RDP-LOAM: Remove-Dynamic-Points LiDAR Odometry and Mapping

Xingyu Cao
School of Mechanical Engineering
Beijing Institute of Technology
Beijing, China
cxy00329@163.com

Chao Wei*
School of Mechanical Engineering
Beijing Institute of Technology
Beijing, China
weichaobit@163.com

Jibin Hu
School of Mechanical Engineering
Beijing Institute of Technology
Beijing, China
hujibin1970@163.com

Meng Ding
School of Mechanical Engineering
Beijing Institute of Technology
Beijing, China
1422741028@qq.com

Mengjie Zhang
China North Vehicle Research Institute
Beijing, China
mengjie9898@163.com

Zhong Kang
China North Vehicle Research Institute
Beijing, China
kz19800820@163.com

2023 IEEE International Conference on Unmanned Systems (ICUS) | 979-8-3503-1630-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICUS58632.2023.10318381

***Abstract*—Simultaneous Localization and Mapping (SLAM) is a critical technology for autonomous driving and robotics. However, many SLAM algorithms assume a static environment, leading to reduced robustness and accuracy in highly dynamic environments. In this study, we introduce RDP-LOAM, a real-time and robust LiDAR-based SLAM framework designed for dynamic environments. Our approach incorporates a sliding window-based method to retain historical frame information for comparative analysis. We employ probability estimation to detect and eliminate dynamic objects, and we adjust parameters adaptively based on current velocity. Subsequently, we match the static point cloud with a local submap to achieve precise poses and create static maps in highly dynamic environments. To validate our framework, we conduct extensive experiments utilizing both the open-source UrbanLoco dataset and our self-collected dataset. The results conclusively demonstrate that RDP-LOAM effectively removes dynamic points and significantly enhances odometry accuracy.**

Keywords—SLAM, LiDAR Odometry, static map, dynamic points removal.

I. Introduction

SLAM technology, enabling centimeter-level positioning independent of the Global Navigation Satellite System (GNSS), is of paramount importance in autonomous driving and robotics. With its unrivaled ranging precision, Light Detection and Ranging (LiDAR) gains extensive applications in the SLAM domain [1]-[3]. LiDAR SLAM not only provides accurate pose information but also enables the creation of three-dimensional point cloud maps [4][5]. Over recent years, researchers have proposed several outstanding LiDAR SLAM frameworks, such as LOAM [6], ALOAM [7], and FLOAM [8].

While modern LiDAR SLAM frameworks generally deliver stable and accurate performance under most circumstances, they can encounter challenges in complex scenarios [9]-[12]. For instance, in urban high-dynamic scenes, there are numerous moving objects such as pedestrians, cyclists, and vehicles. However, LiDAR SLAM methods based on registration assume that the point cloud is static, leading to significant misalignment and a decrease in system accuracy and robustness [13]. Consequently, the resulting point cloud maps contain the motion trajectories of many moving objects. Traditional methods like RANSAC [14] are unable to completely remove the point cloud of moving objects. Additionally, algorithms like Octomap [15] are time-consuming and cannot remove dynamic points in real-time. Deep learning-based methods often require extensive training data and have limited adaptability [16]-[17]. Therefore, real-time and effective removal of dynamic points is crucial for LiDAR SLAM. This is a challenging problem that requires further research and development of advanced methods to handle high-dynamic scenes and improve robustness and accuracy of the system.

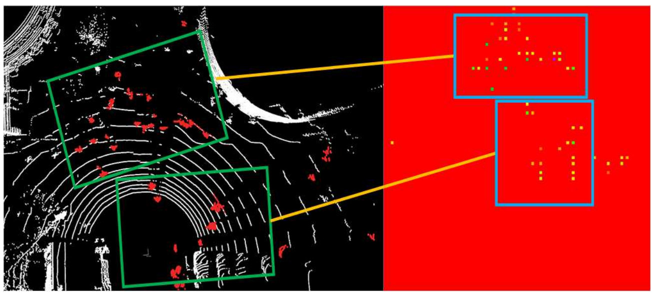


Fig. 1. The red-colored point cloud in the left represents the dynamic objects detected using the method proposed in this paper. The right image displays the corresponding differential *H-value* grid map, where the yellow-green pixels represent dynamic grid cells.

