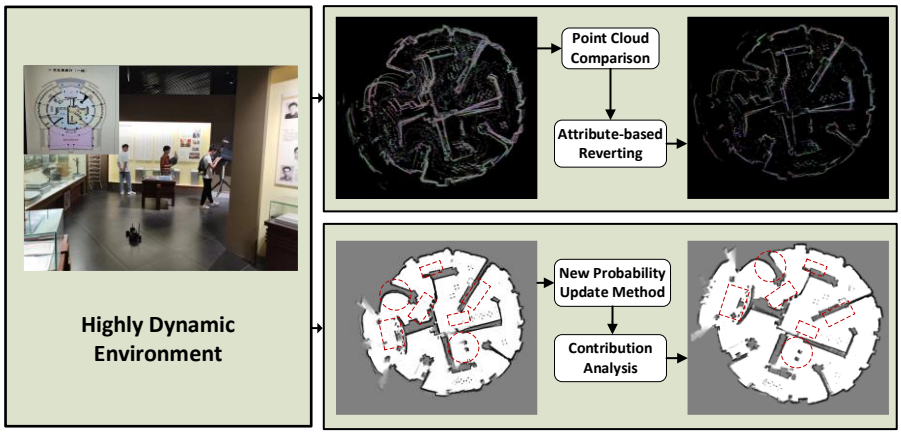
This article has been accepted for publication in IEEE Sensors Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2024.3414661

# DY-LIO：Tightly-coupled LiDAR-Inertial Odometry for dynamic environments

Jingliang Zou, Huangsong Chen, Liang Shao, Haoran Bao, Hesheng Tang, Jiawei Xiang *Member, IEEE*, and Jun Liu

***Abstract*—A limitation of the Simultaneous Localization and Mapping (SLAM) is the lack of consideration of dynamic objects in the environment, resulting in degradation of map quality and positioning accuracy. This paper introduces a 2D tightly-coupled LiDAR-Inertial Odometry and Mapping framework DY-LIO to remove dynamic objects for a better SLAM. The main innovations are based on the process of LiDAR point cloud and occupancy grid map: 1) W.r.t. LiDAR point cloud processing, on the basis of de-Skewing and aligning the point cloud through factor-graph-optimized IMU pre-integration, the alignment errors are utilized to identify dynamic objects. Besides, different property of static and dynamic cloud point is introduced to reduce the mis-identification error. 2) In terms of occupancy grid map processing, a novel probabilistic update method is presented to enhance the filterability of dynamic grid and accelerate the filtering process. Based on these novel strategies, DY-LIO can guarantee accurate localization and clean map estimation in dynamic environments. Experimental results on realistic high dynamic datasets demonstrate that DY-LIO greatly outperforms the current mainstream 2D-SLAM algorithm (Cartographer) in terms of the mapping effect and localization accuracy, with high efficiency in dynamic environments.**




***Index Terms*—Dynamic obstacles, point cloud, scene understanding, scan matching, SLAM**

## I. Introduction

SIMULTANEOUS localization and mapping (SLAM) techniques have played a crucial role in the field of intelligent vehicles [1]. In SLAM system design, LiDAR is widely adopted due to its advantages such as extended range, high accuracy, wide viewing angle, and insensitivity to luminance [2]-[5]. However, motion skewing and the difficulties of establishing the relationship between sparse point clouds make the development of a high-quality LiDAR SLAM system difficult. Inertial measurement unit (IMU) can compensate these shortcomings with its instantaneous motion measurement. Hence, LiDAR and IMU are often combined to build SLAM systems to enhance pose estimation and mapping performance [6]-[12].

Currently, most of the existing LiDAR-Inertial SLAM techniques assumes observed objects are static. However, as described in [13]-[14], the presence of numerous dynamic objects during the vehicle's localization and mapping causes variations in the spatial structure, resulting in a poor localization accuracy and a low-quality map. Therefore, it is necessary to enhance the performance of LiDAR-Inertial SLAM systems in dynamic environments.

In this paper, we focus on building 2D LiDAR-Inertial SLAM which is capable of online removal of dynamic objects for highly dynamic scenes. There are two main types of methods for processing dynamic objects, one based on LiDAR point clouds and the other on occupancy grid maps.

For the processing of LiDAR point clouds, the aim is to differentiate the point clouds of static and dynamic objects. Some Learning-based approaches [15]-[17] can detect point clouds of dynamic objects from a single scan, but they rely on many labeled datasets and may fail once they meet an unlabeled object. The dependence on specific models may detect a static object as a dynamic one, for instance, a long-time parked car on the roadside should be regarded as a static obstacle. Visibility-based approaches do not rely on datasets and semantic labels, and are computationally efficient. Kim et.al [18] transforms point cloud data into distance images to dynamic objects. Based on Kim's work, Qian et.al [19] uses IMU pre-integration to design the initial resolution of the distance image, for a better dynamic object detection. According to a specific field of view (FOV), Yoon et.al [20] reprojects the distance measurements of the 3D point cloud to the image plane for obtaining visibility differences between multiple scans, leading to the detection

This work was supported in part by the National Key R&D Program of China under Grant 2022YFE0117100, and in part by Basic Industrial Science and Technology Project of Wenzhou under Grant G20210006. *(Jingliang Zou and Huangsong Chen contributed equally to this work).(Corresponding authors: Liang Shao.)*

Jingliang Zou, Huangsong Chen, Liang Shao, Haoran Bao, Hesheng Tang and Jiawei Xiang are with the College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou 325035, China (e-mail: shaoliangtjauto@163.com).

Jun Liu is with Shanghai X-AI Auto Technology Co., Ltd.

This article has been accepted for publication in IEEE Sensors Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2024.3414661

of dynamic object. Based on visibility differences, Lázaro et.al [21] merges the point clouds for a better dynamic object identification. However, Visibility-based approaches suffer from incidence angle ambiguity [22], and occlusion [23] issues, which may cause that many static objects are regarded as dynamic objects. Occupancy map-based approaches also do not depend on the datasets, but use the simple priori knowledge- when the LiDAR beam hits the grid, the grid along the way must be empty. Such methods require high computational demand to ray-trace in real time and rely heavily on very accurate positional information. Hornung et.al [24] optimized to reduce the calculation burden. Schauer et.al [25] set a safe distance and cluster the dynamic point clouds to better distinguish the static and dynamic objects. However, map-based approaches are still difficult to handle huge amount of data online.

There is huge number of research to identify dynamic objects based on occupancy grid maps. The well-known Cartographer [6] updates the occupied probability of grid to obtain a dynamic object filtering effect. Danescu et.al [26] proposed a particle-based occupancy grid map to describe its dynamic state. Nuss et.al [27] defined the dynamic grid mapping as a stochastic finite set problem for a better detection of dynamic objects. However, particle-based approach requires large computational power when dealing with many dynamic objects. Recently, several learning-based approaches, such as convolutional neural networks, have used grid maps as input data to detect objects [28], [29], correctly estimate occupancy grid maps [30], and predict future occupancy [31]. Ronneberger et.al [32] uses a combination of convolutional and recursive network layers to estimate filtered occupancy and velocity estimates for each grid. These learning-based approaches demonstrate significant improvements over the particle-based approach with limitation in specific environments.

