DLOAM: Real-time and Robust LiDAR SLAM System Based on CNN in Dynamic Urban Environments

Wenbo Liu, Wei Sun, *Member, IEEE*, and Yi Liu

Abstract : Dynamic object detection, state estimation, and map-building are crucial for autonomous robot systems and intelligent transportation applications in urban scenarios. Most current LiDAR Simultaneous Localization and Mapping (SLAM) systems operate on the assumption that the observed environment is static. However, the overall accuracy and robustness of a SLAM system can be compromised by dynamic objects in the environment. Aiming at the problem of inaccurate odometry estimation and wrong mapping caused by the existing LiDAR SLAM method which cannot detect the dynamic objects, we study the SLAM problem of robots and unmanned vehicles equipped with LiDAR traveling in the dynamic urban scenes. We propose a fast LiDAR-only model-free dynamic objects detection method, which uses the spatial and temporal information of point cloud through a convolutional neural network (CNN), and the detection accuracy is improved by 35% to 86% compared with methods that only use spatial information. We further integrate it into a state-of-the-art LiDAR SLAM framework to improve the SLAM performance. Firstly, the range image constructed by LiDAR point cloud is used for ground extraction and non-ground point clustering. Then, the motion of objects in the scene is estimated by the difference between adjacent frames, and the segmented objects are further divided into dynamic objects and static objects by their motion features. After that, the stable feature points are extracted from the static objects. Finally, the pose transformation of adjacent frames is solved by matching feature point pairs. We evaluated the accuracy and robustness of our system on datasets with different challenging dynamic environments, and the results show our system has significant improvements in accuracy and robustness of odometry and mapping, while still maintain real-time performance, which is sufficient for autonomous robot systems and intelligent transportation applications in urban scenarios.

***Index Terms*—SLAM, CNN, LiDAR odometry, Point cloud registration, Dynamic object detection, Range image segmentation.**

I. INTRODUCTION[1]

Simultaneous Localization and Mapping (SLAM) is a basic prerequisite for intelligent robots and a necessary capability for driverless vehicles. Great efforts have been made before. In many applications, the real-time 6 degree-of-freedom SLAM can be achieved using vision-based [1] or LiDAR-based [2][3] methods. It can also incorporate their advantages [4] or be used with other auxiliary sensors (e.g. Inertial Measurement Unit (IMU) [5], wheel odometry, Global Positioning System (GPS) [6]) for better results.

Intelligent transportation system usually works in a variable environment, even in the environment where offline map has been established, there may still be a large number of dynamic objects interfering with the map. Detecting dynamic objects helps the carrier to realize the tasks of real-time perception and prediction of the surrounding environment, collision prediction and path planning, which can also improve the robustness of vehicle pose estimation, simultaneous localization and mapping in unknown environment [7]. Therefore, detecting dynamic objects in the environment is the key to safe and reliable intelligent transportation and automatic driving. Our motivation comes from the urgent need for the robustness and accuracy of SLAM systems works in dynamic scenarios for autonomous robot systems and intelligent transportation applications. One the on hand, in the real world, there are a lot of dynamic objects, such as inevitable moving objects (e.g., pedestrians, vehicles and multiple robots work together), unstable objects in the environment (e.g., leaves, lawns), and even sudden objects (e.g., animals rushing to the road, rocks flying up) and so on. The existing methods can easily lead to the wrong pose estimation and mapping results due to the wrong correspondence of the feature points on dynamic objects. On the other hand, in order to avoid the dangerous situation caused the by disappearance of external information and failure of auxiliary sensor, these subsystems must also work independently [8]. Although the vision-based methods have the advantages of fast detection frequency and good at identifying loop-closure, their sensitivity to light intensity and view changes will reduce the robustness of the SLAM system. In contrast, LiDAR based methods work even at night, and many high-resolution 3D LiDAR allow the capture of fine details of the environment at a distance [9]. Therefore, in this paper, we focus on using 3D LiDAR to deal with the challenge of SLAM in dynamic urban environments.

Unfortunately, most LiDAR based SLAM methods assume that the surrounding environment is static. These methods are difficult to provide robust and accurate pose estimation and build accurate static maps in dynamic urban scenes. And there are few reliable LiDAR-only SLAM systems. Obviously, we need to design a real-time

This work was supported in part by National Nature Science Foundation of China (NSFC) under Grants No. 61671356, Science and Technology Program of Shaanxi Province under Grants No. 2020GY-136 and 2019ZDLGY14-02-03.

Wenbo Liu is with the School of Aerospace Science and Technology, University of Xidian, Xi'an, China(e-mail: wbliu1@stu.xidian.edu.cn).

Wei Sun, corresponding author, is with the School of Aerospace Science and Technology, University of Xidian, Xi'an, China (e-mail: wsun@xidian.edu.cn).

Yi Liu is with the School of Aerospace Science and Technology, University of Xidian, Xi'an, China (e-mail: 714515894@qq.com).

robust LiDAR-only SLAM system for these situations. The main challenge of this work is to analyze the motion information in the complex dynamic environment and identify the dynamic objects. Specifically, our goal is not to detect all theoretically movable objects, such as vehicles or people, but to separate the actual moving objects (such as driving cars) from static or non-moving objects (such as buildings, parked cars, etc.).

Inspired by the above discussion, we propose a new r LiDAR-only odometry and mapping method suitable for dynamic environments, which not only realizes robust pose estimation, but also builds a static map filtering out dynamic objects in environments, which promotes the development of dynamic object detection and LiDAR SLAM. Our dynamic object detection method can accurately divide the scene into dynamic and static objects, such as moving cars and parked cars. In other words, it neither semantically segments the point cloud, that is, predicts semantic categories, such as vehicles, pedestrians, buildings, roads, etc., nor limits to whether a point is dynamic. Specifically, our proposed method takes the point cloud of the rotating 3D LiDAR as input and the continuous point cloud range image as the intermediate representation to obtain the motion label of the objects in the scene. We use CNN and traditional 3D point cloud processing methods to realize dynamic object detection respectively. We compare the proposed approach and system with several other state-of-the-art approaches and systems that have shown excellent results in dynamic urban environments, and the experimental results show that in dynamic scenarios, the accuracy of dynamic object detection is greatly improved compared with methods that only use spatial information, and the odometry and mapping accuracy are significantly improved. The proposed SLAM system is mainly applied to autonomous robots, intelligent transportation and automatic vehicles in dynamic urban scenarios, but the method is not limited to these fields. To sum up, the main contributions of this paper are as follows:

- We propose a fast model-free dynamic object detection method based on the spatial and temporal information of point cloud, which is realized by CNN and traditional methods respectively. Compared with only using semantic information to remove all possible dynamic objects and dynamic point detection method based on scene flow, this method can detect dynamic objects more accurately and is more suitable for urban dynamic scenes; We also introduce a method based on Random Sample Consensus (RANSAC) to deal with the pseudo-static objects.
- We present a real-time LiDAR-only SLAM framework suitable for both static and dynamic environments, which is easy to be further expanded and developed. By embedding the proposed dynamic object detection module into a state-of-the-art LiDAR SLAM framework, the accuracy of odometry and mapping is greatly improved in dynamic environments compared with methods which are under the static-world assumption.
- In order to verify our method, we have done detailed tests on challenging datasets. The results show that the accuracy of dynamic object detection is improved by 35% to 86% compared with methods that only use spatial information, and the accuracy and robustness of SLAM in these scenarios are significantly improved.

The remainder of this paper is organized as follows: related work is presented in Section II, the proposed methods are described in Section III, quantitative evaluation and analysis between our method and the state-of-the-art method are showed in Section IV, followed by conclusions and future works in Section V.

II. Related Work

We refer to the SLAM review article [10] and KITTI odometry evaluation ranking [11], and we focuses on the methods of recent years because they are more representative of the most advanced methods and the latest research directions.

A. LiDAR Odometry and Mapping

LiDAR-only odometry and mapping methods with good performance are as follows. LOAM [2], no. 2 on KITTI odometry evaluation ranking, has very low motion drift on short-trajectory datasets and is largely responsible for the growth of LiDAR SLAM. This method is novel in partitioning SLAM problem into a high frequency but low accuracy odometry algorithm and a low frequency but high accuracy mapping algorithm. Pose estimation is realized by matching feature points extracted from all points in adjacent scans. The mapping algorithm realizes the fine odometry by registering the feature points with the historical map and registers the corrected point cloud into the map. However, LOAM is unable to eliminates the continuous accumulation of errors due to the absence of loop-closure detection, therefore, significant errors may occur in the case of long trajectories. More importantly, LOAM does not consider dynamic objects, resulting in incorrect mapping in dynamic scenarios. LeGo-LOAM [9] is carried on the optimization to LOAM. It adds loop-closure detection to reduce the motion drift, and adds clustering and segmentation modules to remove the clusters with less than 30 points, so as to realize the small object filtering and more reliable feature extraction and matching in the environment. The original 6 degree-of-freedom pose estimation in one step was optimized to a more reasonable two-step pose estimation. However, its ground segmentation method is too rough (only 10° is selected as the ground segmentation standard), and it is difficult to select the threshold value of noise filtering point (if the threshold is too small, it will be difficult to identify large dynamic objects and distant sparse dynamic objects, if the threshold is too large, the map will be sparse because the features used for registration will be reduced). [8] improves odometry accuracy by detecting geometric similarity between online 3D point cloud and prior offline map, but SLAM problem is more applied in unknown environments, and new objects in the scene will cause inconsistency between the actual map and prior map, so this kind of method which depends on prior information has great limitations in unknown or highly dynamic environments. IMLS-SLAM [29] presents a low-drift SLAM algorithm based on Implicit Moving Least Squares surface representation. S4-SLAM [28] proposes a suitable for outdoor multi-scene applications with sparse features, highly dynamic environments by a coarse-to-fine scan matching strategy. But the two methods cannot achieve real-time performance. LiTAMIN2 [30] proposes an ultra-light SLAM method using geometric approximation applied with KL-Divergence, and it has high computational efficiency and similar accuracy to other state-of-the-art methods.

