2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

ROI-cloud: A Key Region Extraction Method for LiDAR Odometry and Localization

Zhibo Zhou, Ming Yang, Chunxiang Wang and Bing Wang

***Abstract*— We present a novel key region extraction method of point cloud, ROI-cloud, for LiDAR odometry and localization with autonomous robots. Traditional methods process massive point cloud data in every region within the field of view. In dense urban environments, however, processing redundant and dynamic regions of point cloud is time-consuming and harmful to the results of matching algorithms. In this paper, a voxelized cube set, ROI-cloud, is proposed to solve this problem by exclusively reserving the regions of interest for better point set registration and pose estimation. 3D space is firstly voxelized into weighted cubes. The key idea is to update their weights continually and extract cubes with high importance as key regions. By extracting geometrical features of a LiDAR scan, the importance of each cube is evaluated as a new measurement. With the help of on-board IMU/odometry data as well as new measurements, the weights of cubes are updated recursively through Bayes filtering. Thus, dynamic and redundant point cloud inside cubes with low importance are discarded by means of Monte Carlo sampling. Our method is validated on various datasets, and results indicate that the ROI-cloud improves the existing method in both accuracy and speed.**

I. Introduction

Robot localization based on on-board sensors (e.g. light detection and ranging (LiDAR) sensors and cameras) and prior models of the environment (typically metric maps) is a fundamental task. Simultaneous localization and mapping (SLAM) is helpful to build a consistent map of environment, with which a robot can localize itself if the surrounding information is perceived. Great efforts have been devoted to odometry/SLAM and localization problems in recent years. In this paper, the LiDAR odometry problem and the map-based localization problem are mainly concerned.

A powerful and fast matching algorithm plays an important part in these two problems. Registering two point sets takes massive computational resource, and it has been long regarded as a major challenge. Traditional methods (e.g. ICP [1] and NDT [2]) traverse and process every point iteratively to calculate the transformation between two sets. They are usually time-consuming, and one of the significant reasons is that they treat all points or regions as equally important. Though downsampling [3] may reduce its size, an oversized grid can cause loss of accuracy. Therefore, a more compact representation of raw point cloud data is urgently needed. Usually, semantic features [4] (e.g. poles, trees or walls) are extracted to represent the environment by artificial rules or learning-based methods and are of great help to improve matching result in both speed and accuracy. However, these methods often fail in places that lack adequate and generalized semantic feature labels or training dataset of high-quality. Also, impressive contributions have been made to extract primitive geometrical features [5], [6] (e.g. edges, corners or planar surfaces) for better registration. Geometrical features have the overwhelming advantage of being lightweight and robust to various environments but occasionally suffer from wrong pose estimation due to occlusions and disturbances from dynamic objects, such as moving vehicles and pedestrians [7].

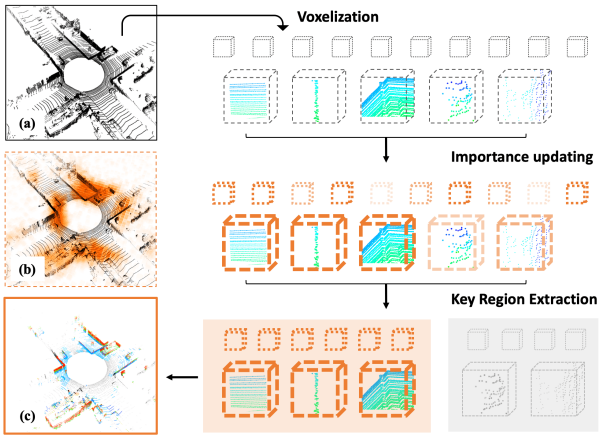


Fig. 1. (a) Original point cloud. The original cloud is voxelized and each cube contains some local points. (b) ROI-cloud in orange. Cubes that have significant geometrical features are assigned with higher weights (deeper orange) and then form an ROI-cloud.(c) Target point cloud. Key regions (orange) are extracted from cubes of high importance, while those of low importance (gray) are discarded. As a subset of the original cloud, the target point cloud is clustered around corners at the crossroad as illustrated in color.

Obviously, different point cloud regions have unequal importance. Inspired by methods in computer vision, we believe that it is sufficient for a scan matching task to use only a small portion of points from those important regions, which we call key regions or regions of interest (ROI). Outside key regions, there exists numerous points in redundant regions which are helpless for registration. Additionally, points in dynamic regions are harmful to matching algorithms. The efficiency, as well as robustness, can be improved if these points are removed in advance.

The aim of this paper is to locate the key regions (ROI). A key region should contain intensive and stable features for matching. Therefore a fast geometrical feature extraction is

This work is supported by the National Natural Science Foundation of China (U1764264/61873165), Shanghai Automotive Industry Science and Technology Development Foundation (1807), International Chair on automated driving of ground vehicle. Ming Yang is the corresponding author.