This paper proposed a tightly coupled LiDAR-Inertial 2D SLAM framework (DY-LIO) for highly dynamic environments, which demonstrates online processing capabilities, maintains a low resource usage, and is model-independent. DY-LIO minimizes the impact of dynamic objects by processing point clouds and maps separately. In front end, the point cloud processing module begins by de-skewing and aligning the point cloud through factor-graph-optimized IMU pre-integration. Subsequently, the alignment errors (point-to-point and point-to-line) are utilized to obtain a preliminary dynamic point set $\tilde{C}_D$ and a static point set $\tilde{C}_S$. The $\tilde{C}_D$ contains a small number of static points due to the visibility issues mentioned above. These static points in $\tilde{C}_D$ are then defined as static points with a new set $\bar{C}_S$ by utilizing different property of static and dynamic cloud point. Then, set $\tilde{C}_S$ and $\bar{C}_S$ are merged together to form the optimized static set $C_S$; In the map processing module, a novel probabilistic update method is designed to maximize the filtering effect of occupied grid maps on dynamic grids. The method eliminates dynamic grids by comparing the number of the LiDAR beams hitting and passing through the grid. At the back end, we use the same process as Cartographer [6] for global optimization by constructing a pose graph and for loop detection with branch delimitation.

We evaluate DY-LIO by qualitative and quantitative analysis on self-collected dynamic datasets. Experimental results demonstrate that DY-LIO greatly outperforms the current mainstream 2D-SLAM algorithm (Cartographer) in terms of the mapping effect and localization accuracy, with high efficiency in dynamic environments.

The entire pipeline is demonstrated in Fig. 1 and the main contributions are summarized as follows:

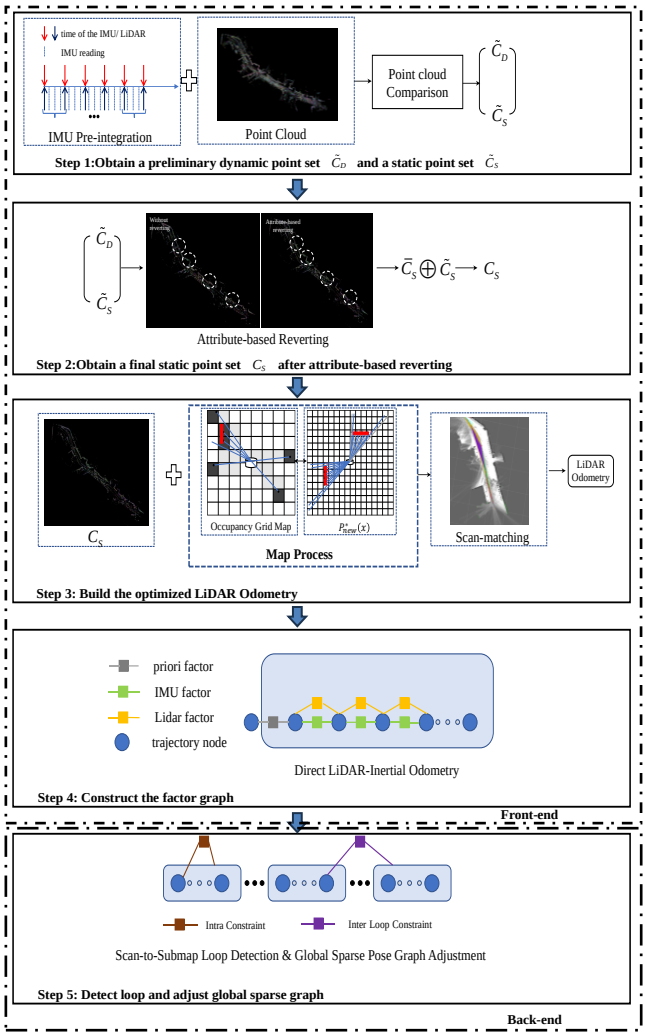



**Fig. 1.** Framework of DY-LIO. $\tilde{C}_D$ and $\tilde{C}_S$ denote the initially obtained dynamic point set and static point set, respectively. $\bar{C}_S$ and $C_S$ denote the repaired and final static point sets, respectively. $P^*_{new}(x)$ denotes probability update of occupancy grid map. $\oplus$ denotes the union.

- We propose the Tightly-coupled LiDAR-Inertial Odometry that efficiently and accurately eliminate dynamic objects by point clouds and maps processing. To our best knowledge, it is the first work that considers both point clouds and maps processing for SLAM in dynamic environments.
- Different property of static and dynamic cloud point is introduced to reduce the miss-identification error, which can mitigate the visibility issues caused by the visibility-based removal.

- We proposed a novel probabilistic update method of occupancy grid map. This method enhances the filterability of dynamic grid, protects the static grid and accelerates the filtering process.

The remainder of this paper is organized as follows: Section II mainly introduces the proposed dynamic objects removed method, including point cloud processing and grid map processing. Section III verifies the effectiveness of the proposed method through experiments, and the work of this paper is summarized in Section IV.

## II. Methodology

Since our work is mainly focused on the front-end, in this section, we describe the methodological part of our front-end in detail, which consists of three main parts: point cloud processing, map processing, and local factor map construction. The back-end consists of constructing pose graph and loop detection, which are the same as the parts in Cartographer [6] and we will not describe them in this paper.

### A. Point Cloud Process

In the point processing stage, we focus on the removal of dynamic points from the consecutive 2D LiDAR scans. The processes are explained as follows:

- IMU pre-integration: Alignment of consecutive scan with IMU pre-integration.
- Point cloud Comparison: Identification of dynamic objects with alignment differences (point-to-point and point-to-line).
- Attribute-based reverting: Repair of static points based on different property of static and dynamic cloud point.

We spend the rest of this section to discuss each step in more detail and highlight its contribution to the pipeline.

*1) IMU Pre-integration:* Forster et al. [33] proposed the well-known IMU pre-integration. It can reduce the computational burden of pose estimation by establishing a correlation between the relative motion and original IMU data. The pre-integration process is described as follows.

$$\Delta\hat{R}_{bj}^{bi} \approx R_{bi}^{wT} R_{bj}^{w} EXP(\delta\phi_{bj}^{bi})$$
$$\Delta\hat{v}_{bj}^{bi} \approx R_{bi}^{wT}(v_{bj}^{w} - v_{bi}^{w} - g^{w}\Delta t_{ij}) + \delta v_{bj}^{bi}$$
$$\Delta\hat{p}_{bj}^{bi} \approx R_{bi}^{wT}(p_{bj}^{w} - p_{bi}^{w} - v_{bi}^{w}\Delta t_{ij} - \frac{1}{2} g^{w}\Delta t_{ij}^{2}) + \delta p_{bj}^{bi} \quad (1)$$

where $\delta\phi_{bj}^{bi}$, $\delta v_{bj}^{bi}$ and $\delta p_{bj}^{bi}$ are Gaussian noise of the pre-integral measurement.

*2) Point Cloud Comparison:* We identify dynamic objects by comparing different point clouds. This can work because the description of the environments by point clouds will be different due to the moving objects.

With IMU pre-integration, we can align two frames of point clouds at any moment. Therefore, we can identify differences by common error metrics in point cloud alignment, namely the point-to-point (PtP) and point-to-line (PtL) metrics. The flowchart is shown in Fig. 2.

For the $i^{th}$ frame, the point cloud ${}^{B}P_i$ is located with local sensor frame $F_B$. By using IMU pre-integration we can obtain the transition matrix ${}_{W}^{B}T_i$ between local sensor frame $F_B$ and global frame $F_W$. Based on ${}_{W}^{B}T_i$, all points in one frame are converted to the global coordinate system $F_W$ then a motion-compensated point cloud is created. Subsequently, given any query point $q_0$ from one frame , if the two close-enough points $p_1$, $p_2$ from other frame can be found, their normal vector $n_p$ is calculated to create a PtL metric $|n_p \cdot (q_0 - p_1)|$, otherwise, a PtP metric $\|p_0 - q_0\|_2$. According to the obtained PtL and PtP metric, we are able to detect dynamic point set $\tilde{C}_D$ and static points set $\tilde{C}_S$.

