# OTD: An Online Dynamic Traces Removal Method Based on Observation Time Difference

Rongguang Wu, Zheng Fang, *Member, IEEE*, Chenglin Pang, and Xuankang Wu

***Abstract*— Three-dimensional point cloud map plays an important role in 3-D reconstruction, autonomous robot navigation, autonomous driving, and environmental monitoring. Nowadays, 3-D point cloud map could be obtained through frame-by-frame accumulation of LiDAR point cloud using SLAM technology. However, during this process, the movements of dynamic objects in the environment will leave a large number of traces on the point cloud map, causing difficulties in the subsequent use of the map, such as city model construction and robot autonomous navigation. Therefore, dynamic traces removal is crucial for building clean static maps. However, existing methods for dynamic traces removal are mostly offline, which inevitably incurs additional time consumption. To address this problem, this article proposes an online dynamic traces removal method. We take voxels as the smallest unit for dynamic traces removal, and voxels containing dynamic traces are called dynamic voxels, otherwise they are called static voxels. Our method is based on the assumption that static voxels always appear and disappear simultaneously with the ground below them. Therefore, we call voxel that appears later than the ground as *suddenly appear* dynamic voxel, and voxel that disappears earlier than the ground as *suddenly disappear* dynamic voxel. We call this method of judging dynamic voxels as observation time difference, and propose downward retrieval and upward retrieval methods to remove these two types of dynamic voxels, respectively. We tested our proposed method on SemanticKITTI, UrbanLoco, and author-collected datasets. Experimental results show that our method is more accurate and robust than existing online dynamic traces removal methods. And compared with other methods, our method shortens the time of processing each frame of point cloud by more than 60%. Our method is open-sourced on GitHub: https://github.com/RongguangWu/OTD.**

***Index Terms*— Dynamic objects, mapping, point cloud.**

## I. Introduction

STATIC maps are the most fundamental and crucial part for numerous applications, including smart cities [1], architectural modeling [2], and autonomous navigation [3], [4]. Nowadays, mobile mapping systems (MMSs) equipped with LiDAR sensors has become a common way to obtain maps due to many advantages [5], [6]. The map drawn by LiDAR sensor is called 3-D point cloud map, which is usually obtained by accumulating the point cloud data collected by LiDAR frame by frame. Unfortunately, urban environments usually contain a large number of dynamic objects [7], [8], which are also scanned by LiDAR into point cloud data. Moreover, as these dynamic objects move, a large number of points will be accumulated in the map along the trajectory of the objects. These points are usually called dynamic points, and the traces formed by these points are called dynamic traces [9]. As shown in Fig. 1, the purple area is the dynamic traces left by the dynamic objects. Those dynamic traces will act as obstacles in the process of autonomous navigation and urban modeling, which will affect the subsequent use of maps [10], [11]. Therefore, dynamic traces removal is crucial in constructing clean point cloud map.

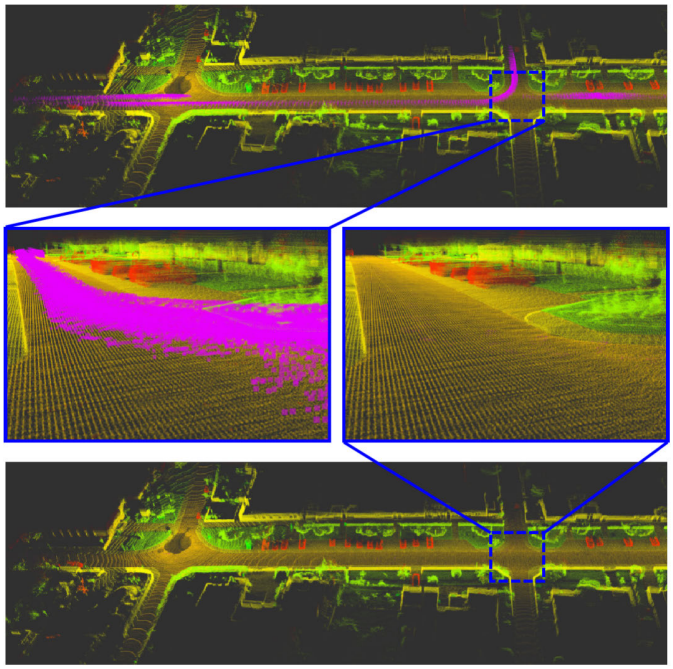


Fig. 1. Results of dynamic traces removal. The upper image is original map. The lower image is static map constructed by our method. The middle image is partially enlarged image, and the purple area is traces left by dynamic objects.

Existing dynamic traces removal methods are usually carried out offline. Generally, a priori map containing dynamic traces is first constructed, and then the original point cloud data is compared with the priori map frame by frame. Finally, the dynamic traces is removed by utilizing the differences between the single-frame point cloud and the prior map, resulting in a map that only contains static point

Manuscript received 27 October 2023; revised 5 May 2024 and 1 June 2024; accepted 16 June 2024. Date of publication 4 July 2024; date of current version 15 July 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62073066, in part by the Fundamental Research Funds for the Central Universities under Grant N2226001, and in part by the 111 Project under Grant B16009. *(Corresponding author: Zheng Fang.)*

Rongguang Wu and Zheng Fang are with the Faculty of Robot Science and Engineering, the National Frontiers Science Center for Industrial Intelligence and Systems Optimization, and the Key Laboratory of Data Analytics and Optimization for Smart Industry, Ministry of Education, Northeastern University, Shenyang 110819, China (e-mail: fangzheng@mail.neu.edu.cn).

Chenglin Pang and Xuankang Wu are with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China.

Digital Object Identifier 10.1109/TGRS.2024.3422084

cloud data. Several effective methods have been proposed, such as [9], [11], and [12]. Although those methods achieve significant dynamic removal performance, they require additional time cost. Therefore, this article focuses on removing dynamic traces while building the map simultaneously, namely online dynamic traces removal.

For online dynamic traces removal, an effective idea is to directly apply the ideas used in offline removal methods, such as [13], [14], and [15]. However, this way of directly applying offline method to online processing has a natural defect: there is no prior map in online processing. 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 traces removal.

In this article, we propose a novel online dynamic traces removal method for ground vehicles, which is an extended work of our conference paper [16]. Our method regards voxels as the smallest unit for dynamic traces removal, and voxels containing dynamic traces are called dynamic voxels, otherwise they are called static voxels. Considering that common dynamic traces are caused by objects in contact with the ground [9] (vehicles and pedestrians), our method judges the dynamic voxels by comparing the observation time difference between the voxels and the ground below them. Compared with other online dynamic traces removal methods, our method is more accurate and robust, and reduces the processing time by at least 60%. Our contributions are as follows.

1) We determine whether a voxel is dynamic by examining whether it appears and disappears simultaneously with the ground below it. We refer to this method as observation time difference.
2) We classify dynamic voxels into two categories: *suddenly appear* and *suddenly disappear*, and propose downward retrieval and upward retrieval to remove them, respectively.
3) We conduct extensive experimental validations on real-world datasets, Urbanloco [17] datasets and SemanticKITTI [18] datasets. Compared to other online dynamic traces removal methods, our approach is more effective and robust, and reduces the average processing time per frame by over 60%.
4) 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 is open-sourced on GitHub.

The rest of the article is organized as follows. Section II reviews the related work about dynamic traces removal. Section III explains the details of our proposed dynamic removal method. Experimental comparisons and analyses are carried out in Section IV. Finally, Section V summarizes the contributions and describes future works.

## II. Related Work

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

### A. Offline Removal Approaches

One mainstream concept of offline dynamic traces removal is to detect dynamic traces 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. Octomap [19], [20] and TSDF [21], [22], [23] are typical methods that use ray tracing to remove dynamic traces. However, checking voxels is computationally expensive. Even though Schauer and Nüchter [11] 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 [8], [22] is proposed, which determines dynamic traces 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. To solve this problem, Kim and Kim [12] added revert module, which can revert misidentified points back to the map. However, the revert module is still based on visibility, therefore, it cannot solve this problem fundamentally.

Additionally, Lim et al. [9] 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 traces by comparing the pseudooccupancy differences between the query frame and the map in the occupied grid. Furthermore, Lim et al. [24] extended this approach by introducing instance segmentation and proposed a method to reject dynamic traces at instance-level. However, this method requires significant time consumption (200 ms/frame), making it unsuitable for real-time online processing.

