4-D SLAM: An Efficient Dynamic Bayes Network-Based Approach for Dynamic Scene Understanding

ZHONGLI WANG1,2, (Member, IEEE), WENYI LI1, YAN SHEN1, (Member, IEEE), AND BAIGEN CAI1,2, (Senior Member, IEEE)

1School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China
2Beijing Engineering Research Center of EMC and GNSS Technology for Rail Transportation, Beijing 100044, China

Corresponding author: Zhongli Wang (zlwang@bjtu.edu.cn)

This work was supported in part by the Natural Science Foundation of China under Grant 61573057 and Grant 61702032.

ABSTRACT Most of the existing simultaneous localization and mapping (SLAM) methods are based on the static environment assumption. The presence of moving objects in the scene will lead to much uncertainty in SLAM results, which also hinders the loop-closure detection (LCD). Although moving object tracking (MOT) is necessary for planning and decisions, it is often accomplished separately. To jointly solve SLAM and MOT for complex urban driving scenarios, this paper presents a high performance method named 4D-SLAM based on the fusion of LiDAR and IMU. The integration of SLAM and MOT is formulated as a joint posterior probability problem based on a dynamic Bayesian network (DBN) and is implemented with the following four sequential stages: preprocessing, moving object detection and tracking, odometry estimation and mapping. In the preprocessing stage, the motion distortion uncertainty caused by LiDAR scanning is first compensated for, and the initial LiDAR motion is estimated. In the moving object detection and tracking stage, we exploit a CNN-based segmentation network to detect the potential moving objects first, then the states of the potential moving objects are optimized by an unscented Kalman filter (UKF). In the odometry estimation stage, the distinctive planar and edge features extracted from a static background point cloud are used for the odometry estimation, and a two-step Levenberg-Marquardt optimization method is adopted to solve the 6DOF pose across consecutive scans. In the mapping stage, the mapping based on the pose estimation result and LCD are realized, and graph-based global optimization is exploited to further improve the map consistency for large scale environment. The comprehensive experiments with the open source dataset KITTI and the data collected by us show that the presented method not only outperforms the SOTA SLAM methods in terms of trajectory and mapping accuracy but can also detect and track moving objects efficiently.

INDEX TERMS Point cloud segmentation, moving object detection and tracking, simultaneous localization and mapping (SLAM), dynamic Bayesian network (DBN).

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) is a fundamental prerequisite for an autonomous mobile robot to navigate an unknown indoor/outdoor environment [1], which is also widely used in virtual reality, autonomous driving, etc. Great efforts have been devoted to achieving real-time SLAM with vision-based [2], [3] or LiDAR-based [4] methods.

The vision-based methods have advantages in terms of LCD and low cost, but their sensitivities to illumination and viewpoint changes limit their application. The LiDAR-based method can even work at night, and its high accuracy and reliability make it a main perception sensor in self-driving vehicles. Thus, this paper focuses on using 3-D LiDAR for scene understanding of autonomous driving.

Although SLAM issues have been studied for many years, some challenges for dynamic large scale environment still exist. For example, most of the proposed methods are based
on the implicit assumption that the surrounding environment is stationary. This assumption is often in conflict with real applications. Taking the self-driving vehicle as example, when it runs on an urban street, there are always other moving cars, pedestrians or cyclists around it. The number and types of moving objects on the road are constantly changing and may even be crowded. To navigate in such a dynamic environment, the self-driving vehicle must be able to detect and track the targets around it in real time while it estimates its own location and builds the map. As discussed in reference [25], the following three main issues regarding SLAM in dynamic environments were identified: how to perform SLAM based on a static background, which we call static SLAM; how to detect and track moving objects; and how to jointly implement static SLAM and motion segmentation. Despite the existing remarkable achievements, most SLAM methods based on the static scene assumption would be jeopardized when faced with moving objects. Additionally, LiDAR normally works at a low spinning rate, and the motion distortion caused by the vehicle moving should be compensated for [9]–[13]. Both the moving objects in a dynamic environment and the motion distortion issue bring large challenges for the LiDAR-based SLAM method.

To alleviate the motion distortion problem, some correction methods have been presented. According to the ego-motion estimation method, these methods can be classified into two groups, ICP-based [9], [10] and external sensor-aided [3], [11]–[13]. The ICP-based method estimates the ego-motion between two scan frames with ICP first, then every scan block in a frame is realigned according to the estimated motion. For example, an ICP-based velocity estimation is presented and followed by a distortion compensation step. Since the ICP-based motion estimation is implicitly based on the assumption of a static environment, this method is limited in a dynamic environment. On the other hand, the external sensor-aided method estimates the ego-motion with the aids of sensors, such as IMU [13] or odometry. Since motion estimation does not depend on the observation of the environment, it is more robust than the ICP-based method in a highly dynamic environment. However, the motion of the external sensor should be aligned to the LiDAR frame with the geometric relationship between the two sensors. Le et al [14] proposed an extrinsic calibration framework for LiDAR-IMU fusion. Ji et al [3], [15] presented a robust camera-IMU fusion pipeline to estimate ego-motion and built a coherent map in various challenging environments. In this paper, we use an IMU sensor for the initial ego-motion estimation, and the principle of motion distortion compensation will be detailed in section 4.

Some methods proposed achieve a robust SLAM result in a dynamic environment, but most of them take moving objects as outliers and try to filter them out, rather than solving the SLAM and MOT simultaneously. For example, Sun [52]–[54] proposed the moving object detection and motion removal approaches with RGB-D SLAM. Walcott [16] considered the time factor in the map construction process and maintained a precise map in dynamic environments. Wang [17] proposed an EKF framework for IMU and 2D LiDAR data fusion, which was also used to estimate pose in a low-texture dynamic environment. Both methods are mainly designed for indoor low dynamic environments and are not suitable for outdoor large-scale highly dynamic environments. François [18] used a Bayesian model to assess whether a point is static or dynamic. Wang [19] and Bahraini [20] used an improved RANSAC algorithm to track moving objects, but it may lose the ability to track static objects in scenes where moving objects are dense. Jiang [21] presented a LiDAR-camera SLAM system, which used a sparse subspace clustering-based motion segmentation method to build a static map in a dynamic environment. However, this method cannot run in real-time and requires sufficient illumination in the environment.

Recently, Zhang et al [15] and Shan et al [22] applied point cloud segmentation to filter out noise, and only the distinct planar and edge features extracted from the original point cloud are used to estimate different parts of the 6DOF transformation. Their method achieved high performance on pose estimation and mapping and became the SOTA method, though they did not explicitly filter out moving objects. It works well with lower dynamic environments or fewer moving objects. Chen et al [23] proposed a semantic SLAM method - SuMa++. This method exploited the semantic consistencies between the scans and the map to filter out dynamic objects with a RangeNet++ network [24], which could provide higher-level constraints during the ICP process and further achieve robust pose estimation and semantic maps. However, it cannot solve the MOT issue.

Generally, SLAM and MOT are two inseparable tasks for scene understanding in a dynamic environment, which are the key prerequisites for making a robot truly autonomous and need to be completed synchronously. Theoretically, MOT can eliminate the influence of moving objects on SLAM, and the robustness and accuracy of SLAM can be improved. Meanwhile, accurate SLAM results are conducive to the MOT solution. Moreover, for a semantic-based SLAM and MOT in 3-D space, both of them share some common fundamental tasks, such as object detection, data association, etc. The integration of SLAM and MOT can avoid redundant computation and save the computing resources for many resource-limited applications. The traditional SLAM based on the static assumption is modeled in 3-D space, and the rigid moving object tracking addresses the environment changes with time; in this paper, we call the integration of SLAM and MOT as 4-D SLAM (3-D+t-time) for a dynamic environment. To address the SLAM with detection and tracking of moving objects problem, some pioneer works include Wang [11], [12], [26]. They decomposed the problem into two separate estimators, and two separate posteriors were maintained for stationary and moving objects.