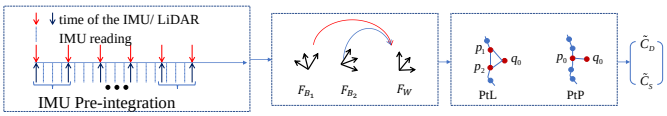



**Fig. 2.** Pipeline of point cloud comparison. $F_B$ denotes local sensor frame. $F_W$ denotes global frame.

*3) Attribute-based Reverting:* Occlusion problems with dynamic objects will lead to some static points being mis-detected as dynamic ones, indicating that the accuracy rate of classification between static and dynamic points based on point cloud comparison is not enough. Yoon et.al [20] performed restoration via free-space queries, and Kim et.al [18] repeatedly performed Visibility-based removal with dynamic submap as input, to recover static points from dynamic submap. However, both recovery methods still suffer from visibility issues. Hence, we introduce attribute-based reverting to improve such situation. The basic intuition is that, static points in dynamic set $\tilde{C}_D$ have the same property as the ones in static set $\tilde{C}_S$. Based on this knowledge, we detect the static points in dynamic point set $\tilde{C}_D$ by utilizing the distance and reflection intensity property of LiDAR point cloud. The flowchart is shown in Fig. 3.

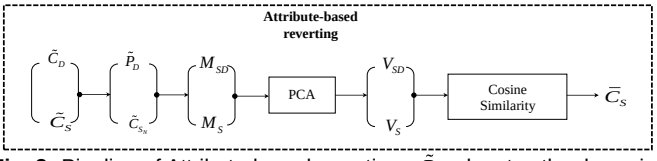



**Fig. 3.** Pipeline of Attribute-based reverting. $\tilde{P}_D$ denotes the dynamic point, and $\tilde{C}_{S_N}$ denotes the nearest static point set for $\tilde{P}_D$. $M_{SD}$ denotes the matrix of dynamic point merged with the static point set. $M_S$ denotes the matrix of static point set. $v_{SD}$ and $v_S$ denote the eigenvectors of $M_{SD}$ and $M_S$, respectively.

In the dynamic point set $\tilde{C}_D$ obtained from point cloud comparison, we judge whether each dynamic point in $\tilde{C}_D$ can be recovered as static point or not. First, for any point $\tilde{P}_D$ in $\tilde{C}_D$, we collect a small static set $\tilde{C}_{S_N}$ from $\tilde{C}_S$ that are close to $\tilde{P}_D$. Then, we construct a matrix $M_S$ containing coordinates and reflection intensity information of $\tilde{C}_{S_N}$. Moreover, we compute the eigenvectors $V_S$ of $M_S$ by principal component analysis (PCA) [34]. Similarly, we can also compute the matrix $M_{SD}$ of the concatenated set of $\tilde{P}_D$ and $\tilde{C}_{S_N}$, as well as the eigenvectors $V_{SD}$ of $M_{SD}$. Finally, we compare the similarity of the two vectors ($V_S$ and $V_{SD}$) by computing their cosine similarity. If the similarity is high enough (cosine similarity > 0.9), the dynamic point $\tilde{P}_D$ is recovered as a static point. After all points in $\tilde{C}_D$ are detected with the same process, we obtain the recovered static

point set $\bar{C}_S$. By merging it with $\tilde{C}_S$, we have the final static point set $C_S$. During this process, the recovery from dynamic points to static ones is not affected by the visibility issues. Meanwhile, since the recovery process introduces the characterization of reflection intensity, it is able to avoid the situation that dynamic points are incorrectly recovered as static points when the dynamic object is very close to the static object. The qualitative results are shown in Fig. 4. As shown in the enlarged view of A-D, after the attribute-based reverting process, the right-side point cloud retains more static features than the left-side point cloud, such as counters and chair legs, which can be easily deleted as dynamic objects by mistake.

After Attribute-based reverting, we obtain a static point set $C_S$, and we will use it in the subsequent scan matching.

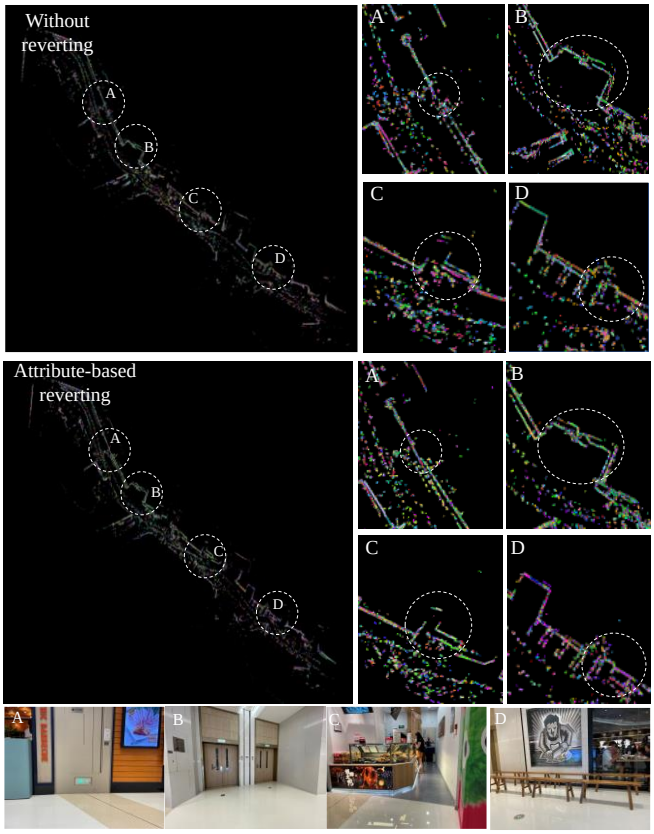



**Fig. 4.** The qualitative result of attribute-based reverting. Top: result without reverting; Middle: result with attribute-based reverting. Bottom: partially enlarged camera images(A~D).

### *B. Map Process*

Due to the online processing of point cloud, we can only utilize short data sequences, which inevitably results in the presence of residual dynamic points. These residual dynamic points are incorporated into the occupancy grid map, forming dynamic grids. Our map processing objective is to eliminate this dynamic grids.

As depicted in Fig. 5, the occupancy grid map leverages the information from long sequences to filter out the dynamic grid. Referring to Cartographer [6], when a frame scan is inserted into the probabilistic grid map, a probability value $p_{hit}$ and $p_{miss}$ is assigned to the hit and miss grids. if the grid has been observed, we update the hit and miss probabilities as

$$odds_{(p_{state})} = \frac{p_{state}}{1 - p_{state}} \tag{2}$$

$$P_{new}(x) = clmap(odds^{-1}(odds(P_{old}(x)) \cdot odds(p_{state}))) \tag{3}$$

where $p_{state}$ respect to $p_{hit}$ or $p_{miss}$, and $clmap()$ limits the result to the range zero to one.

Note that (3) is updated only once for the grid probability in a frame of laser scan. This means that the grid is updated the first time it is hit by or passes through a laser beam, and is not updated for subsequent hits or passes. This prevents the map from being updated repeatedly and reduces the amount of computation when the number of dynamic objects in the environment is small. But its processing speed is notably inadequate in highly dynamic scenes. Therefore, we propose a grid probability updating method that speeds up the filtering effect and is not computationally intensive. Distinguished from the original update method, our method considers all laser beams in a frame of laser scan.

