2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3315-4393-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IROS60139.2025.11246976

LiDAR-Inertial Odometry in Dynamic Driving Scenarios using Label Consistency Detection

Zikang Yuan[1], Xiaoxiang Wang[2], Jingying Wu[2], Junda Cheng[2] and Xin Yang[2*]

***Abstract*— In this paper, a LiDAR-inertial odometry (LIO) method that eliminates the influence of moving objects in dynamic driving scenarios is proposed. This method constructs binarized labels for 3D points of current sweep, and utilizes the label difference between each point and its surrounding points in global map to identify moving objects. The surrounding points in global map are localized by voxel-location-based nearest neighbor search, without involving any massive computations. In addition, the proposed method is embeded into a LIO system (i.e., Dynamic-LIO), and achieves state-of-the-art performance on public datasets with extremlely low computational overhead (i.e., 1∼9ms/sweep). We have released the source code of this work for the development of the community.**

I. Introduction

For LiDAR-inertial odometry (LIO) in actual driving scenarios, moving vehicles or pedestrians can leave ghost tracks in the map, leading to cumulative errors in state estimation and providing erroneous observational information for obstacle avoidance. In addition, given the necessity for real-time state estimation and mapping in a LIO system, the computational cost of removing dynamic objects must be within the processing time budget for a single sweep. Hence, efficiently identifying moving objects from 3D point clouds is crucial.

To address the issues caused by dynamic objects, researchers employ a range of approaches such as point correlation, visibility, occupancy probability and semantic information to identify and remove dynamic 3D points from input LiDAR sweeps. For point correlation [5], the construction and maintenance of a correlation graph involve computing pairwise Euclidean distances of batch 3D points, and the entire process requires substantial computational resources. For visibility [6], [11], [27], the generation of re-projected image plane entails a computational process to map large number of LiDAR points from their original 3D space onto the 2D image plane. For occupancy probability [20], [26], the estimation of an occupancy grid map need to combine multiple nearest static submaps and count the occupancy status of each voxel in each submap. The three aforementioned methods leverage extensive quantitative geometric computations and global statistics to identify moving objects in 3D point clouds, and they incur high computational costs, which limits their integration into LIO systems. Deep learning methods [4], [17] use learned semantic information for rapid qualitative separation of dynamic objects. However, they depend on extensive labeled data, risk failure with unlabeled classes, and necessitate powerful Graphic Processing Units (GPUs) for real-time operation.

In contrast to existing approaches involving extensive quantitative geometric computations, global statistics or prior semantic information to classify moving objects and static environments, in this study, a qualitative identification criterion based on label difference with nearest neighbors is used to identify 3D points on moving objects. Since moving vehicles and pedestrians are both situated on the ground, the potential dynamic points only exist in non-ground space, while the bases of moving objects are adjacent to the ground. Based on this characteristic, a label consistency detection method, which can fastly identify moving objects without any prior semantic informations, is proposed to classify moving objects and static environments in driving scenarios. The core idea of the proposed label consistency detection method consists of two parts. First, a fast 2D connected components [7] is utilized to divide the 3D points of current sweep into binarized labels, i.e., ground points and non-ground points. Then, dynamic points are determined by comparing label consistency with nearest neighbors, which are directly localized by a voxel-location-based nearest neighbor search method. The whole dynamic point identification process including voxel-location-based nearest neighbor search and label consistency comparison, is devoid of any operations about geometric computations or global statistics. This characteristic ensures that the proposed method has the advantage of a expremely low computational overhead (i.e., 1∼9ms/sweep). Finally, a LIO system (i.e., Dynamic-LIO) that uses this label consistency detection method is proposed to eliminate the influence of moving objects in driving scenarios. Experimental results on six public datasets demonstrate the outstanding performance of Dynamic-LIO in both static and dynamic scenes.

To summarize, the main contributions of this work are three aspects: 1) We propose a label consistency detection method for fast identification of 3D dynamic points. It circumvents the operations of extensive geometric computations or global statistics and thus achieves lightweight; 2) We inte-

This research is supported by the National Key R&D Program of China (2024YFE0217700), National Natural Science Foundation of China (62472184) and the Fundamental Research Funds for the Central Universities.

[1]Zikang Yuan is with AI Chip Center for Emerging Smaert Systems, Hong Kong University of Science and Technology, Hong Kong. (E-mail: zikangyuan@ust.hk)

[2]Xiaoxiang Wang, Jingying Wu, Junda Cheng and Xin Yang* are with the Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, 430074, China. (* represents the corresponding author. E-mail: m202272556@hust.edu.cn; m202177065@hust.edu.cn; jundacheng@hust.edu.cn; xinyang2014@hust.edu.cn)

grate the proposed label consistency detection method into a LIO system in a unified manner, improving the accuracy of pose estimation in dynamic scenes by eliminating the ghost tracks in the reconstructed map; 3) We have released the source code of our approach to facilitate the development of the community[1].

