGO-LOAM, while the blue one is ours. The results of the two methods have little difference from a global perspective, but there are quite differences in detail. We choose the most representative location (upper left corner of scenario 1, where pedestrians gathering here) to show the contrast, as can be seen from the green box, and the red line has a distinct jagged edge, which is exactly the wrong odometry estimation caused by the outliers, while the blue line has no such problem.

Figure 10.5 shows the comparison result between the two methods and the true value in scenario 2, and we can see from Fig. 10.5 that the accuracy can still be maintained in the long trajectory scene (with a length of about 3730 m). The relative pose estimation error is calculated by comparing the final pose and the initial pose. Rotation errors and translation errors of LeGo-LOAM are 5.65° and 3.19 m, while our results are 4.52° and 2.75 m. These two groups of experiments illustrate the advantage of our method in odometry estimation.

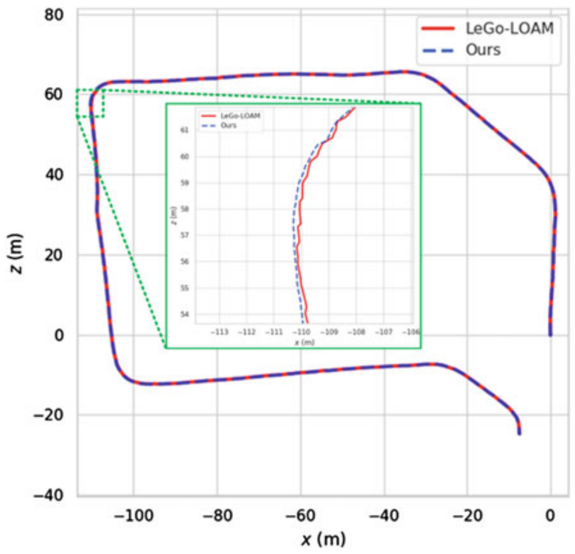

Fig. 10.4 Odometry comparison results in scenario 1 between the two methods, where the green box is enlarged to highlight the differences. Ours is smoother and more accurate than that of LeGo-LOAM

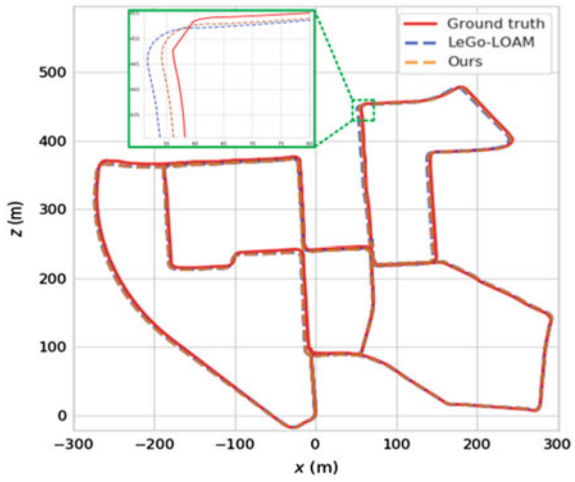

Fig. 10.5 Odometry comparison results in scenario 2 between the two methods. Green box is enlarged to highlight the differences

10.4.3 Mapping

The results of representative scenarios indicate that LeGo-LOAM will produce wrong mapping results because they do not consider outliers, as shown from Figs. 10.6, 10.7, 10.8, and 10.9.

Figure 10.6 shows the mapping result of scenario 1, where Fig. 10.6a is the LeGO-LOAM mapping result. Figure 10.7 shows the details of the mapping result of scene 1, where (a), (b), and (c) are the result of incorrect mapping caused by incorrect

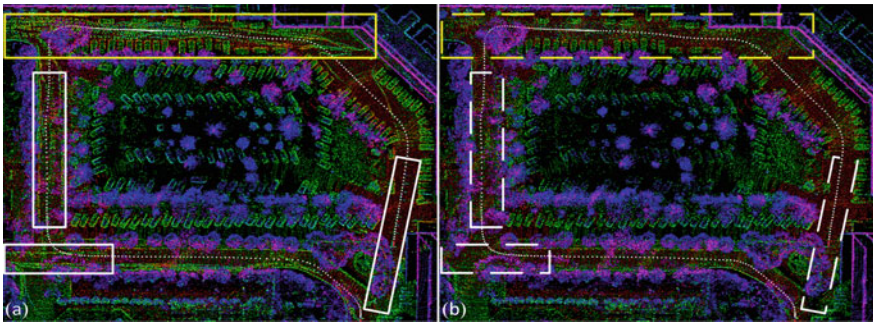

Fig. 10.6 Mapping result of scenario 1. **a** The result of LeGO-LOAM, we can clearly see the shadow left by outliers. **b** Our result, we get a clean map by identifying outliers and avoiding extracting feature points and mapping using them

