2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Observation Time Difference: an Online Dynamic Objects Removal Method for Ground Vehicles

Rongguang Wu, Chenglin Pang, Xuankang Wu, Zheng Fang*

***Abstract*— In the process of urban environment mapping, the sequential accumulations of dynamic objects will leave a large number of traces in the map. These traces will usually have bad influences on the localization accuracy and navigation performance of the robot. Therefore, dynamic objects removal plays an important role for creating clean map. However, conventional dynamic objects removal methods usually run offline. That is, the map is reprocessed after it is constructed, which undoubtedly increases additional time costs. To tackle the problem, this paper proposes a novel method for online dynamic objects removal for ground vehicles. According to the observation time difference between the object and the ground where it is located, dynamic objects are classified into two types: *suddenly appear* and *suddenly disappear*. For these two kinds of dynamic objects, we propose downward retrieval and upward retrieval methods to eliminate them respectively. We validate our method on SemanticKITTI dataset and author-collected dataset with highly dynamic objects. Compared with other state-of-the-art methods, our method is more efficient and robust, and reduces the running time per frame by more than 60% on average. Our method will be open-sourced on GitHub[1].**

I. Introduction

Concise and reliable maps are the basis for long-term autonomy of mobile robots [1], [2]. With the information retrieved from the map, robots can achieve successful localization and navigation [3], [4]. Currently, real-time mapping using SLAM-based technology, especially LiDAR mapping, has become a common way to obtain maps due to many advantages [5], [6]. Unfortunately, there are inevitably a substantial amount of dynamic objects [7], [8] during urban mapping, which will leave undesired traces when constructing the point cloud maps [9]. Moreover, these traces will form a huge impassable area (as shown in Fig. 1), which will make subsequent tasks difficult, such as navigation and path planning. Therefore, dynamic objects removal is crucial in constructing clean point cloud map.

Existing dynamic objects removal methods can be mainly divided into two categories: offline removal and online removal. Offline removal is to perform dynamic removal on the basis of a pre-built map (prior map), in order to obtain a new map with only static objects. Several effective methods have been proposed, such as [9]–[11]. Although these methods achieve significant dynamic removal performance, they require additional time cost. Therefore, this paper focuses on removing dynamic objects while building the map, namely online dynamic objects removal.

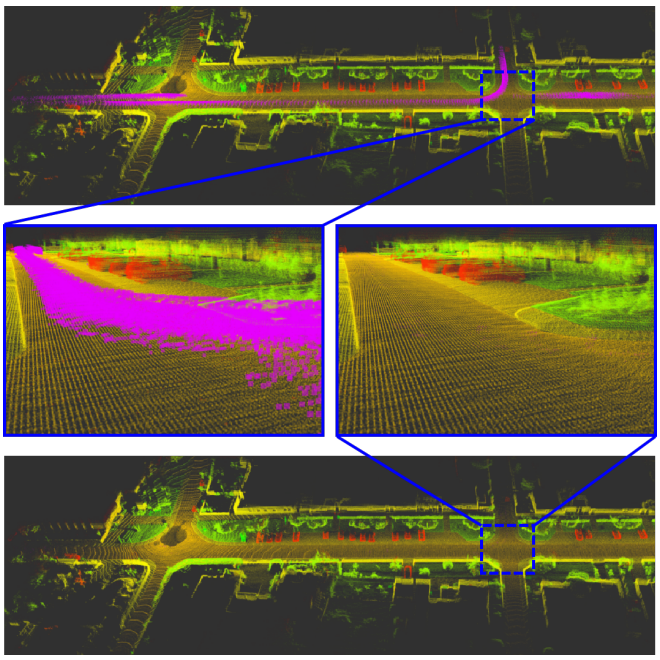


Fig. 1. The results of dynamic objects removal. The upper image is original map. The lower image is static map constructed by our method. The middle image is partially enlarged image, and the purple area is traces left by dynamic objects.

For online dynamic objects removal, an effective idea is to apply and improve similar ideas used in offline removal methods, such as [12], [13]. But the core principle of offline removal method is to judge dynamic objects by comparing the difference between the point cloud of each frame and the prior map. It is infeasible to directly apply offline methods to online removal, since online removal does not have a prior map. Even though online map can be obtained by SLAM algorithms, the map only contains information prior to current frame, and no information after current frame. This will lead to a decrease in the performance of dynamic removal.