II. Related Work

Pomerleau el. al. [18] utilized the motion pattern of point to represent the correlation. They calculated the motion pattern of each 3D point and infer the dominant motion patterns within the map, then determined the points that do not fit the motion pattern as dynamic points. Dai et. el. [5] utlized the relative position between two points to represent the correlation, and then utilized the amplitude of relative position change over time as the criterion of consistency to identify dynamic points. However, the computational overhead of calculating motion pattern and maintaining map point correlation in large-scale outdoor scenarios is prohibitive. Yoon et. al. [27] proposed to simply query one sweep against another, and identify point with evident visibility difference as dynamic point. Removert [11] proposed a multi-resolution range image-based false prediction reverting algorithm. This method first conservatively retained definite static points and iteratively recover more uncertain static points by enlarging the query-to-map association window size. However, visibility-based approaches usually suffers from incidence angle ambiguity and occlusion issues. In addition, the generation of multiple projections on the spherical image plane and the assignment of a static value to each point in visibility-based approaches require high computational overhead. OctoMap [8] firstly proposed a framework to generate volumetric 3D environment model, which is based on octrees and uses probabilistic occupancy estimation. Erasor [15] proposed the concept called pseudo occupancy to express the occupancy of unit space and then discriminate spaces of varying occupancy. Then, the region-wise ground plane fitting (R-GPF) method is adopted to distinguish static points from dynamic points within the candidate bins that potentially contain dynamic points. DORF [3] proposed a novel coarse-to-fine offline framework that exploits global 4D spatial-temporal LiDAR information to achieve clean static point cloud map generation. DORF first conservatively preserved the definite static points leveraging the receding horizon sampling (RHS) mechanism, then gradually recovered more ambiguous static points, guided by the inherent characteristic of dynamic objects in urban environments. [20] proposed to incrementally estimate high confidence free-space areas by modeling and accounting for sensing, state estimation, and mapping limitations [16], [24] during online robot operation. RH-Map [26] proposed a novel map construction framework based on 3D region-wise hash map structure, which adopts the two-layer 3D region-wise hash map structure and the region-wise ground plane estimation for dynamic object removal. Occupancy map-based approaches are usually accompanied by the nearest neighbor search, confidence calculation, occupancy probability statistics, relative spatial position calculation and other operations requiring batch geometric computation, which cause a significant computational burden.

In recent years, a limited number of LIO systems [12]–[14] with online removing dynamic objects have been proposed. RF-LIO [19] utilized an adaptive multi-resolution range images to first remove dynamic objects, and then match LiDAR sweeps to the map for state estimation. ID-LIO [23] proposed a LiDAR-inertial odometry based on indexed point and delayed removal strategy for dynamic scenes. Although RF-LIO and ID-LIO have the ability to perform state estimation in dynamic scenarios, huge computational overhead makes them unable to run stably in real time.

III. Our System Dynamic-LIO

Dynamic-LIO begins with a baseline LIO system, which is based on SR-LIO [30] but without using sweep reconstruction [28], [31]. Fig. 1 illustrates the framework of Dynamic-LIO which consists of four main modules: cloud processing, static initialization, ESIKF based state estimation and dynamic point identification. The yellow rectangles indicate the specific locations of five steps of label consistency detection within the overall system framework and will be introduced in Sec. IV. For the other functional modules, please refer to our previous work SR-LIO [30].

The entire system maintains two global maps: the tracking-map and the output map. The former is utilized for state estimation, while the latter is utilized for label consistency detection and serves as the final reconstruction outcome.

IV. Label Consistency Detection

The core premise of label consistency detection is that the moving objects in driving scenarios are in contact with the ground. Under this premise, we first construct the binarized labels (i.e., ground label and non-ground label) for each 3D point by segmenting the ground points from current input sweep (as illustrated in Fig. 2 (a)). All ground points are inherently static, and the potential dynamic points are exclusively found among non-ground points. If we have already prepared the static global map at the previous moment, aside from the new points to be added at a greater distance, each static point at current moment can find its corresponding nearest neighbor within the global map during map update. For LiDAR points scanned from moving objects, the lack of structural informations within the global map prevents the current position from coinciding with any existing static geometric structures in space. (as illustrated in the green area of Fig. 2 (b-1)). Thus, most LiDAR points scanned from moving objects are often unable to find nearest neighbors during registration and we identify thoes points as dynamic points (shown as the green points in Fig. 2 (b-2)). As for the remaining small subset of LiDAR points (shown as the pink points in Fig. 2 (b-2)), they may find ground points as their nearest neighbors. We then determine whether to classify

[1]https://github.com/ZikangYuan/dynamic_lio

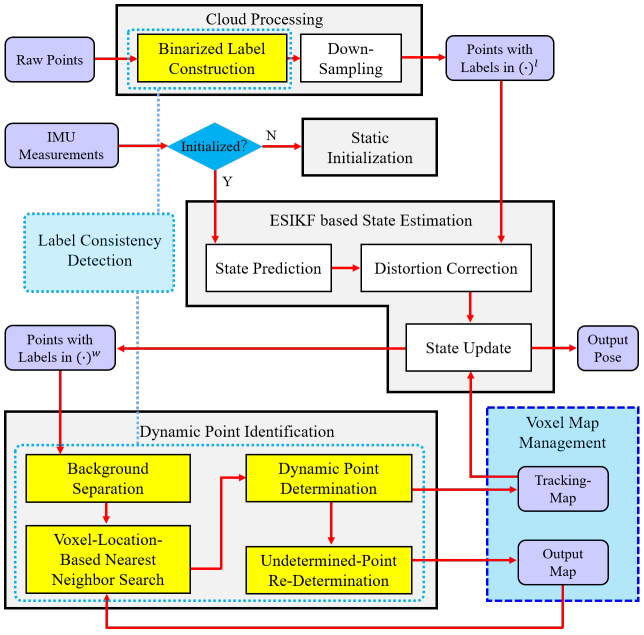


Fig. 1. Overview of our Dynamic-LIO which consists of four main modules: cloud processing, static initialization, ESIKF based state estimation and dynamic point identification. The yellow rectangles indicate the operations of our system that are related to label consistency detection.