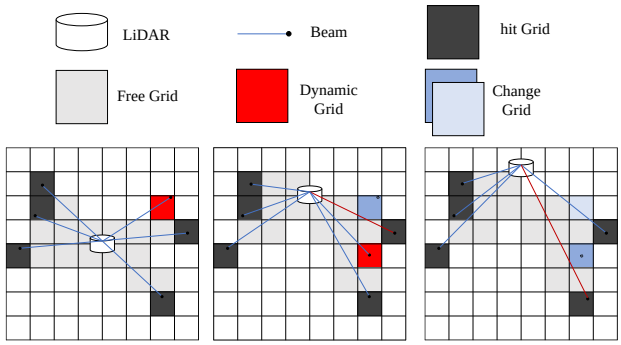



**Fig. 5.** The filtering process of the occupancy grid map for dynamic objects. From left to right, the dynamic grid is gradually updated to a free grid as LiDAR information is inserted.

The method is Algorithm 1, and it will be described in detail in the following sections.

*1) New Probability Update Method:* As depicted in Fig. 6, compared to static objects, the movement of dynamic objects often brings more beams passing through one grid in one frame laser scan. Based on this intuition, we utilize it to speed up the filtering effect.

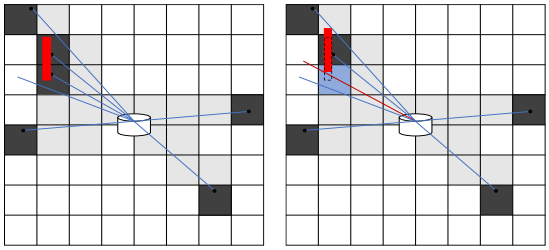


**Fig. 6.** Dynamic objects move in the time of a point cloud in one frame, causing subsequent beams to pass through the grid.

We only consider the hit grids because it is very likely to be affected by the dynamic object. In Fig. 7(b), we examine each hit grid and calculate two key factors: the numbers of hit by different beams in a frame of the point cloud is denoted as $n_{hits}$, and the number of times traversed, is represented by $n_{misses}$. We define the hit coefficient $a_{hits}$ as follows:

$$\alpha_{hits} = \frac{n_{hits}}{n_{hits} + n_{misses}} \tag{4}$$

We add the effect of hit coefficient to the original update strategy. And add weighting coefficients to choose whether to trust more the observations of a frame of laser scans or consecutive laser scans in a sequence. Therefore, we update the final probability of the hit grids to

$$P^*_{New}(x) = \theta\alpha_{hits} + (1-\theta)P_{new}(x) \tag{5}$$

The choice of the weighting factor $\theta$ is determined based on the number of dynamic objects. When the dynamic object is dense, a larger value of $\theta$ can be selected.

**Algorithm 1** New Probability Update Method

Update the grid probability when **a frame of laser scan** comes.
Initialize the contribution of each beam: *contribution* ← 0
Initialize the state of each grid*: firstUpdate* ← *true*
**for** each laser beam **do**
  **for** each grid **do**
    **if** the grid is hit **then**
      **if** *firstUpdate* is true **then**
        $P_{new}(x) = clmap(odds^{-1}(odds(P_{old}(x)) \cdot odds(p_{hit})))$
        $n_{hits}$ ++
        *firstUpdate* ← false
      **else**
        $n_{hits}$ ++
      **end if**
    **end if**
    **if** the grid is passed **then**
      **if** *firstUpdate* is true **then**
        $P_{new}(x) = clmap(odds^{-1}(odds(P_{old}(x)) \cdot odds(p_{miss})))$
        *firstUpdate* ← false
        *contribution* ← *Contribution Analysis ()*
        **if** the *contribution* of the beam is sufficient **then**
          $n_{misses}$ ++
        **end if**
      **else**
        *contribution* ← *Contribution Analysis ()*
        **if** the *contribution* of the beam is sufficient **then**
          $n_{misses}$ ++
        **end if**
      **end if**
    **end if**
  **end for**
**end for**
Now the initial hit grids are obtained.
**for** each hit grid **do**
  calculate hit coefficient $\alpha_{hits}$
  update the hit grid probability again:
  $P^*_{New}(x) = \theta\alpha_{hits} + (1-\theta)P_{new}(x)$
  $P_{old}(x) = P^*_{New}(x)$
**end for**
**return** final hit grids and miss grids.

*2) Contribution Analysis:* Calculating $n_{misses}$ directly will incorrectly treat the static grids as dynamic grids, resulting in the static grids being filtered out. This is because when the beam passes through the grid edge, it can lead to a pseudo-miss phenomenon. As depicted by the red beam in Fig. 7(b), we consider the contribution of this beam to the grid-probability-update as small. If $n_{misses}$ is increased at this point, $\alpha_{hits}$ will decrease, causing incorrect updates of the static grid and leading to unclear and distorted static obstacle edges on the map. Therefore, it is crucial to quantitatively assess the contribution value in real-time. To address this, we propose a simple but effective method which calculates the normal distance from the center of the grid to the beam, as shown in Fig. 7(a). the larger the normal distance is, the closer the beam passes to the edge of the grid, resulting in a smaller contribution value. By regarding the gray beams as invalid, as illustrated in Fig. 7(c), we effectively mitigate the impact of pseudo-misses. The qualitative analysis of these techniques is presented in Fig. 14.

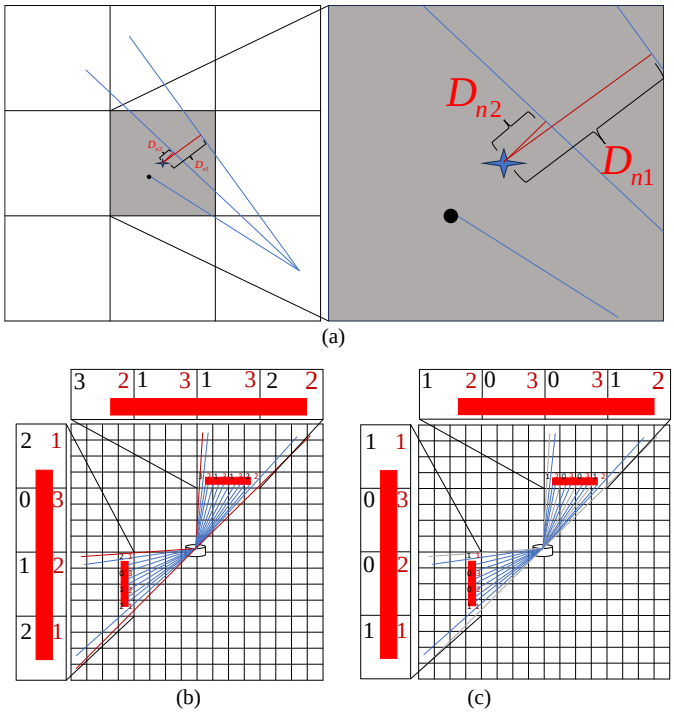



**Fig. 7.** $D_n$ is the normal distance between the center of the grid and the beam. The red and black numbers indicate the number of times the grid was hit and passed by different beams, respectively.

### C. Factor Graph Construction

The LiDAR odometry transforms the pose estimation problem into a least squares optimization problem through scan matching, and solves it using the Gauss-Newton method. We used the method proposed in [6] to optimize the least-squares problem related to the robot pose.

$$\underset{\xi}{argmin}\sum_{k=1}^{K}(1 - M_{smooth}(T_\xi h_k))^2 \tag{6}$$

Where $T_\xi$ is the conversion of scanning points from the LiDAR frame to the submap frame, and $h_k$ is the pose of a series of scanning points.

The LiDAR odometry can be added to the local factor graph as a unit factor. The Jacobian matrix of this factor

regarding the node orientation is an identity matrix. Besides, the associated covariance can be calculated as $Var(\xi) = \sigma^2 \cdot H^{-1}$, in which $H$ can be obtained according to the Taylor expansion term in Eq. (6) and $\sigma$ is related to LiDAR accuracy [35].