To mitigate the issues encountered in high-dynamic environments, we introduce a novel method for dynamic points removal and present the RDP-LOAM framework. To preclude mismatches due to dynamic points, we project the point clouds onto two-dimensional grids, introducing the concept of an *H-value* to represent the z-direction distribution of point clouds within each grid. By comparing the *H-value* of each grid with history frames stored in a sliding window, we obtain a differential *H-value* grid map as showed in Fig. 1. Dynamic grid cells are identified based on an adaptive threshold, and the point clouds are then clustered to effectively remove dynamic points. The point cloud matching process employed is similar to FLOAM's approach, consequently achieving accurate pose estimation and a static map. We compare our proposed method with FLOAM and ALOAM on the UrbanLoco dataset [18] and our own collected dataset. The results demonstrate the effectiveness of our method in removing dynamic points and improving odometry accuracy.

In conclusion, our main contributions are as follows:

- We propose the concept of *H-value* to represent the distribution of the two-dimensional grid in the vertical direction and utilize a differential *H-value* grid map to remove dynamic points.
- We feed the point cloud after filtering out the dynamic points into the SLAM system, which significantly

* The corresponding author.

improves the accuracy of the odometry in dynamic environments and results in a static point cloud map.

II. Related Works

Numerous scholars have conducted research on the removal of dynamic points, and the methods for removing dynamic points in LiDAR SLAM can be categorized as follows:

Map-based methods: This approach aims to remove dynamic points based on a pre-built map and the motion trajectory of the LiDAR. For instance, Lim et al. [19] propose a grid-based method that utilizes the concept of pseudo occupancy to represent the spatial occupancy of objects. This method differentiates between dynamic and static points by fitting a region-based ground model. Johannes Schauer et al. [20] propose a voxel occupancy grid approach, where the grid is traversed along the LiDAR path to detect occupancy differences at different time instances, enabling the identification of dynamic points within the grid. Such methods typically consume significant resources and are generally not capable of real-time execution.

Visibility-based methods: This approach relies on comparing consecutive frames without the need for maintaining large-scale maps. It associates query points with points within a certain viewing angle, considering points that are occluded in the far distance as static and those in close proximity as dynamic. For example, Chenglong Qian et al. [21] estimate the initial pose using an Inertial Measurement Unit (IMU) and subsequently employs adaptive resolution distance images to compare the differences between the current point cloud and the distance images corresponding to previous frames. These differences are then mapped back to the point cloud to differentiate dynamic points from static points. Kim et al. [22] first remove dynamic points as much as possible using distance images and then restore the falsely detected dynamic points as static points through multi-resolution distance images. Such methods typically fail in situations where LiDAR is heavily occluded.

Segmentation-based methods: This approach typically involves clustering of point clouds to remove dynamic points. For instance, Xiao Hu et al. [23] detect and remove dynamic points by calculating the similarity between clusters corresponding to multiple scans. If the position of an object changes between multiple scan frames, it is considered dynamic. David J et al [24] compare the current frame with the previous and subsequent frames to identify dynamic points. Then, they use the dynamic points as seed points for cluster growth. Methods of this kind are heavily influenced by their own motion, resulting in a significant impact on accuracy and a higher likelihood of false detections.

Deep learning-based methods: This approach employs neural networks for the detection and segmentation of dynamic point clouds. For example, Patrick Pfreundschuh et al. [25] utilize the 3D-MiniNet network for real-time dynamic objects detection, followed by conventional LiDAR SLAM using the filtered point cloud. Yu-Kai Lin et al. [26] employ the SE-SSD framework for dynamic objects tracking and removal. These types of methods typically require a large amount of training data and generally exhibit moderate generalization performance.

III. Methods

A. System Overview

Fig. 2 illustrates the overall framework of RDP-LOAM, consisting of three main modules: dynamic points removal, mapping, and history information sliding window.

Firstly, within the dynamic points removal module, the point cloud is projected onto a 2D plane. A comparison ensues between the current 2D grids and the historical data within the sliding window to identify dynamic grids. Cluster-based methods are then employed to remove dynamic objects. The mapping module involves several steps, including feature extraction, point cloud registration, and state estimation. Edge and surface features are extracted and aligned with local submap to optimize the pose. The history information sliding window module primarily maintains a sliding window that stores history frames information, serving as a reference for the dynamic points removal module.

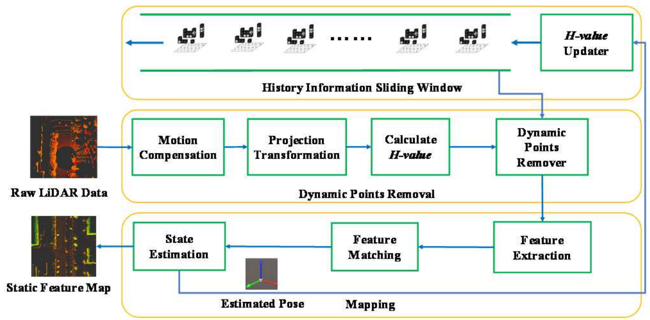


Fig. 2. Overview of our method. The dynamic points removal module is tasked with negating the impact of dynamic points on point cloud registration and building a static map. The mapping module extracts features from the point cloud and conducts matching to acquire optimized poses and static feature maps. The history information sliding window module prevents the need for frequent global point cloud transformations..