them as dynamic points according to the proportion of ground points within the nearest neighbors. It is evident that throughout the process of evaluating label consistency, we only need to calculate the proportion of ground points among the nearest neighbors, without engaging in any batch geometric computations and global statistics. In addition, we utilize the voxel-location-based nearest neighbor search to obtain nearest neighbors, which can be directly located without any quantitative geometric distance calculation. The lightweight of these two core aspects ensures the low computational overhead of our method. Once the dynamic points at the current moment are identified, we utilize the estimated pose from LIO to register the static points into the global map to finish the map update, which can be used to identify dynamic points of next sweep, thereby ensuring the sustainability of our method.

Specially, the label consistency detection method is divided into five steps: binarized label construction, background separation, voxel-location-based nearest neighbor search, dynamic point determination and undetermined-point re-determination. In the following, we will provide a detailed description of each step.

A. Binarized Label Construction

Before dynamic point identification, we first construct binarized descriptors, i.e., ground label and non-ground label, for each 3D point of current sweep. We utilize a fast 2D connected component method [7], which is the same as LeGO-LOAM [21], to separate ground points from current input sweep with very low computational cost.

B. Background Separation

In the process of executing label consistency detection, it is necessary to find the nearest neighbors for each point of current sweep. Points that are close to the vehicle platform can reliably find their nearest neighbors, whereas points that are farther may fail due to the incomplete reconstruction of their locations. In outdoor driving scenarios (excluding extreme occlusion), the map structures within a 30 meter radius around the vehicle platform are usually already reconstructed. Therefore, we set a empirical threshold of 30 meters, and define points within 30 meters of the vehicle platform as fore-points and those beyond 30 meters as back-points. For fore-points and back-points, we employ determinations that are specifically tailored to their characteristics.

C. Voxel-Location-Based Nearest Neighbor Search

For a specific point $\mathbf{p}^w$ with the "non-ground point" label, to determine whether its label is consistent with surrounding points in the global map, we first need to search for the nearest neighbors of $\mathbf{p}^w$. An intuitive alternative is to utilize the 26-nearest neighbor search which is the same as the nearest neighbor search method for point-to-plane distance computation in LIO. Specially, we locate the voxel V to which $\mathbf{p}^w$ belongs and the 26 voxels adjacent to V, and set all points in these voxels as candidate points. Subsequently, the 20 nearest points of $\mathbf{p}^w$ are identified from 9 candidate voxels by comparing the magnitudes of Euclidean distances to $\mathbf{p}^w$.

However, calculating the Euclidean distance of each candidate point to $\mathbf{p}^w$ is an extremely time-consuming process. When LIO builds the point-to-plane distance residuals, in order to ensure that the fitted plane can reflect as much of the geometric information around $\mathbf{p}^w$ as possible, we have to utilize the conventional 26-nearest neighbor search. Fortunately, only 600 point-to-plane distance residuals are required for estimating the pose of each sweep in our system, so the total computational overhead is acceptable. However, in order to ensure that the final output map does not contain dynamic points, it is necessary to determine each point of the current sweep, which requires searching the nearest neighbors for each point from the global map. A single sweep of one 32-line LiDAR can yield more than 50,000 points, making the conventional 26-nearest neighbor search inapplicable here.

In the proposed label consistency detection method, we qualitatively assess whether the non-ground point $\mathbf{p}^w$ is a dynamic point by comparing its label with those of its surround points. Since the surround points are not involved in quantitative calculations, it is not necessary to strictly satisfy the concept of nearest neighbors. Instead, an approximate approach, i.e., voxel-location-based nearest neighbor search, can be adopted to significantly reduce the computational cost. As illustrated in Fig. 3, we locate the voxel V to which $\mathbf{p}^w$ belongs and consider the other points within V as the approximate nearest neighbors. Since the voxel map has a query operation with a computational complexity of $O(1)$, the entire nearest neighbor search process is extremely