### B. Online Removal Approaches

In recent years, with the development of deep learning, the research on point cloud analysis has attracted significant attention [25], [26]. Therefore, numerous moving objects segmentation (MOS) methods have been proposed. For example, Chen et al. [27] generated residual images from previous scans and input them together with the range image of current scan into CNN to distinguish dynamic objects and static objects. Sun et al. [28] 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 [13], [15]. Qian et al. [13] applied the visibility-based method to the online process and combined it with LIO-SAM, using online map obtained by LiDAR odometry to replace the prior map. However, the online map only contains points before the current moment, not after the current moment, which can result in a decrease

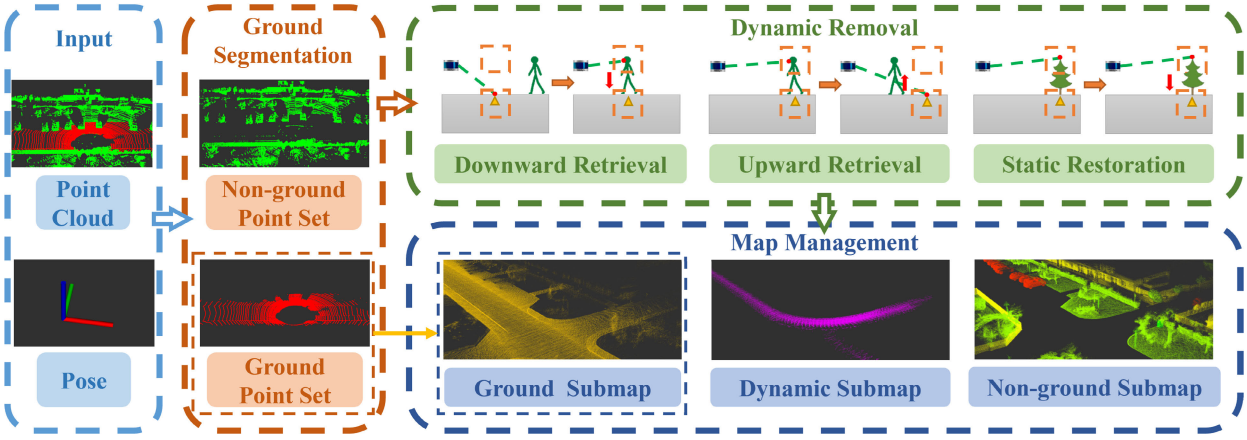



Fig. 2. Overview of our proposed method. Our proposed method includes three modules: ground segmentation, map management, and dynamic removal. The point cloud is divided into ground point set and nonground point set in ground segmentation module, and then are sent to map management module and dynamic removal module together with the pose. After downward retrieval, upward retrieval, and static restoration, dynamic traces can be recognized.

in the performance of dynamic removal. To solve this problem, Fan et al. [15] accumulated multiframe point clouds to build a local submap, and then compared each frame with the local submap to judge dynamic traces. 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 article, we propose a novel online dynamic traces removal method. Our method determines dynamic traces by comparing observation time difference between the point and the corresponding ground, and achieves effective performance in dynamic traces removal.

## III. Methodology

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

Similar to [9], we believe that in most cases, dynamic traces are caused by moving objects that are in contact with the ground, whether they are pedestrians or terrestrial vehicles. Based on this, we assume that there are no dynamic points present in the ground, and the ground can be utilized as a reference to assist in the removal of dynamic traces. Therefore, the input point cloud is first segmented into ground point set and nonground point set in the ground segmentation module. Then, the ground point set is directly added to the ground submap because we assume that there are no dynamic traces in the ground point set. As for the nonground point set, it is fed into the dynamic removal module for further processing. In the dynamic removal module, we remove dynamic traces by comparing the observation time difference between the ground submap and nonground submap. Finally, the removed dynamic traces are consolidated into a dynamic submap, while the remaining nonground submap and ground submap are considered as static map.

### A. Notations and Problem Definition

Let $\mathcal{P}_k = [\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_k]$ be a set of point clouds from start time $t_0$ to current time $t_k$, where $\mathbf{P}_i$ is the point cloud at time $t_i$. For each point cloud $\mathbf{P}$ in $\mathcal{P}_k$, it is defined as $\mathbf{P} = [\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_N]$, where $N$ denotes the size of $\mathbf{P}$ and each point is defined as $\mathbf{p} = [x, y, z]^{\mathrm{T}}$. Using the input pose $\mathbf{T}_i^W \in \mathrm{SE}(3)$ at time $t_i$, the point cloud $\mathbf{P}_i$ at time $t_i$ can be transformed into the world frame $W$

$$^W\mathbf{P}_i = \mathbf{T}_i^W \mathbf{P}_i. \tag{1}$$

Based on this, we can get the map $\mathcal{M}_k$ at the time $t_k$ as follows:

$$\mathcal{M}_k = \bigcup {}^W\mathbf{P}_i, i \in [0, k]. \tag{2}$$

Note that $\mathcal{M}_k$ is in the world frame and contains all measured dynamic points.

When the point cloud $\mathbf{P}_{k+1}$ and the pose $\mathbf{T}_{k+1}^W$ at the time $t_{k+1}$ are received, the problem of online dynamic traces removal can be described as

$$\hat{\mathcal{M}}_{k+1}^{\mathrm{sta}} = \mathcal{M}_k - \hat{\mathcal{M}}_k^{\mathrm{dyn}} + \mathbf{T}_{k+1}^W \left( \mathbf{P}_{k+1} - \hat{\mathbf{P}}_{k+1}^{\mathrm{dyn}} \right) \tag{3}$$

where $\hat{\mathcal{M}}_k^{\mathrm{dyn}}$ refers to the estimated dynamic points in the map at time $t_k$. $\hat{\mathbf{P}}_{k+1}^{\mathrm{dyn}}$ refers to the estimated dynamic points in the point cloud at time $t_{k+1}$. $\hat{\mathcal{M}}_k^{\mathrm{dyn}}$ and $\hat{\mathbf{P}}_{k+1}^{\mathrm{dyn}}$ are the dynamic traces to be removed in this article.

### B. Dynamic Objects Classification

Dynamic traces removal aims to remove all dynamic points obtained by dynamic objects. However, treating each point in the map individually is impractical for online processes. Therefore, we use voxels as the fundamental units for dynamic traces removal. If a voxel contains dynamic points, it is considered as a dynamic voxel, and all points within that voxel are identified as dynamic points and removed. Otherwise, the voxel is considered as a static voxel. Furthermore, if a point in a certain voxel is observed at time $t_k$, then we consider that the voxel can be observed at time $t_k$, otherwise we consider that the voxel cannot be observed at time $t_k$.

Fundamentally, if a voxel in the map is static, it should be observed and disappear from the field of view at the same time as the ground it is located on. As shown in Fig. 3, the voxel represented by the red box contains a static object. At the beginning, the LiDAR is far from the voxel, so both the voxel

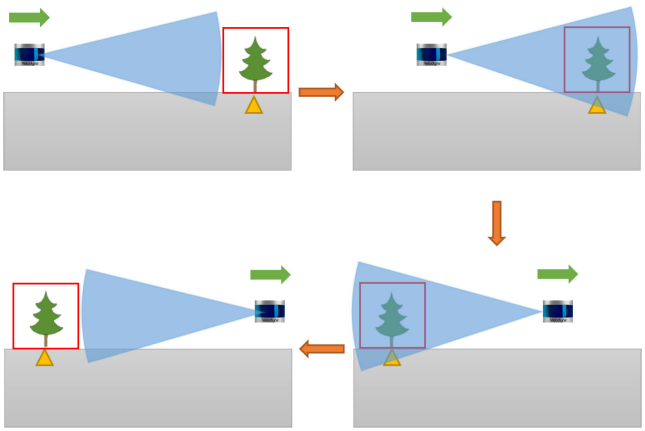
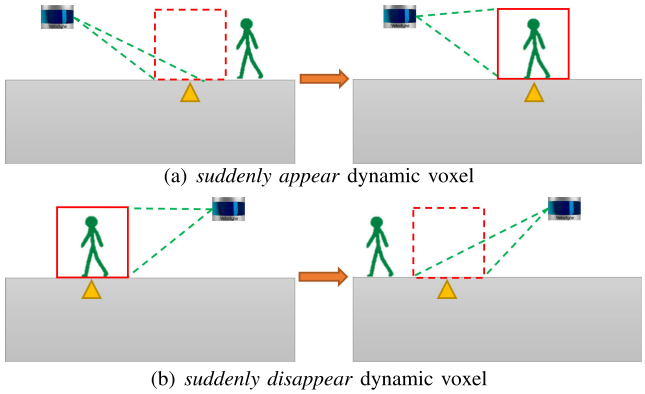


Fig. 3. Static voxel always appears and disappears simultaneously with the ground. The red box represents a voxel, and the yellow triangle represents a specific fixed position in the map. The gray and blue areas represent the ground and detection range of the LiDAR, respectively. The green arrow represents the direction of LiDAR movement, while the orange arrow represents the chronological order of the images.



(a) *suddenly appear* dynamic voxel

(b) *suddenly disappear* dynamic voxel

Fig. 4. Dynamic voxels classification. The red box represents a voxel, and the green stick figure represents a dynamic object. Only ground was observed at the beginning in (a). But after a period of time, both ground and nonground voxel were observed. The voxel is a *suddenly appear* dynamic voxel. On the contrary, both ground and nonground voxel were observed at the beginning in (b). But after a period of time, only ground was observed. The voxel is a *suddenly disappear* dynamic voxel.