B. Definition of H-value

To efficiently represent the distribution of point clouds and minimize memory usage, we introduce the concept of *H-value*. This measure quantifies the distribution of point clouds in the vertical direction relative to the ground.

Specifically, we first divide the point cloud into two-dimensional grids, and convert the points to two-dimensional planes through mapping function $\Gamma(\cdot)$. $\Gamma(\cdot)$ is defined as

$$\Gamma(\boldsymbol{p})=(\mathcal{X},\mathcal{Y})=(floor(x\cdot res_X),floor(y\cdot res_Y)), \quad (1)$$

where res_X and res_Y respectively represent the resolutions of the grids in the x and y directions. $(\mathcal{X},\mathcal{Y})$ is the indices of the grid in which the point $\boldsymbol{p}=(x,y,z)$ resides.

In general, a grid typically contains multiple points. Here, we use $\mathcal{G}_{ij}^{k}$ to represent the grid in the *k-th* frame of the point cloud with the index (i,j) . $\mathcal{G}_{ij}^{k}$ is defined as

$$\mathcal{G}_{ij}^{k}=\{\boldsymbol{p}^{k}\mid\Gamma(\boldsymbol{p}^{k})=(i,j)\}. \quad (2)$$

Expanding upon this foundation, we proceed to partition the grid $\mathcal{G}_{ij}^{k}$ into smaller subgrids in the vertical dimension by mapping function $\Pi(\cdot)$, as illustrated in Fig. 2. We employ the notation g_{ij}^{h} to denote the individual subgrid within $\mathcal{G}_{ij}^{k}$, while b_{ij}^{h} represents the binary value associated with the subgrid g_{ij}^{h}. $\Pi(\cdot)$, g_{ij}^{h} and b_{ij}^{h} are defined as follows

$$\Pi(\boldsymbol{p}) = floor((z - z_{\min}) \cdot res_Z), \tag{3}$$

$$g_{ij}^{h} = \{\boldsymbol{p}_{ij}^{k} | \Pi(\boldsymbol{p}_{ij}^{k}) = h,\ \boldsymbol{p}_{ij}^{k} \in \mathcal{G}_{ij}^{k}\}, \tag{4}$$

$$b_{ij}^{h} = \begin{cases} 1,\ size(g_{ij}^{h}) \geq \lambda_{\min} \\ 0,\ size(g_{ij}^{h}) < \lambda_{\min} \end{cases}, \tag{5}$$

where $\lambda_{\min}$ represents the threshold value for the number of points within the subgrid g_{ij}^{h}. $z_{\min}$ is the minimum value set for the z direction. res_Z is the resolution in the z direction. If the number of points within the subgrid g_{ij}^{h} is at least $\lambda_{\min}$, b_{ij}^{h} is assigned 1; otherwise, b_{ij}^{h} is assigned 0.

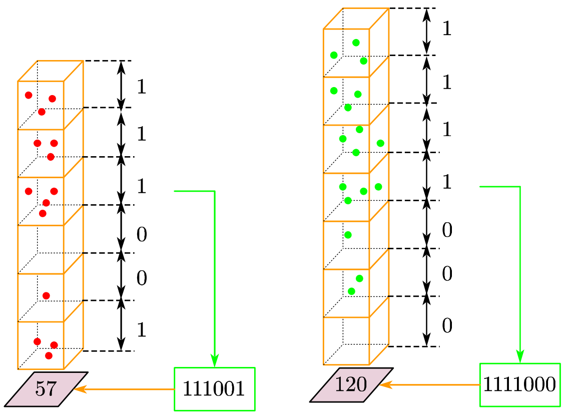


Fig. 3. Map the point cloud to two-dimensional grids. Here we set $\lambda_{\min}$ to be 3.

Then, we define the *H-value* of $\mathcal{G}_{ij}^{k}$ as

$$H\text{-}value(\mathcal{G}_{ij}^{k}) = \sum_{h=0}^{h_{\max}} 2^{h} \cdot b_{ij}^{h}. \tag{6}$$

The binary form of the *H-value* represents the occupancy status of the grid in the vertical direction. By utilizing the *H-value*, three-dimensional point cloud information can be compressed into a two-dimensional grid. Consequently, the disparity in vertical distribution between two grids can be determined by calculating the Hamming distance of their respective *H-value*. Typically, computer processing of bitwise operations is fast and efficient.