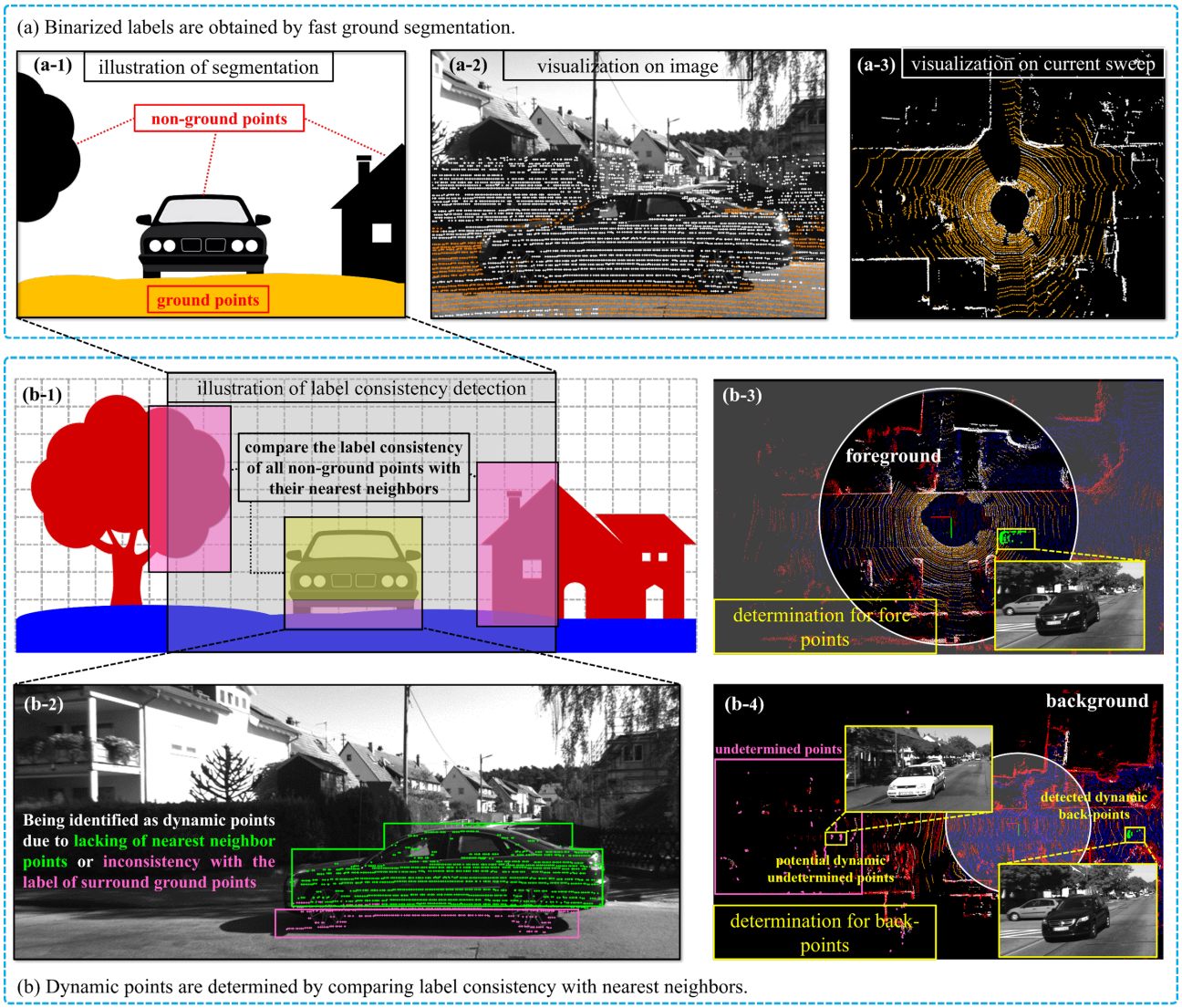


Fig. 2. Illustration and examplar results of the proposed label consistency detection method. (a) Binarized labels are obtained by fast ground segmentation, where all points are divided into ground points (i.e., orange points in (a-2) and (a-3)) and non-ground points (i.e., white points in (a-2) and (a-3)). (b) Dynamic points are determined by comparing label consistency with nearest neighbors. The grid space where the dynamic point is located at the current time was not occupied in the past (i.e., green area in (b-1)). Thus the green points in (b-2) are detected as dynamic points because they are unable to find enough nearest neighbors. The occupied grid space does not exclusively contain static points (i.e., pink area in (b-1)), there may be some dynamic points adjacent to the ground. Thus the pink points in (b-2) are detected as dynamic points because their labels are inconsistent with surround ground points. (b-3) and (b-4) are the illustrations of Sec. IV-D. (b-3) is the visualization of dynamic point determination results for fore-points. The areas obscured in white are the background regions, while those that remain visible are the foreground regions. (b-4) is the visualization of dynamic point determination results for back-points. The areas obscured in white are the foreground regions, while those that remain visible are the background regions.

fast. In addition, the computational overhead associated with Euclidean distance is also saved.

D. Dynamic Point Determination

In Sec. IV-B, we categorize the points of current sweep into fore-points and back-points based on their distance from the vehicle platform. For dynamic point determination of fore-points and back-points, we employ the following two distinct modes.

Mode for fore-points. If the number of nearest neighbors is below a certain threshold (5 in our system), it indicates that the location of $\mathbf{p}^w$ was originally unoccupied, and thus $\mathbf{p}^w$ is classified as a dynamic point. If the number of nearest neighbors is sufficiently large (greater than 5), we calculate the proportion of non-ground points among all nearest neighbors. If this proportion is sufficiently low (less than 30%), $\mathbf{p}^w$ is classified as a static point and added to both the tracking-map and the output map. Conversely, $\mathbf{p}^w$ is classified as a dynamic point and excluded in the map. The visualization of dynamic point determination results for fore-points is shown in Fig. 2 (b-3).

Mode for back-points. If the number of nearest neighbors is scarce, we cannot identify the back-point as a dynamic point, because it is possible that the location has not yet been reconstructed, preventing the obtainment of the nearest

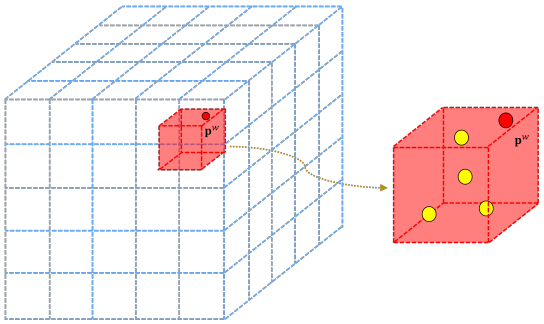


Fig. 3. Illustration of the voxel-location-based nearest neighbor search. The voxel V to which $\mathbf{p}^w$ belongs is directly located, and all points in V (no more than 20) are considered as nearest neighbors (simply represented by 5 yellow points).