Zhibo Zhou, Ming Yang, Bing Wang and Chunxiang Wang are with Department of Automation, Shanghai Jiao Tong University, Shanghai, 200240; Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai, 200240. E-mail: MingYang@sjtu.edu.cn

Authorized licensed use limited to: Shenyang Institute of Automation. Downloaded on October 14,2024 at 10:54:55 UTC from IEEE Xplore. Restrictions apply.

applied to evaluate the importance of the region. Besides, the historical information is of significance for dynamic object removal [8]. Notably, unlike dynamic regions and noises, a key region tends to remain stationary in several consecutive scans. Hence, the history of the extracted regions is so helpful that they should be utilized as prior knowledge.

When locating key regions (ROI) to remove redundant or dynamic objects, one important fact is that the observation of environment is difficult to be mathematically approximated in a parametric way (e.g. Gaussian techniques such as GMM [9]). In other words, both position and number of key regions are uncertain in various scenarios, thus a limited number of parameters may reduce the quality of the approximation. We novelly utilized a nonparametric framework, particle filter [10], to modeling the distribution of key regions more precisely, making it more robust in different environments.

We propose a novel set of cubes, ROI-cloud, for key region extraction of point cloud. ROI-cloud consists of many high-probability voxelized cubes. Under a particle filter framework, 3D space is voxelized into weighted cubes and each particle is exclusively attached to a cube coordinate, therefore the position of key regions is approximated by belief distribution of particles. During filtering, the weights (importance) of particles are updated at each instance with the help of feature extraction result and IMU/odometry data. This system works well because the particles attached to cubes in dynamic or redundant regions are prone to be assigned with small weights, and consequently escape from these areas over time. By this means, most particles accumulate around cubes of high probability, namely key regions. Unlike segmentation of all static and dynamic regions, ROI-cloud manages to merely select several key regions and discard meaningless regions, which is lightweight and timesaving.

This paper presents the following contributions:

- Key regions consisting of high-probability cubes are extracted to effectively remove redundant and dynamic point cloud by novelly utilizing historical information and extracting geometrical features.
- A nonparametric filter framework is applied to approximate the distribution of key regions precisely and efficiently in various scenarios.
- Feasible applications of ROI-cloud with multiple mainstream algorithms in LiDAR odometry and map-based localization experiments which achieve better accuracy and speed in dense urban environments.

The remainder of this paper is organized as follows. Some related work is shown in Section II. Section III introduces the detail of ROI-cloud, and its applications in LiDAR odometry and localization are described in Section IV. Experiment setup and results analysis are shown in Section V. Section VI presents the conclusion with some discussions.

II. Related Work

Generally, LiDAR odometry problem and map-based localization problem are similar since both of them are focusing on point cloud registration and pose estimation. On the one hand, the core of the LiDAR odometry problem is a scan-to-scan matching method. Iterative closest point (ICP) [1] is a typical approach to iteratively aligning two sets of points. Generalized-ICP [11] is one of those ICP variants [12] that are proposed to improve its accuracy and efficiency. Feature-based matching algorithms, such as normal distribution transform (NDT) [2], Point Feature Histograms (PFH) [13] and Viewpoint Feature Histograms (VFH) [14], extract precise representative features to accelerate the matching of dense point cloud. Some algorithms select key points at edges and planes to perform matching. Among them LOAM [6] and LeGO-LOAM [15] are by far the best methods which dividing the estimation task to a two-step algorithm and is capable to achieve real-time performance. Those scan-to-scan matching algorithms estimate transformations among scans and integrate them into odometry. On the other hand, the core of the localization problem is a scan-to-map matching method. Besides the geometrical point set registration algorithms aforementioned, methods based on Gaussian mixture model (GMM) [16], [17] or filtering [18], [19] are also proposed. Monte Carlo localization (MCL) [10], [20] algorithm is popular but occasionally low-precise.

Due to the enormous amount of data, a downsampling filter may reduce the size but can result in poor accuracy. Although there exist some methods simplify point cloud by extracting pole-like [21] or planar characteristics [22], they often fail in environments that lack clear semantic features. SegMatch [23] applies segmentation to a point cloud but has low-frequency output. Things get worse when there are plentiful moving objects in dense urban environments where dynamic noises are non-negligible. Most of the existing matching methods extract features only based on the current scan, neglecting the history of feature information. When considering such informations, algorithms on SLAM with detection and tracking of moving objects (DATMO) [24] have been proposed to solve the problem. They usually maintain two separate maps, localizing the robot as well as tracking multiple targets simultaneously. Nevertheless, they still require algorithms that reduce the computational complexity to achieve optimal results.

III. Region of Interest Cloud

Unlike a traditional point cloud, ROI-cloud is not a set of points, but a set of weighted cubes in 3D space. After recursively updating, the weights (or probabilities) of cubes increase around key regions and eventually form a cloud-like shape, which is the origin of the name in this paper.