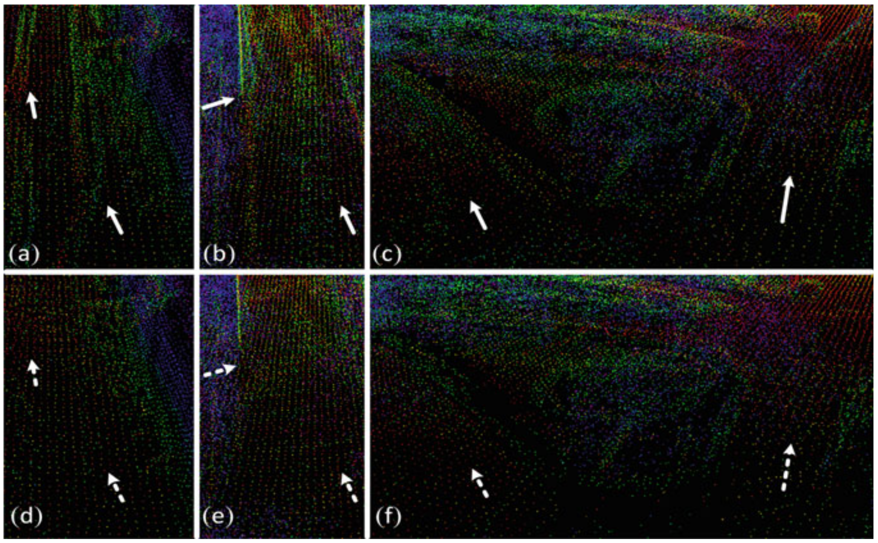

Fig. 10.7 Details comparison in scenario 1. **a**, **b**, and **c** are the results of LeGo-LOAM. **d**, **e**, and **f** are the results of our method. The white arrows highlight the differences

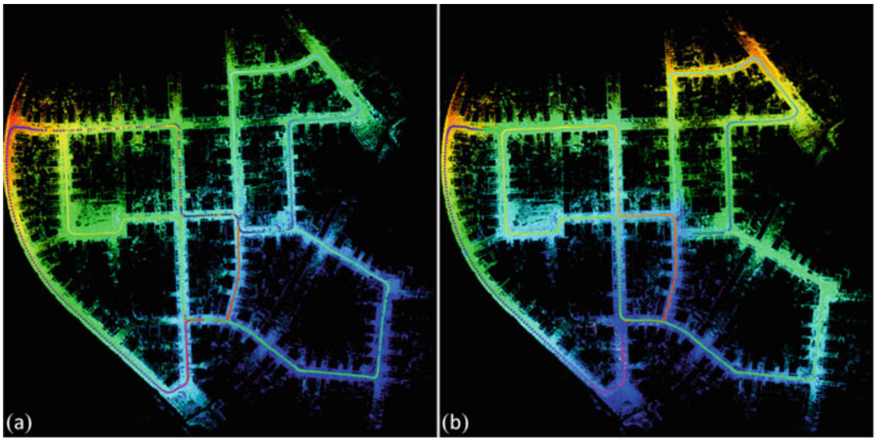

Fig. 10.8 Mapping result of scenario 2. **a** The result of LeGO-LOAM. **b** The result of our method

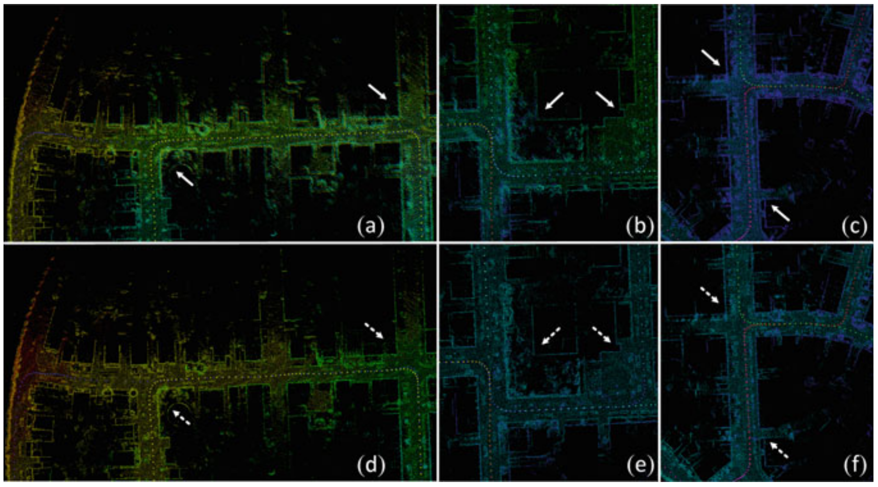

Fig. 10.9 Details comparison in scenario 2. **a**, **b**, and **c** are the results of LeGo-LOAM. **d**, **e**, and **f** are the results of our method. The white arrow highlights the differences

feature extraction. Dynamic pedestrians and vehicles participate in feature extraction and association and are thus added to the map, leaving obvious residual shadows. Among them, Fig. 10.7a is the enlarged result on the top of the map, and the residual shadow is the movement track left by four pedestrians moving forward. Figure 10.7b is a local enlarged image on the left of the map, and the residual shadow is the movement track left by an upward moving vehicle. Figure 10.7c is a local enlarged image of the lower right part of the map, and the residual shadow is left by a moving car. Different from LeGO-LOAM, we first detect and eliminate outliers and focus

Table 10.2 Runtime (ms) comparison of one frame between LeGo-LOAM and ours

Scenario	Lidar odometry		Lidar mapping	
	Ours	LeGO-LOAM	Ours	LeGO-LOAM
1	9.2	10.7	217.5	311.0
2	12.1	14.0	200.4	294.2

on extracting effective features, so we have obvious advantages when mapping in dynamic scenarios.