In this paper, we propose a novel online dynamic objects removal method for ground vehicles. Considering that common dynamic objects (vehicles, pedestrians, etc.) are always in contact with the ground [9], our method judges the dynamic objects by comparing the observation time

This work was supported in part by the National Natural Science Foundation of China under Grants 62073066 and U20A20197, in part by the Fundamental Research Funds for the Central Universities under Grant N2226001, and in part by 111 Project under Grant B16009 (Corresponding author: Zheng Fang).

The authors are all with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: 2202066@stu.neu.edu.cn, 2010690@stu.neu.edu.cn, 2202067@stu.neu.edu.cn, fangzheng@mail.neu.edu.cn).

[1]https://github.com/RongguangWu/OTD

difference between the object and the ground on which it is located. Our contributions are as follows:

- Based on the observation time difference between the object and the ground it is located on, we classify dynamic objects into two categories: *suddenly appear* and *suddenly disappear*.
- For the dynamic objects of *suddenly appear* and *suddenly disappear*, we propose two methods, namely downward retrieval and upward retrieval, to remove them respectively.
- We conduct extensive experimental validations on challenging datasets, and SemanticKITTI [14] datasets. Compared to the state-of-the-art (SOTA) methods, our approach is more effective and robust, and reduces the average processing time per frame by over 60%.
- We have encapsulate our proposed method as an independent dynamic removal module, which can run concurrently with LiDAR mapping algorithm to generate clean static maps. Our method will be open-sourced on GitHub.

The rest of the paper is organized as follows: Section II reviews the related work about dynamic objects removal. In section III, we divide dynamic objects into two categories. Section IV explains the details of our proposed dynamic removal method. Experimental comparisons and analyses are carried out in Section V. Finally, Section VI summarizes the contributions and describes future works.

II. Related Work

This section briefly reviews the relevant studies on dynamic objects removal methods, which can be roughly divided into offline removal approaches and online removal approaches:

A. Offline Removal Approaches

One mainstream concept of offline removal is to detect dynamic objects by comparing the differences between the point cloud of each frame and the prior map. Ray tracing-based method is a commonly used method. Its fundamental idea is to calculate the occupancy probability of a voxel based on whether it is penetrated by the laser beam of the LiDAR [15]. However, checking voxels is computationally expensive. Even though Schauer $et\ al.$ [8] and Pagad $et\ al.$ [7] made some improvements to accelerate the speed of inspecting voxels, the processing time is still long.

In order to reduce the computational cost, visibility-based method is proposed [8], [11], which determines dynamic objects by checking whether the points in the map are occluded by the points in the query frame. However, when the incidence angle is big, the visibility-based method may mistakenly identify distant ground points as dynamic points.

Additionally, Lim $et\ al.$ [9], [16] proposed a visibility-free approach. This method is based on the assumption that dynamic objects are mostly in contact with the ground. It detects dynamic objects by comparing the pseudo-occupancy differences between the query frame and the map in the occupied grid. However, this method requires significant time consumption, making it unsuitable for real-time online processing.

B. Online Removal Approaches

In recent years, with the development of deep learning, numerous online dynamic objects segmentation methods have been proposed. Chen $et\ al.$ [17] generated residual images from previous scans and input them together with the range image of current scan to CNN, in order to distinguish between dynamic objects and static objects. Sun $et\ al.$ [18] exploited a dual-branch structure based range images, which is utilized to process the spatial-temporal information of LiDAR point cloud separately. Then, the motion-guided attention module was employed to detect dynamic objects. However, learning-based methods heavily rely on the training dataset, and they are prone to failure when there is a significant discrepancy between the actual scenario and the training scenario.

Furthermore, there are also some methods that employ a concept akin to offline removal [12], [13]. But the problem is that online removal does not have a prior map. To solve this problem, Park $et\ al.$ [12] accumulate differential range images between the scan data of current frame and that of past few frames to obtain a background model. However, the background model only contain information prior to the current frame, and cannot capture information after current frame. Fan $et\ al.$ [13] accumulated multi-frame point clouds to build a local submap, and then compared each frame with the local submap to judge dynamic objects. Nevertheless, the accumulated number of frames can impact the performance of dynamic removal. Too few frames may not provide sufficient information, while too many frames can increase the computational cost of the algorithm.

In this paper, we propose a novel online dynamic objects removal method. Our method determines dynamic objects by comparing observation time difference between the object and the ground, and achieves effective dynamic object removal without a prior map.