A. System Overview

An overview of the proposed ROI-cloud framework is shown in Fig. 2. Raw point cloud data and IMU/odometry messages are input to the system, and a point cloud of key regions are output. The point cloud of the newest LiDAR scan is popped from LiDAR scan stack and preprocessed through a fast feature extraction module, then the voxelized feature information is obtained as the current measurement. Meanwhile, in the framework of Bayes filtering, the distribution of particles one time step earlier contains prior

Authorized licensed use limited to: Shenyang Institute of Automation. Downloaded on October 14,2024 at 10:54:55 UTC from IEEE Xplore. Restrictions apply.

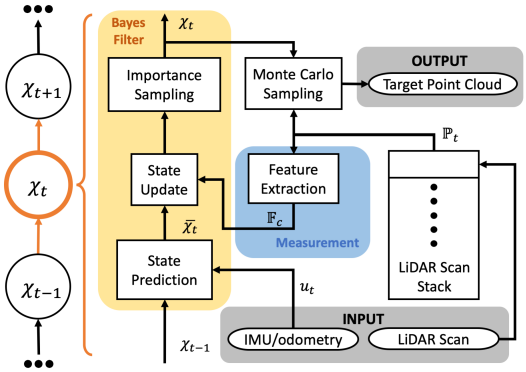


Fig. 2. Block diagram of ROI-cloud system.

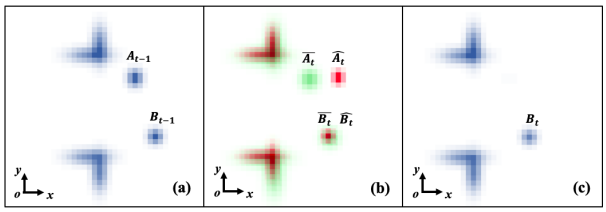


Fig. 3. An illustration of the state update process. The process is simplified to X-Y plane for better visualization. The 2D space is voxelized and the darker color of a grid represents that it contains more particles inside. Moving object A was around the position A_{t-1} at last time step. We assume it to be a static object so that it's predicted to be transformed to $\bar{A}_t$ as the robot moves along x-axis. However, the target actually moves to $\hat{A}_t$ at time t, thus being removed after convolution. Static object B is updated in the same way and finally being reserved.

information and predicts the current ROI by combining the IMU/odometry. Prediction and measurement are filtered to update the state of particles. The resampling step forces particles back to the posterior distribution and outputs the ROI-cloud. Monte Carlo method is finally applied to sample out the target point cloud as ROI-cloud recommended. The details of each module are introduced as follows.

B. Feature Extraction

In a single LiDAR scan $\mathbb{P}$, points on the prime edges and surfaces are suitable and adequate for a simple representation of the environment. The feature extraction method is similar to [6] except that points on the floor are firstly removed by assuming that robots have rarely significant change in pitch and roll angles while moving. Edge features $\mathbb{P}^e$ and surface features $\mathbb{P}^p$ are extracted by evaluating the geometric smoothness ξ of point $p_i = [x_i, y_i, z_i]^{\mathrm{T}}$ and its neighbors S_i in same channel l:

$$\xi_i = \frac{1}{|S_i| \cdot \left\|p_i^l\right\|} \left\| \sum_{j \in S_i, j \neq i} (p_j^l - p_i^l) \right\|. \tag{1}$$

Generally, the value of the smoothness ranges from 10^{-5} to 10^5. We thus calculate the logarithm of original values to obtain a new curve of distribution and search from the left side (right side) of the curve to get enough surface feature (edge feature) points. This operation is a replacement of sorting algorithm and the time complexity is decreased significantly.

For a cube $c_k = [X_k, Y_k, Z_k]^{\mathrm{T}}$ in cube set $\mathbb{C}$, the overall feature of this cube is evaluated by the distribution of feature points inside. For points in cube c_k, its covariance matrix A_{c_k} can be expressed as

$$A_{c_k} = \frac{1}{|\mathbb{P}_{c_k}|} P P^{\mathrm{T}}, \tag{2}$$

where $P = [\hat{p_1}, \hat{p_2}, ..., \hat{p_m}], m = |\mathbb{P}_{c_k}|$ and

$$\hat{p_i} = p_i - \frac{1}{|\mathbb{P}_{c_k}|} \sum_{j \in S} p_j. \tag{3}$$

After eigendecomposition $A_{c_k} = Q \Sigma Q^{\mathrm{T}}$ where Σ is a diagonal matrix: $diag(\lambda_1, \lambda_2, \lambda_3)$, and $\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3$.

Define a term f_{c_k} to evaluate the importance or feature of a local cube:

$$f_{c_k} = \frac{\lambda_1^e}{\sum_{j \in \Sigma^e} \lambda_j^e} \left|\mathbb{P}_{c_k}^e\right| + \eta \frac{\lambda_1^p + \lambda_2^p}{\sum_{j \in \Sigma^p} \lambda_j^p} \left|\mathbb{P}_{c_k}^p\right|, \tag{4}$$

where η is a balance factor between edge and surface points. At one time step, the importance factor set of all cubes $\mathbb{F}_{\mathbb{C}}$, is the current measurement of this LiDAR scan.