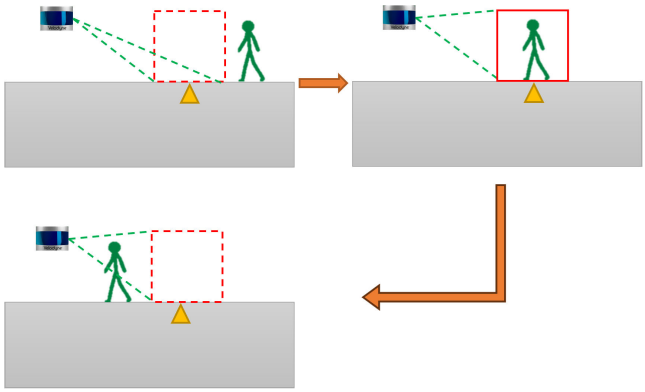


Fig. 5. Continuously moving object. The red box represents a voxel in the map, and the green stick figure represents a continuously moving object. When it moves to the voxel, the voxel meets the definition of a *suddenly disappear* dynamic voxel. When it leaves that voxel, the voxel meets the definition of a *suddenly disappear* dynamic voxel.

and the ground cannot be observed. As the LiDAR moves forward, when the voxel enters the detection range of the LiDAR, both the voxel and the ground should be observed simultaneously. As the LiDAR continues to move, the voxel and the ground will also disappear from the field of view simultaneously. In other words, the voxel and the ground must appear and disappear simultaneously. On the contrary, if either of the two conditions is not met, the voxel is considered as a dynamic voxel. Therefore, we classify dynamic voxels into the following two categories.

1) At a certain position, only ground was observed but not nonground voxel at time $t_0$. Both ground and nonground voxel were observed since time $t_1 (t_1 > t_0)$. Then, the nonground voxel at that position is defined as a *suddenly appear* dynamic voxel.
2) At a certain position, both ground and nonground voxel were observed at time $t_0$. Only ground was observed but not nonground voxel since time $t_1 (t_1 > t_0)$. Then, the nonground voxel at that position is defined as a *suddenly disappear* dynamic voxel.

As shown in Fig. 4(a), at the beginning, only the ground has been observed, but no nonground voxel. However, after some time, the object moved into the voxel. At this moment, both the voxel and the ground can be observed simultaneously. This indicates that the voxel appeared later than the ground, so the voxel is considered as a *suddenly appear* dynamic voxel. Similarly, as shown in Fig. 4(b), both the ground and a nonground voxel were simultaneously observed at the beginning. After some time, the object has left the voxel, which means that the voxel is currently unable to be observed. However, the ground below the voxel can still be observed, indicating that the voxel disappears earlier than the ground. In this case, the voxel is considered as a *suddenly disappear* dynamic voxel.

It should be noted that continuously moving objects often simultaneously satisfy both the definition of *suddenly appear* and *suddenly disappear*. As shown in Fig. 5, at the beginning, only the ground was observed at the voxel. When the object moves to the voxel, the voxel meets the definition of a *suddenly disappear* dynamic voxel. However, when the object leaves that voxel, the voxel meets the definition of a *suddenly disappear* dynamic voxel.

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

Note that we focus more on voxels than on objects because objects are often difficult to distinguish in point cloud maps. Regardless of whether a voxel contains only part of an object or contains many objects, as long as the points contained in it are points of static objects, the voxel will be determined as a static voxel. On the contrary, as long as the points contained in a voxel are points of dynamic objects, the voxel will be determined as a dynamic voxel.

In addition, there is also a situation where a voxel may contain both dynamic and static objects. This voxel will be judged as a static voxel because it meets the condition of appearing and disappearing simultaneously, causing the dynamic parts

within it unable to be removed. This is a limitation caused by using voxels as the smallest processing unit, but compared to the huge efficiency improvement it brings, we believe that this limitation is acceptable.

### C. Ground Segmentation

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 [29] to remove most of the nonground points. Then, we introduce principal component analysis (PCA) proposed in [30] to further refine it and obtain the final ground point set $\mathbf{G}$, while the remaining points are classified as nonground point set $\mathbf{U}$. Since we removed most of the nonground points in the preprocessing, the ground fitting process will be relatively more accurate.

### D. Map Management

In Section III-B, we mentioned using voxels as the fundamental units for dynamic traces removal. Therefore, following [31], we build a voxel map to manage the point cloud map. In order to facilitate the management of ground points, nonground points, and dynamic points, we divide the global map into three parts: ground submap ${}^{g}\mathcal{V}$, nonground submap ${}^{n}\mathcal{V}$, and dynamic submap ${}^{d}\mathcal{V}$. These three submaps share the same coordinate system, but are stored separately.

The map management module receives the ground point set from the ground segmentation module and the static point set and dynamic point set from the dynamic removal module, and adds them, respectively, to the ground submap, nonground submap, and dynamic submap. For convenience, we will refer to the voxels located in the ground submap, nonground submap, and dynamic submap as ground voxels, nonground voxels, and dynamic voxels, respectively, in the following paper. In addition, in order to obtain the observation time of each voxel, we store the index of LiDAR frame for each point besides storing the points in the voxel, and record all the frame indexes in a Set, which is denoted as $\mathcal{S}$. Then, the smallest frame index $\gamma_{\min}$ in the $\mathcal{S}$ can represent the time when the voxel was first observed, while the largest frame index $\gamma_{\max}$ can represent the time when the voxel was last observed.

### E. Dynamic Removal

The purpose of the dynamic removal module is to remove the dynamic points in nonground submap ${}^{n}\mathcal{V}_k$ at $t_k$ and nonground point set $\mathbf{U}_{k+1}$ at $t_{k+1}$. Based on the observation time difference we introduced in Section III-B, we can compare the observation time difference between each nonground 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, therefore we added static restoration process.

**Algorithm 1** Downward Retrieval

**Data:** Nonground Point Set $\mathbf{U}_{k+1}^{W}$ at time $t_{k+1}$; Nonground Submap ${}^{n}\mathcal{V}$ at time $t_k$; Ground Submap ${}^{g}\mathcal{V}$ at time $t_k$
**Result:** Static Point Set $\mathbf{S}_{k+1}^{W}$; Dynamic Point Set $\mathbf{D}_{k+1}^{W}$; Dynamic Submap ${}^{d}\mathcal{V}$

```
for p_i^W ∈ U_{k+1}^W do
    Let ^nV_i be the voxel of p_i^W in nonground submap;
    Let ^nγ_min be the minimum frame index of ^nV_i;
    for k = 1, 2, ... do
        ^gV_i = ^nV_i - k · (0, 0, 1)^T;
        if ^gV_i ≠ ∅ then
            Let ^gγ_min be the minimum frame index of ^gV_i;
        end
    end
    if ^nγ_min − ^gγ_min > τ_ret then
        Find dynamic voxel ^dV_i of p_i^W;
        ^dV_i = ^dV_i ∪ ^nV_i;
        ^nV_i = ∅;
        D_{k+1} += p_i^W;
    else
        S_{k+1} += p_i^W;
    end
end
```

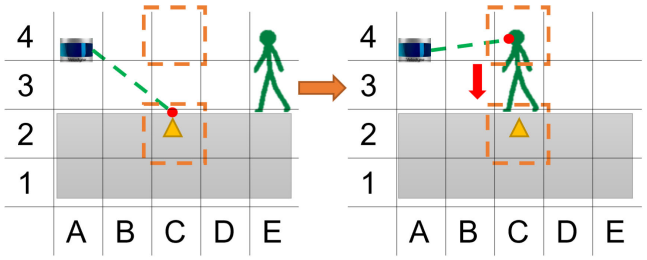



Fig. 6. Downward retrieval. The image on the left represents the $i$th LiDAR frame, while the image on the right represents the $j$th LiDAR frame ($j > i$). The points observed by LiDAR are represented by red dots. The object was at position E at the $i$th LiDAR frame, and it moved to position C at the $j$th LiDAR frame. We first observed the nonground 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.

Using the input pose $\mathbf{T}_{k+1}^{W}$ (usually comes from the SLAM front-end), we can transform $\mathbf{G}_{k+1}$ and $\mathbf{U}_{k+1}$ into the world frame and denote them as $\mathbf{G}_{k+1}^{W}$ and $\mathbf{U}_{k+1}^{W}$, respectively.

*1) Downward Retrieval:* The downward retrieval process we proposed is shown in Algorithm 1 and Fig. 6. Since the *suddenly appear* dynamic voxels are the voxels that can be observed at time $t_{k+1}$, it must containing the points in $\mathbf{U}_{k+1}^{W}$. Therefore, to achieve the removal of *suddenly appear* dynamic voxels, we only need to examine the voxel containing $\mathbf{U}_{k+1}^{W}$.