We performed dynamic object removal on both point cloud and grid maps to optimize the laser odometry, which can achieve high-quality scan matching and thus yield accurate pose estimation results. Then, we incorporate the optimized laser odometry as a unary factor into the factor graph to effectively correct the biases of the Inertial Measurement Unit (IMU), leading to the accuracy improvement of pre-alignment in the point cloud processing stage and better point clouds identification of dynamic objects.

## III. Experiment

In this section, we would like to analyze the mapping and localization performance of DY-LIO in the dynamic environment, and answer four questions through extensive experiments, namely,

- Can point cloud processing effectively remove point clouds from dynamic objects? Is it real-time?
- Can the map processing perform faster and better in filtering dynamic objects compare with Cartographer [6]?
- How is the resource usage of our framework?
- Why does contribution analysis module play a key role in improving the map integrity?

For answering aforementioned questions, we collected several datasets from diverse scenes based on our platform. Besides, we explored the performance in terms of localization and mapping from both qualitative and quantitative perspectives. Moreover, we conducted a comprehensive quantitative analysis of the point cloud process, scan match time efficiency and global CPU usage. Finally, we performed ablation studies on the specially designed modules.

### A. Experiment Protocols

*1) Implementation:* All modules in our frame were implemented in C++. Our frame was constructed and optimized with GTSAM[36]—a widely used factor graph optimization toolbox. We used the communication mechanism of a popular robot operation system (ROS) to transmit messages among processes. Experiments were conducted on a workstation with a 2.92GHz Intel i5-10400 CPU.

*2) Datasets:* The experimental datasets, called "DY datasets", are available in[1]. They were collected from a variety of scenes, as shown in Fig 8. These scenarios contain a varying number of dynamic objects. An intelligent vehicle, shown in Fig 9, was equipped with 2D LiDAR (RPLIDAR A1) and IMU (mpu6050), with frequencies of 5 Hz and 50 Hz, respectively. DY-1 is a simple scenario, where the vehicle is stationary on the ground to collect data, and dynamic pedestrians are constantly walking or jogging around the vehicle. DY-2 and DY-3 are in a closed indoor environment, where the dynamic objects are randomly walking students, and we control the vehicle to move to collect data. DY-4 is at the entrance and exit of a shopping mall, where the flow of people is very high, and we place the vehicle on the ground to collect data.

[1] https://github.com/Zoujingliang/DYLIO.git

### B. Qualitative Analysis

*1) Point Cloud Process:* By removing the point clouds of dynamic objects and recovering the wrongly removed static points, we can significantly reduce the dynamic points and keep the good static features.

Comparing the cloud maps for four dynamic scenes from DY-1 to DY-4 by Cartographer [6] with the cloud maps obtained by our point cloud processing. The results are shown in Fig 10. It can be seen that the dynamic object point cloud removal is very effective in different scenes, although our point cloud processing algorithm is online and only short sequences of data are used. Besides, the static features are well preserved due to the restoration mechanism. More real-world experiments can be found in https://b23.tv/960zQOc.

*2) Map Process:* By processing the grid map, our algorithm can filter the dynamic grid faster, leading to a "cleaner" match (clean point clouds match clean maps). To verify this, we build a grid map of DY-2 by Cartographer [6] and add our processing of the map to its algorithm for comparison. DY-2 was chosen, because, compared to other scenes, DY-2 has more dynamic objects, more complex scenes, and less glasses in the environment. We observed grid map's difference between Cartographer [6] and DY-LIO every 10 frames, and these frames are from the 90th to 140th with enough dynamic point cloud data. The outcomes of this investigation are presented in Fig. 11. Since our algorithm can start filtering the dynamic grid as soon as a point cloud is inserted into the grid map in one frame, the grids of dynamic objects in region A disappear faster and more complete.

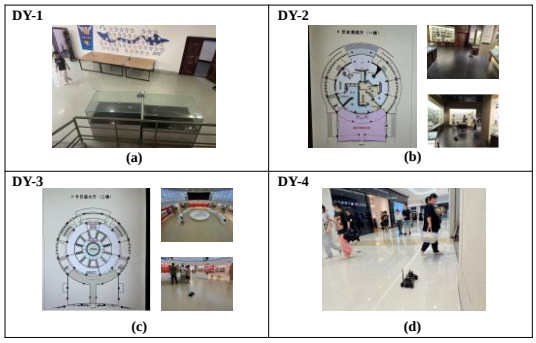



**Fig. 8.** (a) environment of DY-1; (b) environment of DY-2, with diagram on the left; (c) environment of DY-3, with diagram on the left; (d) environment of DY-4.

**Fig. 9.** An intelligent vehicle equipped with 2D LiDAR and IMU.

### C. Quantitative Analysis

To perform a deep quantitative analysis of our frame, we explored its front-end global error and evaluated its time

This article has been accepted for publication in IEEE Sensors Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2024.3414661

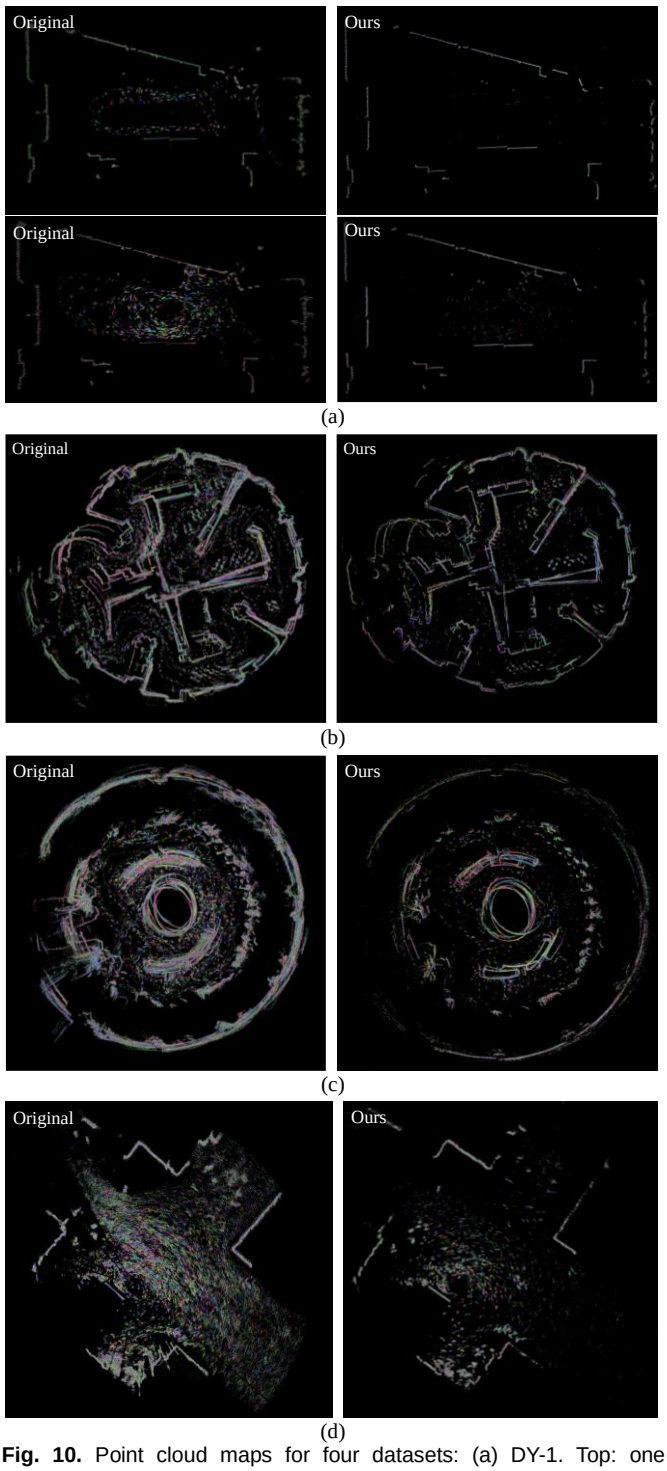



**Fig. 10.** Point cloud maps for four datasets: (a) DY-1. Top: one dynamic object. Bottom: three dynamic objects;(b) DY-2; (c) DY-3; (d)DY-4. The Cartographer [6] results are displayed on the left; our DY-LIO results are displayed on the right.

efficiency and system resource usage. We selected the DY-2, DY-3 and DY-4 datasets. These particular datasets offer several advantages for analysis, with a higher presence of dynamic objects, more intricate environmental settings.