C. Importance updating

The importances of cubes are updated through an application of particle filtering. Each particle is designed to represent a specific position of a cube.

The state prediction of particles can be easily derived from the dead reckoning. The initial distribution of ROI-cloud is achieved through a set of particles drawn at random and uniformly over the entire 3D voxelized space. Particles will gradually accumulate near key regions as new measurements come. Assume that the set χ_{t-1} at time $t-1$ is distributed according to $bel(x_{t-1})$. The goal is to predict the distribution at time t before incorporating the measurement, only based on the previous state posterior. We assume that all particles are clustered in static regions, then from time $t-1$ to time t, a set of m particles $\chi_{t-1} = \{x_{t-1}^{[1]}, x_{t-1}^{[2]}, ..., x_{t-1}^{[m]}\}$ are transformed according to the motion model of a robot, and form a new set $\overline{\chi}_t = \{\overline{x}_t^{[1]}, \overline{x}_t^{[2]}, ..., \overline{x}_t^{[m]}\}$:

$$\overline{x}_t^{[i]} = x_{t-1}^{[i]} + \Delta x_t^{[i]} + w_t, \tag{5}$$

where Gaussian noise w_t is added here since sensors are subject to uncertainty. The belief $\overline{bel}(x_t)$ is approximated by the set of particles $\overline{\chi}_t$, with each roughly corresponding to a cube's position in 3D space:

$$\begin{aligned} \overline{\chi}_t^{[i]} \sim \overline{bel}(x_t) &= p(x_t \mid \mathbb{F}_{\mathbb{C}_{1:t-1}}, u_{1:t}) \\ &= p(x_t \mid \mathbb{F}_{\mathbb{C}_{t-1}}, u_t) \end{aligned}, \tag{6}$$

where u_t is the motion measurement at time t.

The weight of each particle is then updated so that the particles around key regions can be assigned with higher weights of importance. The importance weights of particles $\overline{\chi}_t$ are uniform. Naturally, Markov assumption is applied

Authorized licensed use limited to: Shenyang Institute of Automation. Downloaded on October 14,2024 at 10:54:55 UTC from IEEE Xplore. Restrictions apply.

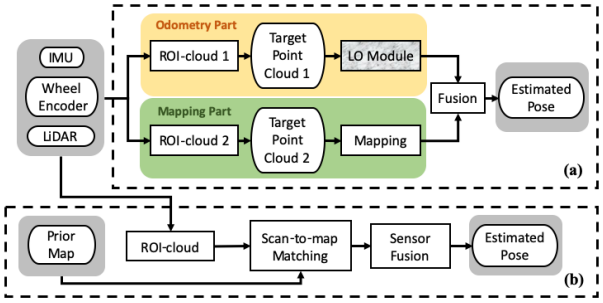


Fig. 4. LiDAR odometry (a) and localization (b) system with ROI-cloud

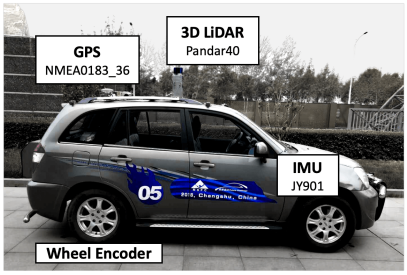


Fig. 5. The vehicle platform for testing in this paper.

here, and the posterior distribution at time t can be calculated by a Bayes filter with newest extracted features $\mathbb{F}_{\mathbb{C}_t}$:

$$\begin{aligned}\chi_t^{[i]} \sim bel(x_t) &= p(x_t \mid \mathbb{F}_{\mathbb{C}_{1:t}}, u_{1:t}) \\ &= \bar{\mu} p(\mathbb{F}_{\mathbb{C}_t} \mid x_t) \overline{bel}(x_t)\end{aligned}, \tag{7}$$

where $\bar{\mu}$ is the factor for normalization. Thus, the weight of particles can be updated through

$$\begin{aligned} w_t^{[i]} &= p(\mathbb{F}_{\mathbb{C}_t} \mid x_t^{[i]}) \\ &= \sum_{k \geqslant 0, \left\| c_{t,k} - x_t^{[i]} \right\| \leqslant D} \mathcal{N}(c_{t,k} - x_t^{[i]} \mid \mu, \sigma^2) \mathbb{F}_{\mathbb{C}_{t,k}}. \end{aligned} \tag{8}$$

Gaussian filter method $\mathcal{N}(\bullet \mid \mu, \sigma^2)$ is applied here for smoothing and noise reducing. Each particle's new weight is set to a weighted average of that cube's neighborhood within distance D. Gaussian filtering is time-consuming that parallel computing may be leveraged for improved efficiency.

Notice the difference of particles' updating between dynamic and static regions, as shown in Fig. 3. After the state prediction, the target of the static region still stays at the center of the high observed probability region, so that the target is reserved. While the moving object escapes away from the predicted region, thus being removed.