III. Dynamic objects classification

Similar to [9], we believe that in most cases, dynamic objects should have contact with the ground, whether they are pedestrians or terrestrial vehicles. Furthermore, if an object on the ground is static, it should be observed and disappear from the field of view simultaneously with the ground it is on. In other words, the object and the ground must appear and disappear simultaneously. If either of the two conditions is not met, the object is considered as a dynamic object. Therefore, we classify dynamic objects into the following two categories:

- As shown in Fig. 3(a), at a certain position, only ground was observed but not non-ground object at time t_0. Both ground and non-ground object were observed since time $t_1\,(t_1 > t_0)$. Then, the non-ground object is defined as a *suddenly appear* dynamic object.
- As shown in Fig. 3(b), at a certain position, both ground and non-ground object were observed at time t_0.

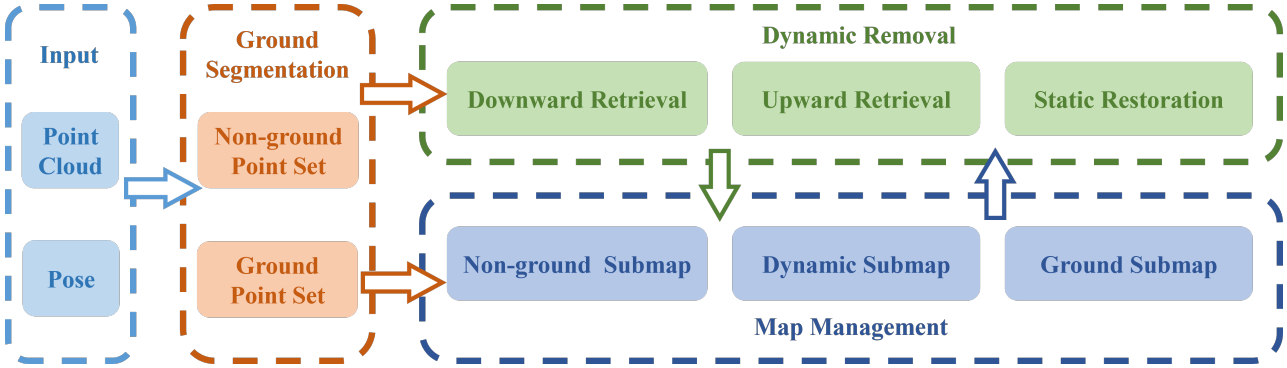
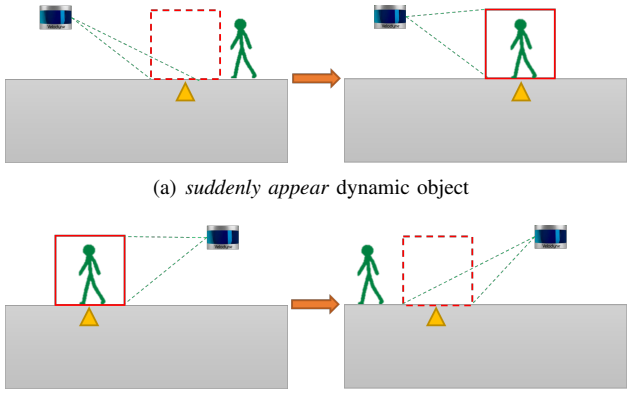


Fig. 2. Overview of our proposed method. Our proposed method includes three modules: ground segmentation, map management and dynamic removal. The point cloud $\mathbf{P}$ is divided into ground point set $\mathbf{G}$ and non-ground point set $\mathbf{U}$ in ground segmentation module, and then is sent to map management module and dynamic removal module together with the input pose $\mathbf{T}^W$. After downward retrieval, upward retrieval and static restoration, dynamic objects can be recognized.

Only ground was observed but not non-ground object since time t_1 $(t_1 > t_0)$. Then, the non-ground object is defined as a *suddenly disappear* dynamic object.



(a) *suddenly appear* dynamic object

(b) *suddenly disappear* dynamic object

Fig. 3. Dynamic objects classification. Only ground was observed at the begining in (a). But after a period of time, both ground and non-ground object were observed. The object is a *suddenly appear* dynamic object. On the contrary, both ground and non-ground object were observed at the beginging in (b). But after a period of time, only ground was observed. The object is a *suddenly disappear* dynamic object.

Note that a continuously moving object simultaneously meets the definitions of *suddenly appear* and *suddenly disappear*. When an object moves to a specific position, it satisfies the definition of *suddenly appear* at that position. However, when it leaves that position, it satisfies the definition of *suddenly disappear*.