*1) Front-end global error:* Since our algorithm processes dynamic objects in the point cloud and map separately, it can achieve a "clean" matching process and improve the localization accuracy in dynamic scenes. As our experiments were conducted indoors, there was no ground truth for DY-2 dataset. However, according to the description in [37], front-end accuracy can be measured by the global error $\chi^2$. A lower value indicates better front-end scan alignment.

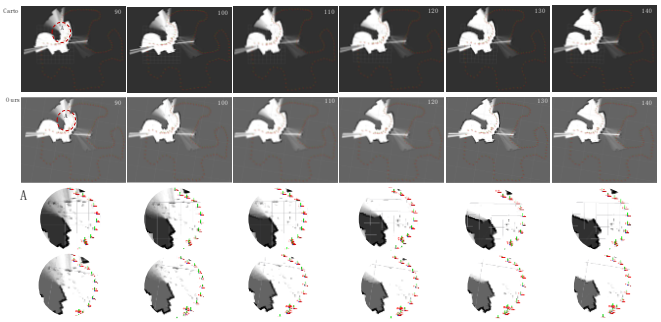


**Fig. 11.** Qualitative analysis of map processing between Cartographer [6] and our DY-LIO.

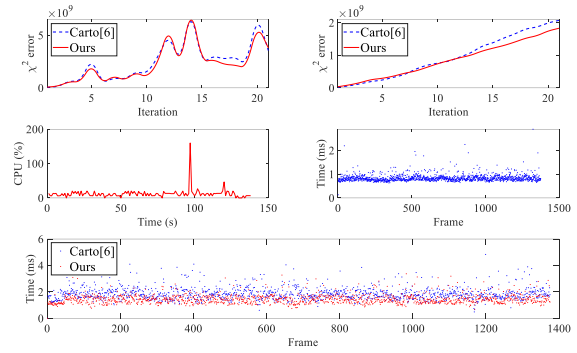
Thus, we compared the $\chi^2$ of Cartographer [6] and our DY-LIO before and after the global optimization. As shown in Fig. 12 and Table I, the $\chi^2$-value of our algorithm outperforms Cartographer for all three scenes. Before optimization, compared to Cartographer [6], our algorithm reduces the $\chi^2$-value by 6.6%, 17.7%, and 43.5% on DY-2, DY-3, and DY-4, respectively. Moreover, the higher the number of dynamic objects is, the more reduction $\chi^2$-value has. After optimization, the $\chi^2$-value of ours decreases by 9.0%, 4.0%, and 41.7%, respectively.

*2) Resource Usage:* In addition to accuracy, analysis of how system resources are utilized is important, such that there is remaining resource for other programs to function. We monitored the CPU usage. After comparison with Cartographer [6], it is evident from Fig 12 and Table 1 that DY-LIO exhibits favorable performance metrics. The CPU usage of ours is in a reasonable range in all three cases. Compared to the maximum usage[2] of Cartographer [6], ours is reduced by 55.3%, 30.6% and 79.5%, respectively. We can conclude, our algorithm has less requirements on hardware.

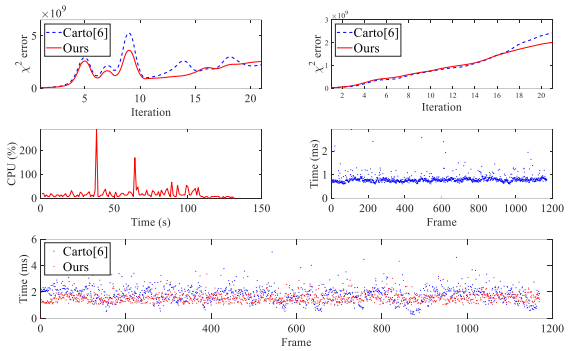
*3) Time Cost:* Apart from localization and mapping accuracy, time cost is a crucial metric for evaluating the performance of a SLAM system. To quantify the processing speed of our framework, we measure the required time of dynamic object removal process and front-end matching process for each point cloud frame, respectively. The results are presented in Fig 12 and Table 1, which have significant improvements in processing efficiency enabled by our IMU pre-integration-based pre-alignment. In all three cases, the overall processing time for point cloud handling typically falls within the range of 0.5-1.0ms, with an average time of 0.839ms, 0.803ms and 0.712ms, respectively. Such processing time satisfies real-time constraints. Besides, as the dynamic point cloud is effectively eliminated, the number of point clouds involved in matching decreases, resulting in improved front-end matching efficiency. Compared to Cartographer [6], our method reduces the scanning matching time by 21.8%, 13.8%, and 28.2% on average for the three cases, respectively. Furthermore, as the number of dynamic

[2] CPU with six physical cores, if the usage of all cores reaches 100%, the total CPU usage will be displayed as 600%.

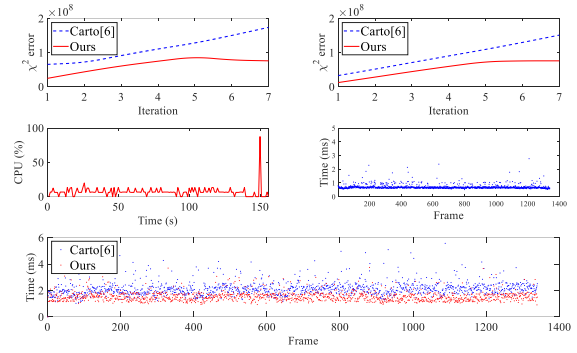
objects increases, our scan matching efficiency surpasses that of Cartographer [6]. Although the point cloud processing takes extra time, the cost is small enough to meet the real-time requirement. Given our enhanced scan matching efficiency, the overall time-cost in the front-end stage is comparable to that of Cartographer [6].




(1)




(2)




(3)

**Fig. 12.** Results of quantitative analysis for Cartographer [6] and DY-LIO on three datasets: (1) DY-2; (2) DY-3;(3) DY-4. In each subgraph, upper left and upper right show the front-end global error; middle left shows the CPU usage of DY-LIO; middle right shows the point cloud process time cost for each point cloud frame of DY-LIO; bottom shows the time cost of scan matching on Cartographer [6] and DY-LIO.

### *D. Ablation Study*

In this section, the quality of map construction is further evaluated by ablation study. Firstly, variation of $\theta$ is conducted for investigating its influence on map building performance. Then, whether we need contribution analysis is also analyzed.

*1) Weighting Factor and Contribution Analysis:* To investigate the influence of different weighting factor $\theta$ on map building performance, we set it as 0.2, 0.4, 0.6, 0.8, and 1.0. And for each $\theta$. We constructed maps using only map processing with and without contribution analysis on the DY-2 dataset. The results are illustrated in Fig. 13. Notably, when the weighting factor $\theta$ was set to 0.6, our map process method exhibited a balanced performance in filtering dynamic grid while retaining static grid. When $\theta$ was set to 0.8 without performing contribution analysis, a larger number of static grids were retained compared to other cases.