Finally, resampling reassigns uniform weights to the particle set, then an increased number of particles are near cubes of key regions, making such cubes high-probability.

D. Other Details

Particle Filter Resampling. Particles are apt to stick to several old key regions, and fails to find new key regions timely as the robot moves. So we use a large portion (80%-90%) of particles for resampling, while the rest is used for random injection in the front view of LiDAR sensor.

Uniform Distribution. To evenly distribute the key regions within the environment, we separate 360° of view into four identical subregions and apply Bayes filter framework respectively in each subregion. An even distribution of features has important significance for pose estimation and localization. Otherwise, the odometry will drift easily and the localization accuracy will decrease.

Avoidance of Excessive Accumulation. One of the natural advantages of particle filtering is that it enables particles to accumulate in high-probability regions, which is extremely helpful in removing large amounts of redundant regions. Whereas an excessive accumulation can cause degeneration of key regions. Therefore the high-probability of those regions should be dumped or limited to a threshold properly.

Monte Carlo sampling. When generating the position of target point cloud through ROI-cloud, Monte Carlo method is applied to select the most likely cubes from numerous ones. It helps to reduce time consuming and unexpected noise.

IV. ROI-cloud Application

ROI-cloud can be easily combined with existing LiDAR localization algorithms and achieves improved results. The block diagram of the ROI-cloud localization system is shown in Fig. 4. Before the traditional algorithm framework, the point cloud is filtered by the ROI-cloud. The dynamic targets and the redundant regions are filtered out, and the filtering result is used as an input of the existing method.

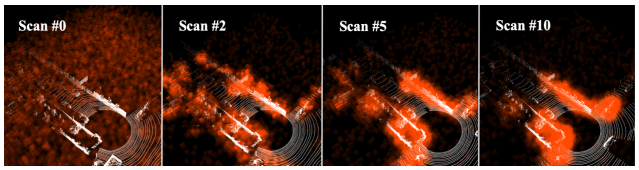
There are notable differences when applying ROI-cloud to aid the LiDAR odometry algorithm. A basic idea of the application is shown in the odometry part of Fig. 4. However, the pose estimation results of the LiDAR Odometry (LO) module will drift over time. A scan-to-map matching algorithm runs at a lower frequency for odometry correction. In our system, by using different parameters, we can obtain a larger ROI-cloud, and use corresponding target point cloud for a low-frequency map matching. This part is called the mapping part as shown in Fig. 4.

V. Experiments

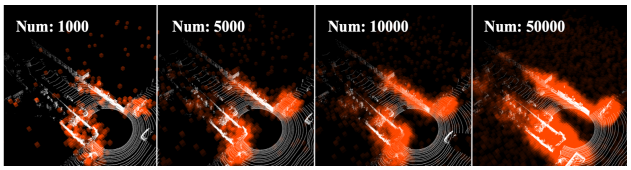
We will present experimental results in this section to verify LiDAR odometry and map-based localization with ROI-cloud. Both KITTI dataset and datasets collected from dense urban environments are used to test the proposed method. Our testing platform and the sensor configuration are depicted in Fig. 5. A multi-channel LiDAR sensor Pandar40 is mounted on top of the vehicle. Low-cost IMU sensor and wheel encoders are equipped to provide motion information. Ground truth from the real-time kinematic (RTK) GPS is used for evaluations. Our algorithms are executed on a hardware platform with a CPU of Intel Core i7-6700HQ and 16GB RAM.

The ROI-cloud algorithm is easy to combine with existing matching methods, which is verified by applying ROI-cloud to various matching algorithms in the following experiments.

Authorized licensed use limited to: Shenyang Institute of Automation. Downloaded on October 14,2024 at 10:54:55 UTC from IEEE Xplore. Restrictions apply.



(a) The convergence of key regions.



(b) The key region clustering result.

Fig. 6. (a) The particles are drawn randomly and uniformly in the first scan and gradually gather around corners of the crossroad after several scans. (b) The key region clustering result over different numbers of particles. Neither too many or too few particles are suitable for fast and accurate clustering.

TABLE I

RUNTIME OF MODULES FOR PROCESSING ROI-CLOUD (UNIT: MS)

Number of Particles	Feature Extraction	Importance Updating	Resampling	Others	Total
1000	20.1	17.2	4.1	17.5	58.9
5000	20.2	17.5	5.7	17.8	61.2
10000	20.5	19.4	9.5	18.7	68.1
50000	22.2	35.4	21.3	25.2	104.1

A. ROI-cloud Performance

The ROI-cloud is generated by the random scattering of particles, and it takes time to complete the convergence to the key regions. An example of the key region convergence process is shown in Fig. 6 (a). Experiments have proved that particles can converge in 5 to 10 scans, that is, within one second.