The distribution disparity of corresponding grids between *k-th* frame and *l-th* frame can be represented by Hamming Distance $\mathcal{D}_{ij}^{k,l}$, $\mathcal{D}_{ij}^{k,l}$ can be computed quickly by

$$\mathcal{D}_{ij}^{k,l} = f\ (H\text{-}value(\mathcal{G}_{ij}^{k}) \oplus H\text{-}value(\mathcal{G}_{ij}^{l})), \tag{7}$$

where $\oplus$ represents the XOR operation, and $f(\cdot)$ calculates the number of occurrences of 1 in each binary bit.

C. Dynamic Points Removal

Leveraging the *H-value* concept, we present a real-time approach for effectively removing dynamic points. To reduce the necessity of frequent pose transformations across the entire point cloud map, we perform comparisons within a unified global coordinate system. Furthermore, we utilize a sliding window to store the *H-value* maps derived from the transformed point clouds of the preceding frames. By assuming a consistent motion state of the carrier within a short temporal window, we can accurately predict the current pose $\tilde{T}_{k}^{W}$:

$$\Delta T = (T_{k-2}^{W})^{-1} \cdot T_{k-1}^{W}, \tag{8}$$

$$\tilde{T}_{k}^{W} = T_{k-1}^{W} \cdot \Delta T. \tag{9}$$

By utilizing the predicted pose, we convert the point cloud of the current frame to the global coordinate system. Subsequently, we compute the *H-value* for the point cloud and generate a differential *H-value* map by comparing it with the N frames stored in the sliding window. We use the probability density function $PbD(\cdot)$ to represent the proportion of dynamic points in the grid, which can be calculated by the differential *H-value* map. The probability density function $PbD(\cdot)$ of a grid can be computed by

$$PbD(\mathcal{G}_{ij}^{k}) = \frac{1}{N} \sum_{l=k-N}^{N} \frac{1}{\sqrt{2\pi\sigma_d^2}} exp\left(-\frac{1}{2}\left(\frac{\mathcal{D}_{ij}^{k,l}}{\sigma_d}\right)^{2}\right), \tag{10}$$

where σ_d is the deviation over the differential *H-value* map for each grid. N is the size of the sliding window, which affects the estimation of the dynamic grid. If N is too large, the coincidence area between the front and back frames is too small, and accurate differential *H-value* map cannot be generated.

σ_d is estimated using the median absolute deviation for all the grids in the sliding window as

$$\sigma_d \approx \frac{K \cdot Median(\mathcal{D}_{ij}^{k,l})}{\sqrt{2}}, \tag{11}$$

where K is a constant parameter. We can set thresholds to distinguish dynamic grids from static ones. We define $\mathcal{L}_{ij}^{k}$ to indicate whether the grid $\mathcal{G}_{ij}^{k}$ is dynamic or static in the following way:

$$\mathcal{L}_{ij}^{k} = \begin{cases} dynamic,\ PbD(\mathcal{G}_{ij}^{k}) \geq \delta_k \\ static,\ PbD(\mathcal{G}_{ij}^{k}) < \delta_k \end{cases}, \tag{12}$$

$$\delta_k = \frac{K'}{\sqrt{\pi \bar{v}}}, \tag{13}$$

where δ_k is the threshold that determines whether a grid is static or dynamic and is affected by the average speed $\bar{v}$ between two frame point clouds. K' is a constant parameter.

D. Ground Segmentation and Clustering

In urban environments, the movement speed of certain large vehicles can be low, leading to minimal variation in measurements across successive frames. Thus, solely using the aforementioned method may not fully eliminate some dynamic points. Therefore, clustering techniques become essential to thoroughly remove dynamic points showing little change.