According to the definition, if the time of the first observation of an object is later than the time of the first observation of the ground below it, then the object is considered as a *suddenly appear* dynamic object. On the contrary, if the time of the last observation of an object is earlier than the time of the last observation of the ground, then the object is considered as a *suddenly disappear* dynamic object. We refer to this method of determining dynamic objects as observation time difference.

IV. Methodology

The overall framework of our proposed dynamic objects removal method is illustrated in Fig. 2, which includes three modules: ground segmentation, map management and dynamic removal.

A. Ground Segmentation

Since our proposed dynamic removal approach needs to constantly compare the observation time difference between ground and non-ground objects, the accuracy of ground segmentation will affect the accuracy of dynamic removal. Therefore, we have adopted a two-step ground segmentation process, whereby candidate ground points are first extracted through preprocessing, and then they are refined to obtain the final ground point set.

In the preprocessing, we refer to the method proposed in [19] to remove most of the non-ground points. Then, we introduce PCA proposed in [20] to further refine it and obtain the final ground point set $\mathbf{G}$, while the remaining points are classified as non-ground point set $\mathbf{U}$. Since we removed most of the non-ground points in the preprocessing, the ground fitting process will be relatively more accurate. This two-step ground segmentation method aims to minimize the number of dynamic points in the segmented ground as much as possible, which is crucial for improving the effectiveness of dynamic removal.

B. Map Management

It is difficult to accurately identify dynamic *object* in point cloud maps, so we use voxels [21], [22] to represent *object*. If a voxel is detected to be dynamic, all points in the voxel are classified as dynamic points.

Therefore, following [22], we build a voxel map to manage the point cloud. In order to facilitate the management of ground points, non-ground points and dynamic points, we divide the global map into three parts: ground submap ${}^{g}\mathcal{V}$, non-ground submap ${}^{n}\mathcal{V}$ and dynamic submap ${}^{d}\mathcal{V}$. These three submaps share the same coordinate system, but are stored separately.

In addition, in order to obtain the observation time of each voxel, we store the index of LiDAR frame for each point

Algorithm 1 Downward Retrieval

Input: Non-ground Point Cloud $\mathbf{U}^W$; Non-ground Submap ${}^n\mathcal{V}$; Ground Submap ${}^g\mathcal{V}$

for $\mathbf{p}_i^W \in \mathbf{U}^W$ **do**
  Let ${}^n\mathbf{V}_i$ be the non-ground voxel of $\mathbf{p}_i^W$;
  Let ${}^n\gamma_{min}$ be the minimum frame index of ${}^n\mathbf{V}_i$;
  for *k = 1, 2, ...* **do**
    ${}^g\mathbf{V}_i$ = ${}^n\mathbf{V}_i$ - $k \cdot (0,0,1)^T$;
    if ${}^g\mathbf{V}_i \neq \emptyset$ **then**
      Let${}^g\gamma_{min}$ be the minimum frame index of ${}^g\mathbf{V}_i$;
      break;
    end
  end
  if ${}^n\gamma_{min} - {}^g\gamma_{min} > \tau_{ret}$ **then**
    Find dynamic voxel ${}^d\mathbf{V}_i$ of $\mathbf{p}_i^W$;
    ${}^d\mathbf{V}_i$ = ${}^d\mathbf{V}_i \cup {}^n\mathbf{V}_i$;
    ${}^n\mathbf{V}_i = \emptyset$;
  end
end

besides storing the points in the voxel, and record all the frame indexes in a Set. Then, the smallest frame index in the set can represent the time when the voxel was first observed, while the largest frame index can represent the time when the voxel was last observed. For convenience, we will refer to the voxels located in the ground submap, non-ground submap, and dynamic submap as ground voxels, non-ground voxels, and dynamic voxels respectively in the following paper.

C. Dynamic Removal

Based on the observation time difference we introduced in Section III, we can compare the observation time difference between each non-ground voxel and ground voxel below it to determine whether the voxel is dynamic or not. Indeed, comparing all voxels with the ground for each iteration is impractical for online processing. Therefore, we propose downward retrieval and upward retrieval methods to remove the *suddenly appear* and *suddenly disappear* dynamic voxels respectively. In addition, there will always be some misidentified static voxels anyway, so we added static restoration process.

Using the input pose $\mathbf{T}^W$ (usually comes from the SLAM front-end), we can transform $\mathbf{G}$ and $\mathbf{U}$ into the world frame and designate them as $\mathbf{G}^W$ and $\mathbf{U}^W$, respectively.