Another key issue is the runtime of the ROI-cloud algorithm. The resolution of the voxel cubes and the number of particles have a huge impact on the result of particle clustering. The resolution of cubes is set to $1.0m$ in urban scenes. After that, the average runtime for each module over different numbers of particles is shown in Table I. Experiments show that an excessive number of particles can cause time consumption as well as unexpected clustering result (shown in Fig. 6 (b)), thus a trade-off should be made for fast and accurate clustering. Note that a runtime less than $100ms$ does not impair the original real-time performance of the subsequent matching algorithms, since all algorithms of out method are executed in parallel structure and ROI-cloud rarely changes significantly between two consecutive scans.

B. LiDAR Odometry with ROI-cloud

We perform quantitative comparisons between LiDAR odometry with and without ROI-cloud over two experiments. Pose estimation accuracy of each method is evaluated by comparing the translational error and rotational error between the odometry and the corresponding ground truth every 100 meters. To align two trajectories, the initial pose of odometry is transformed to that of GNSS ground truth here.

1) Experiment 1: ROI-cloud aided LiDAR odometry is tested on the KITTI odometry dataset. The dataset contains a few moving objects and abundant static features, which is suitable for validation of redundant point cloud removal. All training sequences together with pose ground truth and IMU/odometry data are utilized to evaluate the system.

In this experiment, we extract 3% of the sharpest points (approximately 4000) and 10% of the flattest points (approximately 12000) as feature points, and the number of Monte Carlo sampling times is set to 20000. An example of the target point cloud generating at a crossroad is illustrated in Fig. 1. Objects such as walls, poles, and static vehicles are naturally retained in target point cloud. Actually, the semantic information of these regions is not recognized. They are just judged to have contained more features than other regions, which are sufficient for scan matching. The size of the point cloud has decreased considerably by over 64.62% per scan on average, as shown in Table II, where μ is performance improvement in percentage.

Then the classical NDT algorithm [25] is applied to determine the most probable registration transformation between target point clouds with loop closure. We downsample point cloud with a voxel grid filter before matching. Performances of two trials (with and without ROI-cloud) are shown respectively in Table II. Rotational errors are so small that

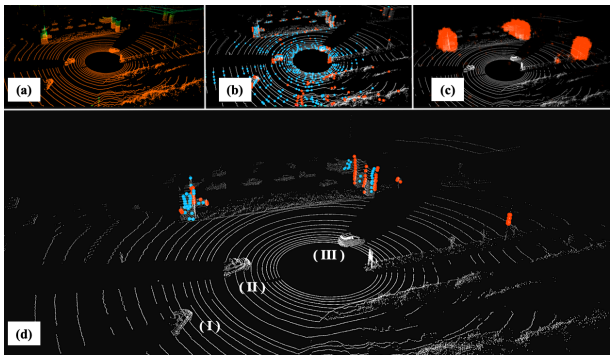


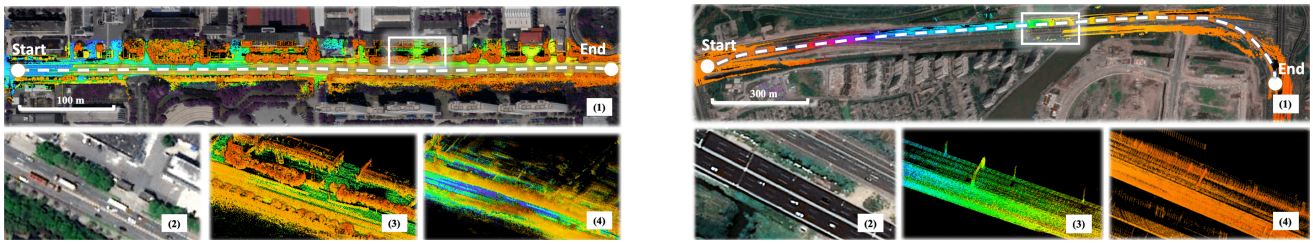
Fig. 7. (a) Original point cloud. (b) LOAM feature points. Edge feature points are in orange and surface feature points are in blue. (c) ROI-cloud. (d) Lite LOAM feature points in key regions. The feature points are clustered around the bridge piers and poles. Dynamic vehicles labeled with (I), (II) and (III) are neglected.

TABLE II

COMPARISONS OF LIDAR ODOMETRY PERFORMANCES BEFORE AND AFTER IMPLEMENTING ROI-CLOUD

KITTI Sequence	Number of Points Per Scan			Matching Runtime (ms)			Translational Error (m)		
	Before	After	μ	Before	After	μ	Before	After	μ
00	121302	43980	63.74%	606	168	72.28%	1.34	1.23	8.21%
01	120934	39870	67.03%	624	167	73.24%	13.89	14.02	-0.94%
02	122399	49734	59.37%	465	144	69.03%	7.23	7.01	3.04%
03	121243	41094	66.11%	626	160	74.44%	1.32	1.18	10.61%
04	121989	44092	63.86%	568	127	77.64%	0.39	0.36	7.69%
05	120324	41243	65.72%	458	141	69.21%	0.89	0.76	14.61%
06	126891	43783	65.50%	496	165	66.73%	0.95	0.86	9.47%
07	128002	39930	68.81%	551	159	71.14%	1.17	1.09	6.84%
08	121093	41951	65.36%	635	167	73.70%	3.96	3.68	7.07%
09	121709	43581	64.19%	636	152	76.10%	1.34	1.28	4.48%
10	120930	46966	61.16%	471	121	74.31%	1.69	1.47	13.02%
Average	122438	43293	64.62%	558	152	72.53%	3.11	2.99	7.65%

Authorized licensed use limited to: Shenyang Institute of Automation. Downloaded on October 14,2024 at 10:54:55 UTC from IEEE Xplore. Restrictions apply.