Generally, before point cloud clustering, interference from ground points should be eliminated. In order to ensure the real-time performance, we use Ray-Ground-Filter [27] for ground segmentation and Breadth First Search (BFS) algorithm for clustering [28]. For clustered point cloud, there is a small number of points that are incorrectly detected as dynamic. We can calculate the percentage of dynamic points within each point cloud cluster, and only consider a point cloud cluster as dynamic if the percentage exceeds a certain threshold ε. The result of whether an object is dynamic is as follows:

$$\mathcal{L}(obj)=\begin{cases} dynamic,\ dynamic/sum \geqslant \varepsilon \\ static,\ dynamic/sum < \varepsilon \end{cases}. \qquad (14)$$

E. Feature Extraction

In order to minimize the computational resources required for point-to-point Iterative Closest Point (ICP) [29] matching, while simultaneously ensuring match accuracy, we deploy a feature extraction approach on point clouds. Specifically, we adopt the FLOAM feature extraction method, where points with low surface roughness are classified as surface features, while points with high surface roughness are classified as edge features. We then search for matching points of the current point cloud in the static map. The distance calculation formulas for point-to-edge and point-to-surface are as follows:

$$d_k^e=\frac{|(\boldsymbol{p}_{i+1,k}^e-\boldsymbol{p}_{i,u}^e)\times(\boldsymbol{p}_{i+1,k}^e-\boldsymbol{p}_{i,v}^e)|}{|\boldsymbol{p}_{i,u}^e-\boldsymbol{p}_{i,v}^e|}, \qquad (15)$$

$$d_k^p=\frac{\left|\begin{matrix}(\boldsymbol{p}_{i+1,k}^p-\boldsymbol{p}_{i,w}^p)\\ (\boldsymbol{p}_{i,w}^p-\boldsymbol{p}_{i,u}^p)\times(\boldsymbol{p}_{i,w}^p-\boldsymbol{p}_{i,v}^p)\end{matrix}\right|}{|(\boldsymbol{p}_{i,w}^p-\boldsymbol{p}_{i,u}^p)\times(\boldsymbol{p}_{i,w}^p-\boldsymbol{p}_{i,v}^p)|}, \qquad (16)$$

where i, k, u and v are indices of feature points.

IV. Experiments

We perform extensive experiments to evaluate the performance of our RDP-LOAM framework and compare it with ALOAM and FLOAM. Firstly, we assess the odometry accuracy on the publicly available high-dynamic UrbanLoco dataset. Additionally, we record our own dataset to evaluate the accuracy of odometry and the effectiveness of dynamic points removal. The algorithms are implemented in C++ and employed the Robot Operating System (ROS) with Ubuntu 20.04. All experiments are performed on a laptop computer equipped with 16GB of RAM and an Intel i5-12500H CPU.

A. Performance on UrbanLoco Dataset

The UrbanLoco dataset comprises a large number of moving pedestrians and vehicles, exhibiting high-dynamic urban scenes, as depicted in Fig. 4. This dataset serves as a means to validate the accuracy and robustness of LiDAR SLAM algorithms in high-dynamic environments. We conduct a comparative analysis between our RDP-LOAM, ALOAM, and FLOAM. The Table I presents the absolute trajectory accuracy of all trajectory accuracy of all methods on this dataset. We use root mean square error (RMSE) to measure absolute trajectory accuracy. Additionally, Fig. 5 illustrates the trajectory comparison between the three methods and the ground truth on one of the datasets, while Fig. 6(a) shows the point cloud maps before and after dynamic points removal.

Fig. 4. Some images from the UrbanLoco dataset, which contains a large number of dynamic objects.

TABLE I. RMSE[M] OF LO SYSTEMS ON URBANLOCO

Dataset	ALOAM	FLOAM	Proposed
2019033HH	1.712	1.827	**0.494**
20190331_NJ_SL	2.942	3.085	**0.516**
20190331WH	2.426	2.463	**0.496**
CAColiTower	6.673	8.249	**3.436**
CALombardStreet	4.853	4.716	**2.370**