For each point $\mathbf{p}_i^{W}$ in $\mathbf{U}_{k+1}^{W}$, we calculate its voxel ${}^{n}\mathbf{V}_i$ in nonground 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.

**Algorithm 2** Upward Retrieval

**Data:** Ground Point Set $\mathbf{G}_{k+1}^{W}$ at time $t_{k+1}$; Nonground Submap ${}^{n}\mathcal{V}$ at time $t_k$; Ground Submap ${}^{g}\mathcal{V}$ at time $t_k$
**Result:** Dynamic Submap ${}^{d}\mathcal{V}$
**for** $\mathbf{p}_i^W \in \mathbf{G}_{k+1}^{W}$ **do**
  Let ${}^{g}\mathbf{V}_i$ be the voxel of $\mathbf{p}_i^W$ in ground submap;
  Let ${}^{g}\gamma_{max}$ be the maximum frame index of ${}^{g}\mathbf{V}_i$;
  **for** $k = 1, 2, \ldots$ **do**
    ${}^{n}\mathbf{V}_i = {}^{g}\mathbf{V}_i + k \cdot (0, 0, 1)^{\mathrm{T}}$;
    **if** ${}^{n}\mathbf{V}_i \neq \emptyset$ **then**
      Let ${}^{n}\gamma_{max}$ be the maximum frame index of ${}^{n}\mathbf{V}_i$;
      **if** ${}^{g}\gamma_{max} - {}^{n}\gamma_{max} > \tau_{ret}$ **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**
  **end**
**end**

Next, in the ground submap, we retrieve the ground voxel ${}^{g}\mathbf{V}_i$ within 3 m 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:

$$ {}^{n}\gamma_{\min} - {}^{g}\gamma_{\min} > \tau_{\text{ret}} \tag{4} $$

where $\tau_{\text{ret}}$ is a predefined threshold. Then, we move all points as well as all frame indexes in ${}^{n}\mathbf{V}_i$ into dynamic submap. Meanwhile, point $\mathbf{p}_i^W$ is identified as a dynamic point and added to the dynamic point set $\mathbf{D}_{k+1}^{W}$. Otherwise, it will be added to the static point set $\mathbf{S}_{k+1}^{W}$.

As shown in Fig. 6, the nonground 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 4, if $j - i > \tau_{\text{ret}}$, then the nonground voxel (C, 4) is considered as a dynamic voxel.

*2) Upward Retrieval:* The upward retrieval process that we proposed is shown in Algorithm 2. In Section III-E1, we eliminated *suddenly appear* dynamic voxels by downward retrieval. However, downward retrieval cannot remove *suddenly disappear* dynamic voxels. Because the *suddenly disappear* dynamic voxels can no longer be observed at time $t_{k+1}$, these voxels must not contain the points in the $\mathbf{U}_{k+1}^{W}$. Indeed, the ground below the *suddenly disappear* dynamic voxels can still be observed. Therefore, we use voxels containing points in $\mathbf{G}_{k+1}^{W}$ to retrieve upward to find *suddenly disappear* dynamic voxels.

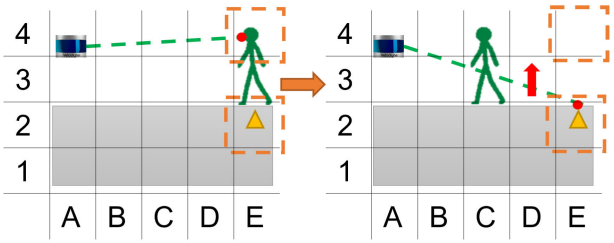



Fig. 7. Upward retrieval. The image on the left represents the $i$th LiDAR frame, while the image on the right represents the $j$th LiDAR frame ($j > i$). The points observed by LiDAR are represented by red dots. The object was at position E at the $i$th LiDAR frame, and it moved to position C at the $j$th LiDAR frame. We last observed the nonground voxel (E, 4) in the $i$th LiDAR frame, but we still can observe the ground voxel (E, 2) below it in the $j$th frame. Therefore, we compare the maximum LiDAR frame index between these two voxels to determine whether (E, 4) is dynamic or not.

For each point $\mathbf{p}_i^W$ in $\mathbf{G}_{k+1}^{W}$, 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 nonground voxels within 3 m by ascending along the $z$-axis in the nonground submap. We check each of these nonground voxels ${}^{n}\mathbf{V}_i$ in turn, and define the largest frame index in it as ${}^{n}\gamma_{\max}$.

Finally, if the following criteria are satisfied:

$$ {}^{g}\gamma_{\max} - {}^{n}\gamma_{\max} > \tau_{\text{ret}}. \tag{5} $$

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

As shown in Fig. 7, the ground voxel (E, 2) was last observed in the $j$th frame. Therefore, the maximum LiDAR frame index for (E, 2) is recorded as $j$. Then, we find the nonground voxel (E, 4) by ascending along $z$-axis from (E, 2). Since (E, 4) was last observed in the $i$th frame, the maximum LiDAR frame index for (E, 4) is $i$. According to 5, if $j - i > \tau_{\text{ret}}$, then the nonground voxel (E, 4) is considered as a dynamic voxel.

*3) Static Restoration:* In Sections III-E1 and III-E2, we used the methods of upward retrieval and downward retrieval to remove dynamic voxels and put dynamic points into the dynamic submap. However, due to the influence of ground slope and LiDAR measurement noise, some static voxels may be mistakenly identified as dynamic voxels. Therefore, we introduce a static restoration module to place these misclassified static voxels back into the nonground submap.

Because static voxels always appear and disappear simultaneously with the ground, their total number of observations with the ground is similar. Therefore, we assume that if the total times of observation of a dynamic voxel and the ground voxel is similar, then the dynamic voxel should be static. Based on this, we designed the process of static restoration (as shown in Algorithm 3).

For each point $\mathbf{p}_i^W$ in $\mathbf{U}_{k+1}^{W}$, we calculate its voxel ${}^{d}\mathbf{V}_i$ in dynamic submap. Then, we define the total number of frame indexes in ${}^{d}\mathbf{V}_i$ as ${}^{d}\gamma_{\text{sum}}$. ${}^{d}\gamma_{\text{sum}}$ can represent the total times the voxel has been observed.

Next, in the ground submap, we retrieve the ground voxel ${}^{g}\mathbf{V}_i$ within 3 m by descending along $z$-axis from ${}^{d}\mathbf{V}_i$. Finally, we define the total number of frame indexes in the ground voxel ${}^{g}\mathbf{V}_i$ as ${}^{g}\gamma_{\text{sum}}$. ${}^{g}\gamma_{\text{sum}}$ can represent total times the ground voxel has been observed.

**Algorithm 3** Static Restoration

**Data:** Nonground Point Set $\mathbf{U}_{k+1}^{W}$ at time $t_{k+1}$; Dynamic Submap ${}^{d}\mathcal{V}$ at time $t_k$; Ground Submap ${}^{g}\mathcal{V}$ at time $t_k$
**Result:** Nonground Submap ${}^{n}\mathcal{V}$
**for** $\mathbf{p}_i^W \in \mathbf{U}_{k+1}^{W}$ **do**
  Let ${}^{d}\mathbf{V}_i$ be the voxel of $\mathbf{p}_i^W$ in dynamic submap;
  Let ${}^{d}\gamma_{sum}$ be the total number of the frame indices in ${}^{d}\mathbf{V}_i$;
  **for** $k = 1, 2, \ldots$ **do**
    ${}^{g}\mathbf{V}_i = {}^{d}\mathbf{V}_i - k \cdot (0, 0, 1)^{\mathrm{T}}$;
    **if** ${}^{g}\mathbf{V}_i \neq \emptyset$ **then**
      Let ${}^{g}\gamma_{sum}$ be the total number of the frame indices in ${}^{g}\mathbf{V}_i$;
    **end**
  **end**
  **if** ${}^{n}\gamma_{sum} - {}^{g}\gamma_{sum} > \tau_{res}$ **then**
    Find nonground voxel ${}^{n}\mathbf{V}_i$ of $\mathbf{p}_i^W$;
    ${}^{n}\mathbf{V}_i = {}^{n}\mathbf{V}_i \cup {}^{d}\mathbf{V}_i$;
    ${}^{d}\mathbf{V}_i = \emptyset$;
  **end**
**end**

If the difference between ${}^{g}\gamma_{\mathrm{sum}}$ and ${}^{d}\gamma_{\mathrm{sum}}$ is less than a predefined threshold $\tau_{\mathrm{res}}$, ${}^{d}\mathbf{V}_i$ 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 to the nonground submap.