To jointly solve the SLAM and MOT problems by the fusion of 3-D LiDAR and IMU, we present a 4D-SLAM approach to improve the accuracy of static SLAM and the
efficiency of tracking in a challenging dynamic environment. The MOT and SLAM problems are formulated as a joint posterior probability based on a dynamic Bayesian network (DBN), which will be discussed in section 3. To improve the accuracy, several measurements in different processing stages are elaborately considered. In the preprocessing stage, the motion distortion of the point cloud is corrected with the aid of IMU integration; then, the entire point cloud is classified into two groups, i.e., static background and moving objects, according to a fully convolutional network (FCN) based on potential moving object detection and tracking. Motivated by the works of LeGO-LOAM [15], [22], only the discriminative features extracted from a static background are exploited to estimate the initial pose, which will be further optimized in the mapping stage. A mapping module performs local and global optimization according to the availability of loop-closure detection (LCD) results. When a loop-closure is detected, either global optimization will be performed or a local submap is optimized. In the presented approach, other than discarded or filtered out the moving objects, the potential moving objects resulted from FCN segmentation are tracked with a nonlinear UKF. This allows us to generate high-quality semantic maps, while at the same time improving the accuracy of the localization. The presented approach is a key part of scene understanding for self-driving techniques. To our knowledge, this is the first work to implement the 3-D SLAM and MOT simultaneously.

The main contributions of this work are as follows:

1) A robust and efficient 4D-SLAM approach based on the fusion of LiDAR and IMU is presented. The integration of 3-D SLAM and MOT is formulated as the joint posterior probability with a dynamic Bayesian network (DBN). The implementation includes four modules, i.e., preprocessing, dynamic object detection and tracking, fusion based odometry and mapping. The four modules work logically in a sequential mode.

2) An efficient MOT approach is presented, which exploits deep-learning-based semantic segmentation to detect potential moving objects and a nonlinear motion estimator based on UKF for tracking. Assuming the process has the Markov property, the object tracking is considered a Joint Probabilistic Data Association Filter (JPDAF). Because it is efficient and theoretically coherent with the framework, it is convenient to integrate with SLAM and they benefit from each other.

3) Comprehensive experiments are conducted on the open dataset and the dataset collected by our self-driving platform, which shows that the presented method can work well in a complex dynamic environment and outperforms the SOTA methods.

The rest of this paper is organized as follows. The main related works are summarized in section 2. The DBN-based formulation of the 4D-SLAM problem is reviewed in section 3. The implementation is outlined in section 4. Section 5 presents the experiment results. Finally, conclusions are made in section 6.

II. RELATED WORKS

The presented method draws on the recent successes of LiDAR-based SLAM in a dynamic environment [15], [22], [23], [26], CNN-based object segmentation and tracking [26], [28]–[35]. In most of the existing SLAM methods, moving objects are viewed as outliers and the methods try to filter them out, while few of them address the integration of SLAM and MOT. Recently, deep learning-based object detection with point cloud has become an active research topic, and many excellent works have been presented; however, most of them are only focused on detection or segmentation tasks. We now unify SLAM and MOT into a common framework, aim to improve the accuracy of localization and the consistency of the mapping, while simultaneously tracking the moving objects, which is a key part of scene understanding for self-driving.

A. SLAM IN DYNAMIC ENVIRONMENTS

A few of studies have been published on jointly solving the static SLAM and detection and tracking of moving objects. Wang [11], [12], [26] has performed some pioneering works that employed a 2D laser scanner to track moving objects using a Bayesian approach; his work mainly focused on representation of the world and data association. Wolf et al [36] proposed to solve SLAM in dynamic environments by maintaining one landmark map for localization purposes and two occupancy grid maps, in which one is for static objects and another is for moving objects. This method is limited to moderately dynamic indoor environments. Huang et al [37] formulated the problem with two interdependent parts, a hierarchical hybrid method to solve SLAM and a straightforward Nearest Neighborhood (NN) algorithm to address the MOT problem, an Enhanced Fuzzy Clustering (EFC) method to segment 2D range images and reliably group objects. Davide et al [38] studied the classification of motion features and achieved accurate tracking of a single object based on MonoSLAM. Their method fails easy when the target is moving at low speed and cannot tracking multiple moving objects. Lin [39] proposed a SLAM and MOT method based on a stereo camera. Shen [40] integrated the instance inference model and temporal feature correlation into a tightly coupled optimization framework based on semantic features and geometric information. These methods are based on a monocular or stereo camera and are easily affected by many factors, such as object occlusion, illumination changes, scene texture, etc. However, these works showed that SLAM and MOT are mutually beneficial to each other; SLAM provides a more accurate pose estimate and a world map, which is used by MOT to detect moving objects more reliably; MOT can detect and predict the locations of the moving objects so that SLAM can filter out moving objects and obtain more accurate localization and a more accurate surrounding map. Based on a scan-to-model matching framework, a low-drift SLAM algorithm was proposed in reference [41]. To remove the noise data and moving objects, they first filter out the point cloud located in the ground plane, then cluster on the left point.
cloud any objects which have sizes less than 14 m$^2$ 14 m$^4$ m, which after being clustered will be removed as outliers; thus, the moving objects on the road will be filtered out.

**B. MOVING OBJECT SEGMENTATION AND DETECTION WITH A POINT CLOUD**

The existing methods of moving object segmentation can be classified into two groups according to data representation and traditional and semantic segmentation. Some traditional methods include statistical model selection [42], low-dimensional subspace clustering [43], and geometric constraint-based [44]. The stability and accuracy of these methods are not satisfactory, and these methods cannot be run online in large-scale scenarios.

Currently, the most widely used method is the semantic segmentation method, which usually adopts a deep learning network; the accuracy and stability of this method are greatly improved compared to those of the traditional methods. These methods can be further divided into projection-based and point-based methods. The core idea of a projection-based method is projecting the 3-D information to a 2D image and making use of the achievement in 2D CNN-based detection, which can be further grouped into the following three categories: multiview representation, spherical representation, and voxel grid representation. DP3D [28] first projected a 3-D point cloud onto 2D planes from multiple virtual camera views. Tatarchenko et al. [29] introduced tangent convolutions for dense point cloud segmentation. SqueezeSeg et al. [30] proposed an end-to-end network based on SqueezeNet and the Conditional Random Field (CRF), which achieves real-time and fast semantic segmentation. Based on Squeezeseg, Milioto et al. [24] proposed RangeNet++, which established a labeled dataset [31] and achieved real-time semantic segmentation of 3-D point clouds. However, these intermediate representations inevitably bring several drawbacks, such as discretization errors and occlusions.

The point-based methods work directly on irregular point clouds. However, point clouds are orderless and unstructured, making it infeasible to directly apply standard CNNs. The most representative works include graph-based methods [32] and PointNet [33]. The core idea of a graph-based method is capturing the basic shape and geometric structure of the 3-D point cloud, and the geometric position between the points is used as the weight of the graph convolution. The Super-Point graph [34] represented a point cloud as a set of interconnected simple shapes and superpoints and used an attributed directed graph (i.e., superpoint graph) to capture the structure and context information.