neighbors. Such points are labeled as undetermined-points, and a determination will be made once the vehicle platform continues to move and the geometric structures of the locations of these points are recovered. To ensure that newly acquired point clouds can be properly registered during state estimation, it is necessary to incorporate the undetermined-points into the tracking-map. This will not significantly affect the accuracy of state estimation, as even if there are dynamic objects among the back-points, the number of LiDAR points scanned onto them is very sparse. As for the final output map, it is imperative to ensure that it contains as few dynamic points as possible, hence the determination for undetermined-points will be conducted subsequently. When the number of nearest neighbors is sufficiently large (greater than 5), the processing approach is the same as that for fore-points, and the static points are added to both the tracking-map and the output map. The visualization of dynamic point determination results for back-points is shown in Fig. 2 (b-4).

E. Undetermined-Point Re-Determination

As mentioned in Sec. IV-D, some back-points may fail to find nearest neighbors due to the incomplete reconstruction of their locations. For such points, we label them as undetermined points and place them in a container. As the vehicle platform continues to move forward, the geometric structure informations of previously unreconstructed positions are recovered. Then we can make re-determination for those undetermined-points. When a point $\mathbf{p}_u^w$ in the undetermined point container is close to the current position of the vehicle platform (less than 30 meter), it is highly likely that the geometric structure information around $\mathbf{p}_u^w$ has been reconstructed. We can then determine whether $\mathbf{p}_u^w$ is a dynamic point. If the number of nearest neighbors is below a certain threshold (5 in our system), it suggests that the location of $\mathbf{p}_u^w$ was originally unoccupied, leading to the classification of $\mathbf{p}_u^w$ as a dynamic point. If the number of nearest neighbors is larger than the threshold of 5, we calculate the proportion of non-ground points among all nearest neighbors. If this proportion is sufficiently low (less than 30%), it is classified as a static point and added to the output map. On the contrary, if the proportion is not smaller than the threshold of 30%, it is classified as a dynamic point and would not be included in output map. If an undetermined-point is more than 30 meters away from the vehicle platform's position for 10 consecutive sweeps, it is likely to be a sparse background point at far distance. Thus, we directly classify it as a static point and add it to the output map.

TABLE I

PR COMPARISON WITH STATE-OF-THE-ART METHODS (UNIT: %)

	offline		online	
	Removert	Erasor	Dynamic Filter	Ours
*kitti*_00	86.83	**93.98**	90.07	**90.36**
*kitti*_01	**95.82**	91.49	87.95	**88.43**
*kitti*_02	83.29	**87.73**	88.02	**88.25**
*kitti*_05	88.17	**88.73**	90.17	**90.31**
*kitti*_07	82.04	**90.62**	87.94	**89.28**

V. EXPERIMENTS

We evaluate the overall performance of our method on six autonomous driving scenario datasets: *semantic-kitti* [1], *ulhk-CA* [22], *urban-Nav* [9], *nclt* [2], *utbm* [25] and *ulhk-HK* [22]. Among them, *semantic-kitti*, *ulhk-CA* and *urban-Nav* are three public datasets collected at dynamic scenes, *nclt*, *utbm* and *ulhk-HK* are three public datasets collected at static scenes. *semantic-kitti* is collected by a 64-line Velodyne LiDAR and each LiDAR point has its unique semantic label. Thus, *semantic-kitti* is used to evaluate the preservation rate (PR) and rejection rate (RR) of proposed label consistency based dynamic point detection and removal method. *ulhk-CA* and *urban-Nav* are used to evaluate the improvement of dynamic point detection and removal method on pose estimation in terms of absolute trajectory error (ATE). Both *nclt*, *utbm* and *ulhk-HK* are used to demonstrate that the proposed dynamic point detection and removal method has no negative effect on the accuracy of LIO in static scenes. A consumer-level computer equipped with an Intel Core i7-11700 and 32 GB RAM is used for all experiments.

A. PR and RR Comparison with the State-of-the-Arts

We compare our label consistency based dynamic point detection method with three state-of-the-art 3D point-based dynamic point detection methods, i.e., Removert [11], Erasor [15] and Dynamic Filter [6], on *semantic-kitti* dataset [1]. Among them, Removert and Erasor are offline methods which need the pre-built map as input, Dynamic Filter and our method are online methods which do not rely on any prior information.

Results in Table I and Table II demonstrate that our method outperforms Dynamic Filter for almost all sequences in terms of higher PR and RR. Although Dynamic Filter achieves higher RR than our method on sequence *kitti*_00, our result is very close to theirs, with only a 0.36% difference.

TABLE II

RR COMPARISON WITH STATE-OF-THE-ART METHODS (UNIT: %)

	offline		online	
	Removert	Erasor	Dynamic Filter	Ours
$kitti\_00$	90.62	**97.08**	**91.09**	90.73
$kitti\_01$	57.08	**95.38**	87.69	**88.41**
$kitti\_02$	88.37	**97.01**	86.10	**86.22**
$kitti\_05$	79.98	**98.26**	84.65	**85.84**
$kitti\_07$	95.50	**99.27**	86.80	**87.34**

TABLE III

RMSE OF ATE COMPARISON WITH STATE-OF-THE-ART METHODS ON DATASETS OF DYNAMIC SCENES (UNIT: M)

	RF-LIO	ID-LIO	Ours
CA-MarktStreet	15.89	28.02	**12.96**
CA-RussianHill	12.17	15.34	**4.84**
TST	-	**1.06**	3.90
Whampoa	-	**3.45**	6.66

Denotations: "-" means the corresponding value is not available.