1) Downward Retrieval: The downward retrieval process we proposed is shown in Algorithm 1. Since the *suddenly appear* dynamic objects are the objects that can be observed at current time, it must exist in $\mathbf{U}^W$. Therefore, to achieve the removal of *suddenly appear* dynamic objects, we only need to examine the voxel containing $\mathbf{U}^W$.

For each point $\mathbf{p}_i^W$ in $\mathbf{U}^W$, we calculate its non-ground voxel ${}^n\mathbf{V}_i$ in non-ground submap. Then, we count all frame indexes in ${}^n\mathbf{V}_i$, and designate the smallest frame index as ${}^n\gamma_{min}$. ${}^n\gamma_{min}$ can represent the time when ${}^n\mathbf{V}_i$ is first observed.

Next, in the ground submap, we retrieve the ground voxel ${}^g\mathbf{V}_i$ within 3 meters by descending along z-axis from ${}^n\mathbf{V}_i$. Finally, we count all the frame indexes in ${}^g\mathbf{V}_i$, and designate the smallest frame index as ${}^g\gamma_{min}$. ${}^g\gamma_{min}$ can represent the time when the ground voxel is first observed.

It can be demonstrated that ${}^n\mathbf{V}_i$ is a *suddenly appear* dynamic voxel if the following criteria are satisfied:

$$ {}^n\gamma_{min} - {}^g\gamma_{min} > \tau_{ret} \tag{1}$$

where τ_{ret} is a pre-defined threshold. Then, we move all points as well as all frame indexes in ${}^n\mathbf{V}_i$ into dynamic submap.

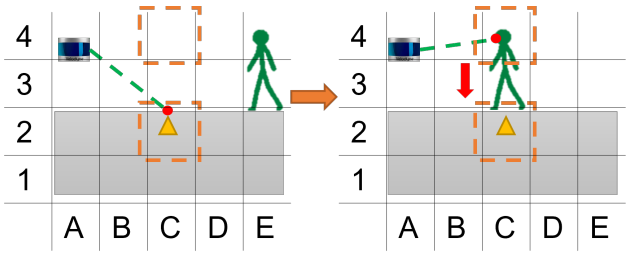


Fig. 4. Downward retrieval. The image on the left represents the i-th LiDAR frame, while the image on the right represents the j-th LiDAR frame ($j > i$). The points observed by LiDAR are represented by red dots. The object was at position E at the i-th LiDAR frame, and it moved to position C at the j-th LiDAR frame. We first observed the non-ground voxel (C, 4) in the j-th LiDAR frame, but we had already observed the ground voxel (C, 2) below it in the i-th frame. Therefore, we compare the minimum LiDAR frame index between these two voxels to determine whether (C, 4) is dynamic or not.

As shown in Fig. 4, the non-ground voxel (C, 4) was first observed in the j-th frame. Therefore, the minimum LiDAR frame index for (C, 4) is recorded as j. Then, we find the ground voxel (C, 2) by descending along z-axis from (C, 4). Since (C, 2) was observed in the i-th frame, the minimum LiDAR frame index for (C, 2) is i. According to Eq. 1, if $j - i > \tau_{ret}$, then the non-ground voxel (C, 4) is considered as a dynamic voxel.

2) Upward Retrieval: Since the *suddenly disappear* dynamic objects can no longer be observed at current time, that is, they do not exist in $\mathbf{U}^W$. Indeed, the ground below them can still be observed. Therefore, we use points in $\mathbf{G}^W$ to retrieve upwards to find *suddenly disappear* dynamic objects.

For each point $\mathbf{p}_i^W$ in $\mathbf{G}^W$, we calculate its corresponding voxel ${}^g\mathbf{V}_i$ in the ground submap. Then, we define the largest frame index in ${}^g\mathbf{V}_i$ as ${}^g\gamma_{max}$. Next, retrieve all the non-ground voxels within 3 meters by ascending along the z-axis in the non-ground submap. We check each of these non-ground voxels ${}^n\mathbf{V}_i$ in turn, and define the largest frame index in it as ${}^n\gamma_{max}$.

Finally, if the following criteria is satisfied:

$$ {}^g\gamma_{max} - {}^n\gamma_{max} > \tau_{ret} \tag{2}$$

It can be demonstrated that ${}^n\mathbf{V}_i$ is a *suddenly disappear* dynamic voxel.