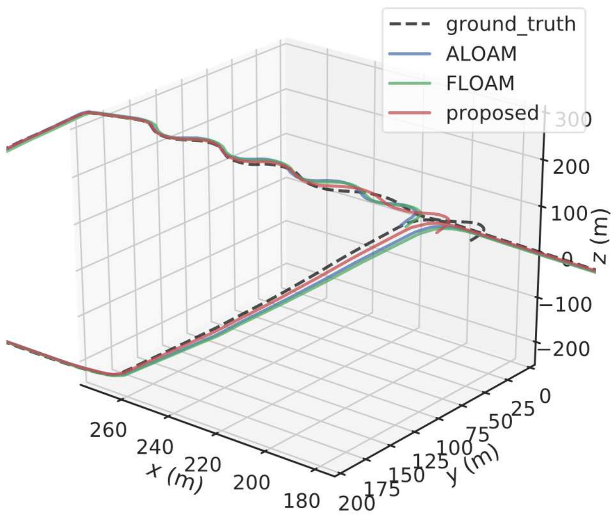
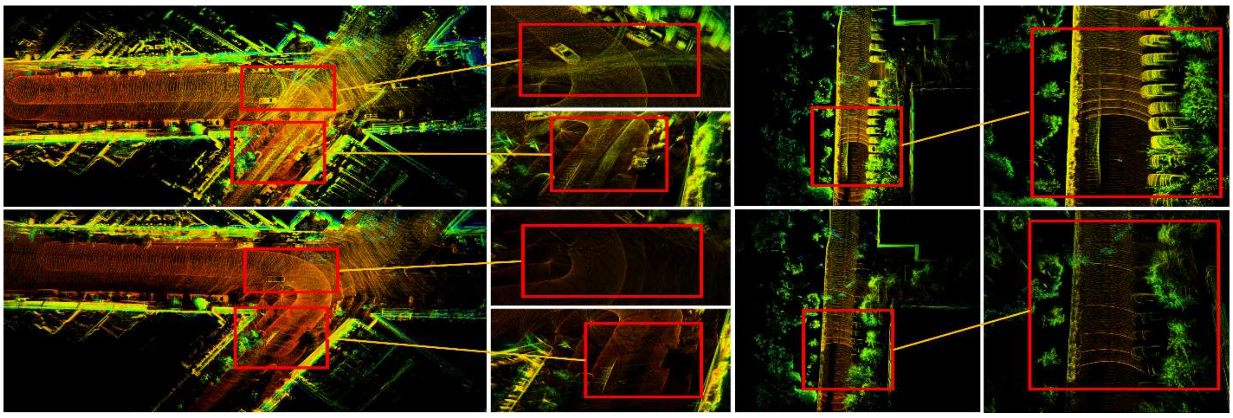


Fig. 5. Comparison details of trajectory between ALOAM, FLOAM, and RDP-LOAM on the CALombardStreet dataset.

From the Table I, it is evident that in dynamic environments, RDP-LOAM performs better than FLOAM


(a) CAColiTower

(b) our dataset

Fig. 6. The figure above (a) and (b) respectively illustrate the effectiveness of our algorithm in dynamic points removal on the UrbanLoco dataset and our dataset. The top images in (a) and (b) represent the maps generated by FLOAM, where (a) showcases intersection scenes with multiple vehicle trajectories, and (b) includes trajectories of oncoming vehicles. The bottom images in (a) and (b) depict the static maps generated by RDP-LOAM, with a detailed comparison before and after dynamic points removal highlighted within the red boxes.

and ALOAM. RDP-LOAM shows an approximately 50% improvement in absolute trajectory accuracy compared to FLOAM. Through Fig. 5, we can visually perceive the differences in accuracy among the three LO systems. Thus, in dynamic environments, effectively eliminating the influence of dynamic points can significantly enhance odometry accuracy.

B. Performance on our dataset

We collect three datasets in congested sections of the Beijing Institute of Technology campus to evaluate the accuracy and dynamic points removal effectiveness of RDP-LOAM. As shown in Fig. 7, we use the miniEV as the experimental vehicle, equipped with a RoboSense 32-line LiDAR for point cloud collection. Additionally, we use the output pose from the onboard Real-Time Kinematic (RTK) device as the ground truth. Our datasets contain a large number of pedestrians and cyclists, which significantly interfere with the SLAM system. By using RDP-LOAM to remove dynamic points, we can obtain static maps that do not include the long point cloud trajectories generated by dynamic objects, as shown in Fig. 6(b). Compared to FLOAM, the maps generated by RDP-LOAM exhibit significantly improved quality. We also evaluate the precision and recall rate of the dynamic points removal algorithm using our own dataset. The calculation methods for precision and recall rate are as follows:

$$\begin{cases} Precision = TP/(TP+FP) \\ \text{Re}call = TP/(TP+FN) \end{cases}, \tag{17}$$

where *TP* (True Positive) and *FP* (False Positive) represent the number of correctly identified dynamic points and incorrectly identified dynamic points, respectively; *FN* (False Negative) represents the number of static points mistakenly identified as dynamic.

We compare our algorithm with ERASOR [20] and Removert [23], and the results are presented in Table II. Our algorithm achieves an average precision and recall rate of 90.2% and 92.1% respectively, surpassing both ERASOR and Removert. These results indicate that our algorithm is capable of correctly and effectively removing dynamic points.

The odometry accuracy of RDP-LOAM is also superior to ALOAM and FLOAM, as shown in Fig. 8 for comparison. Additionally, our dynamic points removal module exhibits excellent performance in terms of processing speed, with the time consumption of each module presented in Table III. Therefore, RDP-LOAM can meet the real-time requirements for dynamic points removal.