It is important to note that comparing the total number of observations is a necessary but insufficient condition to judge whether a voxel is static or not. For example, for a narrow tree trunk, it may not be observed at every moment due to the sparsity of LiDAR scans. In that case, the total observation times of the tree trunk would be much smaller than that of the ground it is located on. However, at the same time, this condition can effectively prevent misclassifying true dynamic voxels as static ones. Therefore, we incorporate this condition into the static restoration process to only restore the more reliable static voxels.

## IV. Experiments

In this section, in order to prove the effectiveness of our proposed method, we conducted online dynamic traces removal experiments on the public SemanticKITTI dataset, Urbanloco dataset, and author-collected 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. In all datasets, we set identical dynamic removal parameters, where $\tau_{\mathrm{ret}} = 15$ and $\tau_{\mathrm{res}} = 5$.

### A. Dataset

To verify the performance of our algorithm, we conduct quantitative experimental evaluation on the SemanticKITTI dataset. However, due to the small proportion of dynamic objects in SemanticKITTI, we additionally performed qualitative demonstrations on Urbanloco and the actual dataset recorded by ourselves.


Fig. 8. Our experimental platform is equipped with LiDAR that has 16 laser beams.

*1) SemanticKITTI Dataset:* The original KITTI dataset [32] provides LiDAR that has 64 laser beams (Velodyne HDL-64E) with an FOV of 30° in vertical and 360° in horizontal. It encompasses various scenarios, including urban areas, suburban areas, and highways. The SemanticKITTI [18] dataset provides pointwise labeled LiDAR points based on KITTI, including 28 different object classes.

Additionally, since dynamic objects are not always present in SemanticKITTI, referring to [9], we selected frames 00 (4390–4530), 01 (150–250), 02 (860–950), 05 (2350–2670), and 07 (630–820) as dynamic removal benchmark where the numbers in parenthesis indicate the start and end frames.

*2) Urbanloco Dataset:* In order to verify the performance of our algorithm on roads with high-speed moving vehicles, we conducted further verification on the HK dataset of Urbanloco [17]. Urbanloco provides LiDAR that has 32 laser beams (Velodyne HDL 32E) with an FOV of 40° in vertical. It contains a large number of moving cars and pedestrians, and there are scenes such as traffic jam, close following of other vehicles and so on.

*3) Author-Collected Dataset:* In addition, we also tested our proposed method on the campus dataset recorded by ourselves. Our experimental platform is shown in Fig. 8, on which a RoboSense RS-LiDAR-16 is equipped. The LiDAR has 16 laser beams with an FOV of 30° in vertical. The test scene is campus, which contains a large number of pedestrians, bicycles, and a small number of cars.

### B. 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 are defined as follows.

1) *PR:* (# of preserved static points)/(# of total static points on the raw map).
2) *RR:* 1 − (# of preserved dynamic points)/(# of total dynamic points on the raw map).

PR and RR represent the preservation rate of static objects and the rejection rate of dynamic objects, respectively. The basis for us to judge whether the point $\mathbf{q}$ in the original map

TABLE I
COMPARISON RESULTS OF DYNAMIC TRACES 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 [12] | 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 [12] | 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 [24] | 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 [15] | 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 [14] | 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 |
| Ours w/o up | 98.915/96.469/0.978 | 99.016/89.209/0.938 | 99.417/57.229/0.726 | 98.522/97.215/0.979 | 96.548/95.368/0.959 |
| Ours w/o down | 99.066/94.272/0.966 | 99.101/93.433/0.962 | 98.739/98.094/0.984 | 99.145/81.834/0.897 | 97.776/96.094/0.969 |

is preserved is as follows:

$$\mathcal{K}(\mathbf{q}, \mathbf{s}) = \text{iVoI}(\mathbf{q}, \mathbf{s}) \wedge \psi(\mathbf{q}, \mathbf{s}) \tag{6}$$

where $\mathbf{s}$ is the nearest neighbor point of $\mathbf{q}$. IVoI$(\cdot)$ and $\psi(\cdot)$ are two bool functions. IVoI$(\cdot)$ will return true if both points belong to the same voxel. Likewise, $\psi(\cdot)$ will return true if the two points are both static points or both dynamic points. Note that both IVoI$(\cdot)$ and $\psi(\cdot)$ are calculated voxel-wise, so the difference in voxel size will affect their computations. Therefore, we set the voxel size to 0.2 for all methods for a fair comparison.

### C. Comparison in SemanticKITTI Datasets

We compare the proposed method with other state-of-the-art dynamic removal methods, including Removert [12], ERASOR [9], and ERASOR2 [24] for offline removal, and DynamicFilter [15] and Nonparametric [14] for online removal. The poses required by our method are provided by SuMa [33] which contains inherent uncertainty. Table I shows the experimental results of ours and other methods on five segments of SemanticKITTI. The upper part of Table I consists of offline methods, while the lower part consists of online methods. 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 [12]. 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]. We use the results or default parameters provided by the authors.

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 traces, resulting in a low PR. In the revert process, some dynamic points will be put back into the map, therefore, 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 five segments while other methods vary greatly in performance across different segments, which means that our method is more robust than other methods.

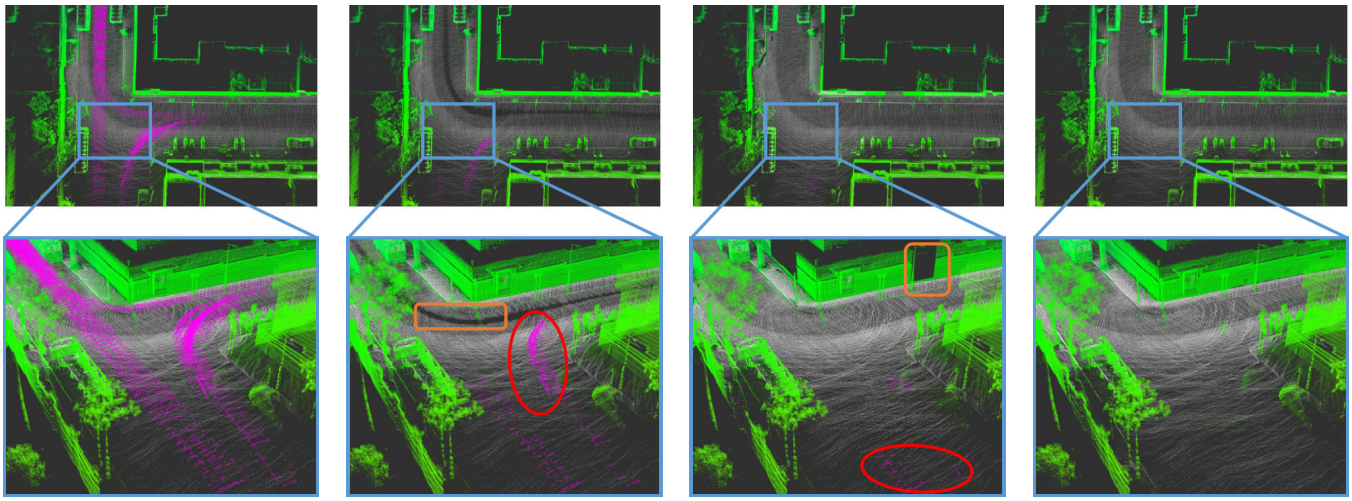
TABLE II
COMPARISON RESULTS OF DYNAMIC TRACES REMOVAL USING URBANLOCO AND AUTHOR-COLLECTED DATASET. METRICS ARE EXPRESSED AS PR[%]/RR[%]/$\mathbf{F}_1$ SCORE

| Method | Urbanloco | campus | challenging |
|---|---|---|---|
| Removert [12] | 96.680/85.030/0.905 | 94.232/84.056/0.889 | **96.780**/91.893/0.943 |
| ERASOR [9] | 93.770/94.682/0.942 | 89.934/86.689/0.883 | 94.465/91.832/0.931 |
| Ours | **98.006/97.897/0.980** | **98.618/98.886/0.988** | 92.868/**98.326/0.955** |
| Ours w/o up | 98.156/85.054/0.911 | 99.232/96.770/0.979 | 94.874/91.176/0.949 |
| Ours w/o down | 98.027/83.850/0.904 | 98.883/95.052/0.969 | 93.798/89.568/0.916 |

