Dynam-LVIO: A Dynamic-Object-Aware LiDAR Visual Inertial Odometry in Dynamic Urban Environments

Jian Shi, Wei Wang, *Senior Member, IEEE*, Mingyang Qi, Xin Li, and Ye Yan

***Abstract*— For most existing simultaneous localization and mapping (SLAM) systems, the surroundings of autonomous vehicle are assumed to be static, which is impractical in urban environments and compromises the effectiveness of existing SLAM algorithms. In this work, a dynamic-object-aware light detection and ranging (LiDAR) visual inertial odometry (LVIO), termed Dynam-LVIO, is introduced to enhance the accuracy of SLAM in dynamic environments by constructing both environment map and object map. Initially, reprojection error and iterative closest point (ICP) error are calculated with object map, serving as observations for the error state iterated Kalman filter (ESIKF) to accurately estimate the object state on manifold. Subsequently, the 2-D bounding boxes derived through YOLO-V5 are tracked with the proposed multiobject tracking (MOT) algorithm, LVI-SORT, to achieve stable MOT in complex scenes. Specifically, to improve the accuracy of MOT in fast moving scenes, the 2-D object flow, calculated with object state, vehicle state and object map, is used to predict object state in the prediction process of LVI-SORT. Furthermore, to mitigate MOT failures caused by temporary object occlusion, a hybrid object association is proposed within the association process of LVI-SORT, which is accomplished by incorporating object map points and intersection-over-union (IoU) of bounding boxes to form the cost matrix in the Hungarian algorithm. Finally, the tracked 2-D bounding boxes are leveraged to segment the recent global map into environment map and object map, thereby reducing the impact of dynamic objects on LiDAR visual inertial SLAM. Compared with other benchmark algorithms, the results indicate that the proposed algorithm can enhance localization accuracy by 5%–10% in dynamic scenarios. Concurrently, the proposed algorithm also enhances object tracking accuracy by nearly 3%.**

***Index Terms*— Dynamic urban environments, inertial, light detection and ranging (LiDAR), sensor fusion, visual.**

I. Introduction

HIGH-PRECISION localization in urban environments is crucial for ensuring the safe operation of autonomous vehicles. Currently, widely used localization methods include satellite navigation [1], inertial navigation [2], [3], [4], etc. In urban environments, satellite navigation may be subject to interference from tall buildings, resulting in lock loss or multipath effects. Inertial navigation uses inertial measurement units (IMUs) to obtain the state of the vehicle. This method remains unaffected by external interference and can operate in any urban scenario. However, due to the adoption of numerical integration algorithm by inertial navigation for state estimation, noise and biases within the IMU will induce drift in the integration results. Consequently, inertial navigation is unable to operate independently for extended periods. In recent years, researchers have explored the utilization of deep learning methods to enhance the accuracy of inertial navigation [5], [6], [7]. Specifically, in [5], self-supervised learning and future-aware inference are used to eliminate noise in the IMU and enhance the accuracy of inertial navigation. In [6], the raw IMU and barometer data are used as inputs of a convolutional neural network (CNN) to predict vehicle speed, enabling vehicle navigation in satellite navigation failure scenarios. In [7], based on the kinematic motion model, a differential error learning strategy is proposed to train the sensor error compensation model, reducing IMU error output during operation. However, current inertial navigation methods based on deep learning exhibit weak generalization capabilities and are not applicable to a wide range of urban scenes.

In the past few decades, simultaneous localization and mapping (SLAM) has been widely investigated in the field of autonomous vehicle to simultaneously estimate the vehicle state and build the environment map [8]. In complex urban scenes, high-precision localization can be achieved by the existing SLAM methods through the utilization of multiple sensors [9], [10], [11]. And most SLAM algorithms operate under the assumption of static environment [12], [13] and achieve vehicle state estimation by registering sensor data from adjacent frames. However, in realistic urban environments characterized by numerous dynamic objects such as cars, bicycles, and pedestrians, environment changes are almost inevitable. These dynamic objects invalidate the static environment assumption, leading to errors in the estimated vehicle states and potentially contributing to ghosting phenomena within the environment map.

In recent years, numerous geometry methods have been proposed to enhance the adaptability of the existing SLAM systems in dynamic environments. Conventional robust estima-

Manuscript received 16 December 2023; revised 4 April 2024; accepted 9 April 2024. Date of publication 30 April 2024; date of current version 10 May 2024. This work was supported by the National Natural Science Foundation under Grant 62271163. The Associate Editor coordinating the review process was Dr. George Dan Mois. *(Corresponding author: Wei Wang.)*

Jian Shi, Wei Wang, Mingyang Qi, and Xin Li are with the College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin 150001, China (e-mail: bznh-sj@hrbeu.edu.cn; wangwei407@hrbeu.edu.cn; qimingyang@hrbeu.edu.cn; xinxin_forever@126.com).

Ye Yan is with the National Innovation Institute of Defense Technology, Academy of Military Sciences China, Beijing 100071, China, and also with Tianjin Artificial Intelligence Innovation Center (TAIIC), Tianjin 300450, China (e-mail: yynudt@126.com).

Digital Object Identifier 10.1109/TIM.2024.3395320

tion methods, including random sample consensus (RANSAC) [14] and its derivative algorithms [15], [16], are used to mitigate the influence of dynamic objects by excluding data associated with dynamic objects from the state estimation process. However, these robust estimation methods will encounter challenges in environments with a significant number of dynamic objects. In [17], capitalizing on the constancy of relative positions among static feature points, point correlation is used to distinguish the static and dynamic map points. In [18], a novel sparse motion removal model is introduced for detecting dynamic regions within a Bayesian framework. Subsequently, the resemblance between the current image frame and the reference image frame is harnessed to alleviate the uncertainty of detected dynamic regions. However, the methods mentioned in [17] and [18] are specifically applicable for the SLAM algorithms that construct sparse maps. Dense maps are crucial for path planning of autonomous vehicles. Applying these methods to dense maps would result in a substantial increase in computational burden. In [19], the stereo scene flow is coupled with IMU to detect the dynamic feature points in stereo images. Owing to the inherent noise in depth measurements derived from stereo images, modeling the uncertainty of stereo scene flow obtained with depth measurements is a challenging task, thus influencing the detection accuracy of dynamic feature points.

The aforementioned geometric methods [14], [15], [16], [17], [18], [19] necessitate manual adjustment of algorithm parameters to ensure robust SLAM performance across various dynamic environments, which may be inconvenient for practical applications. Due to recent advancements in computer vision facilitated by deep learning [20], numerous researchers have embraced methods rooted in deep learning to alleviate the influence of dynamic objects in SLAM. In [21], the dynamic object segmentation is achieved at the pixel level using the CNN. Following this, the residual pixels corresponding to the static environments are used for SLAM. Furthermore, in [22], the instance segmentation method, Mask-region-based convolutional network (RCNN) [23], is applied for eliminating dynamic objects in image. Subsequently, the mask boundary is refined by incorporating the geometric and motion information of dynamic objects. Nevertheless, attaining pixel-level segmentation using the methodologies outlined in [21] and [22] demands considerable computing power and is deemed unsuitable for integration into the SLAM framework.

To ensure real-time execution of the SLAM algorithm in dynamic environments, some lightweight networks, such as the you only look once (YOLO) series [24], are seamlessly integrated into the SLAM framework. Notably, these lightweight networks can achieve object detection frequency of approximately 100 Hz on a desktop computer, rendering them well-suited with the SLAM framework. In [25], a lightweight object detection network, Darknet19-YOLOv3 [26], is adopted to generate regions from which dynamic points are extracted. And then a geometric constraint method is proposed to filter the dynamic point in these areas. In [27], a novel dynamic object boundle adjustment (BA) is proposed to concurrently calculate object state and vehicle state with sparse features correspondence from detected bounding boxes. However, most lightweight object detection networks use one-stage detector [28], resulting in the inevitable occurrence of false and missed detections. Consequently, the dynamic SLAM methods [25], [27] grounded in lightweight object detection are destined to falter in the face of such inaccuracies. In [29], the static object map is constructed with SLAM to provide prior knowledge for object detection, effectively decreasing the occurrence of false and missed detections. Nevertheless, this approach focuses solely on generating static object maps, neglecting dynamic objects and consequently failing to enhance the detection accuracy of dynamic objects.

To mitigate the occurrence of false or missed detections from lightweight object detection networks arising in challenging environments, numerous researchers have endeavored to integrate multiobject tracking (MOT) with the existing SLAM. MOT [30] is designed with the aims of identifying and locating objects within each frame. Subsequently, these objects are associated across image frames to facilitate the continuous tracking of their movements over time. And tracking-by-detection methods [31], [32], [33] are the most effective MOT paradigm, which consists of object detection and tracking step. In [34], the Bayes formulation of SLAM with MOT is derived, establishing a robust foundation for the integration of MOT into SLAM. Recently, visual–inertial SLAM is reconciled with MOT in [35] which focuses on the removal of dynamic people in urban city. Nevertheless, the accuracy of methods mentioned in [34] and [35] is highly correlated with the effectiveness of MOT. Specifically, if the MOT encounters failure, which is quite probable in the situation that vehicle moves rapidly, or the dynamic objects are temporarily obscured, the performance of SLAM integrated with MOT will be significantly affected.

Many recent methods have attempted to improve the accuracy of MOT in these challenging environments. In [36], the conventional image registration is adopted to estimate autonomous vehicle state to overcome the MOT failure that occurs during rapid motion. Since the image registration method only obtains the autonomous vehicle state in 2-D image space, which cannot accurately depict the motion in 3-D real environment. In [37], a novel formulation is proposed to model dynamic scenes in a unified estimation framework over vehicle states, object states, and static and dynamic environment map. This method can achieve accurate object state estimation by considering full information in dynamic environment. In [38], a novel asynchronous factor graph architecture is proposed to achieve SLAM with tightly coupled object tracking. Similarly, in [39], an RGB-D SLAM system is introduced for dynamic scenes, simultaneously estimating the poses of the camera, the map, and the trajectories of the moving objects. However, for the above tightly coupled SLAM method in dynamic environment [37], [38], [39], the introduction of an excessive number of dynamic object state constraints will induce instability in the state estimation process, thereby compromising the accuracy of vehicle state estimation. In [40], a simple 2-D motion model is proposed to describe the nonlinear object motion to achieve robust MOT when the dynamic object is under rapid movement. However, accurately representing the complex 3-D motion of objects with a 2-D motion model is challenging, limiting the improvement in tracking accuracy for fast-moving objects.

Furthermore, in [41], the deep appearance descriptor (a simple CNN) is pretrained for re-identification (Re-ID), which can improve MOT accuracy when the dynamic objects is temporarily obscured. However, the calculation of deep appearance descriptor is time-consuming which will take about 50 ms to extract the appearance descriptor in a desktop computer.

To enhance the accuracy of the MOT-based dynamic-object-aware SLAM framework in the aforementioned MOT challenging scenarios, both light detection and ranging (LiDAR) and visual perception results are used to develop a novel MOT algorithm aimed at achieving accurate perception of dynamic objects. Furthermore, a LiDAR visual inertial odometry (LVIO) capable of robust operation in dynamic environments is proposed in this work. And the main contributions are as follows.

1) To establish a dynamic-object-aware SLAM framework, we propose a novel MOT method named LVI-SORT and integrate it with a standard LVIO to form the Dynam-LVIO in this work. Using the object tracked with LVI-SORT, the global map generated by a standard LVIO can be partitioned into static environment map and object maps. This division mitigates the influence of dynamic objects on LVIO, thereby effectively enhancing localization and mapping accuracy in dynamic environments.
2) To improve the accuracy of MOT for vehicles or dynamic objects in rapidly moving scenes, reprojection errors and iterative closest point (ICP) errors are concurrently used as observations for error state iterated Kalman filter (ESIKF), enabling accurate estimation for the 3-D object state. Subsequently, the 3-D object state and object map points are leveraged to compute 2-D object flow during the prediction phase of the LVI-SORT. This refinement contributes to improve the accuracy of object prediction, particularly in rapid movement vehicle or objects, consequently enhancing overall MOT accuracy.
3) To mitigate MOT failures caused by temporary object occlusion, 2-D object flow is used to track object map points. The tracked object map points are subsequently integrated with the intersection-over-union (IoU) of the bounding boxes to form a hybrid cost matrix within the Hungarian algorithm. This approach guarantees the accurate data association between the predicted object and the YOLO-V5 detected object after a period of occlusion, thereby bolstering the stability of MOT in scenarios involving object occlusion.
4) Multiple datasets are used to validate the effectiveness of the proposed Dynam-LVIO algorithm. The experimental results demonstrate its superiority over the current state-of-the-art SLAM in terms of localization and mapping accuracy in dynamic environments.

The article structure is as follows: Section II provides an overview of the proposed algorithm, Section III details 3-D state estimation for objects, Section IV explains the flow of LVI-SORT for object tracking, Section V covers experiments and discussions, and Section VI summarizes the article and suggests future research directions.

II. Overview

A. Framework

The Dynam-LVIO framework, depicted in Fig. 1, comprises two essential components: 3-D object state estimation and LVI-SORT. The 3-D object state estimation is designed to estimate the object states with ESIKF. And LVI-SORT is proposed to achieve MOT for the 2-D bounding boxes derived from YOLO-V5. The detail process of the proposed Dynam-LVIO is outlined as follows: first, the recent global map and vehicle states are obtained with a standard LVIO. And the object map points are projected onto image to obtain the projection points. These projection points are then tracked using Lucas–Kanade (LK) optical flow to form the reprojection error. Concurrently, the ICP error is derived by comparing object points with the recent global map points. Subsequently, assuming that the object moves at a uniform speed over a brief period, the object state can be effectively predicted. Furthermore, the object state is updated based on the previously calculated ICP error and reprojection error, thereby achieving stable 3-D object state estimation within the framework of ESIKF. Next, the 2-D dynamic object flow is calculated with vehicle states, 3-D object states, and the dynamic object map points. In the proposed LVI-SORT, the 2-D object flow is used for the object propagation. In addition, the predicted object map points obtained with the 2-D object flow are used with IoU to calculate the hybrid cost matrix in the Hungarian algorithm, associating tracked objects with the newly observed objects from YOLO-V5. Finally, a simple method is proposed to determine whether the object is dynamic, and the recent global map obtained with LVIO is divided into dynamic object map points and static map points by the dynamic bounding boxes from LVI-SORT.

B. Preliminaries

The necessary preliminaries for this article are introduced in this section, including notation definitions, coordinate system, and Riemannian geometry.

1) Notations: The sets of real numbers and positive integers are, respectively, denoted as $\mathbb{R}$ and $\mathbb{Z}$. The m-dimensional real vector space is represented as $\mathbb{R}^m$, where $m \in \mathbb{Z}$. Vector are represented as bold lowercase letters, such as $\boldsymbol{t}$, and matrices are represented as bold uppercase letters, such as $\boldsymbol{R}$. The identity matrix is represented as $\boldsymbol{I}$. The transpose of a vector or matrix is represented as $(\cdot)^\top$. And $(\cdot)^\wedge$ maps a vector to a skew-symmetric matrix. Conversely, $(\cdot)^\vee$ maps a skew-symmetric matrix to a vector. The norm of a vector is represented as $\|\cdot\|$. $\hat{(\cdot)}$ represents the prior distribution for the vehicle or object state, and $\check{(\cdot)}$ represents the posterior distribution for the vehicle state or object state. Measurement values for state are represented as $\tilde{(\cdot)}$.

2) Coordinate System: The world frame is represented as w. The nth camera frame is represented as c_n, and the nth image frame is represented as i_n. The kth tracked object frame is represented as o_k. The start image frame is coincide with the world frame. The rotation vector and matrix representing the transformation from frame b to w are, respectively, defined as ${}^w\boldsymbol{\phi}_b$ and ${}^w\boldsymbol{R}_b$.