In this paper, motivated by the work of references [26], [35] and the power of deep learning-based moving object segmentation and detection, the problems of SLAM and MOT are addressed in a unified iterative optimization framework. To tackle this joint problem, how to identify the moving objects and the static background is the key issue. To this end, we use a fully convolutional network for semantic segmentation, which combines the features across different layers to refine the spatial precision, and the moving targets can be recognized and localized with the FCN-based classifier. When the point clouds of the moving target are filtered out, the remaining point cloud is assumed to be the static background and the static SLAM is implemented. The moving targets are further confirmed by the filter-based tracking process. The LCD and graph-based global optimization are used to improve the map consistency. The presented method strikes a balance between the accuracy and stability of object detection and tracking, and the tracking process and static SLAM can benefit from each other. The experiment results show that the map consistency outperforms the SOTA methods.

**III. DBN-BASED FORMULATION OF 4-D SLAM**

**A. BAYESIAN MODELING FOR STATIC SLAM**

We assume that the system is a general nonlinear dynamic system, expressed with a discrete state space model; then, we have the following:

$$ x_k = f(x_{k-1}, u_k) + v_k $$  
$$ z_k = h(x_k) + o_k $$  

(1)

where $f(\cdot)$ and $h(\cdot)$ are the nonlinear state transition model and observation model, respectively, $v_k$ and $o_k$ are noise vectors, which are assumed to be Gaussian, temporally uncorrelated and zero-mean. Then, a SLAM problem is defined as estimating the robot trajectory and the map of the environment as the robot moves in it. The trajectory can be represented by a sequence of random variables as follows:

$$ X_{1:k} \triangleq \{x_0, x_1, \ldots, x_k\} = \{x_{0:k-1}, x_k\} $$

Similarly, let

$$ Z_{1:k} \triangleq \{z_0, z_1, \ldots, z_k\} = \{z_{0:k-1}, z_k\} $$
$$ U_{1:k} \triangleq \{u_0, u_1, \ldots, u_k\} = \{u_{0:k-1}, u_k\} $$

denote the observations of the environment (or landmark) and the odometry measurements acquired, respectively. The discrete time index by the variable $k$, the map $M$ consist of $l$ features; then, solving the full SLAM problem consists of estimating the posterior probability of the robot’s trajectory $X_{1:k}$ and the map $M$ of the environment given all the measurements plus an initial position $x_0$ [45] from a Bayesian point of view, as follows:

$$ p(X_{1:k}, M|U_{1:k}, Z_{1:k}, x_0) $$

(2)

Based on the system model (1) and the assumptions that SLAM problem is a Markov process and the environment is static, then an iterative posterior probability estimation of SLAM problem can be expressed as follows:

$$ p(x_k, M|Z_k, U_k) \propto p(Z_k|x_k, M) $$

(3)

where $p(Z_k|x_k, M)$ is the observation part at time $k$; $p(x_k|x_{k-1}, U_k)$ is the prediction part at $k-1$; $p(x_{k-1}, M|Z_{k-1}, U_{k-1})$ is the posterior part at $k-1$.
Actually, the expression of SLAM with formula (3) assumes that the observation of static feature points is only related to the current robot pose and the state of the feature points.

**B. EXTENDED MODELS FOR SLAM AND MOT IN AN OPEN DYNAMIC ENVIRONMENT**

Based on the Bayesian modeling of static SLAM in the previous section, we now extend it to the dynamic environment where moving objects exist. To describe the states of the moving objects, we add some new random variables as follows,

\[
Y_k \triangleq \{y^1_k, y^2_k, \ldots, y^p_k\} \\
W_k \triangleq \{w^1_k, w^2_k, \ldots, w^q_k\}
\]  

(4)

where \(Y_k\) denotes the states of the moving objects, and \(W_k\) represents the motion models of the moving objects.

Figure 1 illustrates a generative dynamic Bayesian network (DBN) [46], [47] for SLAM with MOT. As shown in this figure, the static map is defined as \(m \triangleq \{m^1, m^2, \ldots, m^p\}\), and the observation for this static part is defined as \(Z^S_{0:k} \triangleq \{z^S_0, z^S_1, \ldots, z^S_k\} = \{z^S_{0:k-1}, z^S_k\}\). The superscript \(s\) indicates the static part. The states \(Y_k\) and motion models \(W_k\) of the moving objects, and the corresponding observations \(Z^m_{0:k} \triangleq \{z^m_0, z^m_1, \ldots, z^m_k\} = \{z^m_{0:k-1}, z^m_k\}\) are added in the graph.

In this figure, the shaded nodes which include the odometry measurements \((u_i, i = 1, \ldots, k)\), the observation node \((z^s_i, z^m_i)\), here \(s\) refers to the observation of static objects and \(m\) indicates the observation of moving object which is highlighted with yellow color. These shaded nodes denote the explicit states. The clear nodes denote the hidden or implicit states, which should be derived from the explicit ones.

To derive the formulation of SLAM with MOT, a special assumption is made to simplify the modeling, that is, the observations can be decomposed into two independent parts, the static background and moving objects. This assumption requires that we can reliably detect and distinguish moving objects from the stationary background. Based on the assumption, we can have the following:

\[
z_k \triangleq z^s_k + z^m_k \Rightarrow z_{0:k} = z^s_{0:k} + z^m_{0:k} \\
\Rightarrow p(z_k | x_k, M, y_k) = p(z^s_k | x_k, M, y_k)p(z^m_k | x_k, M, y_k)
\]  

(5)

where \(z^s_k\) denotes the observation of stationary objects, and \(z^m_k\) denotes the observation of moving objects. Formula (5) defines the joint observation probability.

\[
p(z_k | x_k, M, y_k) = p(z^s_k | x_k, M, y_k)p(z^m_k | x_k, M, y_k)
\]

(6)

Based on the above assumption, the SLAM and MOT problem is derived as follows:

\[
p(Y_k, x_k, M | U_k, Z_k) \propto p(Z_k | Y_k, x_k, M, U_k, Z_{k-1}) \\
\times p(Y_k, x_k, M | Z_{k-1}, U_k)
\]

(7)
Here, \( p(Z_k | Y_k, x_k, M, U_k, Z_{k-1}) \) refers to the observation probability, which is the same as that in formula (6), and \( p(Y_k, x_k, M | Z_{k-1}, U_k) \) is the prediction probability, which can be decomposed into two parts as follows:

\[
p(Y_k, x_k, M | Z_{k-1}, U_k) = p(Y_k | Z_{k-1}, U_k) \cdot p(x_k, M | Z_{k-1}, U_k) \quad (8)
\]

Replace formula (7) with (6) and (8), then we have the formula (9).

According to formula (9), the joint posterior probability for the SLAM and MOT problem is composed of the posterior probability of static SLAM and moving object tracking. Moreover, the misclassification of the moving objects and stationary background could degrade the performance. Based on the assumption 1) in section III.B, a feasible implementation of SLAM and MOT synchronously is that SLAM and MOT can be performed iteratively until the result converges. In this paper, the posterior probability of the MOT part is estimated by a joint probabilistic data association filter (JPDA).

\[
p(Y_k, x_k, M | U_k, Z_k) \propto p(Z_k | Y_k, x_k, M, U_k, Z_{k-1}) \cdot p(Y_k, x_k, M | Z_{k-1}, U_k) = p(z_k^s | x_k, M) p(x_k, M | Z_{k-1}, U_k) \cdot p(z_k^m | y_k, p(Y_k | Z_{k-1}, U_k) \quad (9)
\]

**IV. IMPLEMENTATION**

To estimate the joint posterior probability expressed in formula (9), two more challenging aspects are addressed here based on the maturity of the static SLAM. First, the robust separation between the static background and moving objects is addressed. To solve this problem, we leverage the recent progress in 3-D semantic segmentation and clustering for potential moving object detection. Second, efficient tracking and mapping based on the separation are realized. According to formula (9), the implementation of SLAM and MOT is separated into 4 stages, i.e., scan preprocessing, moving objects detection and tracking, laser-inertial fusion-based odometry and 3-D mapping, as illustrated in Figure 2. They are working in a sequential cooperative way.