(a) Route 1 (b) Route 2

Fig. 8. In both (a) and (b), (1) is a map built from odometry, and (2)-(4) are details around the white rectangle. (2) is the corresponding satellite image. (3) is the result of LOAM with ROI-cloud and (4) is the result of original LOAM which fails in traffic flow.

they are temporarily omitted in the table. Although the target point cloud has a smaller size, it achieves the same or better odometry accuracy with regard to translational error when compared with the original cloud. It is worth noting that the runtime of NDT matching is reduced sharply by 72.53%, which is of great help for low-performance processing units.

2) Experiment 2: Two large-scale datasets are collected in dense urban environments to test the performance of ROI-cloud aided odometry. Both of them are more complex and contain numerous dynamic vehicles. Route 1 is on a busy city road with multiple buildings, trees, and sidewalks, while Route 2 spans a $2.3km$ open highway, which features lampposts, landmarks, and fences. The satellite images in which experiment 2 was performed are illustrated in Fig. 8. Combined with ROI-cloud, the state-of-the-art LOAM [6] is applied as the scan matching method in this experiment. In other words, LOAM extracts feature points in the target point cloud and completes the following pose estimation.

In the dynamic environment, it is an important purpose of our method to remove dynamic regions after clearing the redundant point cloud. Fig. 7 represents a procedure of extracting LOAM feature points in key regions. Numerous superfluous feature points are neglected, including points on the ground and sparse noises. Meanwhile, the wrong feature points adhering to moving vehicles are also removed. After filtering, these lite feature points are input to original LOAM.

When applying ROI-cloud aided LOAM to our datasets, the final translational and rotational errors are $2.87m$ and $0.89°$ (LOAM: $50.34m$ and $11.27°$) on Route 1, and $8.32m$ and $2.54°$ (LOAM: $170.34m$ and $15.92°$) on Route 2. Fig. 8 presents The final point cloud map overlaid atop a satellite image from our method. High consistency is shown among all routes obtained from LOAM with ROI-cloud, especially in dense traffic flow where the original LOAM fails to work.

C. Map-based Localization with ROI-cloud

ROI-cloud aided localization system is tested in outdoor environments. The dataset was logged from a forested u-turn where structured features are rare. The occupancy grid map [26] is built by GNSS/IMU information and serves as a prior map for an ICP-based matching algorithm.

Without ROI-cloud, the original LiDAR data is extremely large with much redundant information (Fig. 9 (a)), thus making it a slower matching process and poorer result. As a comparison, ROI-cloud helps to reduce the runtime of matching per scan by 56.8% (from $174.29ms$ to $75.32ms$), and the vehicle localizes itself successfully without failures. The final localization result of both methods is listed in Table III. More details are shown in Fig. 9.

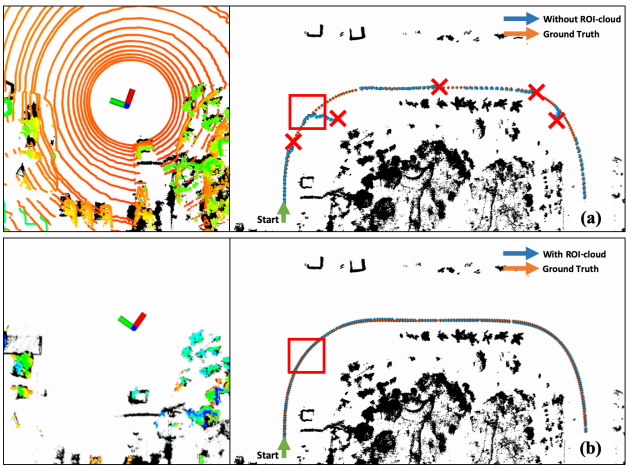


Fig. 9. (a) Localization without ROI-cloud. (b) Localization with ROI-cloud. Pose estimation fails several times with original point cloud, followed by a GNSS as reinitialization, as illustrated with red crosses. Different point sets used for localization around the red rectangle are shown on the left.

TABLE III
LOCALIZATION RESULT WITH/WITHOUT ROI-CLOUD

Methods	Longitudinal Error (m)	Lateral Error (m)	Translational Error (m)	Rotational Error ($°$)	Matching Runtime (ms)
Without	7.26	6.03	10.32	6.65	174.29
With	0.21	0.08	0.24	0.14	75.32

VI. CONCLUSIONS

In this paper, we proposed a key region extraction method of point cloud for LiDAR odometry and localization. It fuses IMU/odometry data by Bayes filtering and has the capability to remove dynamic and redundant point cloud in dense urban environments, which helps to improve speed and accuracy of existing matching algorithms as experiments proved.