TABLE II. COMPARISON OF DYNAMIC POINTS REMOVAL PERFORMANCE USING OUR DATASET

Dataset	Method	Precision	Recall
Crossroad	Removert	60.3%	67.4%
	ERASOR	72.2%	70.1%
	proposed	**90.1%**	**91.7%**
Compass Main Road	Removert	65.8%	62.1%
	ERASOR	78.6%	71.8%
	proposed	**91.3%**	**92.5%**
Canteen Intersection	Removert	70.1%	71.3%
	ERASOR	79.1%	76.9%
	proposed	**90.5%**	**92.0%**

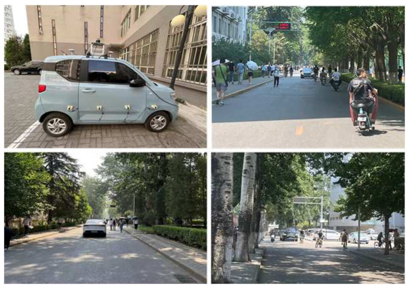


Fig. 7. The vehicle in the top left corner is our data acquisition vehicle, while the others represent our data acquisition environment.

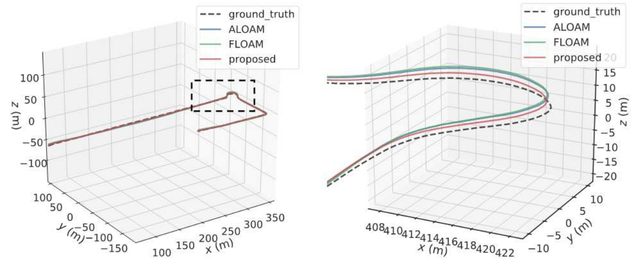


Fig. 8. Comparison details of trajectory between ALOAM, FLOAM, and RDP-LOAM on the Compass Main Road dataset.

TABLE III. RUN TIME [MS] OF EACH MODULE FOR ONE SCAN

Module	*H-value Calculation*	Ground Segmentation and Clustering	Mapping
Run time	2.1	15.3	30.7

V. CONCLUSION

In this paper, we introduce an efficient, real-time methodology for dynamic objects detection that effectively erases dynamic points from the initial point cloud. The resulting static point cloud is then employed as input for a Lidar Odometry (LO) system based on FLOAM. This technique exhibits substantial improvements in odometry accuracy and facilitates the creation of static maps within high-dynamic environments. Notably, our proposed method does not depend on extensive training data and can conform to different types of LiDAR sensors, thereby ensuring reliable performance in various high-dynamic situations. Nonetheless, there exists scope for enhancement within the RDP-LOAM framework. Specifically, we identify a decrease in object removal accuracy when handling objects with slower motion speeds. In forthcoming work, our focus will be to further refine this system, boosting both the accuracy and speed of dynamic point eradication.

REFERENCES

[1] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, “Visual slam for automated driving: Exploring the applications of deep learning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018, pp. 247–257.

[2] Masashi Yokozuka, Kenji Koide, Shuji Oishi and Atsuhiko Banno, “LiTAMIN: LiDAR-based Tracking And MappINg by Stabilized ICP for Geometry Approximation with Normal Distributions” in 2020 IEEE/RSJ InternationalConference on Intelligent Robots and Systems (IROS), 2020, pp. 5143-5150.

[3] W. Xu and F. Zhang, “FAST-LIO: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter,” IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3317–3324, 2021.

[4] A. Thakur, B. Anand, H. Verma and P. Rajalakshmi, “Real Time Lidar Odometry and Mapping and Creation of Vector Map,” in 2022 8th International Conference on Automation, Robotics and Applications (ICARA), 2022, pp. 181-185.

[5] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann et al., “Stanley: The robot that won the darpa grand challenge,” Journal of field Robotics, vol. 23, no. 9, pp. 661–692, 2006.

[6] Y. Li and J. Ibanez-Guzman, “Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems,” IEEE Signal Processing Magazine, vol. 37, no. 4, pp. 50–61, 2020.

[7] J. Zhang and S. Singh, “Loam: Lidar odometry and mapping in real-time,” Robotics: Science and Systems, vol. 2, no. 9, pp. 1–9, 2014.

[8] HKUST-Aerial-Robotics (2019) A-LOAM [Source code]. https://github.com/HKUST-Aerial-Robotics/A-LOAM.

[9] Han Wang, Chen Wang, Chun-Lin Chen, and Lihua Xie, “F-LOAM : Fast LiDAR Odometry and Mapping,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021. pp. 4390-4396.