The list of related work is endless. They use different methods to improve performance, but they either ignore the dynamic objects problem or do not offer good solutions, therefore, these methods have limitations in dynamic environments. We can improve their performance by combining dynamic object detection methods.

B. LiDAR Dynamic Object Detection

3D LiDAR provides more accurate distance measurements than stereo cameras and more abundant measurements than 2D LiDAR, which provides more methods for dynamic object detection. The goal of most existing 3D LiDAR based dynamic object detection approaches is to clean up the point cloud map, and most of these methods do not work online and rely on pre-built maps. Since the algorithm will be embedded into a real-time SLAM system, it has to be real-time and map-free.

To the best of our known, dynamic object detection approaches that meet the requirements can be summarized as object-based and object-independent methods. The object-based methods attempt to extract part of feature points from a certain cluster of points and uses the motions of feature points to represent the motion of the object. [12] compares the current scan with the oldest map in history in a 2D grid, marks obstacles that appear only in the new scan as dynamic obstacles, and detects 3D obstacles by sudden changes in inspection depth. [13] uses a combination of 3D Kalman filter and Hungarian algorithm for state estimation and data correlation to achieve fast motion estimation. The object-independent approaches try to assign speed labels to each point in the point cloud, and relying less on the characteristics of the moving object. [14] extracted point cloud semantic features through CNN, and integrated the semantics into surfel to filter out moving objects, similarly, [18][19] use CNN for semantic segmentation to detect moving objects. [16] proposes a method called LiDAR-flow, which provides robust estimation of dense scene flows by fusing sparse LiDAR with stereo images. [17] and [20] fusing temporal information to detect dynamic objects, they use multi frame point cloud as input to regress the motion behavior of objects on the aerial view through the network, the advantage of this kind of method is to detect all moving objects in the field of vision of LiDAR, including objects not seen in the training set, which is of great significance to the safety of robotics automatic driving.

Our goal is to identify dynamic objects in the dynamic urban scenarios to improve the accuracy of the SLAM system by avoiding extracting feature points from them. Compared to other outdoor scenes, urban scenes usually have a large number of different types of artificial objects (buildings, streets) and different objects usually have clean boundaries. Moreover, dynamic objects in the scene are usually clearly demarcated from the background. Therefore, it is easy to segment the point clouds obtained in urban scenes. Obviously, the object-based approach is more suitable for our system.

C. LiDAR SLAM in Dynamic Environments

Most current SLAM methods operate on the static-world assumption, which limits the performance in dynamic scenarios. Many scholars begin to consider and study the SLAM problem in dynamic scenarios.

Many methods explicitly handle dynamic objects. IMLS-SLAM [29] uses a specific sampling strategy and a new scan to model matching. But it directly discards all objects whose size makes it possible to be dynamic objects. S4-SLAM [28]also perform a small object removal instead of a dynamic object removal. It is difficult for these methods to achieve robust and accuracy pose estimation and map-building in dynamic environments. Some mapping-based methods implicitly handle dynamic objects. For instance, SuMa [14] filter dynamics and estimate the pose transformation by a surfel-based map. In addition, some methods realize dynamic SLAM by combining a CNN-based dynamic object detection and a SLAM algorithm. [26] adds CNN segmentation network to Laser-Inertial odometry and mapping framework to realize robust SLAM in highway environments. SuMa++ [27] improves SLAM accuracy by adding dynamic filtering and semantic matching to SuMa. It can be seen that the problem of LiDAR SLAM in dynamic environment based has not been well solved.

This paper aims to realize real-time, high-precision and robust SLAM in dynamic urban environments. We select the mature LiDAR SLAM system LeGo-LOAM as the baseline method. The odometry and mapping method used in this paper come from LeGo-LOAM, and we have made some optimization to make the system better adapt to the dynamic environments. In order to reduce the accumulated error caused by wrong feature points matching in odometry module and mapping module, we use the difference of adjacent range images to segment dynamic and static objects, and only use the static features to estimate pose transformation and to map. This idea of improving the accuracy and robustness of a SLAM system in dynamic environments by detecting and rejecting dynamic objects to participate in odometry and map building is easy to be applied to other SLAM systems.

III. Proposed System

In this section, we will describe our proposed approach and the data flow between the modules. We introduce the *dynamic object detection* module in detail, while the *segmentation* module, *feature extraction* module and *LiDAR odometry* module in LeGo-LOAM [9] briefly. In addition, we highlight some optimization we made to integrate them.

A. Description and Definitions

The problem addressed in this work is how to use the point cloud observed by 3D LiDAR to estimate its ego-motion and build a global map for the traversed environment.

We define the received point cloud as $\boldsymbol{P}$, LiDAR pose as X, and global map as M. We use the upper right corner to represent the time stamp, and define the kth point cloud as $\boldsymbol{P^k}$, where a point is p_i^k, that is, $\boldsymbol{P^k}=\{p_1^k,p_2^k,\cdots,p_n^k\}$, $p_i^k=(x_i^k,y_i^k,z_i^k,1)^{\mathrm{T}}$, where n is the size of point cloud, $k\in\mathbb{N}^+$.

Under the above definition, the problem can be modeled as: Given map M^{k-1} and LiDAR pose X^{k-1} at scan k-1, and point cloud $\boldsymbol{P^k}$ at scan k, to update X^k and M^k, as can be shown in (1) and (2).

$$X^k=\mathbb{F}(X^{k-1},\boldsymbol{P^k})=\mathbb{F}(X^{k-1},\mathbb{W}(\boldsymbol{P^{*k}},W^k)). \quad (1)$$

$$M^k=\mathbb{G}(M^{k-1},\boldsymbol{P^k},W^k)=\mathbb{G}(M^{k-1},\mathbb{W}(\boldsymbol{P^{*k}},W^k)). \quad (2)$$

where P^{*k} is the description of the real scene in the field of vision at scan k. $W^k=\mathbb{W}(\boldsymbol{P^{*k}},W^k)$ is the sum of many interference factors such as dynamic objects, sensor noise, or abnormal measurement values. $\mathbb{W}$ is the unknown noise function that transforms the real scene into the received point cloud. Next, we define the set of outliers in point cloud caused by W that are harmful to registration as $\boldsymbol{P_{out}}$, and the remaining points form $\boldsymbol{P_{in}}$, namely,

$$\boldsymbol{P}=\boldsymbol{P_{out}}\cup\boldsymbol{P_{in}}=\mathbb{W}(P^*,W). \quad (3)$$

Details of function $\mathbb{F}$ and $\mathbb{G}$ can be found in baseline method [9]. The core task of this paper is to find and eliminate $\boldsymbol{P_{out}}$ to optimize the two functions.

B. System Overview

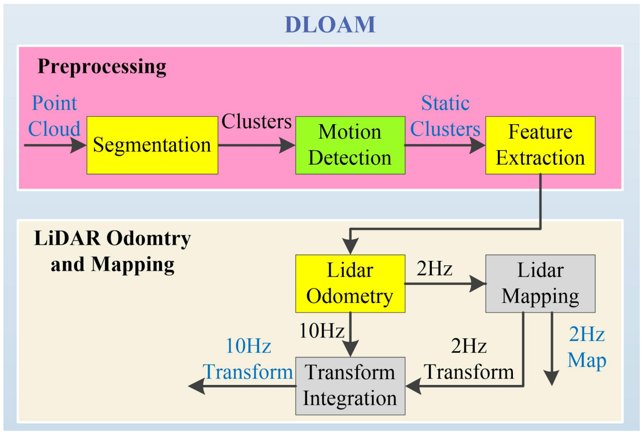

Fig. 1. The pipeline and data flow of the proposed system.