The criterion of defining the feature of cubes could be discussed and improved. Also, our future work involves developing methods to combine ROI-cloud with semantic features to ensure better robustness.

Authorized licensed use limited to: Shenyang Institute of Automation. Downloaded on October 14,2024 at 10:54:55 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] Besl, P. J., and Neil D. McKay. "A method for registration of 3-D shapes." IEEE Transactions on Pattern Analysis and Machine Intelligence 14.2 (1992): 239-256.

[2] Biber, Peter, and Wolfgang Straer. "The normal distributions transform: A new approach to laser scan matching." Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453). Vol. 3. IEEE, 2003.

[3] Rusu, Radu B., and S. Cousins. "Point cloud library (pcl)." 2011 IEEE international conference on robotics and automation. 2011.

[4] Nchter, Andreas, and Joachim Hertzberg. "Towards semantic maps for mobile robots." Robotics and Autonomous Systems 56.11 (2008): 915-926.

[5] Borges, Geovany A., and Marie-Jos Aldon. "Robustified estimation algorithms for mobile robot localization based on geometrical environment maps." Robotics and Autonomous Systems 45.3-4 (2003): 131-159.

[6] Zhang, Ji, and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time." Robotics: Science and Systems. Vol. 2. 2014.

[7] Jian, Rui, et al. "A Semantic Segmentation Based Lidar SLAM System Towards Dynamic Environments." International Conference on Intelligent Robotics and Applications. Springer, Cham, 2019.

[8] Vu, Trung-Dung, Olivier Aycard, and Nils Appenrodt. "Online localization and mapping with moving object tracking in dynamic outdoor environments." 2007 IEEE Intelligent Vehicles Symposium. IEEE, 2007.

[9] Friedman, Nir, and Stuart Russell. "Image segmentation in video sequences: A probabilistic approach." Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1997.

[10] Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. MIT press, 2005.

[11] Segal, Aleksandr, Dirk Haehnel, and Sebastian Thrun. "Generalized-icp." Robotics: science and systems. Vol. 2. No. 4. 2009.

[12] Rusinkiewicz, S., and M. Levoy. "Efficient variants of the ICP algorithm." Proceedings Third International Conference on 3-D Digital Imaging and Modeling. IEEE, 2001.

[13] Rusu, Radu Bogdan, et al. "Learning informative point classes for the acquisition of object model maps." 2008 10th International Conference on Control, Automation, Robotics and Vision. IEEE, 2008.

[14] Rusu, Radu Bogdan, et al. "Fast 3d recognition and pose using the viewpoint feature histogram." 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2010.

[15] Shan, Tixiao, and Brendan Englot. "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

[16] Myronenko, Andriy, and Xubo Song. "Point set registration: Coherent point drift." IEEE transactions on pattern analysis and machine intelligence 32.12 (2010): 2262-2275.

[17] Jian, Bing, and Baba C. Vemuri. "Robust point set registration using gaussian mixture models." IEEE transactions on pattern analysis and machine intelligence 33.8 (2010): 1633-1645.

[18] Sandhu, Romeil, Samuel Dambreville, and Allen Tannenbaum. "Point set registration via particle filtering and stochastic dynamics." IEEE transactions on pattern analysis and machine intelligence 32.8 (2009): 1459-1473.

[19] Moghari, Mehdi Hedjazi, and Purang Abolmaesumi. "Point-based rigid-body registration using an unscented kalman filter." IEEE Transactions on Medical Imaging 26.12 (2007): 1708-1728.

[20] Qin, Baoxing, et al. "Curb-intersection feature based monte carlo localization on urban roads." 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012.

[21] Sefati, Mohsen, et al. "Improving vehicle localization using semantic and pole-like landmarks." 2017 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2017.

[22] Li, Minglei, Peter Wonka, and Liangliang Nan. "Manhattan-world urban reconstruction from point clouds." European Conference on Computer Vision. Springer, Cham, 2016.

[23] Dub, Renaud, et al. "Segmatch: Segment based place recognition in 3d point clouds." 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017.

[24] Siew, Peng Mun, Richard Linares, and Vibhor Bageshwar. "Simultaneous Localization and Mapping with Moving Object Tracking in 3D Range Data using Probability Hypothesis Density (PHD) Filter." 2018 AIAA Information Systems-AIAA Infotech@ Aerospace. 2018. 0507.

[25] Koide, Kenji, Jun Miura, and Emanuele Menegatti. "A Portable 3D LIDAR-based System for Long-term and Wide-area People Behavior Measurement." (2018).

[26] Guo, Lindong, et al. "Occupancy grid based urban localization using weighted point cloud." 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2016.

Authorized licensed use limited to: Shenyang Institute of Automation. Downloaded on October 14,2024 at 10:54:55 UTC from IEEE Xplore. Restrictions apply.