[10] Tingxiang Fan, Bowen Shen, Hua Chen, Wei Zhang and Jia Pan, “DynamicFilter: an Online Dynamic Objects Removal Framework for Highly Dynamic Environments,” in 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 7988-7994.

[11] J. Park, Y. Cho and Y. S. Shin, “Nonparametric Background Model-Based LiDAR SLAM in Highly Dynamic Urban Environments,” IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 12, pp. 24190-24205, 2022.

[12] Z. Chen, Y. Qi, S. Zhong, D. Feng, Q. Chen and H. Chen, “SCL-SLAM: A Scan Context-enabled LiDAR SLAM Using Factor Graph-Based Optimization,” in 2022 IEEE International Conference on Unmanned Systems (ICUS), 2022, pp. 1264-1269.

[13] Min Zhao, Xin Guo, Le Song, Baoxing Qin, Xuesong Shi, Gim Hee Lee and Guanghui Sun, “A General Framework for Lifelong Localization and Mapping in Changing Environment,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 3305-3312.

[14] Martin A. Fischler and Robert C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” Comm. ACM, vol. 24, no. 6, pp. 381–395, 1981.

[15] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard,”OctoMap: An efficient probabilistic 3D mapping framework based on octrees,” Auton. Robots, vol. 34, no. 3, pp. 189–206, 2013.

[16] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, “SuMa++: Efficient LiDAR-based semantic SLAM,” in 2019 IEEE/RSJ International. Conference. Intelligent. Robots and Systems. (IROS), 2019, pp. 4530-4537.

[17] P. Pfreundschuh, H. F. C. Hendrikx, V. Reijgwart, R. Dubé, R. Siegwart and A. Cramariuc, “Dynamic Object Aware LiDAR SLAM based on Automatic Generation of Training Data,” in 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 11641-11647.

[18] W. Wen, Y. Zhou, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, and L.-T. Hsu, “Urbanloco: a full sensor suite dataset formapping and localization in urban scenes,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 2310–2316.

[19] H. Lim, S. Hwang, and H. Myung, “ERASOR: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3D point cloud map building,” IEEE Robotics and Automation. Letters., vol. 6, no. 2, pp. 2272–2279, 2021.

[20] J. Schauer and A. Nüchter, “The Peopleremover—Removing dynamic objects from 3D point cloud data by traversing a voxel occupancygrid.” IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1679–1686, 2018.

[21] Chenglong Qian, Zhaohong Xiang, Zhuoran Wu and Hongbin Sun, “RF-LIO: Removal-First Tightly-coupled Lidar Inertial Odometry in High Dynamic Environments,” in 2021 IEEE/RSJ InternationalConference on Intelligent Robots and Systems (IROS), 2021, pp. 4421-4428.

[22] G. Kim and A. Kim, “Remove, then Revert: Static Point cloud MapConstruction using Multiresolution Range Images,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 10758-10765.

[23] Xiao Hu, Li Yan, Hong Xie, Jicheng Dai, Yinghao Zhao and Shan Su, “A Novel Lidar Inertial Odometry with Moving Object Detection for Dynamic Scenes,” in 2022 IEEE International Conference on Unmanned Systems (ICUS), 2022, pp. 356-361.

[24] David J. Yoon, Tim Y. Tang, and Timothy D. Barfoot, “Mapless Online Detection of Dynamic Objects in 3D Lidar,” in 2019 16th Conference on Computer and Robot Vision (CRV), 2019, pp. 113-120.

[25] Patrick Pfreundschuh, Hubertus F.C. Hendrikx, Victor Reijgwart, Renaud Dub´e,Roland Siegwart and Andrei Cramariuc, “Dynamic Object Aware LiDAR SLAM based on Automatic Generation of Training Data,” in 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 11641-11647.

[26] Yu-Kai Lin, Wen-Chieh Lin and Chieh-Chih Wang, “Asynchronous State Estimation of Simultaneous Ego-motion Estimation and Multiple Object Tracking for LiDAR-Inertial Odometry,” in 2023 International Conference on Robotics and Automation (ICRA), 2023, pp. 10616-10622.

[27] M. Himmelsbach, F. v. Hundelshausen and H. J. Wuensche, “Fast segmentation of 3D point clouds for ground vehicles,” in 2010 IEEE Intelligent Vehicles Symposium (IV), 2010, pp. 560-565.

[28] T. Shan and B. Englot, “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), 2018, pp. 4758–4765.

[29] K. S. Arun, T. S. Huang and S. D. Blostein, “Least-Squares Fitting of Two 3-D Point Sets,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-9, no. 5, pp. 698-700, 1987.