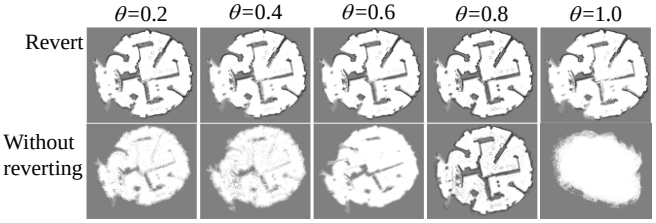



**Fig.13.** The qualitative analysis of weighting factor and contribution analysis.

The impact of contribution analysis proved to be significant. Following the application of contribution analysis, the map exhibited clearer obstacle contours and edges, resulting in an overall cleaner representation. This enhancement can be attributed to the increased retention of static grid, which subsequently improved the accuracy of scan matching and final ray tracing.

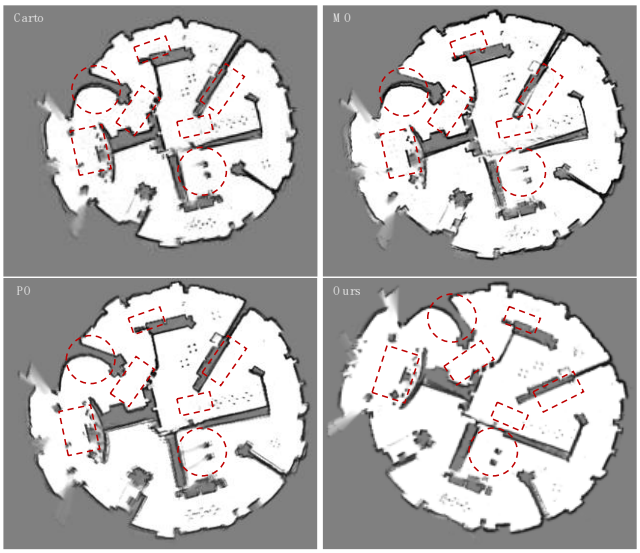



**Fig. 14.** The qualitative analysis of different variants.

*2) Different Variants:* We now investigate two different variants of our approach:

- *Point only (PO)* method only uses the point cloud process;
- *Map only* (MO) method only uses the map process

Similarly, we conducted qualitative analysis by constructing maps using the DY-2 dataset, and the results are shown in the Fig. 14. Both PO and MO outperform Cartographer [6] in removing dynamic objects. PO utilizes

TABLE I
RESULTS OF QUANTITATIVE ANALYSIS BASED ON DY-2, DY-3 AND DY-4 DATASETS FOR CARTOGRAPHER [6] AND DY-LIO

| Metrics \ Datasets | | DY-2 | DY-3 | DY-4 |
|---|---|---|---|---|
| Average Initial $\chi^2$ | Carto [6] | 2423708876.3 | 1842515815.2 | 112810688.6 |
| | Ours | 2262656771.3* | 1515669820.8* | 63732441.2* |
| Average Final $\chi^2$ | Carto [6] | 944262271.01 | 969575591.1 | 90763175 |
| | Ours | 859360834.32* | 930459448.5* | 52915492* |
| Maximum CPU Usage | Carto [6] | 381.2% | 413.3% | 426.7% |
| | Ours | 160.0%* | 286.7%* | 87.5%* |
| Average time of point cloud process | Carto [6] | | | |
| | Ours | 0.839ms | 0.803ms | 0.712ms |
| Average time of scan matching | Carto [6] | 1.860ms | 1.810ms | 2.148ms |
| | Ours | 1.455ms* | 1.561ms* | 1.543ms* |

* Significant result

the initial point cloud and filters dynamic objects solely through map processing, resulting in a slightly lower level of map cleanliness compared to MO. On the other hand, MO does not involve map processing, leading to slower filtering of dynamic grids and potential errors in scan alignment. DY-LIO can simultaneously processes both point cloud and map, enabling a faster and "cleaner" matching process. As a result, it achieves optimal performance in both dynamic object filtering and map coherence.

## IV. Conclusion

We propose DY-LIO to perform real-time and robust state estimation and mapping in highly dynamic environments. DY-LIO considers both LiDAR point cloud and occupancy grid map to remove dynamic objects. Besides, it does not rely on any prior training data, and not limited by the type of moving objects. Hence, DY-LIO can be applied to various scenes and obtain accurate maps robustly.

There are still some not well-solved problems. For dynamic objects in the scene that remain relatively stationary with the vehicle, point cloud processing of DY-LIO does not remove them well. Another problem is that DY-LIO is not suitable when moving objects completely block the FOV of our sensors.

## References


[1]. C. Cadena et al., "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," in IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309-1332, Dec. 2016, doi: 10.1109/TRO.2016.2624754.

[2]. L. Li, Z. Li, S. Liu and H. Li, "Motion Estimation and Coding Structure for Inter-Prediction of LiDAR Point Cloud Geometry," in *IEEE Transactions on Multimedia*, vol. 24, pp. 4504-4513, 2022, doi: 10.1109/TMM.2021.3119872.

[3]. D. V. Nam and K. Gon-Woo, "Solid-State LiDAR based-SLAM: A Concise Review and Application," 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju Island, Korea (South), 2021, pp. 302-305, doi: 10.1109/BigComp51126.2021.00064.

[4]. M. Krivokuća, E. Miandji, C. Guillemot and P. A. Chou, "Compression of Plenoptic Point Cloud Attributes Using 6-D Point Clouds and 6-D Transforms," in IEEE Transactions on Multimedia, vol. 25, pp. 593-607, 2023, doi: 10.1109/TMM.2021.3129341.

[5]. C. Lv, W. Lin and B. Zhao, "Voxel Structure-Based Mesh Reconstruction From a 3D Point Cloud," in IEEE Transactions on Multimedia, vol. 24, pp. 1815-1829, 2022, doi: 10.1109/TMM.2021.3073265.

[6]. W. Hess, D. Kohler, H. Rapp and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 1271-1278, doi: 10.1109/ICRA.2016.7487258.

[7]. J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real time," in Proc. Int. Conf. Robot. Sci. Syst., California, USA, Jul. 2014, doi:10.15607/RSS.2014.X.007.

[8]. T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 4758-4765, doi: 10.1109/IROS.2018.8594299.

[9]. C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang and M. Liu, "LINS: A Lidar-Inertial State Estimator for Robust and Efficient Navigation," 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 8899-8906, doi: 10.1109/ICRA40945.2020.9197567.

[10]. H. Ye, Y. Chen and M. Liu, "Tightly Coupled 3D Lidar Inertial Odometry and Mapping," 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 3144-3150, doi: 10.1109/ICRA.2019.8793511.

[11]. T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 5135-5142, doi: 10.1109/IROS45743.2020.9341176.

[12]. Z. Wang, L. Zhang, Y. Shen and Y. Zhou, "D-LIOM: Tightly-coupled Direct LiDAR-Inertial Odometry and Mapping," in IEEE Transactions on Multimedia, doi: 10.1109/TMM.2022.3168423.

[13]. X. Xia, N. P. Bhatt, A. Khajepour and E. Hashemi, "Integrated Inertial-LiDAR-Based Map Matching Localization for Varying Environments," in IEEE Transactions on Intelligent Vehicles, vol. 8, no. 10, pp. 4307-4318, Oct. 2023, doi: 10.1109/TIV.2023.3298892.

[14]. J. Li, X. Zhang, Y. Zhang, Y. Chang and K. Zhao, "RF-LOAM: Robust and Fast LiDAR Odometry and Mapping in Urban Dynamic Environment," in *IEEE Sensors Journal*, vol. 23, no. 23, pp. 29186-29199, 1 Dec.1, 2023, doi: 10.1109/JSEN.2023.3324429.