B. ATE Comparison with the State-of-the-Arts

We compare our Dynamic-LIO with two state-of-the-art LIO systems for dynamic scenes, i.e., RF-LIO [19] and ID-LIO [23], on $ulhk$-CA [22] and $urban$-Nav [9] datasets. Both RF-LIO, ID-LIO have loop detection module, and use GTSAM [10] to optimize the factor graph. Thus, we also add the same loop detection module and global optimization module to Dynamic-LIO when comparing with them. Both results of RF-LIO and ID-LIO are recorded from their literatures because they have not released the code. The selected four sequences both encompass highly dynamic scenarios and can effectively evaluate the performance of LIO systems in dynamic scenes.

Results in Table III demonstrate that the accuracy of our Dynamic-LIO is superior to that of RF-LIO and ID-LIO on CA-MarktStreet and CA-RussianHill. Since RF-LIO is neither open-sourced nor tested on the $urban$-Nav dataset, we are unable to obtain its results on sequence TST and Whampoa. Although ID-LIO achieves smaller ATE than our system on $urban$-Nav dataset, our act of open-sourcing the code better substantiates the reproducibility of our results.

C. Ablation Study of Undetermined-Points

In our system, the purpose of incorporating undetermined-points is to remove dynamic points as much as possible, thereby increasing the proportion of static points in the output map. In this section, we validate the necessity of incorporating undetermined-points through comparing the

TABLE IV

IMPACT OF UNDETERMINED-POINTS ON PR (UNIT: %)

	Ours w/o Considering Undetermined-Points	Ours
$kitti\_00$	90.09	**90.36**
$kitti\_01$	88.17	**88.43**
$kitti\_02$	87.59	**88.25**
$kitti\_05$	89.31	**90.31**
$kitti\_07$	88.43	**89.28**

TABLE V

IMPACT OF UNDETERMINED-POINTS ON RR (UNIT: %)

	Ours w/o Considering Undetermined-Points	Ours
$kitti\_00$	90.05	**90.73**
$kitti\_01$	87.86	**88.41**
$kitti\_02$	**86.22**	**86.22**
$kitti\_05$	84.08	**85.84**
$kitti\_07$	87.30	**87.34**

TABLE VI

IMPACT OF REMOVING DYNAMIC POINTS ON ATE (UNIT: M)

		Ours w/o Removing Dynamic Points	Ours
Dynamic Scenes	CA-MarktStreet	13.98	**12.96**
	CA-RussianHill	4.88	**4.84**
	TST	5.21	**3.90**
	Whampoa	9.19	**6.66**
Static Scenes	$nclt\_2012\text{-}01\text{-}08$	1.60	**1.54**
	$nclt\_2012\text{-}02\text{-}02$	1.77	**1.74**
	$nclt\_2012\text{-}02\text{-}04$	**2.23**	**2.23**
	$nclt\_2012\text{-}05\text{-}11$	2.55	**1.67**
	$nclt\_2012\text{-}05\text{-}26$	2.44	**2.24**
	$nclt\_2012\text{-}06\text{-}15$	**1.93**	2.05
	$nclt\_2012\text{-}08\text{-}04$	**2.06**	2.13
	$nclt\_2012\text{-}09\text{-}28$	1.92	**1.66**
	$utbm\_2018\text{-}07\text{-}19$	**13.76**	13.92
	$utbm\_2019\text{-}01\text{-}31$	17.05	**16.09**
	$utbm\_2019\text{-}04\text{-}18$	9.66	**9.10**
	$utbm\_2018\text{-}07\text{-}20$	12.97	**9.63**
	$utbm\_2018\text{-}07\text{-}13$	9.74	**9.63**
	$ulhk\_2019\text{-}01\text{-}17$	1.04	**1.03**
	$ulhk\_2019\text{-}04\text{-}26\text{-}1$	3.31	**3.08**

TABLE VII

TIME CONSUMPTION COMPARISON WITH STATE-OF-THE-ART METHODS ON SEMANTIC-KITTI DATASET (UNIT: MS)

	Dynamic Filter (Front-End)	RH-Map	Ours
$kitti\_00$	55.71	63.96	**41.60**
$kitti\_01$		93.23	**34.93**
$kitti\_02$		66.98	**46.00**
$kitti\_05$		64.61	**34.87**
$kitti\_07$		51.02	**27.61**
CPU model	i7-8559U	i7-12700H	i7-11700
clock speed	**2.7GHz**	**2.7GHz**	2.5GHz

TABLE VIII

TIME CONSUMPTION COMPARISON WITH STATE-OF-THE-ART LIO SYSTEMS ON ULHK-CA AND URBAN-NAV DATASETS (UNIT: MS)

	RF-LIO	ID-LIO	Ours
CA-MarktStreet	96	121	**23.33**
CA-RussianHill	121	100	**21.50**
TST	-	96	**16.46**
Whampoa	-	99	**16.41**
CPU model	i5	i7-10700K	i7-11700
clock speed	1.3∼3.7GHz	**3.8GHz**	2.5GHz