The pipeline and data flow of the proposed system are shown in Fig. 1. The input of the system is 3D LiDAR point cloud, and the output is 10Hz 6 degree-of-freedom pose estimation and 2Hz global map of the traversed environment. The whole system is divided into six modules. Grey modules are original in LeGo-LOAM, orange modules represent the ones we have optimized, and the green one is the part we added. The first module is Segmentation. The point cloud of a scan is projected onto the range image for clustering, which is used to segment different objects. The second module is dynamic object detection. The clustering results of the Segmentation module are further divided into static clusters and dynamic clusters by extracting motion labels of objects. The static clusters are then sent to the Feature Extraction module to extract edge points and surface points. Then, the LiDAR Odometry uses feature points to solve the pose transformation between adjacent frames. These features are further processed in LiDAR Mapping, which registers the point clouds without motion distortion into global map and detects loops. Finally, the Transform Integration module fuses the high frequency but low precision transform from the Odometry module and the low frequency but high precision transform results from the Mapping module to obtain the final pose estimation. Compared to the original frameworks in the LOAM and LeGo-LOAM, the system aims to improve the robustness and accuracy of odometry and mapping for robots or driverless vehicles in dynamic environments. Details of these modules are described below.

C. Segmentation

The input of the segmentation module is an original 3D LiDAR point cloud $\boldsymbol{P}=\{p_1,p_2,\cdots,p_n\}$, where $p_i=(x_i,y_i,z_i,1)^{\mathrm{T}}$. The output are m clusters $\{P_{C_1},P_{C_2},\cdots,P_{C_m}\}$ and the corresponding cluster label vector $L=(l_1,l_2,\cdots,l_n)$, where $l_i\in[-1,m]\cap\mathbb{Z}$, label -1 represents the point belongs to the ground, and the positive integer represents the serial number of non-ground clusters.

Point cloud $\boldsymbol{P^k}$ is projected onto a range image $\boldsymbol{P_{U,V}}^{\boldsymbol{k}}$, where $\boldsymbol{P_{U,V}}\triangleq\{p_{u,v}|u\in[1,\frac{2\pi}{R_{hori}}]\cap\mathbb{Z},\ v\in[1,N_{scan}]\cap\mathbb{Z}\}$. The row index v_i of p_i is shown in (4), and the column index u_i is shown in (5).

$$v_i=(\tan^{-1}\frac{z_i}{\sqrt{x_i^2+y_i^2}}+Pit_{bottom})/R_{vert}. \quad (4)$$

$$u_i=\tan^{-1}\frac{y_i}{x_i}/R_{hori}. \quad (5)$$

The value of the image is the distance from p_i to the LiDAR $r_i=\sqrt{x_i^2+y_i^2+z_i^2}$, where R_{hori} is the horizontal resolution of the LiDAR, R_{vert} is the vertical resolution of the LiDAR, Pit_{bottom} is the minimum pitch of the LiDAR beam, and N_{scan} is the number of LiDAR beams. After the projection, a point p_i and its coordinates can be uniquely identified by the v_i and u_i, denoted by:

$$p_i\equiv p_{u_i,v_i}=(x_{u_i,v_i},y_{u_i,v_i},z_{u_i,v_i}). \quad (6)$$

Fig. 2 is an example of a point cloud projected onto a range image.

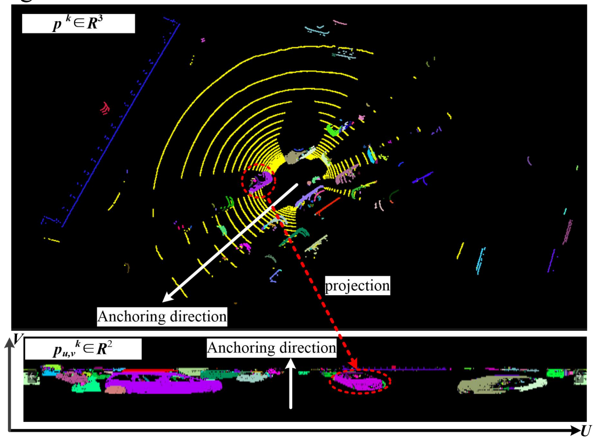

Fig. 2. A 3D point cloud is projected onto a range image and clustered by range image based segmentation method. Using this segmentation result

and the unique correspondence between the elements of two spaces, the classification of a 3D point cloud can be deduced, in which different colors represent different clusters, and yellow points represent the ground.

Then the ground is extracted from the range image $P_{U,V}$. For points $p_{u,v+1}$ and $p_{u,v}$, where $v \in [1, \frac{1}{2}N_{scan}] \cap \mathbb{Z}$, if the pitch of adjacent rows $Pit_{v,v+1}$ (as shown in (7)) are less than 10°, they are denoted as ground points.

$$Pit_{v,v+1} = \tan^{-1}\frac{z_{u,v+1}-z_{u,v+1}}{\sqrt{(x_{u,v+1}-x_{u,v})^2+(y_{u,v+1}-y_{u,v})^2}}. \quad (7)$$

Finally, the range image-based segmentation clustering method [18] is applied to the range image to segment the points into several clusters. Points from the same cluster are assigned a unique label l.

After these processes, for each $p_i^k \in \boldsymbol{P^k}$, we get the ground or non-ground label l_i, row index v_i, column index u_i, and distance r_i, where $i \in [1, N_{pixel}] \cap \mathbb{Z}$. $\boldsymbol{P^k}$ is divided into a ground point set and a non-ground point set, that is, $\boldsymbol{P^k} = \boldsymbol{P_g} \cup \boldsymbol{P_{ng}}$, and it is worth noting that both types of points may contain outliers. The extraction of features from non-ground points can reduce time-consuming of registration and improve the accuracy of registration through label matching and noise filtering. Note that the two sets will continue to be filtered in the next (dynamic object detection) module.

D. Pre-Registration and Dynamic Object Detection

The function of dynamic object detection module is to identify any dynamic objects. This module receives the clustered point cloud from segmentation model, and the output is the motion label of the clustered point cloud, so as to extract the feature points for registration only on the static objects, where static objects are further divided into ground and non-ground objects.

Given only one frame point cloud, it is almost impossible to know which objects are moving. However, given two consecutive frames, it is easy to find the moving objects by their difference. Since we have obtained the range image and corresponding clustering label in the segmentation module, we use the difference of range images of adjacent frames to get the speed matrix, so as to detect dynamic objects according to their speeds.

But please note that we cannot directly difference two point clouds at adjacent time. Because point cloud $\boldsymbol{P^k}$ contains the ego-motion (which is the change of LiDAR pose between adjacent frames), motion distortion of point cloud (which is caused by ego-motion), and the motion of objects in the environment, we should firstly detect ego-motion and point cloud distortion. A natural idea to detect ego-motion is to use ICP-based method to align the adjacent frames and find out the mismatched points, but the computational amount of registration of complete point cloud cannot meet the real-time requirements and the dynamic objects will cause mismatches.

1) Pre-Registration

We are inspired by [22]. The basic idea is to estimate the relative motion of static reference objects from adjacent frames using geometric features. We aim at detecting static vertical (pole-like and vertical planes) objects since our system mainly runs in urban environments with lots of static pole-like structures (such as trees, poles, fences, etc.) and static vertical planes (house, wall, etc.).

Firstly, principal component analysis (PCA) is performed for all clusters with more than N_{PCA} points, we define the eigenvalues from large to small as λ_1, λ_2, and λ_3, and the corresponding eigenvectors as e_1, e_2, e_3 respectively. Clusters whose e_1 is approximately parallel to the normal of the ground n_{ground} and λ_1 is significantly larger than λ_2 and λ_3 are considered as pole-like objects, namely, we use conditions (8) and (9) to detect pole-like objects,

$$\lambda_1 > \lambda_2 \approx \lambda_3, \lambda_1 > \xi_1, \lambda_1 < \xi_2. \quad (8)$$

$$e_1 \cdot n_{ground} / \| n_{ground} \| > \xi_3. \quad (9)$$

While clusters whose e_3 is approximately is perpendicular to n_{ground} and λ_3 is significantly larger than λ_1 and λ_2 are considered as vertical planes, we use conditions (10) and (11) to detect vertical planes.

$$\lambda_1 \approx \lambda_2 > \lambda_3, \lambda_3 < \xi_4. \quad (10)$$

$$e_3 \cdot n_{ground} / \| n_{ground} \| > \xi_5. \quad (11)$$

Secondly, select points belongs to pole-like objects or vertical planes to form a feature point set defined as O. Finally, perform N_{coarse} times RANSAC steps to get coarse transform $\boldsymbol{T_{coarse}^{k-1 \to k}}$ between O^k and O^{k-1}. In each iteration, we use ICP to align two point clouds and discard the first 10% points with the largest registration error. We set N_{PCA} is to 30, ξ_1 to 0.85, ξ_2 to 0.1, ξ_3 to 0.95, ξ_4 to 0.1, ξ_5 to 0.05, and N_{coarse} to 5 empirically.