Figure 10.8 shows the mapping result of scenario 2, when we zoom in on the local map near the loop closure, as shown in Fig. 10.9, we can see the obvious differences as indicated by the white arrows, and the buildings in map of LeGO-LOAM have obviously offset, while ours remains globally consistent, which benefits from two-step of coarse-to-fine registration strategy.

10.4.4 Runtime

The running time of odometry module and mapping module between our method and LeGo-LOAM is shown in Table 10.2. The time consumption of odometry module is reduced by about 15%, while that of mapping module is about 31%. The results demonstrate the advantage of our approach in terms of time consumption, which is because we use two-step coarse-to-fine registration strategy to replace single-step registration.

10.5 Conclusions and Future Works

Future works include further improving the accuracy of SLAM system in complex and highly dynamic scenes, because we find the proposed system will still produce unsatisfactory performance when there are many and dense moving objects around the Lidar. Some methods to solve the problem of robot kidnapping may be the solutions to the raised problems. We will also try to deploy SLAM system on multiple base stations to make up for the deficiency caused by the serious occlusion of a single base station by mobile objects through cooperative tasks. Therefore, we will further combine the semantic information and motion information of the scene, which will also provide a new idea for the construction of semantic map.

Acknowledgements This work was supported by National Nature Science Foundation of China (NSFC) under Grants No. 61671356, Science and Technology Program of Shaanxi Province under Grants No. 2020GY-136 and 2019ZDLGY14-02-03.

References

1. Zhang, J., Henein, M., Mahony, R.E., Ila, V.: VDO-SLAM: a visual dynamic object-aware SLAM system. Int. J. Robot. Res. (2020) [Online]. https://arxiv.org/abs/2005.11052
2. Zhang, J., Singh, S.: LOAM: Lidar odometry and mapping in real time. Proc. Robot.: Sci. Syst. (2014)
3. Zhang, J., Singh, S.: Low-drift and real-time lidar odometry and mapping. Auton. Robot. **41**(2), 401–416 (2017)
4. Zhang, J., Singh, S.: Visual-lidar odometry and mapping: low-drift, robust, and fast. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2174–2181 (2015)
5. Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., Rus, D.: LIO-SAM: tightly-coupled Lidar inertial odometry via smoothing and mapping. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2020
6. Shan, T., Englot, B.: 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 (2018)
7. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: towards the robust-perception age. IEEE Trans Robot. (TRO) 1309–1332 (2016)
8. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)
9. Yang, J., Li, H., Campbell, D., Jia, Y.: Go-ICP: a globally optimal solution to 3D ICP point-set registration. IEEE Trans. Pattern Anal. Mach. Intell. **38**(11), 2241–2254 (2016)
10. Campbell, D., Petersson, L.: Gogma: Globally-optimal gaussian mixture alignment. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5685–5694 (2016)
11. Aiger, D., Mitra, N.J.: Cohen-Or D.4-points congruent sets for robust pairwise surface registration. ACM Trans. Graphics **27**(3), 85, 1–10 (2008)
12. Theiler, P.W., Wegner, J.D., Schindler, K.: Keypoint-based 4-points congruent sets—automated marker-less registration of laser scans. ISPRS J. Photogramm. Remote Sens. **96**, 149–163 (2014)
13. Mellado, N., Aiger, D., Mitra, N.J.: Super 4PCS fast global pointcloud registration via smart indexing. Comput. Graphics Forum **33**(5), 205–215 (2015)
14. Mohamad, M., Ahmed, M.T., Rappaport, D., Greenspan, M.: Super generalized 4PCS for 3D registration. In: 2015 International Conference on 3D Vision, pp. 598–606 (2015)
15. Raposo, C., Barreto, J.P.: Using 2 point+normal sets for fast registration of point clouds with small overlap. In: 2017 IEEE International Conference on Robotics & Automation (ICRA), pp. 5652–5658 (2017)
16. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3354–3361 (2012)
17. Rozenberszki, D., Majdik, A.L.: LOL: Lidar-only odometry and localization in 3D point cloud maps. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, pp. 4379–4385 (2020)
18. Bogoslavskyi, I., Stachniss, C.: Fast range image-based segmentation of sparse 3D Lidar scans for online operation. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 163–169 (2016)
19. Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J.J., Dellaert, F.: iSAM2: incremental smoothing and mapping using the Bayes tree. Int. J. Robot. Res. **31**(2), 216–235 (2012)
20. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: IEEE ICRA Workshop on Open Source Software (2009)