### D. Comparison in Urbanloco and Actual datasets

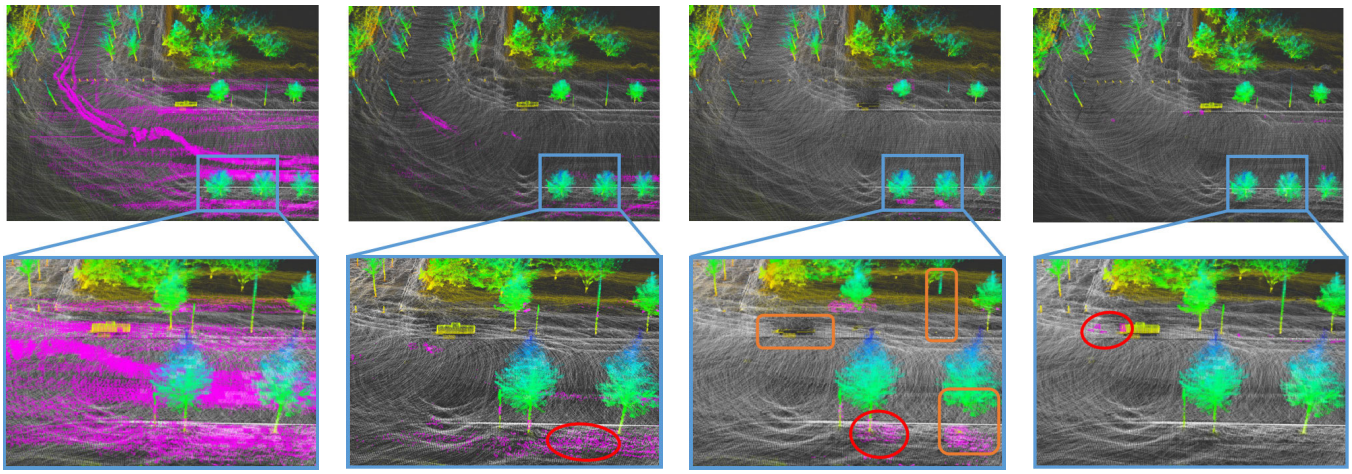
The quantitative result is shown in Table II and the qualitative result is shown in Figs. 9 and 10. As of now, DynamicFilter, Nonparametric, and ERASOR2 have not been open-sourced, we only compared our method with Removert and ERASOR. The poses of both experiments are estimated in real-time by FAST-LIO [34]. Fig. 9 is the road scene in Urbanloco HK dataset, and the bottom part is a partially enlarged image at the corner, where white points represent the ground points and purple points are traces left by dynamic objects. There are cars driving in front, behind, and on the sides of the sensor in the scene of Fig. 9, therefore, a lot of traces are left. Fig. 10 is the actual recorded campus scene, which is recorded outside a highly crowded football field. The scene contains a large number of walkers and cyclists, who leave a large number of dynamic traces. Please check our experiment videos to have a better understanding.[1]

[1]https://youtu.be/vS-Ghlh-NZk



(a) (b) (c) (d)

Fig. 9. Dynamic traces removal results on the urban road in Urbanloco, the partial enlarged image is shown below. The purple points in the picture is dynamic points, and the area with serious error removal is in orange boxes. (a) Original map. (b) Removert. (c) ERASOR. (d) Ours.



(a) (b) (c) (d)

Fig. 10. Dynamic traces removal results in actual recorded campus scene, the partial enlarged images are shown below. The purple points in the picture is traces dynamic points, and the area with serious error removal is in orange boxes. (a) Original map. (b) Removert. (c) ERASOR. (d) Ours.

Removert achieved similar results in both scenarios. Dynamic traces close to the sensor can be effectively removed, but dynamic traces far away from the sensor still remain, such as the points within the red circles in Figs. 9(b) and 10(b). Therefore, the RR of Removert is the lowest in Table II. Furthermore, large areas of ground points are removed in urban scenarios, such as the points within the yellow boxes in Fig. 9(b). This is caused by potential limitations of visibility-based methods. As the range measurement from the ground becomes longer, the incidence angle of a point becomes more ambiguous, therefore, the neighboring ground points may be falsely considered as dynamic points. As shown in Fig. 9(c), ERASOR performs excellently in urban scenes, removing nearly all dynamic points. Unfortunately, ERASOR identified the walls around the corner as dynamic points and removed them at the same time [such as the wall within the orange box in Fig. 9(c)]. Despite lowering the scan ratio threshold to 0.05, this issue cannot be solved. Furthermore, the false-positive issue of ERASOR is more severe in campus scenes. As shown in Fig. 10(c), the tree trunks, pillars, and flower beds within the orange boxes have all been mistakenly identified as dynamic points and removed by ERASOR. Therefore, as shown in Table II, ERASOR achieved the lowest PR in these two scenarios. Moreover, there are still some dynamic points that have not been removed [such as the points within the red circle in Fig. 10(c)]. We believe that the significant performance difference in the two scenarios is due to the reduction in laser beams of LiDAR sensor. In the Urbanloco dataset, a LiDAR sensor with 32 laser beams was used, resulting in a denser point cloud. However, in the author-collected dataset, a LiDAR sensor with 16 laser beams was used, which resulted in a performance decline for ERASOR.

Since our method compares the observation time difference between voxel and the ground, as long as a voxel is observed once within a certain period of time, it will not be classified as a dynamic voxel. Therefore, our method can remove most dynamic traces while preserving static points as much as possible, as shown in Figs. 9(d) and 10(d). Meanwhile, our method achieves the best results in these two scenarios in Table II. In the red circle at the bottom of Fig. 10(d), there are dynamic traces that our method cannot remove. This is because two different pedestrians passed this location at different times.

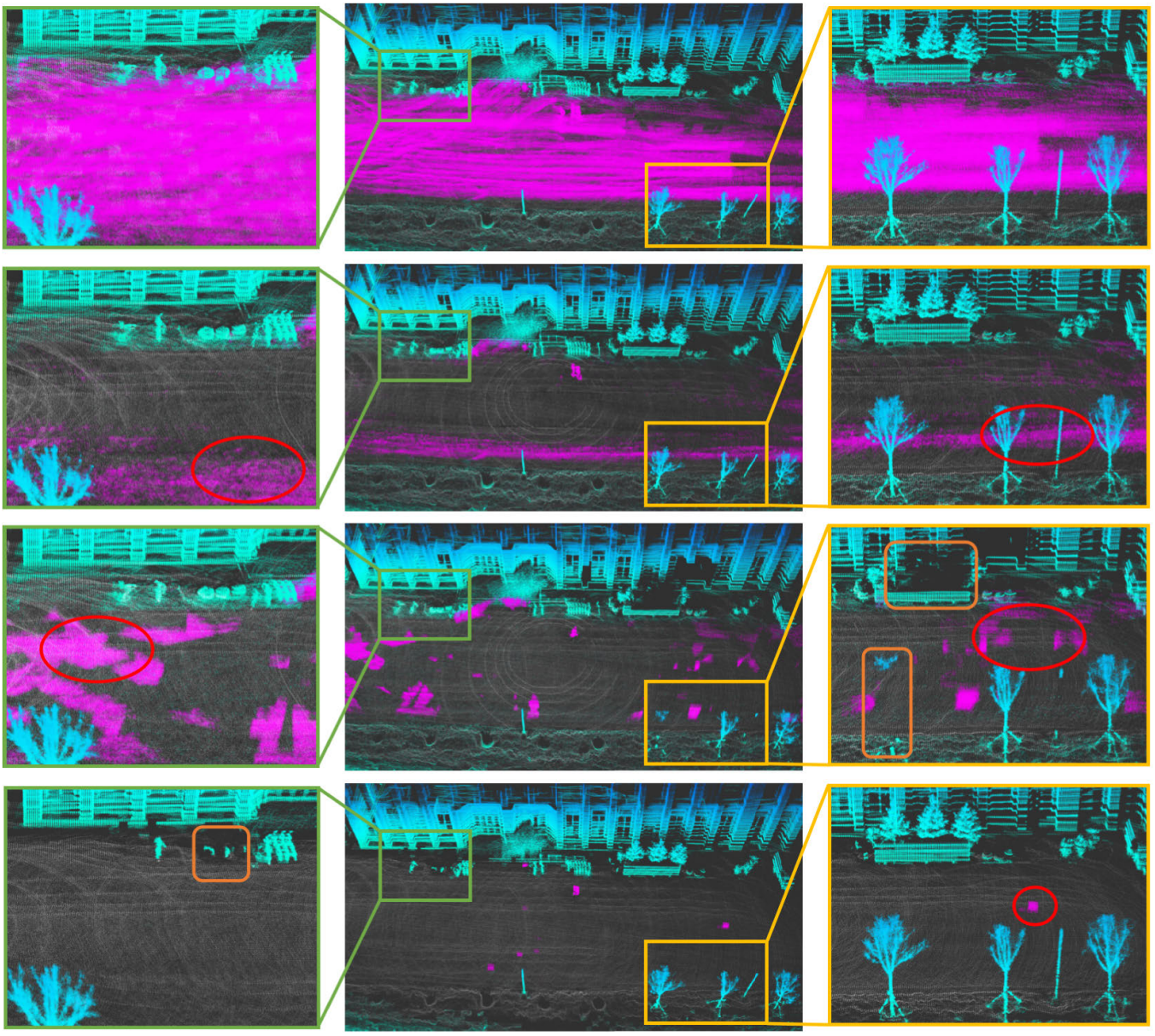


Fig. 11. Dynamic traces removal results in challenging scenarios. The partial enlarged images are displayed on the sides. Top to bottom: original map, Removert [12], ERASOR [9], and Ours. The purple points in the picture is traces left by dynamic objects, and the area with serious error removal is in orange boxes.