2) Remove motion distortion

After the above Pre-Registration steps, $\boldsymbol{T_{coarse}^{k-1 \to k}}$ can be applied to $\boldsymbol{P^k}$ to remove the motion distortion of current point cloud by using the linear interpolation of pose transformation between adjacent frames as mentioned in LOAM. Specifically, let t_i be the stamp of a point $p_i^k \in \boldsymbol{P^k}$, while t_{start} and t_{end} be the starting and ending time of current scan $\boldsymbol{P^k}$ respectively, the pose transform $\boldsymbol{T_i}$ between t_{start} and t_i can be computed by linear interpolation of $\boldsymbol{T_{coarse}^{k-1 \to k}}$ as:

$$\boldsymbol{T_i} = \text{linear}\left(\boldsymbol{T_{coarse}^{k-1 \to k}}\right) \triangleq \frac{t_i - t_{start}}{t_{end} - t_{start}} \boldsymbol{T_{coarse}^{k-1 \to k}}. \quad (12)$$

Fig. 3 illustrates the linear interpolation.

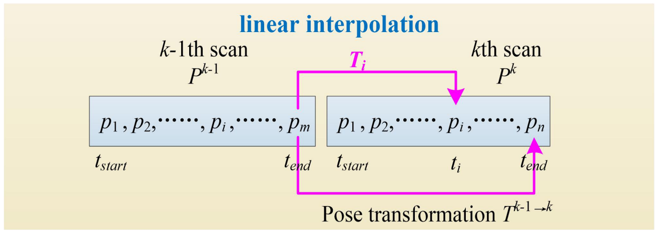
linear interpolation
k-1th scan
P^{k-1}
kth scan
P^k
T_i
$p_1, p_2, \cdots\cdots, p_i, \cdots\cdots, p_m$
$p_1, p_2, \cdots\cdots, p_i, \cdots\cdots, p_n$
t_{start}
t_{end}
t_{start}
t_i
t_{end}
Pose transformation $T^{k-1 \to k}$

Fig. 3. Illustration of linear interpolation of pose transformation between the adjacent frames.

Let the point cloud with motion distortion removed be $\boldsymbol{\bar{p}^k}$, $\boldsymbol{\bar{p}^k} = \{\bar{p}_i \mid \bar{p}_i = \boldsymbol{T_i}^{-1} p_i^k\}$,

where $\boldsymbol{T_i} = \begin{cases} \text{linear}\left(\boldsymbol{T_{fine}^{k-1 \to k}}\right), \text{for previous scan} \\ \text{linear}\left(\boldsymbol{T_{coarse}^{k-1 \to k}}\right), \text{for current scan} \end{cases}$, and $\boldsymbol{T_{fine}^{k-1 \to k}}$ is the pose transform output of LiDAR odometry.

Because we already know the pose transformation $\boldsymbol{T^{j-1 \to j}}$ ($j \in \{[2, k-1] \cap \mathbb{Z}\}$) between the first k-1 scans at current scan k, the transformation from the ath scan to the coordinate frame of the bth ($a<b<=k$) scan is $\boldsymbol{T^{a \to b}} = \prod_{j=a+1}^{b} \boldsymbol{T^{j-1 \to j}}$, and the transformed point cloud is denoted as $\boldsymbol{P^{a \to b}} = \boldsymbol{T^{a \to b}} \boldsymbol{\bar{p}^{k-1}}$, and we define $\boldsymbol{T^{b \to a}} = \boldsymbol{T^{a \to b^{-1}}}$.

3) *Dynamic Object Detection*

We use CNN-based and traditional 3D point cloud processing methods to realize dynamic object detection respectively, which illustrates the feasibility of combining the spatial and temporal information of the point cloud.

a) *Traditional Two Frames Method*

We project the adjacent point clouds into the coordinate frame of the previous one, and then use (4) and (5) to construct the two point clouds into range image, namely, $\boldsymbol{P_{U,V}}^{\boldsymbol{k-1}}$ is constructed from undistorted $\overline{\boldsymbol{p}}^{k-1}$ and $\boldsymbol{P_{U,V}}^{\boldsymbol{k}}$ from $\boldsymbol{P}^{\boldsymbol{k\to k-1}}$. Finally, we can get the speed matrix by their difference,

$$\boldsymbol{S_{U,V}}^{\boldsymbol{k}} \triangleq \boldsymbol{P_{U,V}}^{\boldsymbol{k}} - \boldsymbol{P_{U,V}}^{\boldsymbol{k-1}} = \{s_{u,v}{}^{k} | s_{u,v}{}^{k} = |r_i{}^{k} - r_i{}^{k-1}|, \\ u \in [1,\frac{2\pi}{R_{hori}}] \cap \mathbb{Z}, v \in [1,N_{scan}] \cap \mathbb{Z}\}. \tag{13}$$

We conduct secondary verification on current non-ground points and ground points using speed matrix. Specifically, for non-ground point set, compared with LeGo-LOAM, which directly ignores the cluster with less than 30 points, we will further obtain and check the speed label of each point in the dynamic object detection module. We define the point whose speed is less than the threshold v_{static} as a static point. And if a cluster contains less than α of its total points number static points, it will be considered as a static cluster, otherwise, it will be marked as a dynamic cluster. For ground point set, compared with LeGo-LOAM, which select points with adjacent rows $Pit_{v,v+1}$ less than 10° as ground points, we make a second verification in the dynamic object detection module. Only ground points with speed less than the threshold v_{ground} are selected, and they are not used for clustering.

The parameters are learned empirically, in this paper, v_{ground} is 0.01m/s and v_{static} is 0.03m/s, and α is 50%.

b) *CNN-based Multi-Frames Method*

Define the jth range image pair as $(\boldsymbol{P_{U,V}}^{\boldsymbol{j-1}}, \boldsymbol{P_{U,V}}^{\boldsymbol{j}})$ at scan k, where $j \in \{[2,k] \cap \mathbb{Z}\}$. Each range image pair is used to calculate a speed matrix $\boldsymbol{S_{U,V}}^{\boldsymbol{k}}$. We calculate the nearest m speed matrices and package them and the current range image into a range image bag, where the speed matrix contains the motion information of the objects and the current range image contains the position information of the objects, and take it as the input of a CNN method. The output of CNN is a binary image $\boldsymbol{D_{U,V}}^{\boldsymbol{k}}$, and each element represents whether the corresponding point is dynamic or static.

In order to contrast with the baseline method, we do not change the structure of CNN but retrain it with new datasets. Specifically, we feed a CNN with the range image bags, and manually annotate the static and dynamic points in the scene (mainly for the dynamic vehicles and pedestrians in the urban environments), and we use the same loss function as the base CNN method. The overall work flow of CNN-based multi-frames method is shown in Fig. 4. We choose two popular methods, RangeNet++ and SalsaNext to evaluate the performance of the proposed dynamic object detection method. Please refer to the original paper [18] and [19] for details of the network architecture.

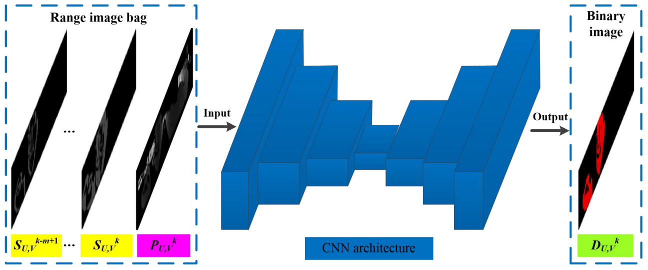

Fig. 4. Overall work flow of the CNN-based multi-frames method. The input is a range image bag consist of a current range image and m recent speed images, and the output is a binary image, where dynamic objects are marked in red.

Here, we have been able to detect the dynamic points in the current point cloud and put them into $\boldsymbol{P_{out}}$. Once again, we only use static points in the adjacent frame for feature extraction and feature points registration.

E. *Feature Extraction*

To match the same object between scans quickly, we use a feature extraction method similar to that used in LeGo-LOAM. Instead of extracting features from the original point cloud, we extract features from ground points and static clusters. We first evaluate the flatness c_i of the point p_i,

$$c_i = \left\|\sum_{j\in S, j\neq i}(r_j - r_i)\right\| / \left(|\boldsymbol{S}| \cdot \|r_i\|\right) \quad , \tag{14}$$

where $\boldsymbol{S}$ is a set of continuous points from the same row of the range image, points in $\boldsymbol{S}$ are uniformly distributed on both sides of p_i, $|\boldsymbol{S}|$ is the number of points in $\boldsymbol{S}$, and r is the range value of points.

In order to extract features uniformly from all directions, we divide the range image horizontally into N_{div} sub-images. Then, we sorted the points in each row of the sub-images according to the c_i value, and called the points where c is greater than the threshold value of c_{th} as edge features, and the points less than c_{th} as plane features. Then the N_{Fe} edge feature points with large c are selected from the static non-ground points. N_{Fp} plane feature points with small c are selected from all static points, and these feature points can be marked as ground points or segmentation points. Let $\boldsymbol{F_e}$ and $\boldsymbol{F_p}$ be sets of all edges and plane features of all sub-images respectively. Then, feature selection is performed again. We select the N_{Fe} edge features with the maximum c_i from $\boldsymbol{F_e}$. Similarly, we select the N_{Fp} plane features of minimum c_i from $\boldsymbol{F_p}$, which must be ground points. Define $\boldsymbol{f_e}$ and $\boldsymbol{f_p}$ as the set of all the edges and plane features of this process, we have $\boldsymbol{f_e} \subset \boldsymbol{F_e}$ and $\boldsymbol{f_p} \subset \boldsymbol{F_p}$. The number of columns per sub-image is 360°/(N_{div}* R_{hori}). $|\boldsymbol{S}|$, N_{div}, N_{fe}, N_{fp}, N_{Fe} and N_{Fp} are set to 6, 2, 4, 40 and 80 respectively. The schematic diagram of feature point extraction and association is shown in Fig. 5.