**A. SCAN PREPROCESSING**

This module mainly completes the initial pose estimation and removes motion distortion based on the fusion of the laser scanner and inertial sensor. Because of the aggressive motion of the autonomous vehicle in urban roads environments, the point cloud distortion caused by vehicle motion during each laser scanning can greatly affect the accuracy of the pose estimation mapping. Therefore, removing this point cloud distortion is essential for building a high definition map. Moreover, LiDAR pose estimation can provide an initial value for the optimal estimation of 3-D point cloud segmentation. Because the self-driving car works in a highly dynamic urban area, we use the IMU-aided method to remove the motion distortion. In this method, the motion is first estimated by the IMU preintegration according to the time interval from scanning start to end; then, the motion is aligned to the LiDAR frame with the geometric relationship between two sensors. The point cloud will be recalculated according to the motion. The specific steps are briefed as follows:
1) An extrinsic calibration between LiDAR and IMU is performed [14], which obtains the rigid body transformation from LiDAR to the IMU coordinate. Then, we project the LiDAR data into the IMU coordinate as follows:

$$T_{IMU} = T_{LIDAR} \times R + t$$  \hspace{1cm} (10)$$

where $R$ and $t$ represent the rotation and translation, respectively, in the world coordinate $W$.

2) All laser points in the laser scan are numbered with timestamps, as illustrated in Figure 3. The mechanical scanning LiDAR rotates at a fixed frequency, and all the laser points of each frame of the point cloud can be numbered by the angle difference, as follows:

$$t_{curr} = t_{start} + \alpha \times \frac{\theta_{curr}}{\theta_{start}}$$  \hspace{1cm} (11)$$

where $\alpha$ represents the scanning period of the laser scan.

3) Find the successive IMU measurements with timestamps $k$ and $k+1$, which is the one closest to the timestamp $T_{current}$ of the current laser point.

4) The pose of the IMU at time $k$ and $k+1$, i.e., $T^W_k$ and $T^W_{k+1}$, respectively, in the world coordinate $W$ can be expressed as follows:

$$T^W_k = [p_x, p_y, p_z, v_x, v_y, v_z, roll, pitch, yaw]$$  \hspace{1cm} (12)$$

5) A linear interpolation method is used to compute the pose of the IMU in the world coordinate at time $t_{curr}$, which is equivalent to finding the current pose $T^W_{curr}$ of the current laser point in the world coordinate. The specific formulas are as follows:

$$r_{front} = \frac{t_{curr} - t_k}{t_{k+1} - t_k}$$  \hspace{1cm} (14)$$

$$r_{back} = \frac{t_{k+1} - t_{curr}}{t_{k+1} - t_k}$$  \hspace{1cm} (15)$$

$$T^W_{curr} = T^W_{k+1} \times r_{front} + T^W_k \times r_{back}$$  \hspace{1cm} (16)$$

6) Since we can calculate the current point cloud position $P^W_{curr}$ and the velocity $V^W_{curr}$ with $T^W_{curr}$, similarly the start position $P^W_{start}$ and the velocity $V^W_{start}$ can be deduced from $T^W_{start}$, then, the motion distortion $\Delta P^W_{start}$ of the laser point related to the start point coordinate can be estimated by Eq.17 and Eq.18, which are as follows:

$$\Delta P^W_{curr} = P^W_{curr} - (P^W_{start} + V^W_{start} \times \alpha \frac{\theta_{curr}}{\theta_{end}})$$  \hspace{1cm} (17)$$

$$\Delta P^W_{start} = T^W_{start} \Delta P^W_{curr}$$  \hspace{1cm} (18)$$

7) All the laser points are transformed to the start point coordinate and the motion distortion $\Delta P^W_{start}$ is subtracted for each laser point; then, the motion distortion caused by ego-motion is compensated.

Since the time synchronization is crucial in this stage, the first frame of a sweep of LiDAR output is synchronized with the IMU data whose timestamp is close to that of LiDAR, generally, two frames of IMU data, which are ahead and after the timestamp of LiDAR data, then a linear interpolation is used for the alignment. This process repeats for every sweep of LiDAR.

The effect of motion distortion on the local submap is shown in Figure 4. The left in this figure is a submap built without motion distortion compensation; the curb and the wall are blurred, the trunks and leaves of trees on the roadside are also blurred. However, in the right figure, the curb, the wall and the trunk are looked deblurred. Particularly we can see the effects from curbs. It is obvious that the map accuracy is improved by motion distortion compensation.

B. MOVING OBJECTS DETECTION AND TRACKING

The tasks in this stage include the detection of a potential moving object and then tracking it. Since the point cloud belong to the potential moving object will be filtered out in this stage, it makes the following static SLAM easier. Accurate segmentation is essential for a robust MOT. For the current methods, The PointNet-based convolutional network requires a fixed-size point cloud input and loses the spatial geometric information of the input point cloud. To avoid this problem, we exploit the fully convolutional semantic segmentation network (FCNN) [48] based segmentation method. The FCNN-based network can receive input matrices of any size and preserve the spatial geometric information of the input point cloud while predicting the semantic label of each point. We first use FCNN to accurately detect and segment potential moving objects, such as cars, pedestrians, cyclists, etc. Then, a tracker takes the segmentation result as the input and performs the MOT with a probabilistic data association based on the multimodal interactive unscented Kalman filter method [49]. Based on the state estimation of potential moving objects, we can recognize moving and static targets. Meanwhile, according to the results of the point cloud segmentation and MOT, the point cloud is classified into the following two groups: static background and moving objects. Only the point cloud belonging to static background will be forwarded to the odometry module.
1) DETECTION OF POTENTIAL MOVING OBJECTS
To track multiple moving objects reliably, we must be able to detect them reliably first. To detect the potential moving objects from the distortion-free point cloud, the 3-D point cloud is first projected into a 2D grid plane; then, the features are extracted for segmentation, and a FCNN is applied for semantic segmentation. Classification and clustering are imposed to further improve the detection accuracy. The procedures are detailed as follows.

\( a: \text{FEATURE EXTRACTION} \)

Because of the large volume of the input point cloud, to reduce the computing load, a fundamental step for feature extraction is to reduce the dimensionality. All the input point cloud data are projected to a 2D pseudo image, and a 2D index for each unit of the 2D pseudo image is built. The specific steps are as follows:

Let \( Q_t \) be a distortion-free point cloud from the scan preprocessing module at time \( t \); it is projected into a 2D grid plane according to the range of \((x, y)\) in \( Q_t \). The size of the grid is preset and the resulting grid is \( H \times W \) (the grid size is set to 0.25m \( \times \) 0.25m in this paper). The height of each grid is filtered by the Z value of those points that are located in this grid. This projection can be expressed as follows:

\[
p'_x = c + \frac{d - c}{b - a}(p_x - a)
\]

where \( p_x \) is the original coordinate of \( x \), whose range is \([a,b]\), and \( p'_x \) is the projected coordinate of \( x \), whose range is \([c,d]\).

After the projection transformation, multiple geometric features of each grid are calculated. Considering the computational efficiency and segmentation performance, 8 features are selected, including the maximum height of the midpoint of the cell, the intensity, the average height, the average intensity, the number of points, the angle of the cell center relative to the origin, the distance between the center of the cell and the origin, and whether the grid cell is empty or occupied. The extracted features are converted to several feature matrices and are used as the semantic segmentation input.