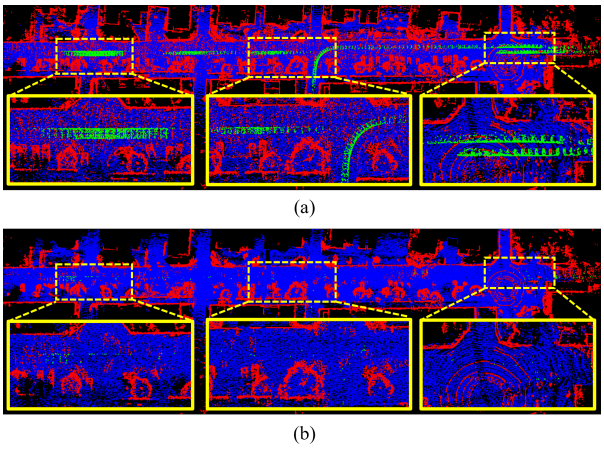
Denotations: "-" means the corresponding value is not available.

TABLE IX

IMPACT OF NEAREST NEIGHBOR SEARCH METHOD ON TIME CONSUMPTION OF LABEL CONSISTENCY DETECTION (UNIT: MS)

	LCD with 26-Nearest Neighbor Search	LCD with our Nearest Neighbor Search
*kitti*_00	78.53	**6.13**
*kitti*_01	115.99	**8.35**
*kitti*_02	81.04	**2.82**
*kitti*_05	43.22	**8.28**
*kitti*_07	46.63	**3.01**
CA-MarktStreet	21.69	**4.40**
CA-RussianHill	21.58	**1.13**
TST	15.20	**3.08**
Whampoa	15.50	**2.24**

Denotations: "LCD" is the abbreviation of "Label Consistency Detection".



(a)

(b)

Fig. 4. Visualization of (a) output map before removing dynamic points and (b) output map after removing dynamic points of the examplar sequence *kitti*_05. The green points are the ghost tracks of moving objects.

PR and RR value of our Dynamic-LIO with and without considering undetermined-points.

Results in Table IV and Table V demonstrate that incorporating undetermined-points can slightly improve PR and RR of our dynamic point detection and removal method.

D. Ablation Study of Dynamic Point Removal for Pose Estimation

In this section, we evaluate the effectiveness of removing dynamic points for pose estimation by comparing the ATE results of our Dynamic-LIO with and without removing dynamic points. Results in Table VI demonstrate that removing dynamic points can enhance the pose estimation accuracy of our Dynamic-LIO in dynamic scenes, especially on *urban-Nav* dataset. In static scenes, our label consistency based dynamic point detection and removal has no negative effect on the pose estimation accuracy, which also reflects the robustness and practicability of the proposed method.

E. Time Consumption Comparison with the State-of-the-Arts

We compare the time consumption of our label consistency based dynamic point detection and removal method with two state-of-the-art online 3D point-based dynamic point detection and removal methods, i.e., Dynamic Filter [6] and RH-Map [26], on *semantic-kitti* dataset [1]. Then, we compare the time consumption of our Dynamic-LIO with RF-LIO and ID-LIO. Both results of other approaches are recorded from their literatures because they have not released the code.

Results in Table VII demonstrate that the time consumption of our dynamic point detection and removal method is much smaller than Dynamic Filter and RH-Map. The front-end of Dynamic Filter requires 55.71ms to process the data of a single sweep. When accounting for the back-end overhead, the total duration for processing a single sweep will be even longer. Given that the current LiDAR acquisition frequency is typically between 10∼20 Hz, this implies that the processing time for a single sweep must be within 50 ms to ensure the real-time performance. It can be seen that neither Dynamic Filter nor RH-Map can guarantee real-time capability, whereas our method can rum in real time stably. Since the *semantic-kitti* dataset does not include IMU data, we complete pose estimation and mapping in a LiDAR-only odometry mode while simultaneously detecting and removing dynamic objects online. Therefore, the time consumption recorded in Table VII represents the total time cost for both LiDAR-only odometry and dynamic point detection and removal, and the results for Dynamic Filter and RH-Map are the same. Although the testing platforms are not entirely the same, the CPUs used for testing Dynamic Filter and RH-Map have a higher clock speed than the one we used. This also serves to a certain extent as evidence of the reference value of our experimental results. Results in Table VIII demonstrate that the time consumption of our Dynamic-LIO is much smaller than RF-LIO [19] and ID-LIO [23], while our system operates at a speed approximately 5X faster than that of RF-LIO and ID-LIO. Since RF-LIO is neither open-sourced nor tested on the *urban-Nav* dataset, we are unable to obtain its results on sequence TST and Whampoa. As stated in ID-LIO [23], the authors tested their system using an i5 CPU. However, no specific model was provided in [23], thus we record the range of clock speeds for the entire i5 series of CPUs in Table VIII. Furthermore, although the i7-11700 has an advantage over the i5 series and i7-8559U in terms of thread count, our entire Dynamic-LIO system is implemented based on a single thread and does not take advantage of the multi-threading capability.

F. Ablation Study of Nearest Neighbor Search

Table IX demonstrates that the voxel-location-based nearest neighbor search we employ has achieved an order-of-magnitude reduction in time consumption compared to the conventional 26-nearest neighbor search, primarily for the following two reasons: (1) The voxel-location-based nearest neighbor search only requires to process the voxel [29] to which the current point belongs; (2) The voxel-location-based nearest neighbor search does not necessitate the computation of the Euclidean distance between candidate points and the current point.