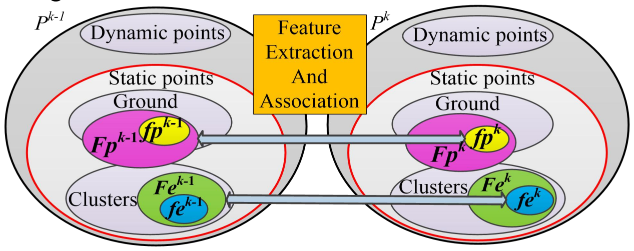

Fig. 5. The correspondence of feature points at adjacent frames. We divide the complete point cloud into static points and dynamic points. The static points include ground and non-ground objects, the green set includes the

blue one, and the pink set includes the yellow one. The edge feature points at scan k are matched in the edge feature points at scan k-1, and the plane feature points are similar.

However, we have noticed a key problem. The static clusters in the dynamic object module only represents that the current moment is static, and its previous moment may still be static or dynamic. We define the objects whose two adjacent states as inconsistent pseudo-static objects, and they are in the transition state of motion. Static objects can be divided into two types, we call objects that change from static to dynamic as the first type, and vice versa as the second type. For the first type, since it participated in feature extraction in the static state of the previous moment, but did not participate in current moment, it can be found corresponding at the previous moment. However, for the second type, the current features cannot be found corresponding to the previous moment, which will lead to a wrong match. To solve this problem, we make two optimizations. First, in the feature extraction module, double feature points are selected from the sub-image where the second kind of pseudo-static objects are located. The second is to use a RANSAC based pseudo-static feature point elimination method in LiDAR odometry module.

F. LiDAR Odometry

The input of LiDAR odometry module is the feature points extracted from current point cloud, and output is pose transform estimation. This module estimates the ego-motion between adjacent scans by matching feature points.

Fig. 5 shows feature association process. We need to find the corresponding features of the $\boldsymbol{f_e^k}$ and $\boldsymbol{f_p^k}$ points in $\boldsymbol{F_e^{k-1}}$ and $\boldsymbol{F_p^{k-1}}$. Then the constraint equation is constructed by using distance of feature point pairs, and the equation is usually solved by LM or other optimization algorithm [23]. A detailed process can be found in LeGo-LOAM [9].

LeGo-LOAM gives the feature point a weight s_i which is inversely proportional to the distance between LiDAR and the feature point. We also consider the influence of the point cloud holes in the background caused by the occlusion of dynamic objects on the stability of the system. Therefore, we reduce the weight of the feature points in a certain range of the yaw of the dynamic objects as:

$$s_i^{comp} = s_i \cdot (1 - r_{hori}/|\theta|), \quad (15)$$

where s_i^{comp} is the feature weight after compensation, r_{hori} is the horizontal resolution of LiDAR, θ is the minimum yaw difference between feature points and dynamic objects.

Aiming at the problem of the second type of pseudo-static objects, our inspiration comes from [24] and we propose a method of eliminating pseudo-static objects based on RANSAC. Firstly, the N_{pseudo} times RANSAC iteration step of are carried out, and part of the feature points are randomly selected to calculate a pose T_{pseudo}, define the matching points in accordance with this model are recorded as inliers, otherwise they are recorded as outliers. Then the point clouds are aligned according to T_{pseudo}, and the first β features with large matching error are removed. Finally, the remaining feature points are used for the next iteration. When the number of outliers and the number of remaining feature points are less than β, the iteration is terminated early. In this way, the influence of the second kind of pseudo-static objects on the result of feature extraction can be eliminated. We set N_{pseudo} to 5 and β to 10% empirically.

G. LiDAR Mapping

This module is consistent with LeGo-LOAM, please refer to [9]. It is worth noting that we have obtained the accurate odometry output here. In some offline methods or map based methods, the known odometry can be used to precisely align the point clouds of adjacent frames to accurately identify the dynamic objects and remove them from the known map, although it does not improve the accuracy of the odometry. The purpose of our method is to improve the accuracy and robustness of SLAM system, so we don't use this post-processing method to clean up the map.

IV. Experiments

All experiments were performed on a laptop equipped with an Intel (R) Core (TM) i7-4700 4 cores @2.4GHz CPU, and a NVIDIA Jetson TX2 with a CortexA57 CPU. The algorithms are implemented in C++ and run on robot operating system (ROS) [25]. The datasets include outdoor scenarios and KITTI odometry scenarios. We selected representative results to display in pictures, and the others are listed in the form of data. The first representative scenario is about 336 meters with 51 pedestrians and 3 vehicles collected by HDL-32 called Scenario 1 and the second one is about 4268 meters with 45 vehicles collected by HDL-64E called Scenario 2.

The selected experiments are to illustrate two key points. Firstly, the proposed dynamic object detection method using spatial and temporal information is better than the baseline methods using only space information. Secondly, the accuracy and robustness of SLAM system which combines the proposed dynamic object detection method are greatly improved in dynamic scenes. We claim that the proposed method is not limited to the baseline method LeGo-LOAM, and it is easy to be further expanded and developed. In addition, the proposed SLAM system is mainly applied to autonomous robots, intelligent transportation and automatic vehicles in dynamic urban scenarios, but it is not limited to these fields. We refer readers to see a video demonstration in Scenario 1 at: https://youtu.be/rlmHGi-mA9g, and a demo in Scenario 2 at: https://youtu.be/_R3wQqwKHFc, they illustrate the details of odometry and mapping comparison between our system and LeGo-LOAM.

A. Ablation Study w.r.t. Components and CNN

Three set of experiments on i7 platform are used to verify the performance improvement of the proposed method.

The first group of experiments were carried out in two representative dynamic scenes, which are Scenario 1(336 meters' parking scene with 51 pedestrians and 3 moving vehicles) and Scenario 2 (4268 meters' highway scene with 45 moving vehicles at high speed), to evaluate the influence on the results of the two components, which are weight compensation and eliminating pseudo-static objects. Pre-registration and the CNN-based dynamic object detection method using RangeNet++[18] are available, and all parameters are same. Table 3 shows the relative pose error under different configurations. We can see that all components have been proved to be effective in dynamic scenes. Among them, eliminating pseudo-static objects component improves the performance more obviously in dynamic scenes similar to Scenario 1 with stop-and-go objects, and weight compensation component improves the performance more obviously in Scenario 2, because we find that the features of highway scenes are sparse, Therefore, the holes generated by a large number of moving objects are very easy to affect the feature point matching of adjacent frames. Weight compensation can solve this problem effectively.

TABLE I
RPE Result of Ablation Study w.r.t. Two Proposed Components

Component		Scenario	
Weight compensation of feature points	Eliminating pseudo-static objects	1 (Urban)	2 (Highway)
w/o	w/o	0.72	3.15
w/o	w.	0.56	1.26
w.	w/o	0.60	1.02
w.	w.	0.55	0.98

The second set of experiments is carried out to illustrate the influence of the proposed traditional two frames method and two different CNN methods on the dynamic object detection accuracy, including the results with and without Pre-registration component. We use Intersection over Union (IoU) to measure the performance, where IoU of dynamic object is defined as:

$$IoU = TP/(TP+FP+FN)\,, \qquad (16)$$

where TP, FP, and FN represent the number of true positive, false positive, and false negative detected dynamic objects respectively. The results are shown in TABLE II.

TABLE II
Mean IoU of Ablation Study w.r.t. Different Dynamic Object Detection Methods (-: Not available, Fail: Fail or close to 0)

Input		Component	Method		
		Pre-registration	Traditional	RangeNet++ [18]	SalsaNext [19]
Single frame	Current frame	w.	-	34.0	53.7
		w/o	-	Fail	Fail
Two frames	Range image pair with m=1	w.	56.8	59.5	65.4
		w/o	12.7	22.1	20.9
Multi frames	Range image bag with m=2	w.	-	63.2	68.4
		w/o	-	21.9	21.0

In the above configuration, multi frames with component and SalsaNext achieves the best dynamic object detection performance because it combines spatial and temporal information, compared with the configurations that only uses spatial information or a small amount of temporal information. At the same time, we found that the pre-registration process is necessary for dynamic object detection. Because if the two point clouds are not aligned in advance, the difference process will not work correctly.

In the third set of experiments, we select the best performance configuration (proposed components + CNN based + SalsaNext) for in-depth experiment to explore the influence of m on IoU, and the results are shown in Fig. 6.