\( b: \text{FCNN-BASED SEMANTIC SEGMENTATION} \)

Caffe is a mature framework and is open-source, and we build the FCNN on the Caffe framework. To improve the generalization ability of the model, we train it on a large scale dataset with different types of traffic scenes. Specifically, we choose the object data subset of KITTI [50]. The KITTI object subset contains many scenes, such as multiple traffic flow intersection scenes, roundabout scenes, rural road intersection scenes, and right overtaking scenes. The KITTI Object subset contains 7481 frames of point cloud data for training and 7518 frames of point cloud data for testing. The dataset provides accurate annotations for cars, pedestrians, and cyclists in each frame of the point cloud. There are a total of 80256 targets marked in the entire object subset. An FCNN can receive input matrices of any size and can perform an end-to-end semantic segmentation of the point cloud. The network is mainly composed of the following three parts: feature encoder, feature decoder, and classification predictor. As illustrated in Figure 5, the feature encoder takes the grid as the input and continuously downsamples its spatial resolution as the feature extraction increases. Based on the sampling results, the attribute predictor predicts the offset displacement (center offset), objectivity, positivity, and object heights of the obstacles in the grid. The feature decoder performs upsampling according to the spatial resolution of the original 2D grid according to the prediction result, restores the details of the original grid, and finally outputs the labels of each grid cell.

\( c: \text{CLUSTERING} \)

The semantic segmentation assigns four attributes to each point in the grid, including the object center offset, center...
offset direction, object attribute probability value, and object height. According to the object center offset value and the center offset direction of the grid, a directed graph can be built, and the grids indicated by multiple grids in the directed graph are potential classification objects, as shown on the left side of Figure 6. To further improve the classification accuracy, reliable clustering objects can be selected based on the attribute probability value and object height. The right side of Figure 6 presents the clustering result.

Figure 7 is a bird’s eye view (BEV) of the clustering result with a scan frame, which is from a VLP-16 LiDAR. It is obvious that all the moving car are detected.

2) MULTIPLE-OBJECT TRACKING (MOT)
In an urban driving scenario, due to the variety of moving objects, the MOT task will face many challenges, such as the uncertainty caused by the mutual occlusion of moving objects, the uncertainty caused by different motion models of moving objects, etc. Therefore, a tracker is required to track the multiple maneuvering objects detected from uncertain sensor measurements.

To deal with different motion modes of different moving objects, the interactive multiple model (IMM) method is adopted. A nonlinear kinematics model is used, and the states of the moving objects are estimated with UKF [51]. To achieve accurate object tracking in the presence of detection noise, a joint probabilistic data association filter (JPDA) [52] is used for reliable data association.

The MOT process is mainly composed of two parts, i.e., JPDA and IMM. The process is detailed as follows.

a: THE OBJECT TRACKING MODEL
A state vector \( X_t \) and a measurement vector \( Z_t \) at time \( t \) are defined as in formulas (20) and (21); the motion model and observation model are defined as (22) and (23), respectively; \( u_t \) is the input vector.

\[
X_t = [\text{pos}_x, \text{pos}_y, v_x, \Psi_t, \Psi_j]^T \tag{20}
\]

\[
Z_t = [\text{pos}_x, \text{pos}_y]^T \tag{21}
\]

\[
X_{t+1} = f_j(X_t, u_t) + w_{j,t} \tag{22}
\]

\[
Z_t = h_j(X_t, u_t) + v_{j,t} \tag{23}
\]

where \( w_{j,t} \) and \( v_{j,t} \) are the process and measurement noise, respectively, both of which are assumed to be white Gaussian noise with an average value of zero. Let \( p_{i,j} \) be the predefined transition probability from model index \( i \) to index \( j \); then, the model transition is controlled by the transition probability matrix \( \Pi \), which is defined as follows:

\[
\Pi = \begin{bmatrix}
    p_{1,1} & \cdots & p_{1,r} \\
    \vdots & \ddots & \vdots \\
    p_{r,1} & \cdots & p_{r,r}
\end{bmatrix} \tag{24}
\]

b: INTERACTION
The main task of interaction is to initialize the initial state of each filter. A fusion of IMM probability is used. Let \( \mu_{(i,j),t-1} \) be the posterior mode probability that the object motion matches the dynamics of the motion model run by filter \( j \) at time \( t \), which can be defined as follows:

\[
\mu_{(i,j),t-1} = \frac{p_{ij} \bar{X}_{i,t}}{\bar{X}_{j,t}} \tag{25}
\]

Then, assuming that there are \( j \) filters, each filter at time \( t-1 \) has the mixed initial state \( \hat{X}_{j,t-1} \) and the mixed covariance matrix \( P_{j,t-1} \), which can be fused to obtain the initial state \( \hat{X}_{j,t-1} \) and the covariance matrix \( P_{j,t-1}^e \) as follows:

\[
\hat{X}_{j,t-1} = \sum_{i=1}^{r} \mu_{(i,j),t-1} \hat{X}_{i,t-1} \tag{26}
\]

\[
P_{j,t-1}^e = \sum_{i=1}^{r} \mu_{(i,j),t-1} [P_{t,k-1} + AA^T] \tag{27}
\]

where \( A = \hat{X}_{j,t-1} - \hat{X}_{j,t-1} \).

c: PREDICTION AND OBSERVATION UPDATING WITH UKF
The sigma points are propagated with the motion model of \( f(\cdot) \) in formula (22), and the weighted sigma points will be recomputed into the predicted state and its corresponding covariance matrix. Subsequently, we select a new set of sigma points, which will be propagated into the measurement function \( h \) according to the prior \( P_{i,k|k-1} \). Finally, the weighted recomputation of sigma points is used to generate the covariance matrix \( Z_k \) and the predicted metric \( P_{i,k} \).

The specific steps are as follows:

Step1: form a set of sigma point \( \chi' \):

\[
\chi'_{i-1|t-1} = \chi_{i-1|t-1} - (\sqrt{(L+\lambda)P_{i-1|t-1}}) \tag{28}
\]

where \( i = 1, 2, \ldots, L \).

\[
\chi'_{L+1|t-1} = \chi_{L+1|t-1} - (\sqrt{(L+\lambda)P_{L+1|t-1}}) \tag{29}
\]

where \( i = L+1, \ldots, 2L \).

Then, the sigma points are transformed and propagated through the motion model function as follows:

\[
\chi_{i+1|t} = f(\chi'_{i|t-1}) \tag{30}
\]

Step2: According to the weights, the corresponding prediction state and covariance matrix are generated, which replace
FIGURE 6. The center offset and clustering results based on the directed graph. (left) The arrows indicate the prediction of the center offset of each cell object; the blue fill corresponds to a possible object with a probability not less than 0.5; (right) the potentially clustered objects are indicated inside the solid red polygon, the stars indicate the clustering center.

FIGURE 7. Object clustering result with a bird’s eye view. (All the cars on the road are clustered and detected).

Propagate 2L+1 sigma points through the observation model h as follows:

\[ \hat{z}_t = h(\chi_t) \]  

Producing the predicted measurement and measurement covariance as follows:

\[ \hat{z}_t = \sum_{i=0}^{2L} W_i^s z_i \]  

The UKF gain is given by the following

\[ K_t = P_{x|t} P_{z|t}^{-1} \]  

Step4: Finally, the prediction results updated by the UKF are forward to the verification gate to determine whether the prediction is valid, as in paper [48].

d: DATA ASSOCIATION AND MODEL FILTERING

The function of this process is roughly the same as the conventional probabilistic data association (PDA) filter, but in this paper, we need to incorporate each model in the process, i.e., \( \hat{x}_t \) and \( P_t \). The updated state is simply the predicted state plus the innovation weighted by the Kalman gain. However, the associated innovation term is multiplied with association probabilities \( \beta \).

\[ \hat{x}_t = \hat{x}_t - K_t (v(t)) \]

\[ v_t = \sum_{m=1}^{N_v} \beta_{m,t} \bar{z}_t - \bar{z}_t \]

and the updated covariance is defined as follows:

\[ P_t = \beta_0 P_t + (1 - \beta_0) \left( K_t S_t K_t^T + Q_t - \beta_0 \right) \]

where \( B = \chi_t - \hat{x}_t \) is the weight coefficients of the state and covariance matrix, respectively.