3) Static restoration: In Section III, we proposed the assumption that static objects should appear and disappear simultaneously with the ground. However, in real-world scenarios, this assumption does not always hold true due to the

influence of ground slope and LiDAR measurement noise. This inevitably leads to some static voxels being mistakenly identified as dynamic voxels. Therefore, we introduce a static restoration module to place these misclassified static voxels back into the non-ground submap. We assume that if the total times of observation of a non-ground voxel and the ground voxel is similar, then the voxel should be static. Based on this, we designed the process of static restoration.

We find the dynamic voxel ${}^{d}\mathbf{V}_i$ in the dynamic submap during the downward retrieval and define the total number of frame indexes in ${}^{d}\mathbf{V}_i$ as ${}^{d}\gamma_{sum}$. ${}^{d}\gamma_{sum}$ can represent the total times the voxel has been observed. Correspondingly, we define the total number of frame indexes in the ground voxel ${}^{g}\mathbf{V}_i$ as ${}^{g}\gamma_{sum}$.

If the difference between ${}^{g}\gamma_{sum}$ and ${}^{d}\gamma_{sum}$ is less than a pre-defined threshold τ_{res}, ${}^{d}\mathbf{V}_i$ is proved to be a static voxel that has been misidentified. Thus, we move all points and all frame indexes in ${}^{d}\mathbf{V}_i$ back into the non-ground submap.

V. Experiments

In this section, in order to prove the effectiveness of our proposed method, we conducted online dynamic object removal experiments on the public SemanticKITTI dataset and challenging dataset. Qualitative and quantitative comparisons with state-of-the-art algorithms are conducted to evaluate our proposed method. In order to fairly test the performance of our algorithm, all experiments were performed on the same laptop with an AMD R7-5800H CPU and 16GB RAM.

A. Evaluation Metrics

We use $preservation\ rate$ (PR) and $rejection\ rate$ (RR) proposed in [9] and their harmonic mean $\mathbf{F}_1$ score to evaluate performance of dynamic removal. PR and RR represent the preservation rate of static objects and the rejection rate of dynamic objects respectively, and their definitions are as follows:

- PR: $\frac{\text{\# of preserved static points}}{\text{\# of total static points on the raw map}}$
- RR: $1 - \frac{\text{\# of preserved dynamic points}}{\text{\# of total dynamic points on the raw map}}$

Since PR and RR are calculated voxel-wise, we set the voxel size to 0.2 for all methods for a fair comparis.

B. Comparison in SemanticKITTI Datasets

We compare the proposed method with other state-of-the-art dynamic removal methods, including Removert [11], ERASOR [9] and ERASOR2 [16] for offline removal, and DynamicFilter [13] and Nonparametric [12] for online removal. The poses required by our method are provided by SuMa [23] which contains inherent uncertainty. Additionally, since dynamic objects are not always present in SemanticKITTI, referring to [9], we select frames 00 (4,390 - 4,530), 01 (150 - 250), 02 (860 - 950), 05 (2,350 - 2,670), and 07 (630 - 820) as dynamic removal benchmark where the numbers in parenthesis indicate the start and end frames.

In all segments, we set identical dynamic removal parameters, where $\tau_{ret} = 7$ and $\tau_{res} = 15$. Table I shows the experimental results of ours and other methods on five segments of SemanticKITTI. In Table I, RM3 denotes the results after three remove stages with per-pixel resolutions of $0.5°$, $0.45°$ and $0.4°$ in Removert [11]. RM3+RV1 means the result of RM3 followed by a revert stage with the resolution per pixel of $0.55°$. ERASOR-0.2 and ERASOR-0.4 denote the different sizes of scan ratio threshold in ERASOR [9].

As shown in Table I, our method achieves comparable performance to ERASOR2 and outperformed other methods in semanticKITTI. By introducing instance segmentation, ERASOR2 rejects dynamic points at an instance-level. Therefore, ERASOR2 outperforms our method in some segments. Particularly, in the Seq.02 segment, it achieves an impressive $\mathbf{F}_1$ score of 0.99. However, ERASOR2 is an offline algorithm that requires a prior map and has a higher time consumption, making it unsuitable for online processing. As for other methods, Removert-RM3 is too confident in removing dynamic objects, resulting in a low PR. In the revert process, some dynamic points will be put back into the map, so the RR of Removert-RM3+RV1 is the lowest. The score of ERASOR in PR is also unsatisfactory, especially after the scanning ratio threshold is increased, the PR of ERASOR-0.4 has dropped sharply. Compared with DynamicFilter and Nonparametric, which are online dynamic removal methods, our method achieves better results because it is not limited by the information of map. It should be noted that our method is stable across all 5 segments while other on-line methods vary greatly in performance across different segments, which means that our method is more robust than other methods.