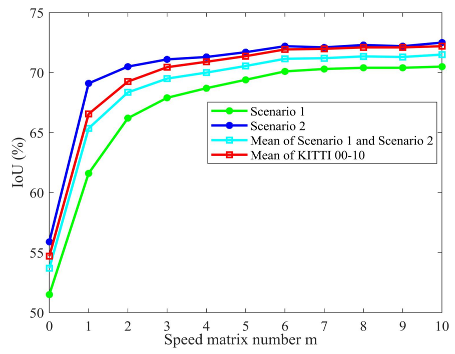

Fig. 6. Influence of speed matrix number m on IoU.

We can see that in the two scenes, m = 1 has made a great improvement in the dynamic object detection performance. At the same time, increasing m can further improve the dynamic object detection performance, but this improvement gradually flattens when m>5. Compared with methods that only use spatial information (m=0), the detection accuracy is improved by 35% at m=10 using SalsaNext and 86% at m=2 using RangeNet++. In addition, compared with the high-speed dynamic scene, for the scene with low-speed dynamic objects, the performance improvement is relatively obvious by increasing m. This result is in line with the expectation, because the motion state of low-speed objects is relatively not obvious and needs more observation.

In the subsequent experiments, we will use the best performing configuration, namely, the proposed components plus SalsaNext with m=6.

B. Feature Extraction Comparison

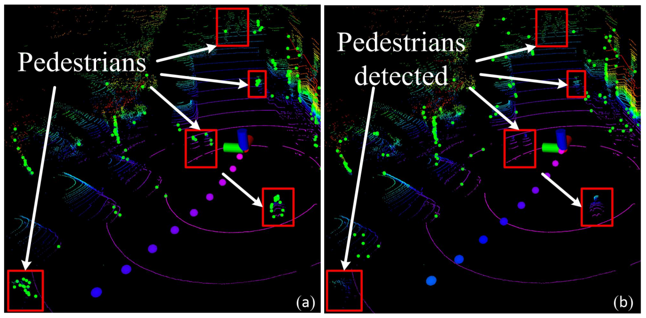

Fig. 7. Dynamic object detection and feature extraction results of a scan between the two methods were compared. The color of the point represents the elevation change, and the green points represent edge feature points. We use the red box to highlight the key points. (a) LeGo-LOAM extracts features from all points. (b) We only extract stable features from static objects.

Fig. 7 shows a scan (in color points) and the edge feature points (in green) in Scenario 1 on i7 platform, including six moving pedestrians (circled in red box), where the red cylinder indicates the moving direction of LiDAR. Fig. 7(a) shows the wrong feature extraction result of LeGo-LOAM. It can be seen that edge feature points are distributed on all points including unstable dynamic points belong to the pedestrians. Fig. 7(b) is our result, which is more reasonable. We detect dynamic objects and avoid extracting feature points from them, so as to avoid the wrong odometry and mapping results caused by mismatch of unstable feature points. Feature extraction results of the two methods are recorded at TABLE III. We define the feature points on dynamic object as ineffective feature points and the rest as effective ones. Note that the real dynamic points are found by coarse registration plus fine ICP registration. Firstly, we record the number of feature points distributed on the object in each scan, then count the points distributed on the static objects into the effective feature point set, and finally calculate the ratio of the number of effective feature points to the total corresponding feature points. Each result is the average of ten experiments. There are 51 pedestrians and 3 moving vehicles in Scenario 1, so the effective feature points ratio of LeGo-LOAM is very low, which show that our method has the advantage of extracting effective features in dynamic environments.

We notice that the effective point ratio of all types of feature points in our method is higher than LeGo-LOAM, where the ratio of $\boldsymbol{F_e}$, $\boldsymbol{f_e}$ and $\boldsymbol{F_p}$ and is significantly higher than the baseline, while the ratio of $\boldsymbol{f_p}$ is slightly higher, because $\boldsymbol{f_p}$ is only extracted from ground points which are rarely affected by dynamic objects. In addition, the ratio of our method is close but not reached to 1, it is caused by primary factor of defect of the ground extraction algorithm in the segmentation module, that is, a small number of ground points are wrongly distributed on objects, while secondary factor is the missed detection of dynamic objects.

TABLE III
EFFECTIVE FEATURE POINTS EXTRACTION COMPARISON BETWEEN LEGO-LOAM AND OURS

Scenario	$\boldsymbol{F_e}$		$\boldsymbol{f_e}$		$\boldsymbol{F_p}$		$\boldsymbol{f_p}$	
	Ours	LeGo LOAM	Ours	LeGo LOAM	Ours	LeGo LOAM	Ours	LeGo LOAM
1	0.96	0.47	0.97	0.45	0.97	0.75	0.97	0.91
2	0.95	0.62	0.97	0.58	0.97	0.51	0.98	0.89

C. Odometry Comparison

In order to further evaluate the performance of LiDAR odometry, we compare our proposed method with the most advanced ones. For LOAM [3], SuMa [14], SuMa++ [27], S4- SLAM [28], IMLS [29], we directly report their results in the paper. Among them, S4-SLAM, IMLS cannot achieve real-time performance, we only take their results as reference. For the open source LeGo-LOAM, because their authors did not publish the test results on the KITTI dataset, we use their code for experiments. At the same time, we mainly compare with LeGo-LOAM on the same platform to prove the effectiveness of our proposed dynamic object detection method for improving the accuracy and robustness of the odometry. Again, we emphasize that our proposed dynamic object detection method can be easily embedded into other SLAM systems to improve their odometry accuracy and robustness in dynamic environments.

Fig. 8 shows the odometry comparison results on i7 platform in Scenario 1 between LeGo-LOAM and our method. The red one is the result of LeGo-LOAM and the blue one is ours. It seems they have little difference in the global perspective. However, as shown in the green box in the center of the image, we choose the most illustrative location (upper left corner) to show the contrast in detail. As can be seen from the green box, the red curve has a distinct jagged edge, which is exactly the wrong odometry estimation caused by a large number of moving pedestrians, while the blue one has no such problem, which is enough to illustrate the advantage of our method in the odometry estimation. This is because we detect dynamic objects and avoid using them for registration, therefore, we can avoid the influence of unstable points on odometry. We can see the process of SLAM and the impact of dynamic objects on SLAM through the video demonstration of Scenario 1.

Fig. 9 shows odometry comparison results on i7 platform in Scenario 2, we can see from Fig. 9 (a) that the accuracy can still be maintained in the long-distance loop-contained scenes, and it is similar for other KITTI sequences. Fig. 10 show odometry and mapping comparison results between the two methods. The wrong map in Fig. 10 (b) is caused by an unstable dynamic vehicles driving at high speed, but our method has no such problem because we detect the dynamic objects and avoid extracting features from them, and we can see clear differences between the red and green boxes. And results on KITTI sequences 00-10 are showed in TABLE IV.

TABLE IV
RELATIVE POSE ERROR RESULTS ON KITTI DATASETS WITH DYNAMIC OBJECTS BETWEEN DIFFERENT METHODS (BOLD: THE BEST AMONG REAL-TIME METHODS)

Sequence	Distance (m)	Environment	Dynamic Objects Number		Method (*: with loop closure) & Type of removed objects & Equipment (CPU/ cores/ clock rate (GHz), -: unknown) & Run time (s)											
			Vehicle	Pedestrian	Ours_ CNN*		LeGo-LOAM* [9]		LOAM (Open Source) [30]		LOAM [3]	SuMa* [14]	SuMa_ no movable* [14]	SuMa++* [27]	S4-SLAM* [28]	IMLS-SLAM [29]
					moving		small		none		none	none	movable	moving	small	small
					TX2	i7	TX2	i7	TX2	i7	-/ 2/2.5	i7-6700/ 4/3.4	i7-6700/ 4/3.4	W-2123/ 8/3.6	i5-3210 2/3.0	-/ 1/4.0
					0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	1.25
00*	3714	Urban	2	9	0.90	0.71	1.16	0.73	1.78	1.25	0.78	0.68	58.0	**0.64**	0.62	0.50
01	4268	Highway	45	0	1.19	**0.92**	5.47	2.98	2.75	2.03	1.43	1.70	1.70	1.60	1.11	0.82
02*	5075	Urban Country	6	1	1.22	0.95	1.24	0.95	1.80	1.37	**0.92**	1.20	63.0	1.00	1.63	0.53
03	563	Country	1	0	0.97	0.95	0.97	0.96	1.49	1.26	0.86	0.74	0.67	**0.67**	0.82	0.68
04	397	Country	14	0	0.88	0.57	1.03	0.66	1.55	1.05	0.71	0.44	0.37	**0.37**	0.95	0.33
05*	2223	Urban	6	4	0.69	0.45	0.72	0.51	1.24	1.02	0.57	0.43	36.0	**0.40**	0.50	0.32
06*	1239	Urban	1	1	1.10	0.66	1.09	0.66	1.58	1.13	0.65	0.54	0.47	**0.46**	0.65	0.33
07*	695	Urban	7	3	0.75	0.64	0.79	0.68	1.15	1.08	0.63	0.74	0.34	**0.34**	0.60	0.33
08*	3225	Urban Country	8	22	1.44	**0.98**	2.10	1.16	2.52	1.71	1.12	1.20	32.0	1.10	1.33	0.80
09*	1717	Urban Country	8	1	0.99	0.69	1.02	0.74	1.61	1.17	0.77	0.62	45.0	**0.47**	1.05	0.53
10	919	Urban Country	5	0	1.19	0.80	1.30	0.87	1.75	1.29	0.79	0.72	19.0	**0.66**	0.96	0.55