Step3: This step is the same as in the prediction model; 2L + 1 sigma points are derived and this set is projected with the measurement model. Finally, the weighted sigma points are recomputed, which generate the measurement value and measurement covariance;
e: MODEL PROBABILITY UPDATING AND FUSION
The mode probabilities are updated based on the likelihood that the measurement fits to a model; it is given by the following:

\[ \mu_{i,t} = \frac{\mu_{i,t} \lambda_{i,t}}{\sum b_{i,t} \mu_{i,t} \lambda_{i,t}} \] (39)

where the Gaussian mixture likelihood \( \lambda_{i,t} \) is given as follows:

\[ \lambda_{i,t} = \frac{1 - (P_D P_G)}{(V_K)^N_c} + \frac{P_D V_i^{1-N_c}}{N_c \sqrt{2\pi}} \sum_{m=1}^{N_c} e^{-\frac{1}{2} C_T^{-1}} \] (40)

where \( C = z_{i,t} - z_{j,t|i-1} \)

Finally, the states updated by the filters are to be combined again into a single final state and covariance estimate, as follows:

\[ \hat{x} = \sum_{j=1}^{r} \mu_{j,t} \hat{x}_{j,t} \] (41)

\[ P_t = \sum_{j=1}^{r} \mu_{j,t} [P_{j,t} + (\hat{x}_{j,t} - \hat{x}_t)(\hat{x}_{j,t} - \hat{x}_t)^T] \] (42)

3) LiDAR-INERTIAL FUSION BASED ODOMETRY
Based on the potential moving object detection and tracking method discussed in the previous section, the moving objects in each scan frame can be filtered out. The rest of the point cloud is assumed to be the static background, which can be used to estimate the pose of a self-driving car. To optimize the vehicle’s trajectory, nonlinear optimization is used. This module consists of two parts, i.e., feature extraction and association.

a: FEATURE EXTRACTION
The static background point cloud is processed to provide different distinctive features. As shown in Figure 8, these features include planar and edge features, which are also used in the LOAM method [15]. The difference is that a ground point cloud extraction operation is performed in the urban scene. In the urban scene, the road surface and sidewalk are mainly composed of ground point clouds. If we directly apply RANSAC-based planar point extraction to segment it, we may obtain a false segmentation result because the sidewalk edge may be treated as the ground. To solve this problem, we discriminate the ground based on the angle difference between points. First, the angles between different laser beams in the vertical direction are calculated. Then, all the beam combinations are traversed and the beam points whose angle differences are less than a threshold are regarded as the ground points. Then, the point cloud is classified into the ground and nonground points, and only the ground point is regarded as a subset of the planar point.

As in [15], the nonground points are further classified into several subsets, and every subset is composed of its adjacent points. The roughness of each point in the subset is calculated; then, the points are sorted based on the roughness threshold into different types. The points with roughness values larger than the high threshold are regarded as edge features, the points with roughness values smaller than the low threshold are taken as planar features, and the points with roughness values between the edge and planar features are outlier points. The edge feature and planar feature points are used for feature association, and the outlier points will be directly forwarded to the mapping module.

b: FEATURE ASSOCIATION
The transformation between two scans is found by performing point-to-edge and point-to-plane scan-matching. We assume that the planar points and edge feature points at time \( t \) are \( f_{p}^{t} \) and \( f_{E}^{t} \), and they are \( f_{p}^{t+1} \) and \( f_{E}^{t+1} \) at time \( t+1 \). Then, we need to find the correspondence for points \( f_{p}^{t+1} \) and \( f_{E}^{t+1} \) from the previous points \( f_{p}^{t} \) and \( f_{E}^{t} \). The difference from reference [15] is that the labels of the feature points constituting the plane and line at time \( t \) are determined after segmentation. For the planar features in \( f_{p}^{t+1} \), only the points that are labeled as ground points in \( f_{p}^{t} \) are used for finding a planar patch as the correspondence. For the edge features...
in \( f_k^{t+1} \), its corresponding edge line is found in \( f_{t}^k \) from segmented clusters.

The Levenberg-Marquardt method is employed to find the minimum-distance transformation between the two consecutive scans. A two-step L-M optimization method is used. The optimal transformation \( T \) is found in two steps as follows:

- \( [t_z, \theta_{roll}, \theta_{pitch}] \) is estimated by matching the planar features in \( f_{p_{t+1}}^k \) and their correspondence in \( f_{p_{t}}^k \).
- \( [t_x, t_y, \theta_{yaw}] \) is then estimated using the edge features in \( f_{E_{t+1}}^k \) and their correspondences in \( f_{E_{t}}^k \) while using \( [t_z, \theta_{roll}, \theta_{pitch}] \) as constraints.

Finally, the 6D transformation between two consecutive scans is found by fusing \( [t_z, \theta_{roll}, \theta_{pitch}] \) and \( [t_x, t_y, \theta_{yaw}] \). The data processing flow is illustrated in Figure 8.

4) 3-D MAPPING

With the pose estimation result from the odometry module, the point cloud of the current scan is aligned and stored in a local submap first. Let \( Q^t \) be the last submap before the current point cloud is added, which is defined as follows:

\[
Q^{t-1} = \{f_{p_{k}}^{t-k}, f_{E_{k}}^{t-k}, f_{p_{k+1}}^{t-k+1}, \ldots, f_{p_{t}}^{t-1}, f_{E_{t}}^{t-1}\}
\]

where \( k \) denotes the window size of the local submap. An NDT-based registration is used to align the current feature point \( f_{p_{t}}^{t} \) to the submap \( Q^t \), and the updated transformation \( T^t \) is the final result and is stored for further global optimization if a loop closure is found. Only the newest \( k \) frames are kept in the submap for efficiency. With the updated local submap \( Q^t \), we can stitch it with the global map at a low frequency.

A graph-based optimization with GTSAM [14] is introduced for global optimization to eliminate the drift caused by long-term mapping. The final pose estimation from the alignment of the submap is treated as a factor, and the corresponding node is defined as the feature points \( f_{p_{t}}^{t}, f_{E_{t}}^{t} \).

To eliminate the drift in 3-D mapping, a LCD process which is same as in reference [15] is adopted. A new constraint will be added to the factor graph if a match between the current feature set and a previous one using the ICP algorithm is found. Once the LCD is found, the global optimization is triggered and all the poses or trajectories will be updated.

V. EXPERIMENTS AND DISCUSSIONS

To validate the performance of the proposed method in complex urban road scenes, we compared the proposed method with two SOTA methods, LeGO-LOAM [22] and SuMa [27]. The scenarios for comparison mainly include urban roads with high-speed moving objects, multiple types of objects and large scale environments. The open benchmark dataset, i.e., KITTI [52], and the dataset collected by our platform are used. The performance evaluation includes the localization accuracy, mapping details, MOT accuracy and computation cost.

A. EVALUATION OF THE POSE ESTIMATION ACCURACY

There are many challenging scenes in the KITTI dataset, such as aggressive motion of the data collector, lack of geometric features, and complicated traffic intersections. The sequences used for accuracy evaluation include 00, 03, 05, 07, 08, and 10, with a total of 6 sequences. The evaluation of the global localization accuracy and robustness of the local pose estimation are carried out, respectively.