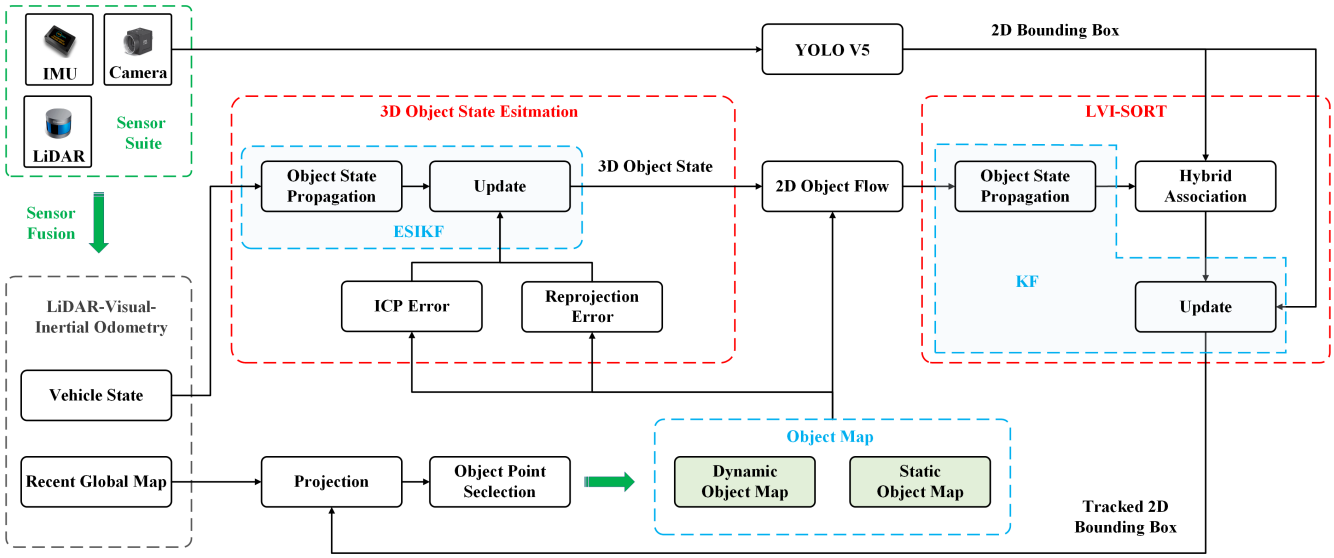


Fig. 1. Block diagram of the proposed Dynam-LVIO framework.

For a 3-D point $\boldsymbol{p}$ described in camera frame c_n, the projection of this point on the image frame i_n can be written as

$$\boldsymbol{\pi}(\boldsymbol{p}) = \left[f_x \frac{\boldsymbol{p}_x}{\boldsymbol{p}_z} + c_x,\ f_y \frac{\boldsymbol{p}_y}{\boldsymbol{p}_z} + c_y \right] \tag{1}$$

where f_x, f_y, c_x, c_y are the camera intrinsic and $\boldsymbol{p} = [\boldsymbol{p}_x, \boldsymbol{p}_y, \boldsymbol{p}_z]$. And $\partial \boldsymbol{\pi}(\boldsymbol{p})/\partial \boldsymbol{p}$ is introduced here for the calculation of latter equation

$$\frac{\partial \boldsymbol{\pi}(\boldsymbol{p})}{\partial \boldsymbol{p}} = \frac{1}{\boldsymbol{p}_z} \begin{bmatrix} f_x & 0 & -f_x \dfrac{\boldsymbol{p}_x}{\boldsymbol{p}_z} \\ 0 & f_y & -f_y \dfrac{\boldsymbol{p}_y}{\boldsymbol{p}_z} \end{bmatrix}. \tag{2}$$

3) Special Orthogonal Group: SO(3) describes the group of 3-D rotation and is defined as follows:

$$\mathrm{SO}(3) \triangleq \left\{ \boldsymbol{R} \in \mathbb{R}^{3\times 3} : \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}, \det(\boldsymbol{R}) = 1 \right\}. \tag{3}$$

The tangent space to SO(3) is denoted as $\mathfrak{so}(3)$. For notational convenience, the same version of the "vectorized" exponential and logarithm map as presented in [42] is adopted to describe the relationship between SO(3) and $\mathfrak{so}(3)$

$$\mathrm{Exp}(\boldsymbol{\phi}) = \boldsymbol{I} + \frac{\sin(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|} \boldsymbol{\phi}^\wedge + \frac{1-\cos(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|^2} \left(\boldsymbol{\phi}^\wedge\right)^2 \tag{4}$$

$$\mathrm{Log}(\boldsymbol{R}) = \left(\frac{\psi \cdot \left(\boldsymbol{R} - \boldsymbol{R}^\top\right)}{2\sin(\psi)} \right)^\vee \tag{5}$$

$$\psi = \cos^{-1}\left(\frac{\mathrm{tr}(\boldsymbol{R}) - 1}{2} \right) \tag{6}$$

where ϕ is the rotation vector, and $\boldsymbol{\phi}^\wedge \in \mathfrak{so}(3)$, $\mathrm{Exp}(\boldsymbol{\phi}) \in \mathrm{SO}(3)$.

Furthermore, three important properties are introduced here and will be used in Sections III and IV. The first property is an important first-order approximation

$$\mathrm{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) \approx \mathrm{Exp}(\boldsymbol{\phi})\mathrm{Exp}(\boldsymbol{J}_r(\boldsymbol{\phi})\delta\boldsymbol{\phi}) \tag{7}$$

where $\boldsymbol{J}_r(\boldsymbol{\phi})$ is the right Jacobian of SO(3)

$$\boldsymbol{J}_r(\boldsymbol{\phi}) = \boldsymbol{I} - \frac{1-\cos(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|^2} \boldsymbol{\phi}^\wedge + \frac{\|\boldsymbol{\phi}\| - \sin(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}^3\|} \left(\boldsymbol{\phi}^\wedge\right)^2. \tag{8}$$

The second useful property is

$$\boldsymbol{R}\mathrm{Exp}(\boldsymbol{\phi})\boldsymbol{R}^\top = \mathrm{Exp}(\boldsymbol{R}\boldsymbol{\phi}). \tag{9}$$

The third property is

$$\boldsymbol{R}\mathrm{Exp}(\delta\boldsymbol{\phi})\boldsymbol{t} \approx \boldsymbol{R}\left(\boldsymbol{I} + \delta\boldsymbol{\phi}^\wedge\right)\boldsymbol{t} = \boldsymbol{R}\boldsymbol{t} - \boldsymbol{R}\boldsymbol{t}^\wedge\delta\boldsymbol{\phi}. \tag{10}$$

4) Manifold Operator: $\mathcal{M} = \mathrm{SO}(3) \times \mathbb{R}^n$ represents the manifold on which state vector lies. And the manifold operators $\boxplus$ and $\boxminus$ are used to describe the encapsulated operation on $\mathcal{M}$, facilitating succinct expressions in the mathematics of state estimation. The details of the manifold operator $\boxplus$ and $\boxminus$ are defined as [43]

$$\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{a}_1 \end{bmatrix} \boxplus \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{a}_2 \end{bmatrix} \triangleq \begin{bmatrix} \boldsymbol{R} \cdot \mathrm{Exp}(\boldsymbol{\phi}) \\ \boldsymbol{a}_1 + \boldsymbol{a}_2 \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{a}_1 \end{bmatrix} \boxminus \begin{bmatrix} \boldsymbol{R}_2 \\ \boldsymbol{a}_2 \end{bmatrix} \triangleq \begin{bmatrix} \mathrm{Log}\left(\boldsymbol{R}_2^\top \boldsymbol{R}_1\right) \\ \boldsymbol{a}_1 - \boldsymbol{a}_2 \end{bmatrix} \tag{12}$$

where $\boldsymbol{R}_1, \boldsymbol{R}_2 \in \mathrm{SO}(3)$, $\boldsymbol{\phi}$ is the rotation vector, $\boldsymbol{a}_1, \boldsymbol{a}_2 \in \mathbb{R}^n$, $\mathrm{Exp}(\cdot)$ represents the exponential mapping from rotation vector to rotation matrix, while $\mathrm{Log}(\cdot)$ represents the logarithmic mapping from the rotation matrix to the rotation vector.

III. 3-D State Estimation for Objects

In urban environments, dynamic objects are treated as rigid objects exhibiting uniform velocity within a brief period [44]. And the object states can be propagated on manifold under this assumption. Subsequently, a specific number of dynamic object map points are projected onto the current image frame, and the LK optical flow [45] is used to track these projection points, allowing the calculation of the reprojection error. Moreover, by iteratively identifying the closest point of each dynamic object map point in the recent global map, the ICP error can be determined. The reprojection error and ICP error

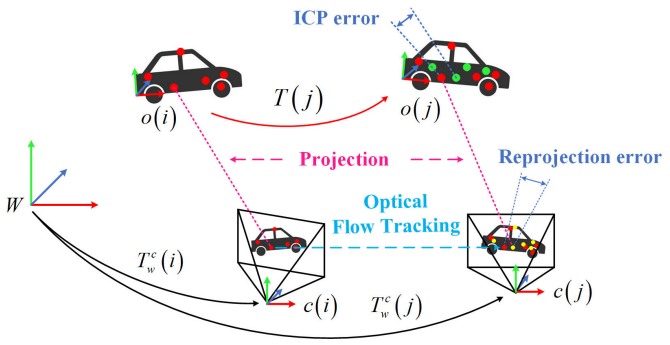


Fig. 2. Measurements of reprojection error and ICP error used in ESIKF.

are depicted in Fig. 2. Due to the highly nonlinear nature of the objective function for object state estimation, ESIKF [46] is adopted to estimate the 3-D object state, considering both accuracy and computational efficiency. Using the aforementioned errors, the update process of ESIKF can be executed, achieving 3-D object state estimation.

1) State Vector: In this work, the nth object pose is estimated using ESIKF, and the object state vector $\boldsymbol{x}^n$ follows the same definition as in [47], as shown below:

$$\boldsymbol{x}^n \triangleq \begin{bmatrix} \boldsymbol{\phi}^n & \boldsymbol{t}^n & \boldsymbol{\omega}^n & \boldsymbol{v}^n \end{bmatrix} \tag{13}$$

where $\boldsymbol{\omega}^n$ is the angular rate between object frames $o(i)$ and $o(j)$, and $\boldsymbol{v}^n$ is the velocity of the nth object in world frame w. $\boldsymbol{\phi}^k$ is the rotation vector between object frames $o(i)$ and $o(j)$, and $\boldsymbol{t}^n$ is the translation between object frames $o(i)$ and $o(j)$. $\boldsymbol{\phi}^n$ and $\boldsymbol{t}^n$ can be consisted to form the pose transformation matrix $\boldsymbol{T}^n$ as follows:

$$\boldsymbol{T}^n = \begin{bmatrix} \boldsymbol{R}^n & \boldsymbol{t}^n \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{Exp}(\boldsymbol{\phi}^n) & \boldsymbol{t}^n \\ 0 & 1 \end{bmatrix} \in \boldsymbol{SE}(3). \tag{14}$$

And $\boldsymbol{T}^n$ can be used to transform object points in various moments as follows:

$$\boldsymbol{p}^n(j) = \boldsymbol{T}^n \boldsymbol{p}^n(i) \tag{15}$$

where $\boldsymbol{p}^n(i)$ describes the point on the nth object in world frame w at timestamp i, and $\boldsymbol{p}^n(j)$ describes the point on the nth object in world frame w at timestamp j.

To simplify the derivation of the formula, the upper corner marker n in the state vector is omitted and the state vector can be rewritten as

$$\boldsymbol{x} \triangleq [\boldsymbol{\phi} \quad \boldsymbol{t} \quad \boldsymbol{\omega} \quad \boldsymbol{v}] \in \mathbb{R}^{12}. \tag{16}$$

2) State Transition Model: With the assumption that the objects are rigid and undergo uniform motion for a short period of time, the kinematic model of object can be written as follows:

$$\dot{\boldsymbol{R}} = \boldsymbol{R}(\boldsymbol{\omega})^{\wedge} \quad \dot{\boldsymbol{t}} = \boldsymbol{v} \quad \dot{\boldsymbol{\omega}} = \boldsymbol{0} \quad \dot{\boldsymbol{v}} = \boldsymbol{0}. \tag{17}$$

By integrating (17) between two consecutive moments i and j, the object state at moment j can be obtained as follows:

$$\begin{aligned} \boldsymbol{R}(j) &= \boldsymbol{R}(i)\mathrm{Exp}(\boldsymbol{\omega}(i)\Delta t) \\ \boldsymbol{t}(j) &= \boldsymbol{t}(i) + \boldsymbol{v}(i)\Delta t \\ \boldsymbol{\omega}(j) &= \boldsymbol{\omega}(i) \quad \boldsymbol{v}(j) = \boldsymbol{v}(i) \end{aligned} \tag{18}$$

where Δt is the time interval between i and j.

The above object state transition model described in (18) can be rewritten as follows:

$$\boldsymbol{x}(j) = \boldsymbol{x}(i) \boxplus (\boldsymbol{f}(\boldsymbol{x}(i)) \cdot \Delta t) \tag{19}$$

where the function $\boldsymbol{f}(\boldsymbol{x}(i))$ is defined as

$$\boldsymbol{f}(\boldsymbol{x}(i)) = \begin{bmatrix} \boldsymbol{\omega}(i) & \boldsymbol{v}(i) & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{\top} \in \mathbb{R}^{12}. \tag{20}$$

3) Propagation: In this section, the error state concept is used to prevent singularity during the minimal parameterization process with the original object state [48]. The object error state, denoted as $\delta\hat{\boldsymbol{x}}(j)$, is defined as follows:

$$\begin{aligned} \delta\hat{\boldsymbol{x}}(j) &\triangleq \boldsymbol{x}(j) \boxminus \hat{\boldsymbol{x}}(j) \\ &= \left[\delta\hat{\boldsymbol{\phi}}(j), \delta\hat{\boldsymbol{t}}(j), \delta\hat{\boldsymbol{\omega}}(j), \delta\hat{\boldsymbol{v}}(j)\right] \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(j)}\right) \end{aligned} \tag{21}$$

where $\boldsymbol{x}(j)$ is the true object state at moment j. And $\hat{\boldsymbol{x}}(j)$ is the prior estimate of $\boldsymbol{x}(j)$, which can be propagated as follows:

$$\hat{\boldsymbol{x}}(j) = \hat{\boldsymbol{x}}(i) \boxplus \left(\boldsymbol{f}\left(\hat{\boldsymbol{x}}(i)\right) \cdot \Delta t\right). \tag{22}$$

Furthermore, $\delta\hat{\boldsymbol{\phi}}(j), \delta\hat{\boldsymbol{t}}(j), \delta\hat{\boldsymbol{\omega}}(j), \delta\hat{\boldsymbol{v}}(j)$ in (21) can be calculated as (12)

$$\begin{aligned} \delta\hat{\boldsymbol{\phi}}(j) &= \mathrm{Log}\left(\hat{\boldsymbol{R}}(j)\boldsymbol{R}(j)\right) \\ \delta\hat{\boldsymbol{t}}(j) &= \boldsymbol{t}(j) - \hat{\boldsymbol{t}}(j) \\ \delta\hat{\boldsymbol{\omega}}(j) &= \boldsymbol{\omega}(j) - \hat{\boldsymbol{\omega}}(j) \\ \delta\hat{\boldsymbol{v}}(j) &= \boldsymbol{v}(j) - \hat{\boldsymbol{v}}(j). \end{aligned} \tag{23}$$

With (19), (21), and (22), the error state $\delta\hat{\boldsymbol{x}}(j)$ can be represented as follows:

$$\begin{aligned} \delta\hat{\boldsymbol{x}}(j) &= \boldsymbol{x}(j) \boxminus \hat{\boldsymbol{x}}(j) \\ &= (\boldsymbol{x}(i) \boxplus (\boldsymbol{f}(\boldsymbol{x}(i))\Delta t)) \boxminus \left(\hat{\boldsymbol{x}}(i) \boxplus \left(\boldsymbol{f}\left(\hat{\boldsymbol{x}}(i)\right)\Delta t\right)\right) \\ &= \begin{bmatrix} \mathrm{Log}\left(\begin{matrix} \left(\hat{\boldsymbol{R}}(i)\mathrm{Exp}(\boldsymbol{\omega}(i)\Delta t)\right)^{\top} \cdot \\ \hat{\boldsymbol{R}}(i)\mathrm{Exp}\left(\delta\hat{\boldsymbol{\phi}}(i)\right)\mathrm{Exp}\left((\boldsymbol{\omega}(i) + \delta\hat{\boldsymbol{\omega}}(i))\Delta t\right) \end{matrix}\right) \\ \delta\hat{\boldsymbol{t}}(i) + \delta\hat{\boldsymbol{v}}(i)\Delta t \\ \delta\hat{\boldsymbol{\omega}}(i) \\ \delta\hat{\boldsymbol{v}}(i) \end{bmatrix}. \end{aligned} \tag{24}$$