C. Comparison in challenging scenarios

To further validate the algorithm's robustness, we conducted experiments on dynamic object removal in challenging scenarios with high pedestrian density. As of now, ERASOR2, DynamicFilter, and Nonparametric algorithms have not been open-source, so we only compared our method with ERASOR and Removert, and the comparative results are shown in Fig. 5. The selected scene is the road in front of the school cafeteria, and the data recording time coincides with the students' dismissal time. At this time, almost all students who finish classes head to the cafeteria for meals, leaving behind a significant amount of traces in the image. As shown in Fig. 5(a), the middle area of the image is almost filled with dynamic points.

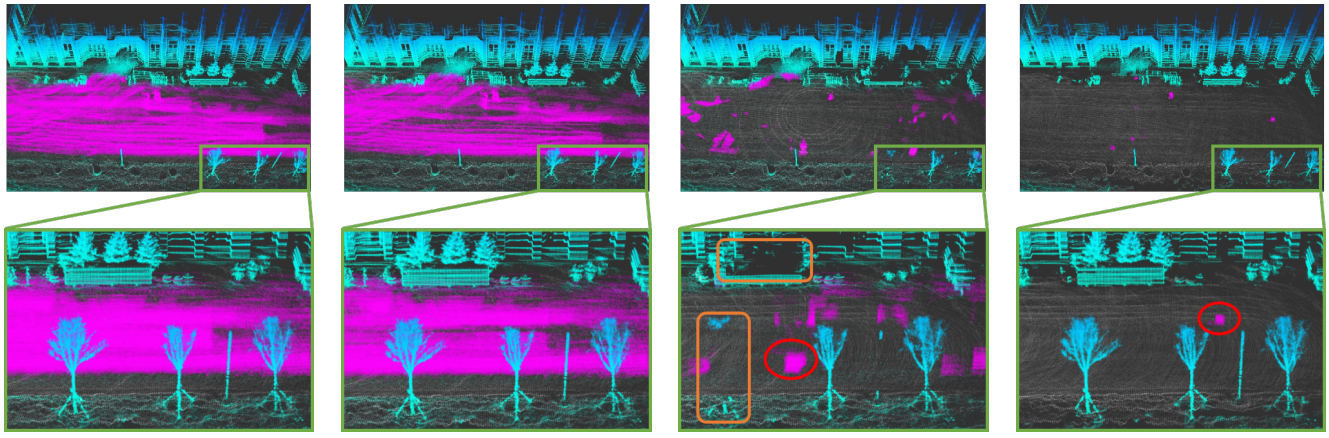
The dynamic object removal effectiveness of Removert in such a crowded scene is limited. While Removert managed to remove some dynamic objects, there are still numerous traces left in the image attributable to the presence of dynamic objects. The results of ERASOR are shown in Fig. 5(c), where most of the dynamic traces have been removed, but there are still some traces remaining. Additionally, ERASOR mistakenly removed a significant number of static objects, such as the wall and tree trunk highlighted in the orange boxes in the image.

By comparing the observation time difference between the objects and the ground, our method achieved a more significant dynamic object removal effect. As shown in Fig.

TABLE I

COMPARISON RESULTS OF DYNAMIC REMOVAL USING SEMANTICKITTI DATASET. METRICS ARE EXPRESSED AS PR[%]/RR[%]/$\mathbf{F}_1$ SCORE.

Method	Seq.00	Seq.01	Seq.02	Seq.05	Seq.07
Removert-RM3 [11]	85.502/**99.354**/0.919	94.221/93.608/0.939	76.319/96.799/0.853	86.900/87.880/0.874	80.689/98.822/0.888
Removert-RM3+RV1 [11]	86.829/90.617/0.887	95.815/57.077/0.715	83.293/88.371/0.858	88.170/79.981/0.839	82.038/95.504/0.883
ERASOR-0.2 [9]	93.980/97.081/0.955	91.487/95.383/0.934	87.731/97.008/0.921	88.730/98.262/0.933	90.624/**99.271**/0.948
ERASOR-0.4 [9]	80.769/98.353/0.887	79.807/95.080/0.868	78.319/98.464/0.872	72.440/97.831/0.832	82.875/98.221/0.899
ERASOR2 [16]	98.788/98.582/0.987	96.879/94.629/0.957	98.523/**99.709**/**0.991**	**97.582**/98.992/0.983	**98.977**/98.459/**0.987**
DynamicFilter [13]	90.070/91.090/0.906	87.950/87.690/0.878	88.020/86.100/0.871	90.170/84.650/0.873	87.940/86.800/0.874
Nonparametric [12]	94.050/97.190/0.956	91.815/94.096/0.929	91.208/95.510/0.933	93.820/95.740/0.947	91.146/97.284/0.941
Ours	**98.819**/98.686/**0.988**	**99.013**/**95.494**/**0.972**	**98.682**/98.954/0.988	97.316/**99.520**/**0.984**	96.490/98.492/0.975