1) EVALUATION ON GLOBAL LOCALIZATION ACCURACY

The metric value of absolute trajectory error (ATE) is used to evaluate the accuracy of each method and all the sequences are 6-DoF aligned with the ground truth. ATE indicates the difference between the estimated trajectory by SLAM and the ground-truth. The qualitative comparisons are shown in Figure 9.

As shown in Figure 9, for the sequences 03 and 10, which are collected with high speed and no loop-closure exist, the proposed method can overcome the effects of aggressive motion by motion compensation. Compared with LeGO-LOAM and SuMa, our method can achieve better pose estimation results. For the sequences 00, 05, 07, and 08, which are collected with low speed and have rich loop-closure, both the proposed method and LeGO-LOAM can achieve accurate localization. Since our method can filter out the moving objects from the scene, so we perform a better static background extraction. Additionally, LCD at the crossroad improves the global localization accuracy.

To intuitively reflect the comparison of the global trajectory, the root mean square error (RMSE) and maximum value of the ATE on the trajectories are used for quantitative comparison; the result is shown in Table 1.

2) ROBUSTNESS COMPARISON OF THE POSE ESTIMATION

Relative pose error (RPE) is used to calculate the difference between two local upper displacement changes. After the time stamp is aligned, we can calculate the sum of the distance between the ground-truth pose and the estimated pose of the trajectory. Taking the equidistant distance of 500 meters as an example, we mainly focus on the mean value and max value of RPE. The RPE values of the proposed method and LeGO-LOAM for each sequence are shown in Table 2.

It can be observed from Table 2 that for most of the sequences, the proposed method is much better than LeGO-LOAM and SuMa. For the sequences 07, 08, there exist many corners in the point cloud, Suma performs LCD with segmented features, and shows high performance in these cases. But only ICP algorithm is used in the presented approach, and some loop closure detections are missed, this leads to the performance decrease.

B. EVALUATION OF THE 3-D MAPPING

The 3-D mapping performance is qualitatively manifested through the removal of moving objects and the map
FIGURE 9. The accuracy comparisons of the global trajectories for the sequence no 00, 03, 05, 07, 08, and 10 in the Kitti dataset. The dotted lines are the ground-truth of each sequence, the blue ones are the global trajectory of our method, the red ones are the global trajectory of LeGO-LOAM, and the green ones are from SuMa.
TABLE 1. Performance comparison of RMSE and max of the ate for the KITTI dataset.

| Sequence | LeGO-LOAM | SuMa | Ours | LeGO-LOAM | SuMa | Ours |
|----------|-----------|------|------|-----------|------|------|
| 00       | 2.91      | 1.89 | 1.3  | 17.92     | 8.39 | 3.58 |
| 03       | 0.92      | 1.03 | 0.85 | 1.81      | 2.09 | 1.41 |
| 05       | 3.81      | 2.97 | 1.19 | 12.69     | 4.16 | 7.19 |
| 07       | 3.41      | 0.58 | 0.52 | 8.44      | 1.21 | 1.03 |
| 08       | 3.59      | 3.2  | 2.75 | 17.4      | 13.6 | 14.25|
| 10       | 2.29      | 1.33 | 1.35 | 3.95      | 3.21 | 2.42 |

TABLE 2. RPE results comparison between the proposed method and LeGO-LOAM for the selected sequences.

| Sequence | LeGO-LOAM | SuMa | Ours | LeGO-LOAM | SuMa | Ours |
|----------|-----------|------|------|-----------|------|------|
| 00       | 0.023     | 0.02 | 0.019 | 1.13      | 0.30 | 0.29 |
| 03       | 0.98      | 0.07 | 0.03 | 1.62      | 1.30 | 0.54 |
| 05       | 1.12      | 0.03 | 0.01 | 1.71      | 0.11 | 1.5  |
| 07       | 5.53      | 0.7  | 1.71 | 0.09      | 0.39 | 0.08 |
| 08       | 0.06      | 0.02 | 1.11 | 6.41      | 0.24 | 1.81 |
| 10       | 1.07      | 0.05 | 0.01 | 2.15      | 0.27 | 0.24 |

FIGURE 10. Comparison of the mapping details for sequence 07. The upper row contains the results of the presented method, LeGO-LOAM and SuMa from left to right. The lower row is the corresponding enlarged details for the red box in the first row.

(a) Mapping result of the proposed method. (b) Mapping result of LeGO-LOAM
(c) Mapping result of SuMa.
(d) Enlarged local submap of the proposed method. (e) Enlarged local submap of the LeGO-LOAM method.
(f) Enlarged local submap of the SuMa method.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.

As shown in Figure 10, the first row is the whole maps and the mapping details are shown in Figure 10 for sequence 07 and Figure 11 for sequence 05.
SuMa. The maps are shown with the original outputs of these methods. There is a busy intersection, which is indicated by the red box. The results of this area are enlarged in the lower row. The maps constructed by LeGO-LOAM and SuMa do not filter out the outlier points, so the maps contain many lines, which are the trajectories of moving objects. With the removal of moving objects performed by the multi-object detection and tracking module, the map built by the presented method only contains the static background and is clearer.

The improvement of the map consistency of the presented method can be seen from Figure 11, which is the mapping results of sequence 05. In this figure, the left column shows the result of the presented method, and the right column is the LeGO-LOAM results. The upper one is the whole map, and the lower is the enlarged details of the red box in first row. It is clear that the presented method outperforms LeGO-LOAM.

The self-driving platform is shown in Figure 12. It is equipped with a Velodyne VLP-16 on the top of the vehicle.
and an OXTS inertial + IMU installed inside the trunk. The two sensors are precalibrated and the collected data is synchronized. The collected dataset includes the urban road scenes around campus during afternoon rush hour. The VLP-16 runs at a frequency of 10 Hz, and the sampling rate of IMU is approximately 200 Hz.

Figure 13 shows the experimental results. It can be observed from Figure 13.a that our proposed method achieves accurate global localization even when the traffic is complex and the number of laser beams is less than that of KITTI. However, the trajectory output by LeGO-LOAM has an obvious offset in the z direction, resulting in a significant deviation in the altitude in the global map. Using the mapping details in the upper right corner as an example, the proposed method eliminates the impact of moving objects and improves the consistency of the mapping.

C. SEGMENTATION-BASED MULTIPLE-OBJECT DETECTION AND TRACKING

Some semantic segmentation results are provided in Figure 14. The results are displayed with Rviz, a tool of the Robot Operating System (ROS). The lighter-colored point cloud in the figure indicates the static background, and the darker-colored point cloud sets are the segmented potential moving objects. In this figure, even if only a part of the point clouds that belong to an object is input into the detection network, the proposed method can still accurately segmented the static background and potential moving objects.

The MOT result is illustrated in Figure 15. The moving objects, such as cars, cyclists and pedestrians, are detected and tracked. The estimated states of the moving objects, including the distance, velocity, heading angle, etc., are indicated in the figure; for example, the green arrow above the moving object indicates its direction of movement, and the id of the moving object is uniquely identified. All these states are the key parameters for driving planning.

To quantitatively evaluate the MOT performance, we adopted the evaluation metrics used by the visual MOT and tested them on the tracking dataset of KITTI. The performance is shown in Table 3. There are 512 trajectories in the entire dataset. It can be observed that the indicators are close to the current MOT method based on deep learning, and the average computation cost is much lower than that of the SOA deep learning-based method. The proposed method strikes a balance between the accuracy of the results and the computing cost.
TABLE 3. Multi-target tracking performance based on the multi-model interaction.

| Objects category | MT    | ML    | MOTA  | MOTP  | MOTAL | Runtime |
|------------------|-------|-------|-------|-------|-------|---------|
| Pedestrian       | 65.5% | 2.86% | 64.1% | 88.6% | 64.8% | 25 ms   |
| Car              | 69.2% | 5.85% | 73.1% | 89.6% | 83.4% | 30 ms   |
| Bike             | 66.1% | 2.03% | 67.8% | 87.2% | 68.4% | 23 ms   |

**FIGURE 14.** Semantic segmentation and object detection results. Left – all the segmented objects are highlighted. Right – Object detection results with segmented points.