With (7) and (9), (24) can be approximated in a simplified form as follows:

$$\delta\hat{\boldsymbol{x}}(j) \approx \begin{bmatrix} \mathrm{Exp}(-\boldsymbol{\omega}(i)\Delta t)\delta\hat{\boldsymbol{\phi}}(i) + \boldsymbol{J}_r(\boldsymbol{\omega}(i)\Delta t)\delta\hat{\boldsymbol{\omega}}(i)\Delta t \\ \delta\hat{\boldsymbol{t}}(i) + \delta\hat{\boldsymbol{v}}(i)\Delta t \\ \delta\hat{\boldsymbol{\omega}}(i) \\ \delta\hat{\boldsymbol{v}}(i) \end{bmatrix}. \tag{25}$$

Furthermore, transforming (25) into matrix form and considering the propagating noise, (25) can be rewritten as

$$\delta\hat{\boldsymbol{x}}(j) \approx \boldsymbol{F}_{\delta\hat{\boldsymbol{x}}(i)}\delta\hat{\boldsymbol{x}}(i) + \boldsymbol{w}(i) \tag{26}$$

where

$$\boldsymbol{F}_{\delta\hat{\boldsymbol{x}}(i)} = \begin{bmatrix} \mathrm{Exp}(-\boldsymbol{\omega}(i)\Delta t) & \boldsymbol{0} & \boldsymbol{J}_r(\boldsymbol{\omega}(i)\Delta t)\Delta t & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{I}\Delta t \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}. \tag{27}$$

And $\delta\hat{\boldsymbol{x}}(j) \sim \mathcal{N}(\mathbf{0}_{6\times1}, \boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(j)})$, $\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(j)}$ is the covariance of the error state $\delta\hat{\boldsymbol{x}}(j)$, which can be calculated with

$$\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(j)} = \boldsymbol{F}_{\delta\hat{\boldsymbol{x}}(i)}\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(i)}\boldsymbol{F}^{\top}_{\delta\hat{\boldsymbol{x}}(i)} + \boldsymbol{Q}_i \tag{28}$$

where $\boldsymbol{Q}(i)$ is the covariance of the propagation noise $\boldsymbol{w}(i)$.

4) Reprojection Error Measurement: The reprojection error measurement can be calculated with the projection points, which are obtained by projecting object map points onto the image. The object map points of the nth object are represented as $\mathcal{P}^n(i) = \{\boldsymbol{P}^n_1(i), \ldots, \boldsymbol{P}^n_m(i)\}$. With the posteriori object state estimation $\check{\boldsymbol{x}}(j)$, the object map point $\mathcal{P}^n(j)$ can be obtained as

$$\mathcal{P}^n(j) = \left\{\check{\boldsymbol{R}}(j)\mathcal{P}^n_s(i) + \check{\boldsymbol{t}}(j), s = 1, \ldots, m\right\} \tag{29}$$

where $\check{\boldsymbol{R}}(j) = \mathrm{Exp}(\check{\boldsymbol{\phi}}(j))$.

Furthermore, the object map points $\mathcal{P}^n(i)$ can be projected on the ith image frame b_n and the projection points $\boldsymbol{p}^n(i)$ can be obtained as follows:

$$\boldsymbol{p}^n_s(i) = \boldsymbol{\pi}\left(\boldsymbol{R}^c_w(i)\mathcal{P}^n_s(i) + \boldsymbol{t}^c_w(i)\right) \tag{30}$$

where $\boldsymbol{\pi}(\cdot)$ describes the projection from the camera frame to the image frame, and

$$\boldsymbol{R}^c_w(i) = \left(\boldsymbol{R}^w_c(i)\right)^{\top} \tag{31}$$

$$\boldsymbol{t}^c_w(i) = -\left(\boldsymbol{R}^w_c(i)\right)^{\top}\boldsymbol{t}^w_c(i). \tag{32}$$

$(\boldsymbol{R}^w_c(i), \boldsymbol{t}^w_c(i))$ is the camera state and obtained with a standard LVIO.

According (29) and (30), the project points $\boldsymbol{p}^n(j) = \{\boldsymbol{p}^n_1(j), \ldots, \boldsymbol{p}^n_m(j)\}$ can be obtained through two methods. One method involves tracking $\boldsymbol{p}^n(i)$ with LK optical flow. The other method involves projecting $\mathcal{P}^n(j)$ on the jth image frame $\boldsymbol{I}_j$. By comparing $\boldsymbol{p}^n(j)$ obtained from the above two methods, the following reprojection error can be derived:

$$\begin{aligned}
&\boldsymbol{r}^{\mathrm{rep}}\left(\boldsymbol{x}(j), \boldsymbol{p}^n(j), \mathcal{P}^n(i)\right)\\
&= \sum_{s=1}^{m}\left(\boldsymbol{p}^n_s(j) - \check{\boldsymbol{p}}^n_s(j)\right)\\
&= \sum_{s=1}^{m}\left(\boldsymbol{p}^n_s(j) - \boldsymbol{\pi}\left(\boldsymbol{R}^c_w(j)\mathcal{P}^n_s(j) + \boldsymbol{t}^c_w(j)\right)\right)\\
&= \sum_{s=1}^{m}\left(\boldsymbol{p}^n_s(j) - \boldsymbol{\pi}\left(\boldsymbol{R}^c_w(j)\left(\check{\boldsymbol{R}}(j)\mathcal{P}^n_s(i) + \check{\boldsymbol{t}}(j)\right) + \boldsymbol{t}^c_w(j)\right)\right).
\end{aligned} \tag{33}$$

The sth 3-D point $\mathcal{P}^n_s(i)$ and its projection $\boldsymbol{p}^n_s(j)$ are selected as example to calculate reprojection error $\boldsymbol{r}^{\mathrm{rep}}(\boldsymbol{x}(j), \boldsymbol{p}^n(j), \mathcal{P}^n(i))$ in (33). The measurements of $\mathcal{P}^n_s(i)$ and $\boldsymbol{p}^n_s(j)$ are represented as $\tilde{\mathcal{P}}^n_s(i)$ and $\tilde{\boldsymbol{p}}^n_s(j)$. And the measurements noise are as follows:

$$\tilde{\mathcal{P}}^n_s(i) = \mathcal{P}^n_s(i) + \boldsymbol{n}_{\mathcal{P}_s} \tag{34}$$

$$\tilde{\boldsymbol{p}}^n_s(j) = \boldsymbol{p}^n_s(j) + \boldsymbol{n}_{\boldsymbol{p}_s} \tag{35}$$

where $\boldsymbol{n}_{\mathcal{P}_s} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{P}_s})$, and $\boldsymbol{n}_{\boldsymbol{p}_s} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{p}_s})$.

Therefore, the true zero reprojection error can be obtained with (34) and (35) as follows:

$$\begin{aligned}
\mathbf{0} &= \boldsymbol{r}^{\mathrm{rep}}_s\left(\boldsymbol{x}(j), \boldsymbol{p}^n_s(j), \mathcal{P}^n_s(i)\right)\\
&= \boldsymbol{r}^{\mathrm{rep}}_s\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}^n_s(j) - \boldsymbol{n}_{\boldsymbol{p}_s}, \tilde{\mathcal{P}}^n_s(i) - \boldsymbol{n}_{\mathcal{P}_s}\right).
\end{aligned} \tag{36}$$

Then, using a first-order expansion, (36) can be simplified as follows:

$$\begin{aligned}
\mathbf{0} &= \boldsymbol{r}^{\mathrm{rep}}_s\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}^n_s(j) - \boldsymbol{n}_{\boldsymbol{p}_s}, \tilde{\mathcal{P}}^n_s(i) - \boldsymbol{n}_{\mathcal{P}_s}\right)\\
&\approx \boldsymbol{r}^{\mathrm{rep}}_s\left(\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}^n_s(j), \tilde{\mathcal{P}}^n_s(i)\right) + \boldsymbol{H}^{\mathrm{rep}}_s\delta\check{\boldsymbol{x}}(j) + \boldsymbol{\alpha}_s
\end{aligned} \tag{37}$$

where

$$\boldsymbol{H}^{\mathrm{rep}}_s = \left.\frac{\partial \boldsymbol{r}^{\mathrm{rep}}_s\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}^n_s(j), \tilde{\mathcal{P}}^n_s(i)\right)}{\partial\delta\check{\boldsymbol{x}}(j)}\right|_{\delta\check{\boldsymbol{x}}(j)=\mathbf{0}} \tag{38}$$

and $\boldsymbol{\alpha}_s \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\alpha}_s})$, and $\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_s}$ can be obtained as

$$\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_s} = \boldsymbol{\Sigma}_{\boldsymbol{p}_s} + \boldsymbol{F}_{\mathcal{P}_s}\boldsymbol{\Sigma}_{\mathcal{P}_s}\boldsymbol{F}^{\top}_{\mathcal{P}_s} \tag{39}$$

$$\boldsymbol{F}_{\mathcal{P}_s} = \left.\frac{\partial \boldsymbol{r}^{\mathrm{rep}}_s\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}^n_s(j), \tilde{\mathcal{P}}^n_s(i)\right)}{\partial\tilde{\mathcal{P}}^n_s(i)}\right|_{\delta\check{\boldsymbol{x}}(j)=\mathbf{0}}. \tag{40}$$

The detail calculation process of $\boldsymbol{H}^{\mathrm{rep}}_s$ and $\boldsymbol{F}_{\mathcal{P}_s}$ can be found in Appendix A.

5) ICP Error Measurement: The reprojection error measurement mentioned above may be unreliable when the object is positioned at a considerable distance from the vehicle. To address this issue, the utilization of ICP error measurement, which uses object map points and recent global map points, can enhance the accuracy of long-distance object state estimation. For each point in the nth object map points $\mathcal{P}^n(i)$, the nearest point is searched in recent global map points $\mathcal{P}^w(j)$ with kd-tree and the error can be calculated as follows:

$$\begin{aligned}
&\boldsymbol{r}^{\mathrm{icp}}\left(\boldsymbol{x}(j), \mathcal{P}^n(i), \mathcal{P}^w(j)\right)\\
&= \sum_{q=1}^{u}\left(\mathcal{P}^w_q(j) - \check{\mathcal{P}}^n_q(j)\right)\\
&= \sum_{q=1}^{u}\left(\mathcal{P}^w_q(j) - \left(\check{\boldsymbol{R}}(j)\mathcal{P}^n_q(i) + \check{\boldsymbol{t}}(j)\right)\right).
\end{aligned} \tag{41}$$

And the qth points $\mathcal{P}^n_q(i)$ and $\mathcal{P}^w_q(j)$ are selected as example to calculate the ICP error $\boldsymbol{r}^{\mathrm{icp}}(\boldsymbol{x}(j), \mathcal{P}^n(i), \mathcal{P}^w(j))$ in (41). The measurements of $\mathcal{P}^n_q(i)$ and $\mathcal{P}^w_q(j)$ are represented as $\tilde{\mathcal{P}}^n_q(i)$ and $\tilde{\mathcal{P}}^w_q(j)$, respectively, and the noise is depicted as follows:

$$\tilde{\mathcal{P}}^n_q(i) = \mathcal{P}^n_q(i) + \boldsymbol{n}_{\mathcal{P}^n_q} \tag{42}$$

$$\tilde{\mathcal{P}}^w_q(j) = \mathcal{P}^w_q(j) + \boldsymbol{n}_{\mathcal{P}^w_q}. \tag{43}$$

The true zero ICP error can be obtained as

$$\begin{aligned}
\mathbf{0} &= \boldsymbol{r}^{\mathrm{icp}}_q\left(\boldsymbol{x}(j), \mathcal{P}^n_q(i), \mathcal{P}^w_q(j)\right)\\
&= \boldsymbol{r}^{\mathrm{icp}}_q\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\mathcal{P}}^n_q(i) - \boldsymbol{n}_{\mathcal{P}^n_q}, \tilde{\mathcal{P}}^w_q(j) - \boldsymbol{n}_{\mathcal{P}^w_q}\right).
\end{aligned} \tag{44}$$

Then, using the first-order expansion, (44) can be simplified as follows:

$$\begin{aligned}
\mathbf{0} &= \boldsymbol{r}^{\mathrm{icp}}_q\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\mathcal{P}}^n_q(i) - \boldsymbol{n}_{\mathcal{P}^n_q}, \tilde{\mathcal{P}}^w_q(j) - \boldsymbol{n}_{\mathcal{P}^w_q}\right)\\
&\approx \boldsymbol{r}^{\mathrm{icp}}_q\left(\check{\boldsymbol{x}}(j), \tilde{\mathcal{P}}^n_q(i), \tilde{\mathcal{P}}^w_q(j)\right) + \boldsymbol{H}^{\mathrm{icp}}_q\delta\check{\boldsymbol{x}}(j) + \boldsymbol{\beta}_q
\end{aligned} \tag{45}$$

where

$$\boldsymbol{H}^{\mathrm{icp}}_q = \left.\frac{\partial \boldsymbol{r}^{\mathrm{icp}}_q\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\mathcal{P}}^n_q(i), \tilde{\mathcal{P}}^w_q(j)\right)}{\partial\delta\check{\boldsymbol{x}}(j)}\right|_{\delta\check{\boldsymbol{x}}(j)=\mathbf{0}} \tag{46}$$

and $\boldsymbol{\beta}_q \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}_q})$, and $\boldsymbol{\Sigma}_{\boldsymbol{\beta}_q}$ can be obtained as

$$\boldsymbol{\Sigma}_{\boldsymbol{\beta}_q} = \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{P}}_q^w} + \boldsymbol{F}_{\boldsymbol{\mathcal{P}}_q^n} \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{P}}_q^n} \boldsymbol{F}_{\boldsymbol{\mathcal{P}}_q^n}^{\top} \tag{47}$$

$$\boldsymbol{F}_{\boldsymbol{\mathcal{P}}_q} = \left. \frac{\partial \boldsymbol{r}_q^{\text{icp}}\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_q^n(i), \tilde{\boldsymbol{\mathcal{P}}}_q^w(j)\right)}{\partial \tilde{\boldsymbol{\mathcal{P}}}_q^n(i)} \right|_{\delta\check{\boldsymbol{x}}(j)=\mathbf{0}}. \tag{48}$$

The detail calculation process of $\boldsymbol{H}_q^{\text{icp}}$ and $\boldsymbol{F}_{\boldsymbol{\mathcal{P}}_q^n}$ can be found in Appendix B.