(a) Original Map (b) Removert (c) ERASOR (d) Ours

Fig. 5. The dynamic removal results in challenging scenarios, the partial enlarged image is shown below. The purple points in the picture is traces left by dynamic objects, and the area with serious error removal is in orange boxes.

TABLE II

RUNTIME PER ITERATION OF SEMANTICKITTI DATASET.

Method	Runtime/iteration [ms]
Removert [11]	73.4
ERASOR [9]	72.6
Nonparametric [12]	59.9
Ours	**23.8**

TABLE III

EACH MODULE'S RUNTIME OF OUR METHOD.

Module	Runtime/iteration [ms]
Ground Segmentation	13.0
Map Management	5.3
Dynamic Removal	5.5
Total	23.8

5(d), our algorithm successfully removed almost all dynamic objects and preserved important details such as tree trunks and buildings. The remaining purple points in the image are caused by objects that remained stationary throughout, so our algorithm did not consider it as a dynamic object.

D. Algorithm Speed

In our algorithm, no matter downward retrieval or upward retrieval, it only needs to compare the observation time of voxels without additional calculation. Moreover, the number of voxels to be retrieved in each frame is much smaller than the number of points. Therefore, compared to other state-of-the-art algorithms, the computing time of our method is reduced by at least 60%, as shown in the Table II.

In addition, in our method, the dynamic removal modules can complete the task within only several milliseconds, and more computing resources are actually used in other modules, as shown in the Table III. The ground segmentation module takes the longest time, up to 13 milliseconds, because we have adopted a two-step ground segmentation algorithm. However, this two-step ground segmentation algorithm minimizes the number of dynamic points in the ground as much as possible, which greatly improves the performance of the dynamic removal.

VI. CONCLUSION

In this paper, we propose a new method for online dynamic objects removal for ground vehicles. Our method identifies dynamic objects by comparing the observation time difference between objects and the ground. Our method achieves excellent dynamic removal performance by only using the map at the current moment, and is more robust than other methods. But since our method relies heavily on the comparison with the ground, it will perform poorly in uneven terrain scenarios and unable to detect aerial objects.

REFERENCES

[1] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term lidar localization using scan context image," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1948–1955, 2019.

[2] E. Stefanini, E. Ciancolini, A. Settimi, and L. Pallottino, "Efficient 2d lidar-based map updating for long-term operations in dynamic environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 832–839.

[3] A. Adkins, T. Chen, and J. Biswas, "Probabilistic object maps for long-term robot localization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 931–938.

[4] H. Wang, C. Xue, Y. Tang, W. Li, F. Wen, and H. Zhang, "Ltsr: Long-term semantic relocalization based on hd map for autonomous vehicles," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2171–2178.

[5] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[6] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[7] S. Pagad, D. Agarwal, S. Narayanan, K. Rangan, H. Kim, and G. Yalla, "Robust method for removing dynamic objects from point clouds," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 765–10 771.

[8] 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.

[9] 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.

[10] J. Schauer and A. Nüchter, "The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid," *IEEE robotics and automation letters*, vol. 3, no. 3, pp. 1679–1686, 2018.

[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. 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. 24 190–24 205, 2022.

[13] 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.

[14] 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.

[15] 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.

[16] H. Lim, L. Nunes, B. Mersch, X. Chen, J. Behley, H. Myung, and C. Stachniss, "Erasor2: Instance-aware robust 3d mapping of the static world in dynamic scenes," in *Robotics: Science and Systems (RSS 2023)*. IEEE, 2023.

[17] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, "Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6529–6536, 2021.

[18] J. Sun, Y. Dai, X. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen, "Efficient spatial-temporal information fusion for lidar-based 3d moving object segmentation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 456–11 463.

[19] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 163–169.

[20] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5067–5073.

[21] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized gicp for fast and accurate 3d point cloud registration," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 054–11 059.

[22] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4861–4868, 2022.

[23] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." in *Robotics: Science and Systems*, vol. 2018, 2018, p. 59.