G. Visualization for Trajectory and Map

Fig. 4 shows the ability of Dynamic-LIO to reconstruct a static point cloud map on the exemplar sequence $kitti\_05$. As illustrated in Fig. 4 (a), before removing dynamic points, the ghost tracks of moving objects (green points) are clearly visible. In Fig. 4 (b), after removing dynamic points, the output map almost no longer contains ghost tracks.

VI. Conclusion

This paper proposes a LIO system with label consistency detection, which can fastly eliminate the influence of moving objects in driving scenarios. Different from existing approaches involving batch geometric computation or global statistics to identify moving objects, the proposed method is more lightweight. We embed the proposed label consistency detection method into a LIO system to achieve dynamic object removal with extremely low time consumption.

Experimental results show that the proposed method can achieve comparable PR and RR to state-of-the-art dynamic point detection and removal methods. In addition, Dynamic-LIO operates at a speed approximately 5X faster than state-of-the-art dynamic LIO systems.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9297–9307.

[2] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016.

[3] Z. Chen, K. Zhang, H. Chen, M. Y. Wang, W. Zhang, and H. Yu, "Dorf: A dynamic object removal framework for robust static lidar mapping in urban environments," *IEEE Robotics and Automation Letters*, 2023.

[4] T. Cortinhal, G. Tzelepis, and E. Erdal Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part II 15*. Springer, 2020, pp. 207–222.

[5] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "Rgb-d slam in dynamic environments using point correlations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 373–389, 2020.

[6] T. Fan, B. Shen, H. Chen, W. Zhang, and J. Pan, "Dynamicfilter: an online dynamic objects removal framework for highly dynamic environments," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7988–7994.

[7] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *2010 IEEE Intelligent Vehicles Symposium*. IEEE, 2010, pp. 560–565.

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[9] L.-T. Hsu, N. Kubo, W. Wen, W. Chen, Z. Liu, T. Suzuki, and J. Meguro, "Urbannav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas," in *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, 2021, pp. 226–256.

[10] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[11] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 758–10 765.

[12] J. Li, T.-M. Nguyen, S. Yuan, and L. Xie, "Pss-ba: Lidar bundle adjustment with progressive spatial smoothing," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 1124–1129.

[13] J. Li, W. Wu, B. Yang, X. Zou, Y. Yang, X. Zhao, and Z. Dong, "Whu-helmet: A helmet-based multisensor slam dataset for the evaluation of real-time 3-d mapping in large-scale gnss-denied environments," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–16, 2023.

[14] J. Li, X. Xu, J. Liu, K. Cao, S. Yuan, and L. Xie, "Ua-mpc: Uncertainty-aware model predictive control for motorized lidar odometry," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3652–3659.

[15] 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.

[16] L. Liu, M. Feng, J. Cheng, J. Xiang, X. Zhu, and X. Yang, "Prior-flow: Enhancing primitive panoramic optical flow with orthogonal view," *arXiv preprint arXiv:2506.23897*, 2025.

[17] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[18] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3d map maintenance in dynamic environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3712–3719.

[19] C. Qian, Z. Xiang, Z. Wu, and H. Sun, "Rf-lio: Removal-first tightly-coupled lidar inertial odometry in high dynamic environments," *arXiv preprint arXiv:2206.09463*, 2022.

[20] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart, "Dynablox: Real-time detection of diverse dynamic objects in complex environments," *IEEE Robotics and Automation Letters*, 2023.

[21] 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)*. IEEE, 2018, pp. 4758–4765.

[22] 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 for mapping and localization in urban scenes," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 2310–2316.

[23] W. Wu and W. Wang, "Lidar inertial odometry based on indexed point and delayed removal strategy in highly dynamic environments," *Sensors*, vol. 23, no. 11, p. 5188, 2023.

[24] J. Xiang, X. Zhu, X. Wang, Y. Wang, H. Zhang, F. Guo, and X. Yang, "Depthor: Depth enhancement from a practical light-weight dtof sensor and rgb image," *arXiv preprint arXiv:2504.01596*, 2025.

[25] Z. Yan, L. Sun, T. Krajník, and Y. Ruichek, "Eu long-term dataset with multiple sensors for autonomous driving," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 697–10 704.

[26] Z. Yan, X. Wu, Z. Jian, B. Lan, and X. Wang, "Rh-map: Online map construction framework of dynamic object removal based on 3d region-wise hash map structure," *IEEE Robotics and Automation Letters*, 2024.

[27] D. Yoon, T. Tang, and T. Barfoot, "Mapless online detection of dynamic objects in 3d lidar," in *2019 16th Conference on Computer and Robot Vision (CRV)*. IEEE, 2019, pp. 113–120.

[28] Z. Yuan, J. Deng, R. Ming, F. Lang, and X. Yang, "Sr-livo: Lidar-inertial-visual odometry and mapping with sweep reconstruction," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5110–5117, 2024.

[29] Z. Yuan, F. Lang, J. Deng, H. Luo, and X. Yang, "Voxel-svio: Stereo visual-inertial odometry based on voxel map," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 6352–6359, 2025.

[30] Z. Yuan, F. Lang, T. Xu, and X. Yang, "Sr-lio: Lidar-inertial odometry with sweep reconstruction," pp. 7862–7869, 2024.

[31] Z. Yuan, Q. Wang, K. Cheng, T. Hao, and X. Yang, "Sdv-loam: Semi-direct visual–lidar odometry and mapping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 11 203–11 220, 2023.