6) Update of ESIKF: By leveraging reprojection error, ICP error, and the prior estimate of $\delta\boldsymbol{x}(j)$ obtained with (22), (37), and (45), the MAP estimation of $\delta\check{\boldsymbol{x}}(j)$ can be obtained by optimizing the following objection function:

$$\begin{aligned} \min_{\delta\check{\boldsymbol{x}}(j)} \Bigg(& \left\| \check{\boldsymbol{x}}(j) \boxminus \hat{\boldsymbol{x}}(j) + \boldsymbol{\mathcal{H}}\delta\check{\boldsymbol{x}}(j) \right\|_{\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(j)}}^2 \\ & + \sum_{s=1}^{m} \left\| \boldsymbol{r}_s^{\text{rep}}\left(\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}_s^n(j), \tilde{\boldsymbol{\mathcal{P}}}_s^n(i)\right) + \boldsymbol{H}_s^{\text{rep}}\delta\check{\boldsymbol{x}}(j) \right\|_{\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_s}}^2 \\ & + \sum_{q=1}^{u} \left\| \boldsymbol{r}_q^{\text{icp}}\left(\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_q^n(i), \tilde{\boldsymbol{\mathcal{P}}}_q^w(j)\right) + \boldsymbol{H}_q^{\text{icp}}\delta\check{\boldsymbol{x}}(j) \right\|_{\boldsymbol{\Sigma}_{\boldsymbol{\beta}_q}}^2 \Bigg) \end{aligned} \tag{49}$$

where $\boldsymbol{\mathcal{H}}$ can be obtained as

$$\begin{aligned} \boldsymbol{\mathcal{H}} &= \left. \frac{\partial\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j) \boxminus \hat{\boldsymbol{x}}(j)\right)}{\partial\delta\check{\boldsymbol{x}}(j)} \right|_{\partial\delta\check{\boldsymbol{x}}(j)=0} \\ &= \begin{bmatrix} \boldsymbol{J}_r^{-1}\left(\text{Log}\left(\left(\hat{\boldsymbol{R}}(j)\right)^{\top}\check{\boldsymbol{R}}(j)\right)\right) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{3\times3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_{3\times3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I}_{3\times3} \end{bmatrix}. \end{aligned} \tag{50}$$

The optimal solution under the constraints of (49) can be solved using quadratic programming, which is essentially the ESIKF update process. Following [49], the ESIKF gain $\boldsymbol{K}$ can be computed as

$$\boldsymbol{K} = \left(\boldsymbol{H}^{\top}\boldsymbol{R}^{-1}\boldsymbol{H} + \boldsymbol{P}^{-1}\right)^{-1}\boldsymbol{H}^{\top}\boldsymbol{R}^{-1} \tag{51}$$

where

$$\boldsymbol{H} = \left[\boldsymbol{H}_1^{\text{rep}}, \ldots, \boldsymbol{H}_m^{\text{rep}}, \boldsymbol{H}_1^{\text{icp}}, \ldots, \boldsymbol{H}_u^{\text{icp}}\right] \tag{52}$$

$$\boldsymbol{R} = \text{diag}\left(\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_1}, \ldots, \boldsymbol{\Sigma}_{\boldsymbol{\alpha}_m}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}_1}, \ldots, \boldsymbol{\Sigma}_{\boldsymbol{\beta}_u}\right) \tag{53}$$

$$\boldsymbol{P} = (\boldsymbol{\mathcal{H}})^{-1}\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(j)}(\boldsymbol{\mathcal{H}})^{-\top}. \tag{54}$$

Then, the posteriori estimation of object state $\check{\boldsymbol{x}}(j)$ can be obtained as

$$\check{\boldsymbol{x}}(j) = \check{\boldsymbol{x}}(j) \boxplus \left(-\boldsymbol{K}\check{\boldsymbol{z}}(j) - (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{\mathcal{H}}^{-1}\left(\check{\boldsymbol{x}}(j) \boxminus \hat{\boldsymbol{x}}(j)\right)\right) \tag{55}$$

where

$$\begin{aligned} \check{\boldsymbol{z}}(j) = \Big\{ & \boldsymbol{r}_1^0\left(\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}_1^n(j), \tilde{\boldsymbol{\mathcal{P}}}_1^n(i)\right), \ldots, \\ & \boldsymbol{r}_m^{\text{rep}}\left(\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}_m^n(j), \tilde{\boldsymbol{\mathcal{P}}}_m^n(i)\right), \\ & \boldsymbol{r}_1^{\text{icp}}\left(\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_1^n(i), \tilde{\boldsymbol{\mathcal{P}}}_1^w(j)\right), \ldots, \\ & \boldsymbol{r}_u^{\text{icp}}\left(\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_u^n(i), \tilde{\boldsymbol{\mathcal{P}}}_u^w(j)\right)\Big\}. \end{aligned} \tag{56}$$

Algorithm 1 3-D Object State Estimation With ESIKF

Input: recent global map points $\boldsymbol{\mathcal{P}}^w(j)$,
nth object map points $\boldsymbol{\mathcal{P}}^n(i)$,
image $\boldsymbol{I}_i$ and $\boldsymbol{I}_j$,
camera state $\left(\boldsymbol{R}_c^w(j), \boldsymbol{t}_c^w(j)\right)$,
object state $\boldsymbol{x}(i)$.
Output: object state $\boldsymbol{x}(j)$,
nth object map points $\boldsymbol{\mathcal{P}}^n(j)$.

1: Propagate the object state $\boldsymbol{x}(i)$ as (22) to obtain $\hat{\boldsymbol{x}}(j)$.
2: Project nth object map points $\boldsymbol{\mathcal{P}}^n(i)$ onto image $\boldsymbol{I}_i$ as (30) to obtain $\boldsymbol{p}^n(i)$, and then track $\boldsymbol{p}^n(i)$ with LK optical flow to obtain $\boldsymbol{p}^n(j)$.
3: Calculate the reprojection error measurements $\left\{\boldsymbol{r}_s^{\text{rep}}, s = 1, \ldots, n\right\}$ with $\hat{\boldsymbol{x}}(j)$, $\boldsymbol{p}^n(j)$, and $\boldsymbol{\mathcal{P}}^n(i)$ as (37).
4: Calculate the ICP error measurements $\left\{\boldsymbol{r}_q^{\text{icp}}, q = 1, \ldots, u\right\}$ with $\hat{\boldsymbol{x}}(j)$, $\boldsymbol{\mathcal{P}}^n(i)$ and $\boldsymbol{\mathcal{P}}^w(j)$ as (45).
5: **while** the object state updation in (55) less than threshold **do**
6: Update object state $\check{\boldsymbol{x}}(j)$ as (55).
7: **end while**
8: Update object state $\hat{\boldsymbol{x}}(j)$ as (57).
9: Update covariance of object error state $\delta_{\hat{\boldsymbol{x}}(j)}$ as (58).
10: Update object map points as (29).

The update procession depicted in (55) is iterated until convergence. And then the convergence state is used to update the object map points as (29). Furthermore, the convergence state is served as the initial state for the propagation for the next cycle

$$\hat{\boldsymbol{x}}(j) = \check{\boldsymbol{x}}(j) \tag{57}$$

$$\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(j)} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{\Sigma}_{\delta\hat{\boldsymbol{x}}(j)}. \tag{58}$$

The detail process of the object state estimation is displayed in Algorithm 1.

IV. LVI-SORT

The proposed MOT algorithm, LVI-SORT, is introduced in this section to track 2-D bounding boxes generated by YOLO-V5. In the proposed LVI-SORT, "LVI" stands for LiDAR–visual–inertial, while "SORT" originates from [31], referring to "simple online and real-time tracking." SORT is a pioneering MOT algorithm implemented with the tracking-by-detection paradigm. And the proposed LVI-SORT is built upon SORT, presenting localization enhancements within the LVIO framework. The concept of 2-D object flow, akin to the object scene flow as presented in [50], is determined by leveraging the object state $\boldsymbol{x}(j)$ and object map points $\boldsymbol{\mathcal{P}}^n(i)$ obtained from Section III, with the aim of enhancing the effectiveness of MOT.

Specifically, the 2-D object flow is initially used to predict the object state, ensuring precise object prediction in conditions of rapid motion. Subsequently, the object map points, transformed by the 3-D object state $\boldsymbol{x}(j)$, are used in conjunction with the IoU of bounding boxes to compute the

hybrid cost matrix in the Hungarian algorithm, establishing a stable association between the predicted object and the detected object. Finally, the object state in the Kalman filter is updated using the associated detected object. Given the similarity among the terms "object state," "object map points," and "object flow" in this article, it is necessary to distinguish these terms. The object state, defined by (13), represents the change in position and attitude of the object relative to the initial observation. The object map points are derived from splicing object point cloud data observed multiple times using the object state. The object flow, calculated with (60), represents the movement trends of object in image frame.

A. Prediction

1) 2-D Object Flow: With the nth object state $\boldsymbol{x}^n(j)$ calculated in Section III, the object map points $\mathcal{P}_s^n(i)$ can be transformed from moment i to moment j as follows:

$$\mathcal{P}_s^n(j) = \boldsymbol{R}^n(j)\mathcal{P}_s^n(i) + \boldsymbol{t}^n(j), \quad s = 1, \ldots, m \tag{59}$$

where m is the points' size of the nth object map.

By projecting $\mathcal{P}_s^n(j)$ and $\mathcal{P}_s^n(i)$ to the corresponding image frame with camera state $(\boldsymbol{R}_c^w, \boldsymbol{t}_c^w)$, 2-D object flow can be obtained as follows:

$$\begin{aligned}\boldsymbol{f}_s^n(j) = \boldsymbol{\pi}\left(\boldsymbol{R}_w^c(j)\mathcal{P}_s^n(j) + \boldsymbol{t}_w^c(j)\right) \\ -\boldsymbol{\pi}\left(\boldsymbol{R}_w^c(i)\mathcal{P}_s^n(i) + \boldsymbol{t}_w^c(i)\right)\end{aligned} \tag{60}$$

where $\boldsymbol{\pi}(\cdot)$ represents the transformation from the camera frame to the image frame as described in (1), $(\boldsymbol{R}_w^c(j), \boldsymbol{t}_w^c(j))$ is the camera state in moment j, and $(\boldsymbol{R}_w^c(i), \boldsymbol{t}_w^c(i))$ is the camera state in moment i.

Furthermore, the mean and covariance of 2-D object flow can be obtained as follows:

$$\boldsymbol{f}^n(j) = \frac{1}{m}\sum_{s=1}^{m}\left(\boldsymbol{f}_s^n(j)\right) \tag{61}$$

$$\mathrm{cov}_{f^n} = \frac{1}{m}\sum_{s=1}^{m}\left(\boldsymbol{f}^n(j) - \boldsymbol{f}_s^n(j)\right)\left(\boldsymbol{f}^n(j) - \boldsymbol{f}_s^n(j)\right)^\top. \tag{62}$$

2) Bounding Box State: The state vector for the mth object bounding box is defined as follows:

$$\boldsymbol{x}_b^m \triangleq \left[x_c, y_c, a, h, \dot{x_c}, \dot{y_c}, \dot{a}, \dot{h}\right] \tag{63}$$

where x_c and y_c are the image coordinates of the bounding box center, $\dot{x_c}$ and $\dot{y_c}$ are the velocities of the bounding box, a is the area of the bounding box, h is the height of the bounding box, and $\dot{a}$ and $\dot{h}$ are the derivatives of a and h, respectively. To simplify the derivation of the follow formula, the upper corner marker m is omitted and the bounding box state can be rewritten as $\boldsymbol{x}_b$. And to distinguish from 3-D object state $\boldsymbol{x}(j)$ in Section III, the lower corner marker will not be omitted.

3) State Transition Model: The kinematic equation with the 2-D object flow $\boldsymbol{f}(j)$ as active control is written as follows:

$$\hat{\boldsymbol{x}}_b(j) = \boldsymbol{F}(j)\check{\boldsymbol{x}}_b(i) + \boldsymbol{C}(j)\boldsymbol{f}(j) + \boldsymbol{n}(i) \tag{64}$$

where $\boldsymbol{F}(j)$ is the state transition matrix, and $\boldsymbol{C}(j)$ is the control matrix. The detail of $\boldsymbol{F}(j)$ and $\boldsymbol{C}(j)$ can be written as follows:

$$\boldsymbol{F}(j) = \begin{bmatrix} 1 & 0 & 0 & 0 & \alpha\cdot\Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \alpha\cdot\Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{65}$$

$$\boldsymbol{C}(j) = \begin{bmatrix} 1-\alpha & 0 \\ 0 & 1-\alpha \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{66}$$

And $\boldsymbol{n}(i)$ is the predict noise which is assumed to be independent and identically distributed with the normal distribution, e.g., $\boldsymbol{n}(i) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}(i))$. And α is a weighting factor, which can be calculated with the eigenvalue of the covariance cov_{f^n} calculated in (62). The detail of α is defined as follows:

$$\alpha = \begin{cases} 0.5, & \lambda_{\mathrm{cov}_f}^{\max} \geq t_\alpha \\ 0, & \lambda_{\mathrm{cov}_f}^{\max} < t_\alpha \end{cases} \tag{67}$$

where t_α is the threshold, and $\lambda_{\mathrm{cov}_f}^{\max}$ is the max eigenvalue of the covariance of the 2-D object flow $\boldsymbol{f}(j)$.

The distinction between (64) and the previous works [31], [33] lies in the method for calculating the predicted position of the bounding box. In contrast to assuming a constant velocity for the bounding box, as done in [31] and [33], where the object position prediction is solely based on the estimated velocity $\{\dot{x_c}, \dot{y_c}\}$, our approach incorporates the weighted sum of the estimated velocity and 2-D object flow $\boldsymbol{f}(j)$. The constant velocity assumption in [31] and [33] may introduce errors, especially during prolonged tracking failures or when the vehicle is in rapid motion conditions [36]. In LVI-SORT, we replace the constant velocity assumption with the 2-D object flow $\boldsymbol{f}(j)$, calculated using (60), and use it as the active control item for the Kalman filter [51]. This improvement offers two advantages: first, the 2-D object flow, computed with the 3-D object state, tends to be more accurate than the results obtained by assuming a constant velocity for the 2-D bounding box. Second, the computation of 2-D object flow compensates for vehicle motion, enhancing the accuracy of object prediction, especially in cases of rapid movement.

4) State Prediction: The state prediction is achieved with the following formula:

$$\hat{\boldsymbol{x}}_b(j) = \boldsymbol{F}(j)\check{\boldsymbol{x}}_b(i) + \boldsymbol{C}(j)\boldsymbol{f}(j) \tag{68}$$

$$\hat{\boldsymbol{P}}(j) = \boldsymbol{F}(j)\check{\boldsymbol{P}}(i)\boldsymbol{F}(j)^\top + \boldsymbol{Q}(j) \tag{69}$$

where $\hat{\boldsymbol{x}}_b(j)$ is the state prediction of bounding box in moment j, $\check{\boldsymbol{x}}_b(i)$ is the posteriori estimate in moment i, $\boldsymbol{F}(j)$ is the state transformation matrix which is obtained with (65), $\boldsymbol{C}(j)$ is the control matrix which is obtained with (66), $\boldsymbol{f}(j)$ is the 2-D object flow, $\hat{\boldsymbol{P}}(j)$ is the covariance of state prediction, and $\check{\boldsymbol{P}}(i)$ is the posteriori estimation of bounding box state in moment i.

B. Association

The Hungarian algorithm [52] is used to associate the predicted objects with the newly detected objects in LVI-SORT. The newly detected objects are obtained with YOLO-V5 [53]. And the association cost matrix is crucial for the Hungarian algorithm, serving as the basis for association. The intersection of union between the predicted bounding boxes and the newly detected bounding boxes is used in [31] and [33] to calculate the association cost matrix. However, the association cost matrix based solely on IoU will fail when the YOLO-V5 fails for a while. Therefore, a simple way is proposed by adding the term of object map points $\mathcal{P}^n(j)$ into the association cost matrix to solve this problem.

Based on N predicted objects and M detected objects, the association cost matrix C_{IoU} can be calculated by referring [31]. Furthermore, for all the object maps $\{\mathcal{P}^n(j), n = 1, \ldots, N\}$, the ratio of the points in the detected bounding boxes to all the tracked points is used to calculate the association cost matrix C_{obj}

$$C_{\text{obj}}^{nm} = \frac{\textbf{size}\big(\mathcal{P}^n(j) \in \det_m\big)}{\textbf{size}\big(\mathcal{P}^n(j)\big)} \tag{70}$$

where $n = 1, \ldots, N$ is the index of the object map, and $m = 1, \ldots, M$ is the index of the detected object. And $\det_m$ represents the mth bounding box detected by YOLO-V5. And **size**$(\cdot)$ is a function used to calculate the number of object map points.

Then, the fused association cost matrix can be written as follows:

$$C = \lambda C_{\text{IoU}} + (1 - \lambda) C_{\text{obj}} \tag{71}$$

where λ is a weight factor and set as 0.5.