**FIGURE 15.** Multiple-object tracking results.

### D. EVALUATION ON EFFICIENCY

We also evaluate the runtime performance of the main module of the presented method. The hardware platform is a 3.2-GHz i7-8700M Intel processor and an integrated GPU (Nvidia GeForce GTX 1060), using the Robot Operating System (ROS) in Ubuntu 18.04. With the help of IMU measurement, the laser-inertial odometry runs at a frequency of approximately 10 Hz. The mapping module runs at a frequency of approximately 1 Hz, including factor graph optimization, global mapping and loop-closure detection. All the modules run on different threads in parallel. The average runtime of each module is shown in Table 4.

With the same hardware configuration, we made an efficiency comparison of the two main modules of SLAM, i.e., odometry and mapping. The run times for the three methods are listed in Table 5. Both the odometry and mapping modules of the presented method can run faster than LeGO-LOAM and SuMa.
TABLE 4. Runtime performances of the main modules.

| MAIN MODULES                  | Average Time |
|-------------------------------|--------------|
| Preprocessing                 | 5 ms         |
| Moving objects detection and tracking | 40 ms        |
| Fusion based odometry         | 10 ms        |
| Mapping                       | 80 ms        |

TABLE 5. Runtime performance comparison for the two main modules.

| method              | odometry | mapping |
|---------------------|----------|---------|
| The presented       | 15 ms    | 80 ms   |
| LeGO-LOAM           | 18.5 ms  | 116.7 ms|
| SuMa                | 19.7 ms  | 123.5 ms|

VI. CONCLUSION

In this paper, a 4D-SLAM approach that jointly solves the SLAM and MOT is presented, which is a key part of scene understanding for self-driving in a dynamic environment. Detection of moving objects plays a key role in the whole system. An FCNN is first employed to detect the potential moving objects, and the states of the potential moving objects are further confirmed by the UKF tracker. A semantic result is output, which is beneficial for the following planning and decisions of the self-driving car. At the same time, by applying the static background point cloud to the standard 3D map maintenance in dynamic environments, the inconsistency problem of SLAM in a dynamic environment is solved. The experiments with the open dataset KITTI and the dataset we collected show that, in terms of the accuracy and consistency of the obtained map, the presented method outperforms the state-of-the-art methods, such as LeGO-LOAM and SuMa++. Since the presented method can complete MOT and SLAM simultaneously, and can run in real-time, it provides a compact and efficient way to obtain scene understanding during self-driving.

Despite these encouraging results, there are some avenues for the next work. For example, only simple ICP is used for the LCD detection, in the future, we plan to investigate the usage of semantics for LCD and use more fine-grained semantic information to estimate, such as road structure and type, which are the basic elements for the HD map of self-driving. Additionally, we try to port the approach to run on the embedded system.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for all the suggestions and questions for improving this manuscript.

REFERENCES

[1] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[2] D. Gálvez-López, M. Salas, J. D. Tardós, and J. M. M. Montiel, “Real-time monocular object SLAM,” Robot. Auto. Syst., vol. 75, pp. 435–449, Jan. 2016.

[3] R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras,” IEEE Trans. Robot., vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[4] J. Zhang and S. Singh, “Laser-visual-inertial odometry and mapping with high robustness and low drift,” J. Field Robot., vol. 35, no. 8, pp. 1242–1264, Dec. 2018.

[5] P. Pritsker, “SLAM tutorial,” in Proc. Winter Simul. Conf., Dec. 1981, pp. 9–11.

[6] J. J. O’Reilly and J. P. Whitford, “SLAM II tutorial,” in Proc. Winter Simulation Conf., Dec. 1988, pp. 85–90.

[7] 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: Toward the robust-perception age,” IEEE Trans. Robot., vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[8] Z. Lu, Z. Hu, and K. Uchimura, “SLAM estimation in dynamic outdoor environments: A review,” in Intelligent Robotics and Applications, vol. 5928, Berlin, Germany: Springer, 2009, pp. 255–267.

[9] S. Hong, H. Ko, and J. Kim, “VICP: Velocity updating iterative closest point algorithm,” in Proc. IEEE Int. Conf. Robot. Autom., May 2010, pp. 1893–1898.

[10] F. Moosmann and C. Stiller, “Veloodyne SLAM,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2011, pp. 393–398.

[11] C.-C. Wang, “Simultaneous localization, mapping and moving object tracking,” Doctoral dissertation, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-04-23, Apr. 2004.

[12] C.-C. Wang and C. Thorpe, “Simultaneous localization and mapping with detection and tracking of moving objects,” in Proc. IEEE Int. Conf. Robot. Autom., vol. 3, May 2002, pp. 2918–2924.

[13] M. Bosse, R. Zlot, and P. Flick, “Zebepose: Design of a spring-mounted 3-D range sensor with application to mobile mapping,” IEEE Trans. Robot., vol. 28, no. 5, pp. 1104–1119, Oct. 2012.

[14] C. Le Gentil, T. Vidal-Calleja, and S. Huang, “3D lidar-IMU calibration based on upsampled preintegrated measurements for motion distortion correction,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2018, pp. 2149–2155.

[15] J. Zhang and S. Singh, “Low-drift and real-time lidar odometry and mapping,” Auto. Robots, vol. 41, no. 2, pp. 401–416, Feb. 2017.

[16] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, “Dynamic pose graph SLAM: long-term mapping in low dynamic environments,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2012, pp. 1871–1878.

[17] Z. Wang, Y. Chen, Y. Mei, K. Yang, and B. Cai, “IMU-assisted 2D SLAM method for low-texture and dynamic environments,” Appl. Sci., vol. 8, no. 12, p. 2534, Dec. 2018.

[18] F. Pomerleau, P. Krusi, F. Colas, P. Furgale, and R. Siegwart, “Long-term 3D map maintenance in dynamic environments,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2014, pp. 3712–3719.

[19] R. Wang, Y. Xu, M. A. Soto, Y. Ma, T. Sarkodie-Gyan, Z. Li, and W. Li, “A robust registration method for autonomous driving pose estimation in urban dynamic environment using LiDAR,” Electronics, vol. 8, no. 1, pp. 43, Jan. 2019.

[20] M. S. Bahraini, A. B. Rad, and M. Bozorg, “SLAM in dynamic environments: A review,” in Proc. Winter Simulation Conf., Dec. 2012, pp. 1309–1332, Dec. 2016.

[21] C. Jiang, D. P. Paudel, Y. Fougerolle, D. Fofi, and C. Demonceaux, “Static-map and dynamic object reconstruction in outdoor scenes using 3-D motion segmentation,” IEEE Robot. Autom. Lett., vol. 1, no. 1, pp. 324–331, Jan. 2016.

[22] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2018, pp. 4758–4765.

[23] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, “SuMa++: Efficient LiDAR-based semantic SLAM,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Nov. 2019, pp. 4530–4537.

[24] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “RangeNet++: Fast and accurate LiDAR semantic segmentation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Nov. 2019, pp. 4213–4220.

[25] M. R. U. Saputra, A. Markham, and N. Trigoni, “Visual SLAM and structure from motion in dynamic environments: A survey,” ACM Comput. Surv., vol. 51, no. 2, p. 37, 2018.

[26] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, “Simultaneous localization, mapping and moving object tracking,” Int. J. Robot. Res., vol. 26, no. 9, pp. 889–916, Sep. 2007.