[15]. A. Milioto, I. Vizzo, J. Behley and C. Stachniss, "RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),

Macau, China, 2019, pp. 4213-4220, doi: 10.1109/IROS40897.2019.8967762.
[16]. Z. Meng, X. Xia, R. Xu, W. Liu and J. Ma, "HYDRO-3D: Hybrid Object Detection and Tracking for Cooperative Perception Using 3D LiDAR," in *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4069-4080, Aug. 2023, doi: 10.1109/TIV.2023.3282567.
[17]. Cortinhal, T., Tzelepis, G., Erdal Aksoy, E. (2020). SalsaNext: Fast, Uncertainty-Aware Semantic Segmentation of LiDAR Point Clouds. In: Bebis, G., *et al.* Advances in Visual Computing. ISVC 2020. Lecture Notes in Computer Science(), vol 12510. Springer, Cham, doi: 10.1007/978-3-030-64559-5_16
[18]. G. Kim and A. Kim, "Remove, then Revert: Static Point cloud Map Construction using Multiresolution Range Images," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 10758-10765, doi: 10.1109/IROS45743.2020.9340856.
[19]. Qian, C., Xiang, Z., Wu, Z., & Sun, H, "RF-LIO: Removal-First Tightly-coupled Lidar Inertial Odometry in High Dynamic Environments," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 2021, pp. 4421-4428, doi: 10.1109/IROS51168.2021.9636624.
[20]. D. Yoon, T. Tang and T. Barfoot, "Mapless Online Detection of Dynamic Objects in 3D Lidar," 2019 16th Conference on Computer and Robot Vision (CRV), Kingston, QC, Canada, 2019, pp. 113-120, doi: 10.1109/CRV.2019.00023.
[21]. M. T. Lázaro, R. Capobianco and G. Grisetti, "Efficient Long-term Mapping in Dynamic Environments," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 153-160, doi: 10.1109/IROS.2018.8594310.
[22]. F. Pomerleau, P. Krüsi, F. Colas, P. Furgale and R. Siegwart, "Long-term 3D map maintenance in dynamic environments," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014, pp. 3712-3719, doi: 10.1109/ICRA.2014.6907397.
[23]. H. Lim, S. Hwang and H. Myung, "ERASOR: Egocentric Ratio of Pseudo Occupancy-Based Dynamic Object Removal for Static 3D Point Cloud Map Building," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 2272-2279, April 2021, doi: 10.1109/LRA.2021.3061363.
[24]. 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, no. 3, pp. 189–206, 2013.
[25]. J. Schauer and A. Nüchter, "The Peopleremover—Removing Dynamic Objects From 3-D Point Cloud Data by Traversing a Voxel Occupancy Grid," in IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1679-1686, July 2018, doi: 10.1109/LRA.2018.2801797.
[26]. R. Danescu, F. Oniga and S. Nedevschi, "Modeling and Tracking the Driving Environment With a Particle-Based Occupancy Grid," in IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 4, pp. 1331-1342, Dec. 2011, doi: 10.1109/TITS.2011.2158097.
[27]. D. Nuss, S. Reuter, M. Thom, T. Yuan, G. Krehl, M. Maile, A. Gern, and K. Dietmayer, "A random finite set approach for dynamic occupancy grid maps with real-time application," The International Journal of Robotics Research, vol. 37, no. 8, pp. 841–86
[28]. S. Hoermann, P. Henzler, M. Bach and K. Dietmayer, "Object Detection on Dynamic Occupancy Grid Maps Using Deep Learning and Automatic Label Generation," 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 2018, pp. 826-833, doi: 10.1109/IVS.2018.8500677.
[29]. S. Wirges, T. Fischer, C. Stiller and J. B. Frias, "Object Detection and Classification in Occupancy Grid Maps Using Deep Convolutional Networks," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 2018, pp. 3530-3535, doi: 10.1109/ITSC.2018.8569433.
[30]. M. Schreiber, V. Belagiannis, C. Gläser and K. Dietmayer, "Dynamic Occupancy Grid Mapping with Recurrent Neural Networks," 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 2021, pp. 6717-6724, doi: 10.1109/ICRA48506.2021.9561375.
[31]. J. Dequaire, P. Ondr´uˇska, D. Rao, D. Wang, and I. Posner, "Deep tracking in the wild: End-to-end tracking using recurrent neural networks," The International Journal of Robotics Research, vol. 37, no. 4-5, pp. 492–512, 2018.
[32]. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Medical Image Computing and Computer-Assisted Intervention, 2015, pp. 234–241.
[33]. A. Forster, L. Carlone, F. Dellaert and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual--Inertial Odometry," in IEEE Transactions on Robotics, vol. 33, no. 1, pp. 1-21, Feb. 2017, doi: 10.1109/TRO.2016.2597321.
[34]. Jian Yang, D. Zhang, A. F. Frangi and Jing-yu Yang, "Two-dimensional PCA: a new approach to appearance-based face representation and recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 1, pp. 131-137, Jan. 2004, doi: 10.1109/TPAMI.2004.1261097.
[35]. B. Antoni, G. Yolanda, & O. Gabriel. On the use of likelihood fields to perform sonar scan matching localization. Auton Robot, 26:203-222, doi: 10.1007/s10514-009-9108-0.
[36]. F Dellaert, "Factor graphs and Gtsam: A hands-on introduction[R]", *Georgia Institute of Technology*, 2012.
[37]. K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai and R. Vincent, "Efficient Sparse Pose Adjustment for 2D mapping," 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010, pp. 22-29, doi: 10.1109/IROS.2010.5649043.

**Jingliang Zou** received the B.S. degrees from the College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China, in 2021. Starting from 2021, he is pursuing his M.S. degree at the College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China. His research interests include SLAM and motion planning.

**Huangsong Chen** received the B.S. degrees from the College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China, in 2022. Starting from 2022, he is pursuing his M.S. degree at the College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China. His research interests include SLAM and motion planning.

**Liang Shao** received the B.S. degree from Wuhan University, China, in 2011, the M.Eng. degree from Tongji University, China, in 2014, and the Ph.D. degree from the Graz University of Technology, Austria, in 2019, all in mechanical engineering.

He currently works in Wenzhou University and his research interests include nonlinear state estimation, vehicle dynamics control, and motion planning.

**Haoran Bao** received the B.S. degrees from the College of Optical, Mechanical and Electrical Engineering, Zhejiang A&F University, Wenzhou, China, in 2018 and 2022. Starting from 2022, he is pursuing his M.S. degree at the College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China. His research interests include SLAM and vehicle dynamics control.

**Hesheng Tang** received the Ph.D. degree in mechatronics engineering from Tongji University, Shanghai, China, in 2016.

He is currently an Associate Professor with the School of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China. His research interests include fault detection/diagnosis theory, models, and techniques for high-end hydraulic components.

**Jiawei Xiang** (Member, IEEE) received the B.S. degree in mechatronics from Hunan University, Changsha, China, in 1997, the M.S. degree in mechanical engineering from Guangxi University, Nanning, China, in 2003, and the Ph.D. degree in mechanical engineering from Xi'an Jiaotong University, Xi'an, China, in 2006. He is currently a Professor with the College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China. His research interests include the health monitoring of mechanical systems using numerical simulation and signal processing techniques. Dr. Xiang is a member of the American Society of Mechanical Engineering.

**Jun Liu** received the B.S. degree from Hainan University, China, in 2012, the M.Eng. degree from Tongji University, China, in 2015. theory, models, and techniques for high-end hydraulic components.

He is currently the technical vice president of Shanghai X-AI Auto Technology Co., Ltd. He has many experiences in research and application in state estimation and vehicle dynamics control for automated driving.