C. Update

The observation model for the LVI-SORT is

$$\boldsymbol{z}(j) = \boldsymbol{H}(j)\boldsymbol{x}(j) + \boldsymbol{v}(j) \tag{72}$$

where $\boldsymbol{z}(j)$ is the measurements of the newly detected object bounding boxes, e.g., $\boldsymbol{z}(j) = [\tilde{x}_c, \tilde{y}_c, \tilde{a}, \tilde{h}]$. And $\boldsymbol{v}(j)$ is the measurements noise which is assumed to be independent and identically distributed with the normal distribution, e.g., $\boldsymbol{v}(j) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}(j))$. $\boldsymbol{H}(j)$ is the measurement matrix and written as follows:

$$\boldsymbol{H}(j) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{73}$$

Furthermore, the posteriori estimation of object state can be achieved as follows:

$$\check{\boldsymbol{x}}_b(j) = \hat{\boldsymbol{x}}_b(j) + \boldsymbol{K}(j)\big(\boldsymbol{z}(j) - \boldsymbol{H}(j)\hat{\boldsymbol{x}}_b(j)\big) \tag{74}$$

$$\check{\boldsymbol{P}}(j) = (\boldsymbol{I} - \boldsymbol{K}(j)\boldsymbol{H}(j))\hat{\boldsymbol{P}}(j) \tag{75}$$

where $\check{\boldsymbol{x}}_b(j)$ is the posteriori estimation of object bounding box state, and $\check{\boldsymbol{P}}(j)$ is the covariance of $\check{\boldsymbol{x}}_b(j)$. And $\boldsymbol{K}(j)$ is the Kalman gain, which can be calculated as follows:

$$\boldsymbol{K}(j) = \hat{\boldsymbol{P}}(j)\boldsymbol{H}(j)^\top \big(\boldsymbol{H}(j)\hat{\boldsymbol{P}}(j)\boldsymbol{H}(j)^\top + \boldsymbol{R}(j)\big)^{-1}. \tag{76}$$

The detail process of the LVI-SORT is displayed in Algorithm 2.

Algorithm 2 LVI-SORT

Input: object map $\{\mathcal{P}^n(i), n = 1, \ldots, N\}$, object state $\{\check{\boldsymbol{x}}^n(j), n = 1, \ldots, N\}$, 2D bounding box state $\{\check{\boldsymbol{x}}_b^n(i), n = 1, \ldots, N\}$, 2D bounding box observation from YOLO-V5 $\{det_m, 1, \ldots, M\}$.

Output: 2D bounding box state $\{\check{\boldsymbol{x}}_b^n(j), n = 1, \ldots, N\}$.

1: Calculate the 2D object flows $\{\boldsymbol{f}_s^n(j), s = 1, \ldots, m\}$ for each object map as (60).
2: Calculate the mean $\boldsymbol{f}^n(j)$ and covariance cov_{f^n} of 2D object flows $\{\boldsymbol{f}_s^n(j), s = 1, \ldots, m\}$ as (61) and (62).
3: Predict 2D bounding box state $\hat{\boldsymbol{x}}_b(j)$ and covariance $\hat{\boldsymbol{P}}(j)$ with $\boldsymbol{f}^n(j)$ as (68) and (69).
4: Calculate the hybrid cost matrix C as (71) and associate the predicted bounding box $\hat{\boldsymbol{x}}_b(j)$ with detected 2D bounding boxes $\{det_m, m = 1, \ldots, M\}$ from YOLO-V5 with Hungarian algorithm.
5: Update the 2D bounding box state $\check{\boldsymbol{x}}_b(j)$ and covariance $\check{\boldsymbol{P}}(j)$ with associated detection object as (74) and (75).

V. Experiments and Discussion

To assess the performance of the proposed Dynam-LVIO in dynamic environments, a series of experiments are conducted using the KITTI dataset [54] and data collected from our platform. We select a set of benchmark algorithms, namely, LiDAR odometry and mapping (LOAM) [55], fast LiDAR-inertial odometry (FAST-LIO) [9], LiDAR-inertial odometry-incremental smoothing and mapping (LIO-SAM) [56], LIO-SEGMOT [38], and robust, real-time, RGB-colored, LiDAR-inertial-visual tightly-coupled state estimation and mapping package (R3LIVE) [57], to demonstrate the superior accuracy of the proposed Dynam-LVIO in localization and mapping. Among these benchmark algorithms, LOAM is a LiDAR odometry that uses point-plane and point-line constraints to formulate an objective function for maximum a posteriori estimation. FAST-LIO is a LiDAR–inertial odometry system using ESIKF for state estimation. LIO-SAM is a LiDAR–inertial odometry using factor graphs for state estimation. LiDAR-inertial odometry via simultaneous ego-motion estimation and multiple object tracking (LIO-SEGMOT) is a LiDAR–inertial odometry with a dynamic object tracking algorithm. It uses self-ensembling single-stage object detector (SE-SSD) [58] for object detection and uses factor graph to estimate the object state. R3LIVE is an LVIO system that also uses ESIKF for state estimation. All the experiments are conducted on a desktop computer equipped with an Intel Core i7-9700 processor running at 3.0 GHz, RTX 3080Ti GPU, and 32 GB of RAM. All the proposed Dynam-LVIO and the benchmark algorithms operate within the robot operating system (ROS) [59], an open-source communication framework. The evaluation of localization accuracy is performed using the absolute trajectory error (ATE) metric, as outlined in [60]. The ATE metric is calculated using the following equation:

$$\text{ATE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{t}_{\text{est}}^i - \boldsymbol{t}_{\text{gt}}^i\right\|} \tag{77}$$

TABLE I
ATE OF THE TRAJECTORY OUTPUT FROM COMPARISON ALGORITHMS IN THE KITTI DATASET

ATE (m)	LOAM	FAST-LIO	LIO-SAM	LIO-SEGMOT	R3LIVE	Dynam-LVIO	Improvement
0926-0009	0.64	0.51	0.51	0.50	0.56	**0.48**	4.00%
0926-0013	0.28	0.24	0.26	0.25	0.24	**0.23**	4.17%
0926-0014	0.91	0.65	0.78	0.61	0.61	**0.52**	14.75%
0926-0051	0.45	0.41	0.39	0.38	0.36	**0.34**	5.56%
0926-0101	8.36	6.94	6.75	6.47	6.63	**6.02**	6.96%
0930-0016	0.41	0.37	0.39	0.32	0.31	**0.26**	16.13%
1003-0047	1.83	1.56	1.47	1.35	1.41	**1.31**	7.09%

where $\boldsymbol{t}^i_{\text{est}}$ is the position obtained with odometry, and $\boldsymbol{t}^i_{\text{gt}}$ is the ground-truth position.

A. KITTI Dataset

The KITTI dataset is widely used in SLAM research field, providing us with the opportunity to compare the proposed Dynam-LVIO with a significant body of previous algorithm. The sensor parameters in the KITTI dataset are detailed in [54]. To evaluate the effectiveness of the proposed Dynam-LVIO in dynamic environments, data segments 0926-0009, 0926-0013, 0926-0014, 0926-0051, 0926-0101, 0930-0016, and 1003-0047 are selected. Furthermore, the naming convention for the KITTI data segments mentioned in this article adheres to the naming convention of "month day-serial number," facilitating the identification of the corresponding data within the KITTI dataset.

The trajectories and corresponding errors obtained by various algorithms are illustrated in Figs. 3–6. In Figs. 3–6, the left side displays the trajectory plot, and the right side showcases the trajectory error. To prevent redundancy in experimental results, only partial trajectories and error plots are displayed. In Figs. 3–6, the trajectory of LOAM is depicted in blue, FAST-LIO in orange, LIO-SAM in green, LIO-SEGMOT in purple, R3LIVE in yellow, the ground-truth trajectory in black, and Dynam-LVIO in red. To enhance the intuitive comparison of trajectories obtained by various algorithms, specific details within the trajectories are magnified using red boxes. From the magnified area, it is evident that the trajectory of the proposed Dynam-LVIO is closest to the ground-truth trajectory, demonstrating that the proposed Dynam-LVIO algorithm can yield more accurate localization results than other algorithms. In addition, the trajectory error in Figs. 3–6 clearly indicates that the error obtained by the proposed Dynam-LVIO algorithm is closer to zero, emphasizing that the proposed algorithm can achieve more accurate localization results.

To quantitatively assess the effectiveness of the proposed Dynam-LVIO algorithm, we calculated the ATE of trajectories obtained by various algorithms, and the results are presented in Table I. Within the selected KITTI data segments, the proposed Dynam-LVIO demonstrates a notably lower ATE compared with other algorithms, as indicated in Table I. This quantitative evidence substantiates the claim that the proposed algorithm is more effective than the existing SLAM algorithms in dynamic environments. In addition, the enhancement of the proposed algorithm over the most accurate algorithm among those compared is calculated. The results indicate that the proposed algorithm can achieve an improvement of approximately 5%–15% in localization. Furthermore, it is noteworthy that in data segment 0926-0101, the ATE for all the algorithms is relatively high compared with other segments. This is primarily attributed to the data segment being collected on a highway, where the geometric texture surrounding the road is absent, and the scene exhibits high repetition, thereby diminishing LiDAR performance. Consequently, the ATE for this data segment is comparatively elevated.

The excellent localization accuracy achieved by the proposed Dynam-LVIO algorithm is primarily attributed to the following enhancements: first, the integration of the MOT algorithm and the standard LVIO algorithm effectively eliminates dynamic objects in the map constructed by the standard LVIO. This enhancement ensures that the environment meets the static environment assumption required by the SLAM system, thereby improving the localization accuracy of Dynam-LVIO. Second, the simultaneous utilization of reprojection error and ICP error as observations enables ESIKF to reliably estimate the 3-D object state. Subsequently, the 3-D object state and object map are used to derive the 2-D object flow, effectively enhancing MOT performance under high-speed movement conditions of vehicles or objects. Moreover, 2-D object flow is used to track object map points, which are then combined with IoU to ensure stable object association during MOT. This approach enhances MOT accuracy in the presence of object occlusion conditions and facilitates accurate tracking of dynamic objects. The utilization of these methods enhance the robustness of MOT and ensure stable tracking of dynamic objects, thereby contributing to the localization accuracy of Dynam-LVIO in dynamic environments.

To effectively verify the object tracking performance of the proposed Dynam-LVIO algorithm, fast-moving and occlusion scenes data are selected from the KITTI dataset for verification. The selected object tracking comparison algorithms are OC-SORT [40] and Bytetrack [33]. For fast-moving scenes, the object tracking results of different algorithms are depicted in Fig. 7. Each column in Fig. 7 represents the object tracking results of two adjacent frames, where the upper subplot illustrates the previous moment and the lower subplot illustrates the later moment. From Fig. 7, it is evident that only the proposed LVI-SORT algorithm can accurately track the vehicle object

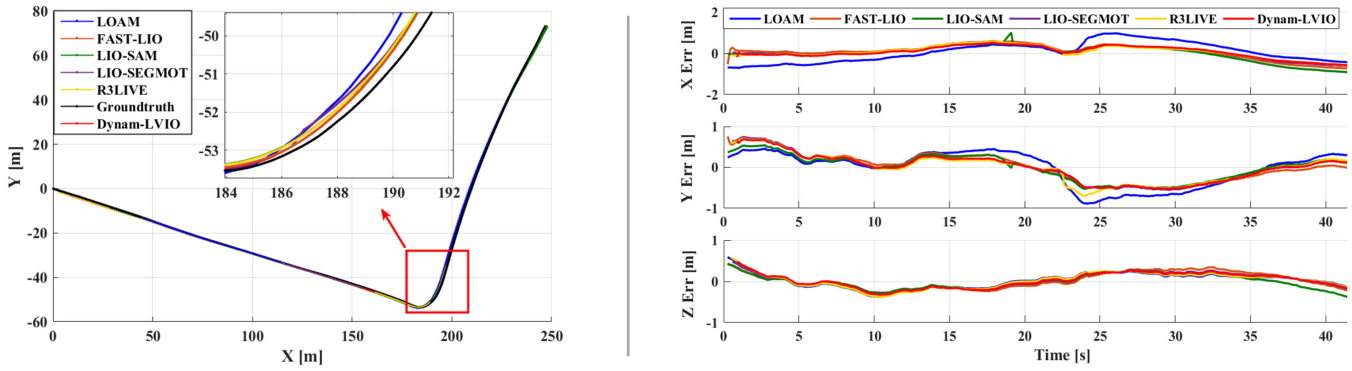


Fig. 3. Trajectory and corresponding error output from comparison algorithms and Dynam-LVIO in segment **0926-0009** of the KITTI dataset.

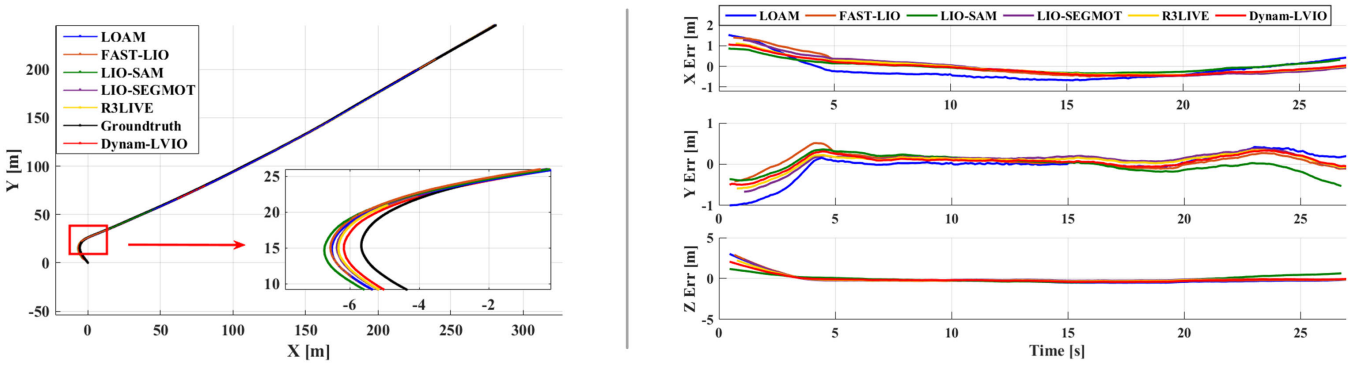


Fig. 4. Trajectory and corresponding error output from comparison algorithms and Dynam-LVIO in segment **0926-0014** of the KITTI dataset.

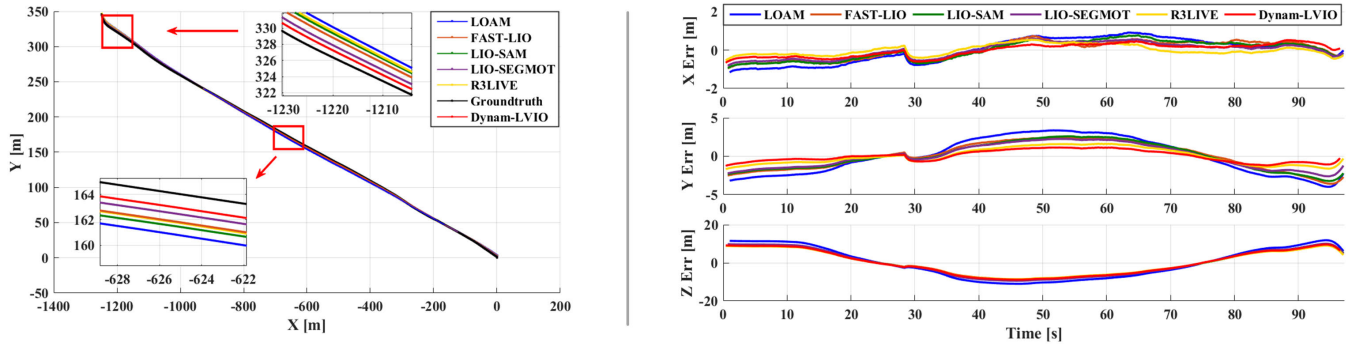


Fig. 5. Trajectory and corresponding error output from comparison algorithms and Dynam-LVIO in segment **0926-0101** of the KITTI dataset.

with ID 2, maintaining ID consistency despite the continuous fast movement of the platform. The main reason for this is that the proposed algorithm accurately estimates the object state and carrier state, effectively addressing the 2-D object flow to enhance object tracking accuracy in fast-moving scenes. In addition, when the object undergoes occlusion, the object tracking outcomes from various algorithms are depicted in Figs. 8 and 9. Within each column of Figs. 8 and 9, the three images, respectively, illustrate the object tracking results before occlusion, during occlusion, and after occlusion. The occluded object IDs in the two selected object occlusion scenes are 139 and 119. The results demonstrate that the proposed method consistently preserves the object ID before and after occlusion, ensuring accurate object tracking. The improvement is primarily attributed to the utilization of 2-D object flow in the proposed algorithm for tracking object mapping points and achieving stable object association in conjunction with object IoU. This ensures accurate tracking, especially when objects are occluded.