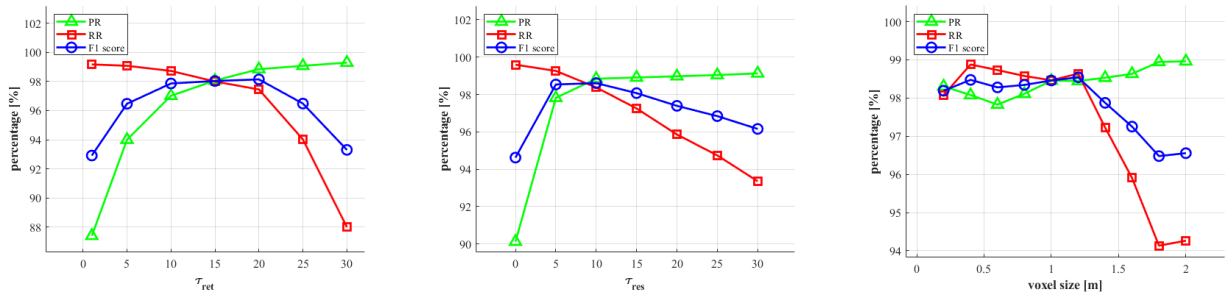
This also reflects the limitations of our method: our method only considers the earliest and latest observation times and ignores the processes between them.

### *E. Comparison in Challenging Scenarios*

To further validate the algorithm's robustness, we conducted experiments on dynamic traces removal in challenging scenarios with high pedestrian density. We still compared our method with the open-source algorithms ERASOR and Removert, and the comparative results are shown in Fig. 11 and Table II. The selected scene is the road in front of the school cafeteria, and the data recording time coincides with the student's 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 the topmost image, the middle area of the image is almost filled with dynamic points.

The effectiveness of dynamic traces removal of Removert in such a crowded scene is limited. Although Removert removes most of the dynamic traces, the dynamic traces on the roadside still cannot be removed, such as the purple points in the second image. The results of ERASOR are shown in the third image, 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 traces, such as the wall and tree trunk highlighted in the orange box in the image.

By comparing the observation time difference between the voxels and the ground, our method achieved a more significant dynamic traces removal effect. Although our algorithm achieves best performance, if we check the result carefully, we can see that our algorithm also exhibits same instances of erroneous removal, such as the bicycle highlighted in the orange box in the zoomed-in image on the left. This is because




(a) (b) (c)

Fig. 12. Impact of the threshold on dynamic removal performance. (a) $\tau_{ret}$. (b) $\tau_{res}$. (c) Voxel size.

the bicycle target is small in size and heavily obstructed in the foreground, making it difficult to be observed accurately. This also results in the lowest PR of our algorithm in this scenario in Table II. Nevertheless, crucial details such as trees and buildings have been successfully preserved in the image.

### F. Threshold Analysis

The impact of the threshold $\tau_{ret}$, $\tau_{res}$ and voxel size on dynamic removal performance is examined on all datasets. The impact of $\tau_{ret}$ is shown in Fig. 12(a). At the beginning, as $\tau_{ret}$ increases, PR is significantly improved but RR does not change significantly. Thereafter, as $\tau_{ret}$ continues to increase, RR will decrease rapidly and PR will be difficult to increase. Similarly, as shown in Fig. 12(b), as $\tau_{res}$ increases, PR will first increase rapidly and then remain stable, while RR will first remain stable and then decrease rapidly.

When $\tau_{ret}$ is set very small, the object can only be considered static when the observation time between the object and the ground is almost exactly the same. However, due to the existence of LiDAR noise, most objects, especially some thin objects such as poles and tree trunks, are difficult to be observed exactly at the same time as the ground. In this case, these objects are easily regarded as dynamic objects, resulting in a low PR. On the contrary, when $\tau_{ret}$ is set very large, the object can only be considered dynamic if the observation time difference between the object and the ground is large. When there are a large number of dynamic objects, due to the influence of occlusion, the observation time difference between some dynamic objects and the ground will not be particularly large. In this case, these objects will not be considered as dynamic objects, resulting in a lower RR.

Similarly, when $\tau_{res}$ is set to a small value, the object will be restored to static only when the total number of observations between the object and the ground is almost the same. In this case, the static restoration module is difficult to play a role, resulting in a lower PR. On the contrary, when $\tau_{res}$ is set to a large value, some dynamic objects, such as closely following vehicles and crowded pedestrians, can also meet the conditions. These dynamic objects will be restored to static, resulting in a lower RR.

Fig. 12(c) shows the impact of voxel size. As the voxel size increases, the metrics of dynamic removal first remains stable and then decreases rapidly. This is because when the voxel size is set larger (such as greater than 1.5 m), it is easy to have voxels that contain both static and dynamic objects. These voxels will be judged as static voxels, resulting in dynamic objects in the voxels cannot be removed, ultimately leading to a reduction in RR.

TABLE III
RUNTIME PER ITERATION OF SEMANTICKITTI DATASET

| Method | Runtime/iteration [ms] |
|---|---|
| Removert [12] | 73.4 |
| ERASOR [9] | 72.6 |
| Nonparametric [14] | 59.9 |
| Ours | **25.9** |

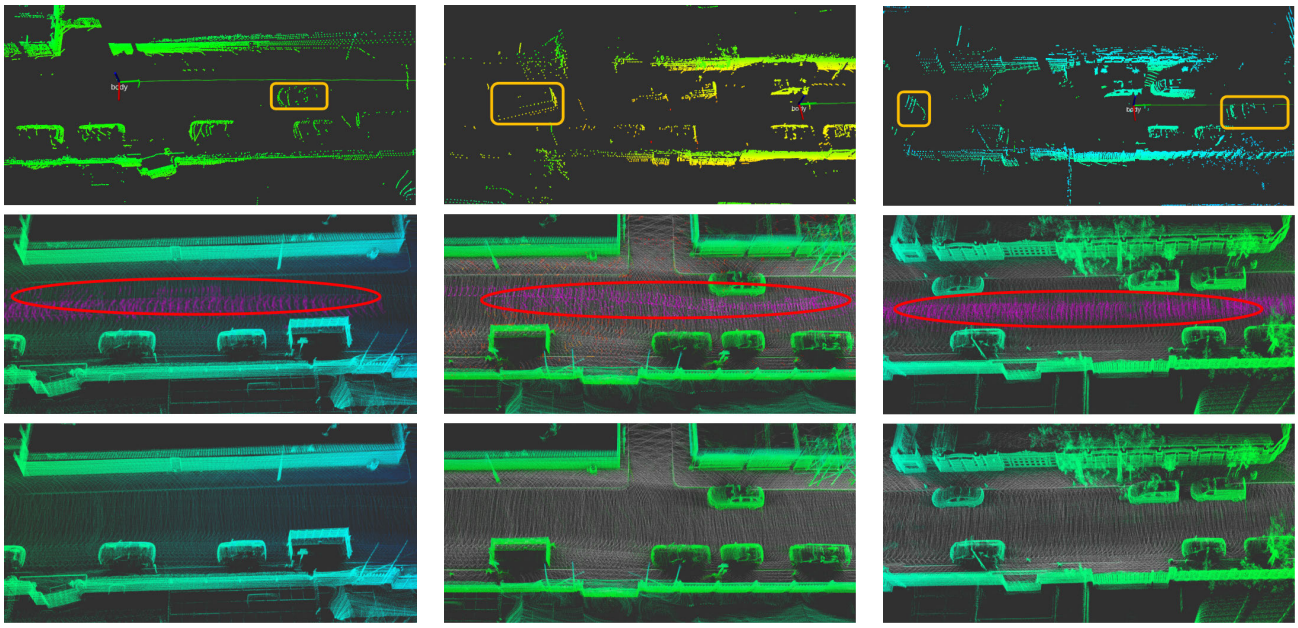
### G. Algorithm Speed

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 III.

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 Table IV.

### H. Ablation Study

The results of the ablation experiment are shown at the bottom of Tables I and II. The w/o up and w/o down represent the result without upward retrieval and downward retrieval respectively. The results show that after removing downward retrieval or downward retrieval, the quantitative results of the algorithm have declined. However, the magnitude of the decline is different in different scenarios, since downward retrieval and downward retrieval are for different dynamic objects. As described in Section III, downward retrieval and upward retrieval in our method are used to remove *suddenly appear* and *suddenly disappear* dynamic voxels, respectively. For the purpose of ablation research, we selected two extreme scenarios: one where a dynamic object is following the sensor from the front, and another one where a dynamic object is following from behind. As shown in Fig. 13.