It can be seen that in the dynamic scenarios, the relative pose error performance of our method on TX2 and i7 laptop is smaller than open-source LeGo-LOAM and LOAM, and this improvement is more obvious in scenes with a large number of dynamic objects such as sequence 01 and 08. This is because we detect dynamic objects and refuse to use them for registration, we can therefore avoid the impact of unstable points on feature matching process, so as to improve the accuracy and robustness of odometry and mapping. Besides, except on 01 sequence, the results of LeGo-LOAM are better than that of LOAM [25]. This is because we observed that there are only fences, vehicles and a large number of trees in the highway environment of sequence 01, therefore, it is almost impossible to extract effective edge feature points using LeGo-LOAM, which leads to the mismatching of edge feature points between adjacent frames. Compared with LeGo-LOAM, the problem rarely occurs in LOAM which extracts feature points from whole current point cloud. Of course, LOAM needs more computation, therefore, it cannot achieve real-time performance on 64-beams KITTI dataset. Specifically, many of the LiDAR scans are skipped, so our test using the open-source LOAM did not achieve real-time performance and the similar accuracy as mentioned in [3].

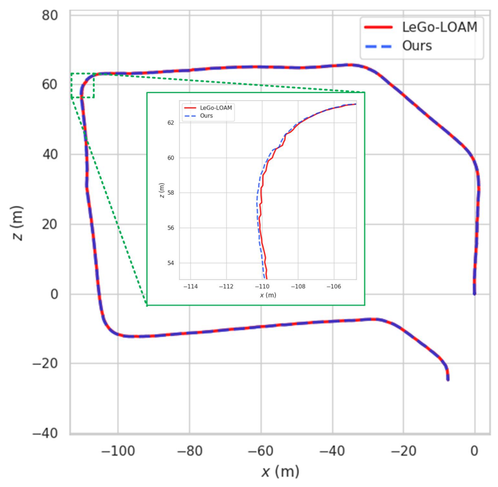

Fig. 8. Odometry comparison results in Scenario 1 between the two methods, the green box is enlarged to highlight the differences, where our odometry curve is smoother than that of LeGo-LOAM because we detect dynamic objects and avoid using them for registration.

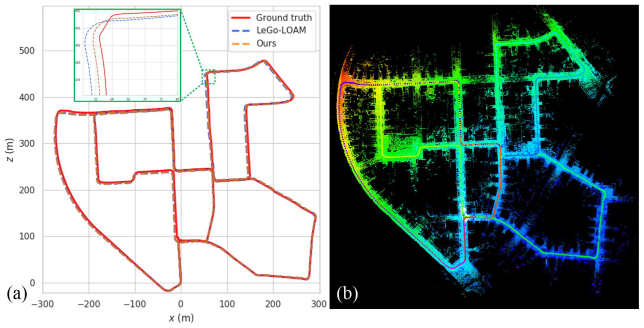

Fig. 9. Results on KITTI 00. (a) Odometry comparison results. (b) Mapping results, where color variation represents elevation change.

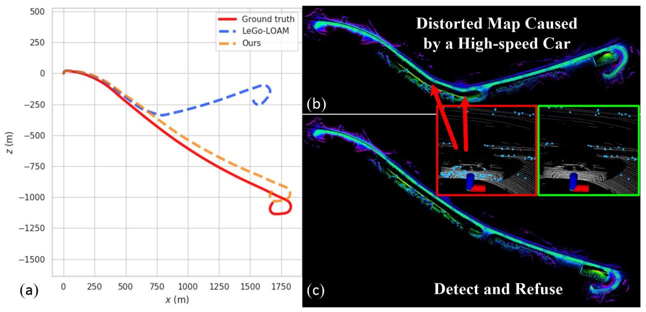

Fig. 10. Results in Scenario 2 (KITTI 01, a 4268m highway scenario with 45 moving vehicles at high speed). (a) Odometry comparison results. (b) Wrong mapping results of LeGo-LOAM, where the red arrows highlight the distorted parts caused by the feature points (colored in blue) extracted from a vehicle at high speed. (c) Our mapping results. Compared with the red one, feature extraction results shown in the green box are more reasonable, where we detect dynamic objects and avoid extracting feature points from them for registration, therefore, our map will not be affected by dynamic objects (vehicles at high speed).

D. Mapping Comparison

The results of representative scenarios show that LeGo-LOAM will produce wrong mapping results because they do not consider dynamic objects, as shown in Fig. 11 and Fig. 12. Fig. 11 is the mapping result of Scenario 1 on i7 platform. The LiDAR begins to move from the lower right and moves counterclockwise to the vicinity of the starting point. Fig. 11 (a) is the LeGo-LOAM mapping result. The residual shadow in the yellow solid line box are the points left by pedestrians, which indicate the wrong mapping result, while the residual shadow in the white solid line box are the points left by vehicles. Fig. 11 (b) is the mapping result of our method, in which the dotted line box is in sharp contrast to that in Fig. 11 (a), which indicate the advantages of our method w.r.t. mapping. fig8shows the details of the mapping result of scene 1. Fig. 12 (a), (b), and (c) are the result of incorrect mapping caused by incorrect feature extraction. Dynamic pedestrians and vehicles participate in feature extraction and are added to the map, leaving obvious residual shadows. Among them, Fig. 12 (a) is the enlarged result on the top of the map, and the residual shadow is the movement track left by at least 3 pedestrians moving forward. Fig. 12 (b) is a local enlarged image on the left of the map, the residual shadow is the movement track left by an upward moving vehicle. Fig. 12 (c) is a local enlarged image of the lower right part of the map, the residual shadow is left by a moving vehicle. Different from LeGo-LOAM, we first separate the dynamic and static objects and focus on extracting features from the static objects, so we have obvious advantages in building dynamic scenes. However, there are still small residual shadows in the map for two reasons. Firstly, dynamic object detection is based on segmentation, while the range image segmentation method is rough, because it ignores some 3D information. Moreover, the experimental results show that the point cloud which we think belongs to one object is often divided into several objects (e.g. vehicles are usually divided into several parts), which may be obstructed by the noise of LiDAR or foreground objects. Secondly, there are drifts in the results of LiDAR odometry, and the accuracy of range image construction will be affected by the errors of point cloud projection and ego-motion detection, which will lead to the wrong speed estimation of real dynamic objects.

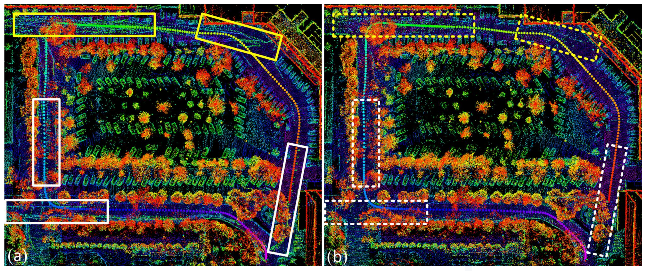

Fig. 11. The complete mapping result in Scenario 1. The color change of map represents the elevation change, with the red track being the starting point of the LiDAR motion and the purple track being the end point. (a) The result of LeGo-LOAM, we can clearly see the shadow left by the moving objects. (b) The result of our method, we get a clean map by detecting dynamic objects and avoiding extracting feature points and mapping them.

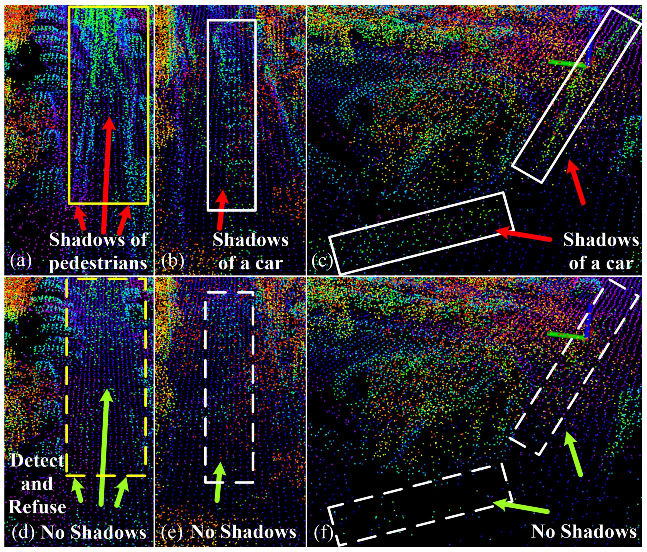

Fig. 12. Detail comparison of the result of the two methods in Scenario 1. (a), (b), and (c) are the results of LeGo-LOAM. (d), (e), and (f) are the results of our method. We add arrows to highlight the differences, where red arrows highlight the shadows (wrong mapping results) left by pedestrians and vehicles, while the green arrows show the result of detecting dynamic objects and refusing to use them for SLAM.