To quantitatively verify the effectiveness of the proposed algorithm in eliminating dynamic objects, we use the accurate object labels in the KITTI dataset to calculate the accuracy and recall rates of dynamic object tracking results. The higher order tracking accuracy (HOTA) metric [61], detection recall (DetRe), detection precision (DetPr), association recall (AssRe), and association precision (AssPr) are used to evaluate the object tracking results across various algorithms. The evaluation results with these metrics are outlined in Table II. And the results demonstrate that the proposed LVI-SORT algorithm performs well across multiple metrics (HOTA, DetRe, DetPr,

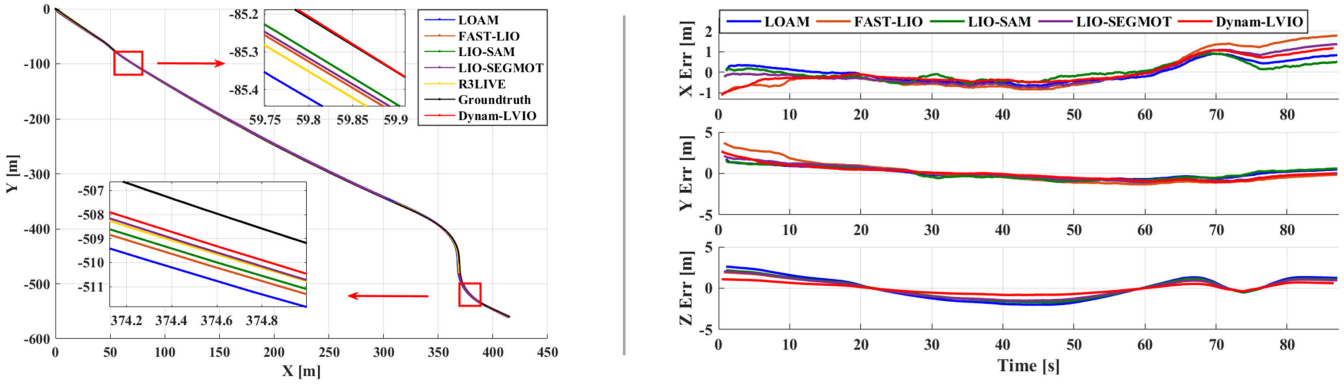


Fig. 6. Trajectory and corresponding error output from comparison algorithms and Dynam-LVIO in segment **1003-0047** of the KITTI dataset.

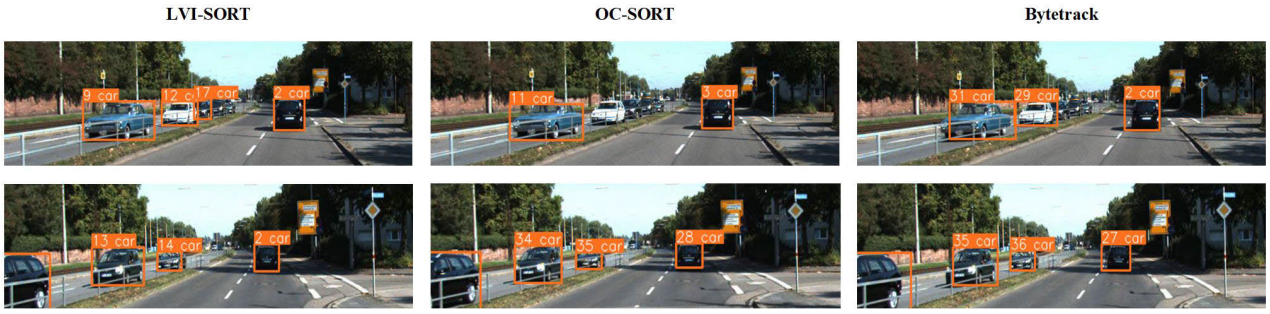


Fig. 7. MOT results in **fast-moving scenes** across various algorithms.

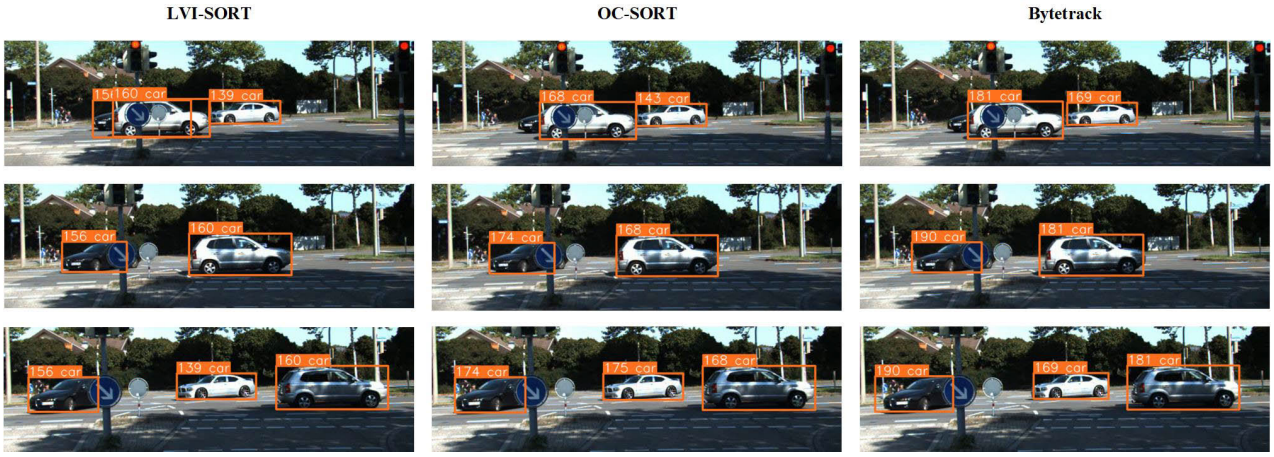


Fig. 8. MOT results in **the first object occlusion scenes** across various algorithms.

AssRe, AssPr). This indicates that the proposed algorithm effectively enhances the accuracy of dynamic object tracking. Specifically, the HOTA metric of the proposed method is 79.92%, which is approximately 3% higher than other algorithms, indicating that the overall object tracking ability of the proposed algorithm is better than other algorithms. Regarding the DetRe metric, the proposed algorithm achieves a result of 81.56%, suggesting effective tracking of all the correct objects compared with other algorithms. Furthermore, for DetPr, the proposed algorithm yields a result of 88.24%, indicating its efficacy in avoiding tracking incorrect objects compared with other algorithms. In terms of AssRe, the proposed algorithm achieves a result of 87.05%, demonstrating its capability to effectively prevent multiple trackers from tracking the same object in comparison to other algorithms. Finally, for AssPr, the proposed algorithm attains a result of 91.52%, signifying its effectiveness in preventing the same tracker from simultaneously tracking multiple objects compared with other algorithms.

To visually illustrate the mapping accuracy of the proposed Dynam-LVIO algorithm in dynamic environments, we conduct a comparison between the map constructed by the proposed Dynam-LVIO and FAST-LIO. As FAST-LIO does not eliminate dynamic objects, the comparison between the map generated by FAST-LIO and the proposed Dynam-LVIO serves as an effective means to validate the mapping accuracy

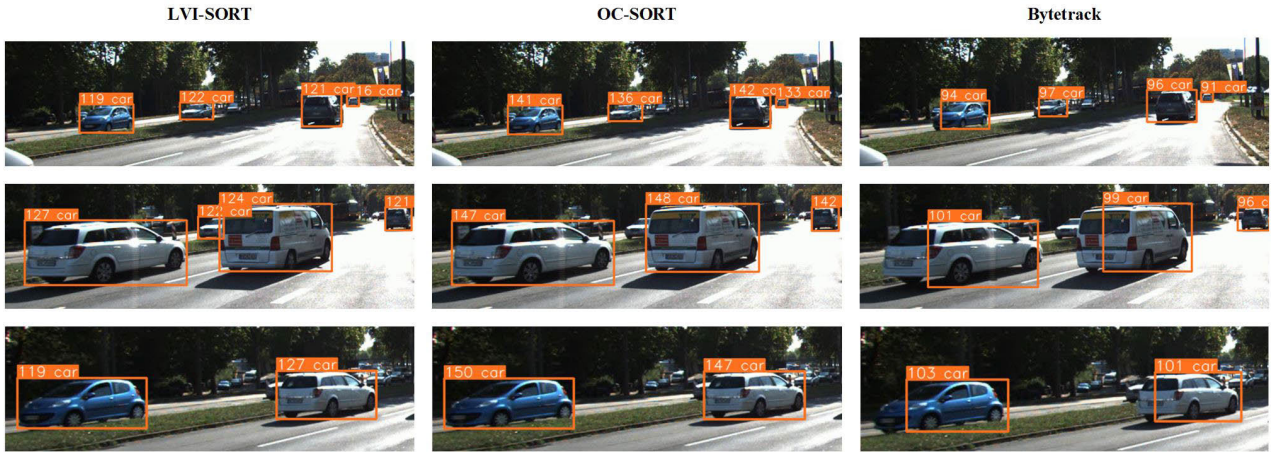


Fig. 9. MOT results in **the second object occlusion scenes** across various algorithms.

TABLE II
COMPARISON OF OBJECT TRACKING RESULTS ACHIEVED BY VARIOUS ALGORITHMS

Metrics	HOTA	DetRe	DetPr	AssRe	AssPr
ByteTrack	77.83%	79.21%	83.31%	84.73%	88.53%
OC-SORT	76.54%	80.64%	86.36%	80.33%	87.17%
LVI-SORT	**79.92%**	**81.56%**	**88.24%**	**87.05%**	**91.52%**

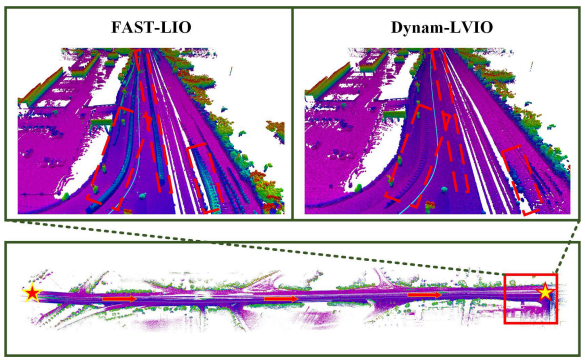


Fig. 10. Maps constructed using FAST-LIO and the proposed Dynam-LVIO in the segment **0926-0101** of the KITTI dataset.

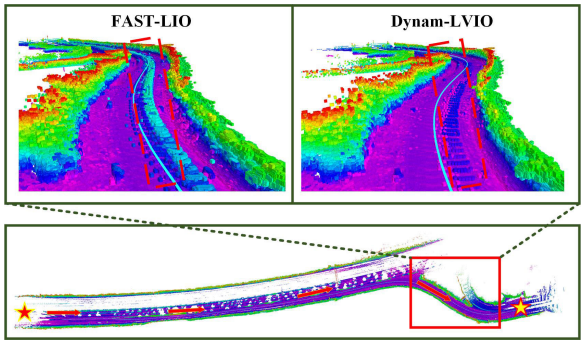


Fig. 11. Maps constructed using FAST-LIO and the proposed Dynam-LVIO in the segment **1003-0047** of the KITTI dataset.

in dynamic environments. Moreover, we select the maps constructed from the 0926-0101 and 1003-0047 data segments in the KITTI dataset for presentation. The constructed maps are depicted in Figs. 10 and 11, where the starting point of the trajectory is marked with a red five-pointed star, the termination point with a yellow five-pointed star, and the running direction of the trajectory is indicated by a series of red arrows. The height value is conveyed through the color of the map. To enhance the intuitive presentation of the dynamic object removal effect, specific areas are selectively zoomed in, demarcated by a red solid line box. Within the magnified map, the dynamic object point cloud is identified by a red dotted box. It can be seen from the results that the proposed Dynam-LVIO can effectively remove dynamic objects in the map and reduce the impact of dynamic objects on SLAM mapping.

To further validate the capability of the proposed Dynam-LVIO algorithm in constructing environmental and dynamic object maps, we present the static environmental maps and dynamic object maps constructed by Dynam-LVIO in Figs. 12 and 13. The data are selected from the KITTI data segment of 0926-0101 and 1003-0047, respectively. As the localization of dynamic object maps is constantly changing relative to static environmental maps, we choose to display these dynamic object maps every 2 s to clearly illustrate the evolution of dynamic object maps. The color of dynamic object maps represents the LiDAR reflectivity. To enhance the clarity of dynamic object maps, we use black dashed lines to outline them and magnify them for display. The movement direction of dynamic objects is indicated by yellow arrows. From Figs. 11 and 13, it can be observed that the geometric textures of dynamic objects and the environment map are clear, accurately describing the geometric shapes of dynamic objects and the environment. This indicates that the proposed Dynam-LVIO algorithm can accurately construct objects and the environment map in dynamic scenes.

To substantiate the efficacy of the proposed algorithm in dynamic scenarios, a comparative analysis is conducted on the accuracy of dynamic object state estimation between

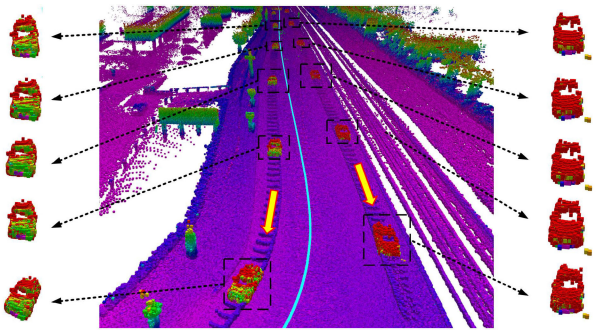


Fig. 12. Environment and object map obtained by Dynam-LVIO in the segment **0926-0101** of the KITTI dataset.

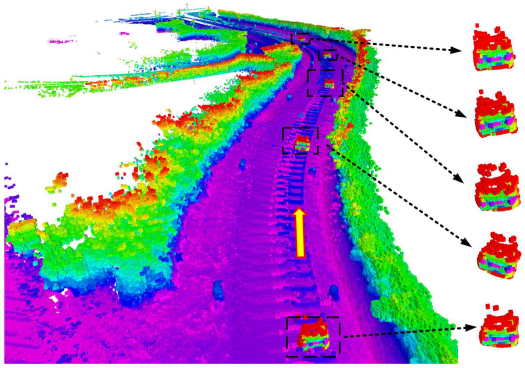


Fig. 13. Environment and object map obtained by Dynam-LVIO in the segment **1003-0047** of the KITTI dataset.

TABLE III
ATE OF OBJECTS TRAJECTORY FROM THE COMPARISON ALGORITHMS IN THE KITTI DATASET

ATE (m)	LIO-SEGMOT	Dynam-LVIO	Improvement
0926-0009 (4)	0.67	**0.58**	13.43%
0926-0013 (2)	0.32	**0.27**	15.63%
0926-0014 (16)	0.86	**0.63**	26.74%

the proposed Dynam-LVIO and LIO-SEGMOT. Objects with ground-truth trajectories are selected for the comparative analysis. The estimation accuracy of dynamic object state is quantified by calculating the ATE of the object trajectory, which can be calculated as (77). The results are delineated in Table III. It can be seen that the proposed Dynam-LVIO can obtain more accurate object position. This improvement can be attributed to the incorporation of both reprojection error measurements and ICP error measurements as observation values within the ESIKF used by the proposed Dynam-LVIO algorithm. The accurate estimation of object states is achieved by maximizing the utilization of the 3-D geometry captured by LiDAR and the environmental gray scale captured by the camera.

Moreover, to validate the feasibility of the proposed algorithm in practical engineering applications, a comparison experiment is conducted to assess the runtime between the proposed Dynam-LVIO and other algorithms. Given that

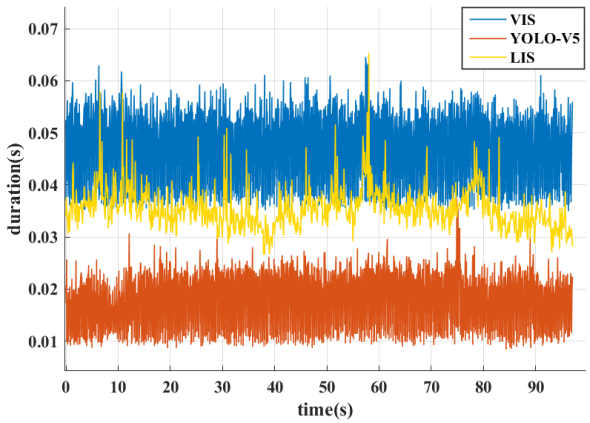


Fig. 14. Runtime of each subsystem within the proposed Dynam-LVIO in the segment **0926-0101** of the KITTI dataset.