(a) Dynamic object behind sensor (b) Dynamic object in front of sensor (c) Dynamic object on both sides of sensor

Fig. 13. Ablation studies. The coordinate axis represents the position of the sensor, and the solid green line represents the movement trajectory of the sensor. The points within the yellow rectangle in the top image are the moving car surrounding the sensor, while the purple points inside the red circle in the middle image represent traces left by the moving car. (a) Situation where there is a car following behind the sensor. In this case, only upward retrieval cannot remove the car. (b) Case where there is a following vehicle in front of the sensor. In this case, only downward retrieval cannot remove the car. (c) Case where there is following vehicles on both sides of the sensor. In this case, both downward retrieval and upward retrieval must be used simultaneously to eliminate all dynamic traces.

TABLE IV
EACH MODULE'S RUNTIME OF OUR METHOD

| Module | Runtime/iteration [ms] |
|---|---|
| Ground Segmentation | 13.1 |
| Map Management | 7.3 |
| Dynamic Removal | 5.5 |
| Total | 25.9 |

In Fig. 13(a), there is a car following behind the sensor (in the yellow box at the top). Due to the occlusion of the following car, the voxel where the vehicle is located and the ground it is on always disappear from the field of view at the same time. The voxel does not meet the definition of *suddenly disappear*. The traces left by this vehicle cannot be removed if only implementing upward retrieval (in the red circle of the middle image). However, the voxel fits the definition of *suddenly appear* and can be removed by downward retrieval (image below).

Correspondingly, the vehicle in Fig. 13(b) is in front of the sensor (in the uppermost yellow box). The voxel where the vehicle is located and the ground where it is located are always observed at the same time due to the occlusion of the car, which does not meet the definition of *suddenly appear*. The dynamic traces cannot be removed if only implementing downward retrieval (middle image). However, the voxel fits the definition of *suddenly disappear* and can be removed by upward retrieval (image below).

In practical scenarios, it is common to have cars both in front and behind the sensor, as shown in Fig. 13(c). In this case, the vehicles in front of the sensor can only be removed by upward retrieval, while the vehicles behind can only be removed by downward retrieval. Without either of them, it is not possible to complete the task of removing dynamic traces (in the red circle of the middle image). Therefore, both downward retrieval and upward retrieval must be implemented simultaneously to remove all dynamic traces (image below).

## V. CONCLUSION

In this article, we propose a new method for online dynamic traces removal for ground vehicles. Our method identifies dynamic traces by comparing the observation time difference between voxels and the ground. Our method achieves excellent dynamic removal performance by only using the map at the current moment. But since our method relies heavily on the comparison with the ground, it will perform poorly in those cases where the ground cannot be detected. In future work, we plan to extend our method to SLAM in highly dynamic scenes, while reducing dependence on ground segmentation to improve the generalization of the proposed method.

## REFERENCES

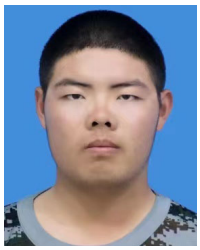
[1] J. Li et al., "WHU-helmet: A helmet-based multi-sensor SLAM dataset for the evaluation of real-time 3D mapping in large-scale GNSS-denied environments," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, May 2023, Art. no. 5702016.

[2] S. Karam, V. Lehtola, and G. Vosselman, "Simple loop closing for continuous 6DoF LiDAR&IMU graph SLAM with planar features for indoor environments," *ISPRS J. Photogramm. Remote Sens.*, vol. 181, pp. 413–426, Nov. 2021.

[3] J. Shao et al., "Single scanner BLS system for forest plot mapping," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 2, pp. 1675–1685, Feb. 2021.

[4] R. Ravi, Y. Lin, M. Elbahnasawy, T. Shamseldin, and A. Habib, "Bias impact analysis and calibration of terrestrial mobile LiDAR system with several spinning multibeam laser scanners," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5261–5275, Sep. 2018.

[5] P. Zhou, X. Guo, X. Pei, and C. Chen, "T-LOAM: Truncated least squares LiDAR-only odometry and mapping in real time," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, Jun. 2022, Art. no. 5701013.
[6] P. Jende, F. Nex, M. Gerke, and G. Vosselman, "A fully automatic approach to register mobile mapping and airborne imagery to support the correction of platform trajectories in GNSS-denied urban areas," *ISPRS J. Photogramm. Remote Sens.*, vol. 141, pp. 86–99, Jul. 2018.
[7] S. Pagad, D. Agarwal, S. Narayanan, K. Rangan, H. Kim, and G. Yalla, "Robust method for removing dynamic objects from point clouds," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 10765–10771.
[8] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3D map maintenance in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 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 Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2272–2279, Apr. 2021.
[10] L. Lin, W. Zhang, M. Cheng, C. Wen, and C. Wang, "Planar primitive group-based point cloud registration for autonomous vehicle localization in underground parking lots," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022.
[11] J. Schauer and A. Nüchter, "The peopleremover—Removing dynamic objects from 3-D point cloud data by traversing a voxel occupancy grid," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1679–1686, Jul. 2018.
[12] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10758–10765.
[13] C. Qian, Z. Xiang, Z. Wu, and H. Sun, "RF-LIO: Removal-first tightly-coupled LiDAR inertial odometry in high dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 4421–4428.
[14] J. Park, Y. Cho, and Y.-S. Shin, "Nonparametric background model-based LiDAR SLAM in highly dynamic urban environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 24190–24205, Dec. 2022.
[15] T. Fan, B. Shen, H. Chen, W. Zhang, and J. Pan, "DynamicFilter: An online dynamic objects removal framework for highly dynamic environments," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 7988–7994.
[16] R. Wu, C. Pang, X. Wu, and Z. Fang, "Observation time difference: An online dynamic objects removal method for ground vehicles," 2024, *arXiv:2406.15774*.
[17] W. Wen et al., "UrbanLoco: A full sensor suite dataset for mapping and localization in urban scenes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 2310–2316.
[18] J. Behley et al., "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9297–9307.
[19] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.
[20] M. Arora, L. Wiesmann, X. Chen, and C. Stachniss, "Static map generation from 3D LiDAR point clouds exploiting ground segmentation," *Robot. Auto. Syst.*, vol. 159, Jan. 2023, Art. no. 104287.
[21] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson surface reconstruction for LiDAR odometry and mapping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 5624–5630.
[22] T. Kühner and J. Kümmerle, "Large-scale volumetric scene reconstruction using LiDAR," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 6261–6267.
[23] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-blox: A scalable and consistent TSDF-based dense mapping approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 995–1002.
[24] H. Lim et al., "ERASOR2: Instance-aware robust 3D mapping of the static world in dynamic scenes," in *Proc. Robot., Sci. Syst.*, Jul. 2023, Art. no. 067. [Online]. Available: https://roboticsconference.org/2023/program/papers/067/
[25] B. Xiang, J. Tu, J. Yao, and L. Li, "A novel octree-based 3-D fully convolutional neural network for point cloud classification in road environment," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 10, pp. 7799–7818, Oct. 2019.
[26] K. Zhang et al., "A dual attention neural network for airborne LiDAR point cloud semantic segmentation," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, Aug. 2022, Art. no. 5704617.
[27] X. Chen et al., "Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6529–6536, Oct. 2021.
[28] J. Sun et al., "Efficient spatial–temporal information fusion for LiDAR-based 3D moving object segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 11456–11463.
[29] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3D laser scans for online operation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 163–169.
[30] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 5067–5073.
[31] 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 Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4861–4868, Apr. 2022.
[32] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[33] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Proc. Robot. Sci. Syst.*, vol. 2018, Jun. 2018, p. 59.
[34] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.



**Rongguang Wu** received the B.S. degree in robot science and engineering from Northeastern University, Shenyang, China, in 2022, where he is currently pursuing the M.S. degree.

His research interests include dynamic point cloud removal and LiDAR mapping on the back-end.

**Zheng Fang** (Member, IEEE) received the B.S. degree in automation and the Ph.D. degree in pattern recognition and intelligent systems from Northeastern University, Shenyang, China, in 2002 and 2006, respectively.

He was a Post-Doctoral Research Fellow with Carnegie Mellon University, Pittsburgh, PA, USA, from 2013 to 2015. He is currently a Professor with the Faculty of Robot Science and Engineering, Northeastern University. His research interests include visual/laser SLAM and the perception and autonomous navigation of various mobile robots.

**Chenglin Pang** received the B.S. degree in automation from Shandong University of Science and Technology, Qingdao, China, in 2016, and the M.S. degree in control engineering from Shandong University, Jinan, China, in 2020. He is currently pursuing the Ph.D. degree with the School of Robotics Science and Engineering, Northeastern University, Shenyang, China.

His research interests include visual-inertial odometry, multisensor fusion SLAM, and autonomous driving.

**Xuankang Wu** received the B.S. degree in robotics science from Northeastern University, Shenyang, China, in 2022, where he is currently pursuing the Ph.D. degree with the School of Robotics Science and Engineering.

His research interests include multisensor fusion SLAM and multimodal autonomous quadrotor.