E. Runtime Comparison

Average running times for each module between our method and LeGo-LOAM are shown in TABLE V. Compared with overall time consumption, dynamic object detection module takes relatively less time, so the runtime of our system is basically same as that of LeGo-LOAM. Our system still shows the real-time performance. In addition, we show the specific time consumption of the proposed components in TABLE VI.

TABLE V
RUNTIME COMPARISON OF MODULES BETWEEN LEGO-LOAM AND OURS (ms)

Equipment	Scenario	Segmentation		Dynamic Object Detection		Feature Extraction		LiDAR Odometry		LiDAR Mapping	
		Ours	LeGo LOAM	Ours	LeGo LOAM	Ours	LeGo LOAM	Ours	LeGo LOAM	Ours	LeGo LOAM
TX2	1	26.4	26.2	52.6	N/A	12.5	13.9	27.9	25.4	295.2	292.7
	2	35.0	34.7	54.3	N/A	14.3	16.0	31.2	27.1	307.0	305.5
i7	1	15.9	17.0	35.5	N/A	6.3	6.5	15.9	13.5	168.3	166.4
	2	20.2	19.8	37.2	N/A	8.5	8.9	17.7	13.8	170.8	169.1

TABLE VI
RUNTIME OF DIFFERENT COMPONENTS (ms)

Equipment	Scenario	Pre-registration		Weight compensation of feature points	Eliminating pseudo-static objects (N_{pseudo}=5)
		PCA	RANSAC (N_{coarse}=5)		
TX2	1	14.8	4.4	0.2	2.3
	2	16.6	5.2	0.5	3.7
i7	1	11.0	3.3	0.2	2.2
	2	12.5	3.7	0.4	3.5

V. CONCLUSION

We propose a fast dynamic object detection method based on the difference between adjacent frames, which is implemented by CNN-based method and traditional method respectively. The experimental results show that the accuracy and robustness of odometry and mapping of the proposed SLAM system is greatly improved in dynamic environments, which benefits from stable and fine features extracted from static objects. Our system is real-time and can meet the needs of autonomous robot systems and intelligent transportation applications in urban environments.

The future works include further improving the performance of the system in highly complex and dynamic scenes by improving the accuracy of dynamic object detection module, because we found our system will still generate wrong results when there are many and dense moving objects around the LiDAR. We notice that the recent registration method TEASER [32], which can quickly and stably register point cloud scans with a large number of noise points, may be suitable for pre-registration or even directly estimating the pose transformation in extremely dynamic environments. Besides, we will also try to deploy the SLAM system on multiple base stations to make up for the deficiency of a single base station seriously blocked by mobile objects by cooperative tasks, in addition, some of the solutions to the robot kidnapping problem may be the feasible solution.

Another problem worthy of discussion is how to deal with objects that are temporarily stationary but may start to move at any time, such as vehicles or pedestrians that are temporarily stationary. The possible and inevitable solution is to extract more semantic information from the scene to make decisions. For example, we should remove the vehicle waiting for the traffic lights in the middle of the road. Therefore, we will further combine the semantic information and motion information of the scene, which will provide a new idea for the construction of semantic map in dynamic scenes.

REFERENCES

[1] J. Zhang, M. Henein, R. E. Mahony, and V. Ila, "VDO-SLAM: A Visual Dynamic Object-aware SLAM System," in *International Journal of Robotics Research*, 2020. [Online]. Available: https://arxiv.org/abs/2005.11052.
[2] J. Zhang and S. Singh, "LOAM: LiDAR Odometry and Mapping in Real time," in *Proceedings of Robotics: Science and Systems*, 2014.
[3] J. Zhang and S. Singh, "Low-drift and real-time LiDAR odometry and mapping," in *Autonomous Robots*, 2017, vol. 41, no. 2, pp. 401-416.
[4] J. Zhang and S. Singh, "Visual-LiDAR odometry and mapping: low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, July 2015, pp. 2174-2181.

[5] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, D. Rus. "LIO-SAM: Tightly-coupled LiDAR Inertial Odometry via Smoothing and Mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 5135-5142.

[6] K. Koide, J. Miura, and E. Menegatti, "A Portable 3D LiDAR-based System for Long-term and Wide-area People Behavior Measurement," in *International Journal of Advanced Robotic Systems*, 2019, vol. 16, no. 2.

[7] M. R. U. Saputra, A. Markham, N. Trigoni, "Visual SLAM and Structure from Motion in Dynamic Environments: A Survey," in *ACM Computing Surveys*, vol.51, no.2, pp. 1557-7341.

[8] D. Rozenberszki and A. L. Majdik, "LOL: LiDAR-only Odometry and Localization in 3D point cloud maps," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 4379-4385.

[9] T. Shan and B. Englot, "LeGo-LOAM: Lightweight and Ground-Optimized LiDAR Odometry and Mapping on Variable Terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 4758-4765.

[10] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age," in *IEEE Trans. on Robotics (TRO)*, Dec. 2016, vol. 32, no. 6, pp. 1309-1332.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2012, pp. 3354-3361.

[12] R. Kiimmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "A navigation system for robots operating in crowded urban environments," in *2013 IEEE International Conference on Robotics and Automation(ICRA)*, Karlsruhe, Germany, 2013, pp. 3225-3232.

[13] X.Weng, J. Wang, D. Held, K. Kitani, "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 7934-7943.

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

[15] S. A. Baur, F. Moosmann, S. Wirges and C. B. Rist, "Real-time 3D LiDAR Flow for Autonomous Vehicles," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, June 2019, pp. 1288-1295.

[16] R. Battrawy, R. Schuster, O. Wasenmüller, Q. Rao and D. Stricker, "LiDAR-Flow: Dense Scene Flow Estimation from Sparse LiDAR and Stereo Images," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 7762-7769.

[17] P. Wu, S. Chen, and D. Metaxas, "MotionNet: Joint Perception and Motion Prediction for Autonomous Driving Based on Bird's Eye View Maps," in *2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11382-11392.

[18] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 4213-4220.

[19] T. Cortinhal, G. Tzelepis, and E.E. Aksoy, "SalsaNext: Fast,Uncertainty-Aware Semantic Segmentation of LiDAR Point Clouds," in *2020 IEEE Vehicles Symposium (IV)*, 2020.

[20] A. F. Yandex, A. R. Yandex, and V. M. Google, "Any Motion Detector: Learning Class-Agnostic Scene Dynamics from a Sequence of LiDAR Point Clouds," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 9498-9504.

[21] I. Bogoslavskyi and C. Stachniss, "Fast Range Image-based Segmentation of Sparse 3D LiDAR Scans for Online Operation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 163-169.

[22] B. Yang, Z. Dong, F. Liang, and Y. Liu, "Automatic registration of large-scale urban scene point clouds based on semantic feature points," in *ISPRS Journal of Photogrammetry and Remote Sensing*, 2016, vol. 113, pp. 43-58.

[23] J. Pan, N. Liu, S. Chu, and T. Lai, "An Efficient Surrogate-Assisted Hybrid Optimization Algorithm for Expensive Optimization Problems," in *Information Sciences*, 2021, vol. 561, pp. 304-325.

[24] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 3126-3131.

[25] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An Open-source Robot Operating System," in *IEEE ICRA Workshop on Open Source Software*, 2009.

[26] S. Zhao, Z. Fang, H. Li, and S. Scherer, "A robust laser-inertial odometry and mapping method for large-scale highway environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 1285–1292.

[27] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, C. Stachniss, "SuMa++: Efficient LiDAR-based Semantic SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 4530-4537.

[28] B. Zhou, Y. He, K. Qian, X. Ma, and X. Li, "S4-SLAM: A real-time 3D LIDAR SLAM system for ground/watersurface multi-scene outdoor applications," in *Autonomous Robots*, 2021, pp. 77-98.

[29] J. Deschaud, "IMLS-SLAM: scan-to-model matching based on 3D data," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2480–2485.

[30] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, "LiTAMIN2: Ultra Light LiDAR-based SLAM using Geometric Approximation applied with KL-Divergence," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 2021.

[31] https://github.com/laboshinl/loam_velodyne.

[32] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration," in *IEEE Transaction on Robotics*, 2020.

Wenbo Liu is currently pursuing his master degree in control science and engineering at Xidian University, Xi'an, China. His current research interests include 3D point cloud processing and 3D LiDAR SLAM.

Wei Sun received the B.S. degree in measuring and control technology and the Ph.D. degree in circuit and system from Xidian University, Xi'an, China, in 2002 and 2009, respectively. He is currently a professor with the School of Aerospace Science and Technology, Xidian University. His current research interests include multi-UAV systems, visual information perception, pattern recognition, and embedded video systems.

Yi Liu is currently pursuing his master degree in space engineering at Xidian University, Xi'an, China. His current research interests include 3D point cloud registration and image processing.