Fig. 15. Sensor suite used in our platform.

the proposed Dynam-LVIO algorithm comprises two parallel subsystems, e.g., visual–inertial subsystem (VIS) and LiDAR–inertial subsystem (LIS), a separate comparison of the runtime for each subsystem is conducted. The results are depicted in Table IV. These results reveal that the runtime of the LIS and VIS in the proposed Dynam-LVIO is 34.79 and 46.87 ms, respectively. And the object detection module (e.g., YOLO-V5) in Dynam-LVIO clocking at 16.52 ms. To provide a more intuitive representation of each module's runtime in the proposed algorithm, we choose to record the runtime of each frame in the 0926-0101 data segment. The results are shown in Fig. 14. The blue curve in Fig. 14 represents the runtime of the VIS, the yellow curve represents the runtime of the LIS, and the brown curve represents the runtime of YOLO-V5. As observed from the results in Fig. 14, the average processing time of VIS is approximately 50 ms, the average processing time of LIS is approximately 40 ms, and the average processing time of YOLO-V5 is approximately 15 ms. Using multithreaded programming technology guarantees real-time operation of the proposed Dynam-LVIO on the tested multicore CPU [62].

B. Our Platform Data

To further substantiate the enhancements achieved by the proposed Dynam-LVIO system in the realms of localization and mapping, we have established an experimental platform, illustrated in Fig. 15. The sensor suite used aligns with the specifications detailed in our prior work [63]. We collected

TABLE IV
RUNTIME OF THE COMPARISON ALGORITHMS IN THE KITTI DATASET

Runtime (ms)	LOAM	FAST-LIO	LIO-SAM	LIO-SEGMOT	R3LIVE	Dynam-LVIO
LIS	43.25	38.24	53.87	55.68	28.37	34.79
VIS	–	–	–	–	23.54	46.87
Object Detection	–	–	–	40.53	–	16.52

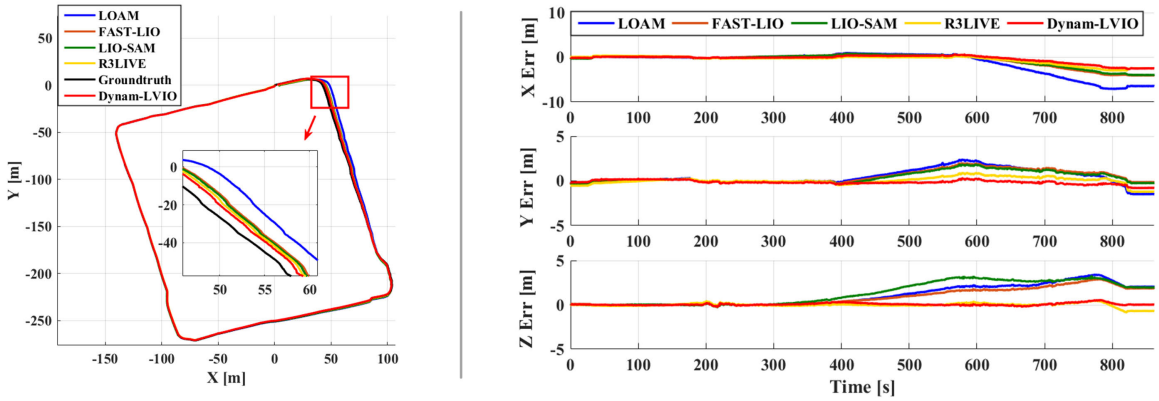


Fig. 16. Trajectory and corresponding error output from comparison algorithms and Dynam-LVIO in **outdoor** environments with our platform.

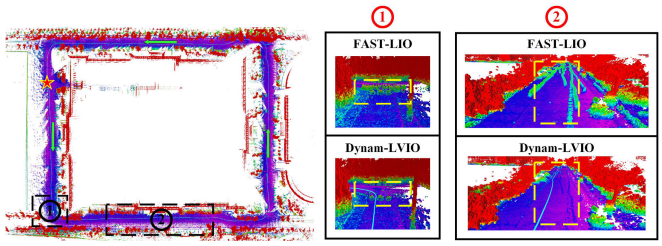


Fig. 17. Maps constructed using FAST-LIO and the proposed Dynam-LVIO in **outdoor** environments with our platform.

TABLE V
ERROR OF THE TRAJECTORY OUTPUT FROM COMPARISON ALGORITHMS WITH DATA FROM OUR PLATFORM

Error (m)	LOAM	FAST-LIO	LIO-SAM	R3LIVE	Dynam-LVIO
outdoor	4.61	2.19	3.26	2.63	**1.62**
indoor	31.08	12.63	14.69	14.77	**5.40**

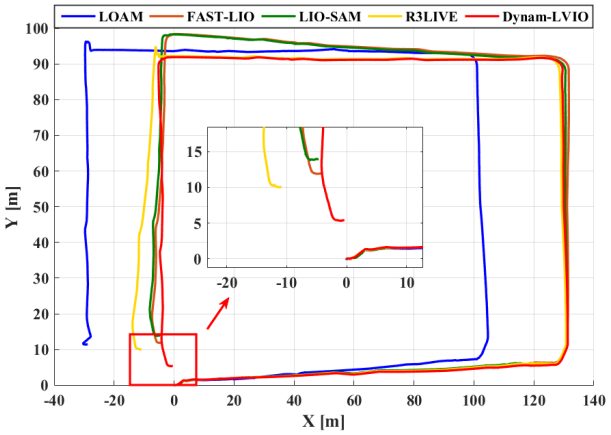


Fig. 18. Trajectory output from comparison algorithms and Dynam-LVIO in **indoor** environments with our platform.

sensor data both in indoor and outdoor environment. Real-time kinematic positioning (RTK) serves as the ground-truth trajectory for outdoor environment, whereas indoor localization accuracy is determined by assessing the positional deviation between the starting and ending points. The selected comparison algorithm aligns fundamentally with that outlined in Section V-A. However, the utilization of the solid-state LiDAR Livox Avia in our platform renders the object detection framework used in LIO-SEGMOT inapplicable. Consequently, LIO-SEGMOT is discontinued as a comparison algorithm. The trajectory colors of the remaining comparison algorithms remain consistent with those presented in Section V-A.

1) Outdoor Environment: In outdoor environment, our platform traverses around the teaching building and returns to a location near the starting point. The trajectories of different algorithms are recorded in Fig. 16. We magnify specific region in Fig. 16 to visually emphasize the trajectories of various algorithms, and the enlarged area is demarcated by red box. From the magnified area, it is evident that the proposed algorithm closely aligns with the ground-truth trajectory output by RTK. Therefore, the proposed method demonstrates a higher level of accuracy in localization compared with other algorithms. In addition, we conduct a comparison between the maps constructed by FAST-LIO and the proposed Dynam-LVIO, and the results are presented in Fig. 17. The red five-pointed star denotes the starting point of the trajectory, while the green arrow indicates the direction of

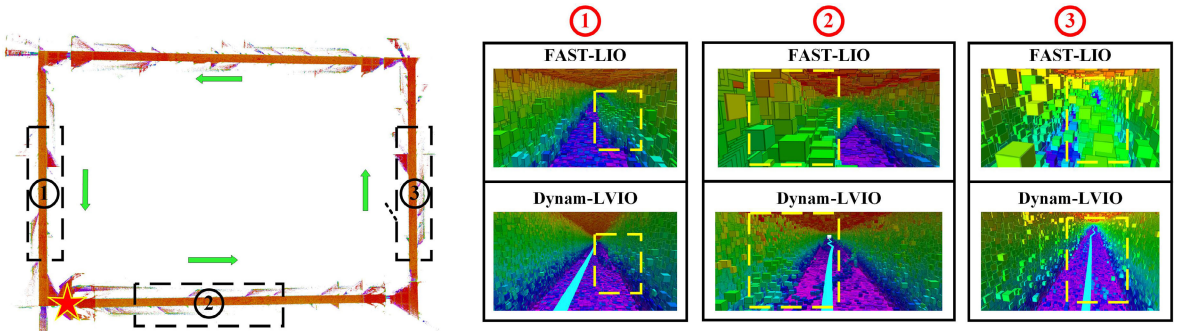


Fig. 19. Maps constructed using FAST-LIO and the proposed Dynam-LVIO in **indoor** environments with our platform.

the trajectory. Similarly, we magnify specific areas on the map to prominently highlight the accuracy of the proposed Dynam-LVIO algorithm in constructing maps in dynamic environments. Observing the region outlined by the yellow dotted line in the magnified image, it is evident that the proposed Dynam-LVIO successfully eliminates dynamic objects from the map. To qualitatively analyze the effectiveness of the proposed algorithm, we calculate the ATE for various algorithms and represent the results in Table V. The results in Table V reveal that the ATE obtained by the proposed algorithm is markedly lower than that of other algorithms. This observation underscores the superiority of the proposed algorithm in comparison to others.

2) Indoor Environment: In indoor environment, our platform traverses the internal corridor of the teaching building and relies on ground markers to facilitate the platform in returning to the starting point of the trajectory, ensuring that the distance between the starting point and the endpoint in the trajectory is less than 1 cm. The trajectories of different algorithms are recorded in Fig. 18. Similarly, we magnify the area near the starting point of the trajectory. It is evident from the zoomed-in image that the localization result obtained by the proposed Dynam-LVIO algorithm is closer to the starting point, providing a qualitative characterization of the localization superiority of the proposed Dynam-LVIO algorithm. We also conducted a comparison of the maps constructed by different algorithms as shown in Fig. 19, and it is evident that the proposed Dynam-LVIO algorithm can effectively eliminate dynamic objects in the indoor environments. We also calculate the difference between the start and end points for various trajectories to quantitatively verify the effectiveness of the proposed algorithm, as shown in Table V. It is evident that the Dynam-LVIO algorithm maintains higher localization accuracy compared with other algorithms.

In comparison to other experiments, errors in indoor environments are relatively significant, as shown in Table V. These discrepancies primarily arise from two factors: first, in Section V-B1, ATE is used to quantitatively assess the localization accuracy of different algorithms. As shown in (77), ATE represents the average errors across all the instances in the trajectory, whereas the calculated error in this section pertains solely to the deviation between the start and end points. In addition, given that the indoor scene comprises a corridor with limited geometric texture and photometric gradient, the performance of both LiDAR and camera deteriorates. As a result, the localization accuracy of both the proposed algorithm and the benchmark algorithm diminishes in this scene. Consequently, the error calculated in this section tends to be larger compared with Section V-B1.

VI. Conclusion

In this work, an LVIO, Dynam-LVIO, is proposed to achieve dynamic-object-aware SLAM in dynamic environments. The proposed Dynam-LVIO algorithm can achieve accurate mapping and localization in dynamic environments by concurrently constructing object and environment maps with the MOT algorithm. And the object map is used to establish reprojection error and ICP error measurements as observation for the ESIKF, thereby facilitating accurate 3-D object state estimation. Subsequently, an improved MOT method, LVI-SORT, is proposed to track 2-D bounding boxes. Based on the 3-D object state and object map points, the 2-D object flow is then calculated and integrated in the prediction process of LVI-SORT. This improvement addresses the challenges of predicting object states in scenarios involving high-speed vehicle or object movement. Moreover, the 2-D object flow is used to track the object map points and combined with the IoU of the object bounding boxes to ensure a stable association between the predicted object and the newly detected object output by YOLO-V5. The aforementioned contributions effectively enhance SLAM accuracy in dynamic environments. To validate the efficacy of the proposed Dynam-LVIO algorithm, extensive experiments are conducted. And the experimental results demonstrate that the proposed Dynam-LVIO algorithm can adeptly enhance localization accuracy by 5%–10% compared with the benchmark algorithms in complex dynamic environments. Concurrently, the proposed LVI-SORT algorithm can enhance the accuracy of object tracking by nearly 3%.

Furthermore, the following challenges are identified during practical experiments. First, although YOLO-V5 can accurately detect object presented in the training datasets, it struggles to detect unknown objects in the environment, thereby impacting the performance of the proposed algorithm in scenes containing these unknown objects. To address this issue, we will use geometric method, such as RANSAC, to detect unknown objects and enhance the accuracy of object detection. Furthermore, despite achieving real-time operation,

overall computational workload of the proposed algorithm is substantial, posing challenges for deployment on lightweight platforms. To address this issue, we will explore a lightweight LVIO method to calculate the vehicle states. In addition, we will propose an algorithm to adaptively adjust the number of ICP error and reprojection error measurements based on the environmental characteristics, effectively reducing computational complexity. Subsequent research efforts will focus on addressing these issues, aiming to enhance the practical engineering significance of the proposed Dynam-LVIO algorithm.

APPENDIX

A. Computation of $\boldsymbol{H}_s^{rep}$ and $\boldsymbol{F}_{\boldsymbol{\mathcal{P}}_s}$

With (33), $\boldsymbol{r}_s^{\text{rep}}(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}_s^n(j), \tilde{\boldsymbol{\mathcal{P}}}_s^n(i))$ in (38) can be written as follows:

$$\begin{aligned}\boldsymbol{r}_s^{\text{rep}}&\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}_s^n(j), \tilde{\boldsymbol{\mathcal{P}}}_s^n(i)\right)\\ &= \check{\boldsymbol{p}}_s^n(j) - \boldsymbol{\pi}\Big(\boldsymbol{R}_w^c(j)\Big(\check{\boldsymbol{R}}(j)\text{Exp}\big(\delta\check{\boldsymbol{\phi}}(j)\big)\tilde{\boldsymbol{\mathcal{P}}}_s^n(i) + \check{\boldsymbol{t}}(j)\\ &\qquad + \delta\check{\boldsymbol{t}}(j)\Big) + \boldsymbol{t}_w^c(j)\Big). \end{aligned} \tag{78}$$

For simplicity, the following definition is given:

$$\begin{aligned}\boldsymbol{p}_s^c(j) = \boldsymbol{R}_w^c(j)&\Big(\check{\boldsymbol{R}}(j)\text{Exp}\big(\delta\check{\boldsymbol{\phi}}(j)\big)\tilde{\boldsymbol{\mathcal{P}}}_s^n(i) + \check{\boldsymbol{t}}(j)\\ &+ \delta\check{\boldsymbol{t}}(j)\Big) + \boldsymbol{t}_w^c(j). \end{aligned} \tag{79}$$

And (78) will be rewritten as

$$\boldsymbol{r}_s^{\text{rep}}\left(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{p}}_s^n(j), \tilde{\boldsymbol{\mathcal{P}}}_s^n(i)\right) = \check{\boldsymbol{p}}_s^n(j) - \boldsymbol{\pi}\left(\boldsymbol{p}_s^c(j)\right). \tag{80}$$

With (10), (79) can be approximated as

$$\begin{aligned}&\boldsymbol{p}_s^c(j)\\ &\approx \boldsymbol{R}_w^c(j)\Big(\check{\boldsymbol{R}}(j)\Big(\boldsymbol{I} + \big(\delta\check{\boldsymbol{\phi}}(j)\big)^\wedge\Big)\tilde{\boldsymbol{\mathcal{P}}}_s^n(i) + \check{\boldsymbol{t}}(j) + \delta\check{\boldsymbol{t}}(j)\Big)\\ &\quad + \boldsymbol{t}_w^c(j)\\ &= \boldsymbol{R}_w^c(j)\big(\check{\boldsymbol{R}}(j)\tilde{\boldsymbol{\mathcal{P}}}_s^n(i) + \check{\boldsymbol{t}}(j) + \boldsymbol{t}_w^c(j)\big)\\ &\quad - \boldsymbol{R}_w^c(j)\check{\boldsymbol{R}}(j)\big(\tilde{\boldsymbol{\mathcal{P}}}_s^n(i)\big)^\wedge\delta\check{\boldsymbol{\phi}}(j) + \boldsymbol{R}_w^c(j)\delta\check{\boldsymbol{t}}(j). \end{aligned} \tag{81}$$

And $\boldsymbol{H}_s^{\text{rep}}$ can be calculated as follows:

$$\boldsymbol{H}_s^{\text{rep}} = -\frac{\partial\boldsymbol{\pi}\left(\boldsymbol{p}_s^c(j)\right)}{\partial\boldsymbol{p}_s^c(j)} \cdot \frac{\partial\boldsymbol{p}_s^c(j)}{\partial\delta\check{\boldsymbol{x}}(j)}\bigg|_{\delta\check{\boldsymbol{x}}(j)=\boldsymbol{0}} \tag{82}$$

where $(\partial\boldsymbol{\pi}(\boldsymbol{p}_s^c(j))/\partial\boldsymbol{p}_s^c(j))$ can be obtained with (2)

$$\frac{\partial\boldsymbol{\pi}\left(\boldsymbol{p}_s^c(j)\right)}{\partial\boldsymbol{p}_s^c(j)} = \frac{1}{\boldsymbol{p}_{z_s}(j)}\begin{bmatrix} f_x & 0 & -f_x\dfrac{p_{x_s}(j)}{p_{z_s}(j)} \\ 0 & f_y & -f_y\dfrac{p_{y_s}(j)}{p_{z_s}(j)} \end{bmatrix} \tag{83}$$

where $\boldsymbol{p}_s^c(j) = [p_{x_s}, p_{y_s}, p_{z_s}]$.

And according to (81), $(\partial\boldsymbol{p}_s^c(j))/(\partial\delta\check{\boldsymbol{x}}(j))$ can be derived as

$$\frac{\partial\boldsymbol{p}_s^c(j)}{\partial\delta\check{\boldsymbol{x}}(j)} = \big[-\boldsymbol{R}_w^c(j)\check{\boldsymbol{R}}(j)\big(\tilde{\boldsymbol{\mathcal{P}}}_s^n(i)\big)^\wedge \quad \boldsymbol{R}_w^c(j) \quad \boldsymbol{0} \quad \boldsymbol{0}\big]. \tag{84}$$

Then, $\boldsymbol{F}_{\boldsymbol{\mathcal{P}}_s}$ can be calculated as follows:

$$\boldsymbol{F}_{\boldsymbol{\mathcal{P}}_s} = -\frac{\partial\boldsymbol{\pi}\left(\boldsymbol{p}_s^c(j)\right)}{\partial\boldsymbol{p}_s^c(j)} \cdot \frac{\partial\boldsymbol{p}_s^c(j)}{\partial\tilde{\boldsymbol{\mathcal{P}}}_s^n}\bigg|_{\delta\check{\boldsymbol{x}}(j)=\boldsymbol{0}} \tag{85}$$

where

$$\frac{\partial\boldsymbol{p}_s^c(j)}{\partial\tilde{\boldsymbol{\mathcal{P}}}_s^n}\bigg|_{\delta\check{\boldsymbol{x}}(j)=\boldsymbol{0}} = \boldsymbol{R}_w^c(j)\check{\boldsymbol{R}}(j) \tag{86}$$

and $(\partial\boldsymbol{\pi}(\boldsymbol{p}_s^c(j))/\partial\boldsymbol{p}_s^c(j))$ can be obtained with (83).

B. Computation of $\boldsymbol{H}_q^{icp}$ and $\boldsymbol{F}_{\boldsymbol{\mathcal{P}}_q^n}$

With (41), $\boldsymbol{r}_q^{\text{icp}}(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_q^n(i), \tilde{\boldsymbol{\mathcal{P}}}_q^w(j))$ in (46) can be obtained as follows:

$$\begin{aligned}&\boldsymbol{r}_q^{\text{icp}}\big(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_q^n(i), \tilde{\boldsymbol{\mathcal{P}}}_q^w(j)\big)\\ &\quad = \tilde{\boldsymbol{\mathcal{P}}}_q^w(j) - \Big(\check{\boldsymbol{R}}(j)\text{Exp}\big(\delta\check{\boldsymbol{\phi}}(j)\big)\tilde{\boldsymbol{\mathcal{P}}}_q^n(i) + \check{\boldsymbol{t}}(j) + \delta\check{\boldsymbol{t}}(j)\Big). \end{aligned} \tag{87}$$

With (10), (87) can be approximated as

$$\begin{aligned}&\boldsymbol{r}_q^{\text{icp}}\big(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_q^n(i), \tilde{\boldsymbol{\mathcal{P}}}_q^w(j)\big)\\ &\approx \tilde{\boldsymbol{\mathcal{P}}}_q^w(j) - \Big(\check{\boldsymbol{R}}(j)\Big(\boldsymbol{I} + \big(\delta\check{\boldsymbol{\phi}}(j)\big)^\wedge\Big)\tilde{\boldsymbol{\mathcal{P}}}_q^n(i) + \check{\boldsymbol{t}}(j) + \delta\check{\boldsymbol{t}}(j)\Big)\\ &= \tilde{\boldsymbol{\mathcal{P}}}_q^w(j) - \big(\check{\boldsymbol{R}}(j)\tilde{\boldsymbol{\mathcal{P}}}_q^n(i) + \check{\boldsymbol{t}}(j)\big)\\ &\quad + \check{\boldsymbol{R}}(j)\big(\tilde{\boldsymbol{\mathcal{P}}}_s^w(i)\big)^\wedge\delta\check{\boldsymbol{\phi}}(j) - \delta\check{\boldsymbol{t}}(j). \end{aligned} \tag{88}$$

And $\boldsymbol{H}_q^{\text{icp}}$ can be calculated as follows:

$$\begin{aligned}\boldsymbol{H}_q^{\text{icp}} &= \frac{\partial\boldsymbol{r}_q^{\text{icp}}\big(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_q^n(i), \tilde{\boldsymbol{\mathcal{P}}}_q^w(j)\big)}{\partial\delta\check{\boldsymbol{x}}(j)}\bigg|_{\delta\check{\boldsymbol{x}}(j)=\boldsymbol{0}}\\ &= \Big[\check{\boldsymbol{R}}(j)\big(\tilde{\boldsymbol{\mathcal{P}}}_q^n(i)\big)^\wedge \quad -\boldsymbol{I} \quad \boldsymbol{0} \quad \boldsymbol{0}\Big]. \end{aligned} \tag{89}$$

Then, $\boldsymbol{F}_{\boldsymbol{\mathcal{P}}_q^n}$ can be calculated as

$$\begin{aligned}\boldsymbol{F}_{\boldsymbol{\mathcal{P}}_q^n} &= \frac{\partial\boldsymbol{r}_q^{\text{icp}}\big(\check{\boldsymbol{x}}(j) \boxplus \delta\check{\boldsymbol{x}}(j), \tilde{\boldsymbol{\mathcal{P}}}_q^n(i), \tilde{\boldsymbol{\mathcal{P}}}_q^w(j)\big)}{\partial\tilde{\boldsymbol{\mathcal{P}}}_q^n(i)}\bigg|_{\delta\check{\boldsymbol{x}}(j)=\boldsymbol{0}}\\ &= -\check{\boldsymbol{R}}(j). \end{aligned} \tag{90}$$

REFERENCES

[1] Y. Yang, Y. Mao, and B. Sun, "Basic performance and future developments of BeiDou global navigation satellite system," *Satell. Navigat.*, vol. 1, no. 1, pp. 1–8, Jan. 2020.

[2] T. Seel, M. Kok, and R. S. McGinnis, "Inertial sensors—Applications and challenges in a nutshell," *Sensors*, vol. 20, no. 21, p. 6221, Oct. 2020.

[3] Y. Li et al., "Inertial sensing meets machine learning: Opportunity or challenge?" *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 9995–10011, Aug. 2022.

[4] N. El-Sheimy and Y. Li, "Indoor navigation: State of the art and future trends," *Satell. Navigat.*, vol. 2, no. 1, pp. 1–23, Dec. 2021.

[5] K. Yuan and Z. J. Wang, "A simple self-supervised IMU denoising method for inertial aided navigation," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 944–950, Feb. 2023.

[6] J. Wang, D. Weng, X. Qu, W. Ding, and W. Chen, "A novel deep odometry network for vehicle positioning based on smartphone," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–12, 2023.

[7] H. Liu, X. Wei, M. Perusquía-Hernández, N. Isoyama, H. Uchiyama, and K. Kiyokawa, "DUET: Improving inertial-based odometry via deep IMU online calibration," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[8] C. Cadena et al., "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.

[9] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

[10] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[11] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[12] T. Sun, Y. Liu, Y. Wang, and Z. Xiao, "An improved monocular visual-inertial navigation system," *IEEE Sensors J.*, vol. 21, no. 10, pp. 11728–11739, May 15, 2021.

[13] M. Zhong, Y. You, S. Zhou, and X. Xu, "A robust visual-inertial SLAM in complex indoor environments," *IEEE Sensors J.*, vol. 23, no. 17, pp. 19986–19994, Sep. 2023.

[14] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 486–492.

[15] M. S. Bahraini, M. Bozorg, and A. B. Rad, "SLAM in dynamic environments via ML-RANSAC," *Mechatronics*, vol. 49, pp. 105–118, Feb. 2018.

[16] D. Barath and J. Matas, "Graph-cut RANSAC," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6733–6741.

[17] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "RGB-D SLAM in dynamic environments using point correlations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 373–389, Jan. 2020.

[18] J. Cheng, C. Wang, and M. Q.-H. Meng, "Robust visual localization in dynamic environments based on sparse motion removal," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 2, pp. 658–669, Apr. 2020.

[19] H. Yin, S. Li, Y. Tao, J. Guo, and B. Huang, "Dynam-SLAM: An accurate, robust stereo visual-inertial SLAM method in dynamic environments," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 289–308, Feb. 2023.

[20] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, 2018.

[21] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018.

[22] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "MID-fusion: Octree-based object-level multi-instance dynamic SLAM," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5231–5237.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[25] W. Wu, L. Guo, H. Gao, Z. You, Y. Liu, and Z. Chen, "YOLO-SLAM: A semantic SLAM system towards dynamic environment with geometric constraint," *Neural Comput. Appl.*, vol. 34, pp. 6011–6026, Jan. 2022.

[26] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[27] P. Li and T. Qin, "Stereo vision-based semantic 3D object and ego-motion tracking for autonomous driving," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 646–661.

[28] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023.

[29] F. Zhong, S. Wang, Z. Zhang, C. Chen, and Y. Wang, "Detect-SLAM: Making object detection and SLAM mutually beneficial," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1001–1010.

[30] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *Computer Vision—ECCV 2020* (Lecture Notes in Computer Science), vol. 12356, A. Vedaldi, H. Bischof, T. Brox, and J. M. Frahm, Eds. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-58621-8_7.

[31] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3464–3468.

[32] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3645–3649.

[33] Y. Zhang et al., "ByteTrack: Multi-object tracking by associating every detection box," in *Computer Vision—ECCV 2022* (Lecture Notes in Computer Science), vol. 13682, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham, Switzerland: Springer, 2022, doi: 10.1007/978-3-031-20047-2_1.

[34] 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.

[35] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans," 2020, *arXiv:2002.06289*.

[36] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "BoT-SORT: Robust associations multi-pedestrian tracking," 2022, *arXiv:2206.14651*.

[37] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: A visual dynamic object-aware SLAM system," 2020, *arXiv:2005.11052*.

[38] Y.-K. Lin, W.-C. Lin, and C.-C. Wang, "Asynchronous state estimation of simultaneous ego-motion estimation and multiple object tracking for LiDAR-inertial odometry," in *Proc. IEEE Int. Conf. Robotics Automat.*, May 2023, pp. 1–7.

[39] B. Bescos, C. Campos, J. D. Tardos, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5191–5198, Jul. 2021.

[40] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-centric SORT: Rethinking SORT for robust multi-object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 9686–9696.

[41] Y. Du et al., "StrongSORT: Make DeepSORT great again," *IEEE Trans. Multimedia*, vol. 25, pp. 8725–8737, 2023.

[42] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[43] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Inf. Fusion*, vol. 14, no. 1, pp. 57–77, Jan. 2013.

[44] D. Marr, *Vision: A Computational Investigation Into the Human Representation and Processing of Visual Information*. Cambridge, MA, USA: MIT Press, 2010.

[45] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell. (IJCAI)*, Vancouver, BC, Canada, Aug. 1981, pp. 674–679. [Online]. Available: https://hal.science/hal-03697340

[46] J. Lin, C. Zheng, W. Xu, and F. Zhang, "R2LIVE: A robust, real-time, LiDAR-inertial-visual tightly-coupled state estimator and mapping," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7469–7476, Oct. 2021.

[47] J. Zhang, M. Henein, R. Mahony, and V. Ila, "Robust ego and object 6-DoF motion estimation and tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5017–5023.

[48] G. Lu and F. Zhang, "IMU-based attitude estimation in the presence of narrow-band noise," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 2, pp. 841–852, Apr. 2019.

[49] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021.

[50] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3061–3070.

[51] D. E. Catlin, *Estimation, Control, and the Discrete Kalman Filter*, vol. 71. Berlin, Germany: Springer, 2012.

[52] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, pp. 83–97, Mar. 1955.

[53] Ultralytics. (2022). *YOLOv5*. [Online]. Available: https://github.com/ultralytics/yolov5.git

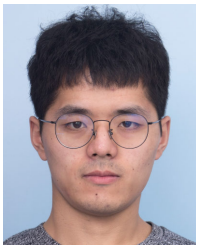
[54] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[55] J. Zhang and S. Singh, "Loam: LiDAR odometry and mapping in real-time," in *Proc. Robot., Sci. Syst.*, 2014, vol. 2, no. 9, pp. 1–9.

[56] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5135–5142.

[57] J. Lin and F. Zhang, "R3LIVE: A robust, real-time, RGB-colored, LiDAR-inertial-visual tightly-coupled state estimation and mapping package," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 10672–10678.

[58] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu, "SE-SSD: Self-ensembling single-stage object detector from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 14494–14503.
[59] M. Quigley et al., "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, 2009, vol. 3, no. 3, p. 5.
[60] M. Grupp. (2017). *EVO: Python Package for the Evaluation of Odometry and SLAM*. [Online]. Available: https://github.com/MichaelGrupp/evo
[61] J. Luiten et al., "HOTA: A higher order metric for evaluating multi-object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 548–578, Feb. 2021.
[62] 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.
[63] J. Shi, W. Wang, X. Li, Y. Yan, and E. Yin, "Motion distortion elimination for LiDAR-inertial odometry under rapid motion conditions," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–16, 2023.



Jian Shi was born in Heilongjiang in 1995. He received the B.S. degree from the College of Intelligent Systems Science and Engineering, Harbin Engineering University (HEU), Harbin, China, in 2018, where he is currently pursuing the Ph.D. degree in control science and engineering.

His research interests include sensor fusion and SLAM.

Wei Wang (Senior Member, IEEE) received the Ph.D. degree in navigation, guidance, and control from Harbin Engineering University (HEU), Harbin, China, in 2005.

He was a Post-Doctoral Research Associate at Harbin Institute of Technology, Harbin, from July 2006 to April 2009. From August 2008 to 2010, he was an Associate Professor with Harbin Engineering University, where he is currently a Professor. From January 2010 to December 2010, he was an Academic Visitor with Loughborough University, Loughborough, U.K. He has authored about more than 80 referred journal articles and conference papers. His current research interests include location, mapping, and image processing.

Mingyang Qi was born in Heilongjiang in 1995. He received the master's degree from the College of Science, Harbin Engineering University (HEU), Harbin, China, in 2021, where he is currently pursuing the Ph.D. degree with the Department of Intelligent Science and Engineering.

His research interests include indoor multisource information fusion and their applications in navigation technology, such as UWB indoor positioning and integrated navigation.

Xin Li received the Ph.D. degree in pattern recognition and artificial intelligence from Harbin Engineering University, Harbin, China, in 2011.

She was a Post-Doctoral Research Associate with Harbin Engineering University from July 2006 to April 2009, where she is currently an Associate Professor. She has authored about 30 refereed journal articles and conference papers. Her current research interests include wireless navigation systems and array signal processing.

Ye Yan received the B.S. and M.S. degrees from the Department of Automatic Control, National University of Defense Technology, Changsha, China, in 1994 and 1997, respectively, and the Ph.D. degree in aircraft design from the National University of Defense Technology in 2000.

He was a Lecturer, an Associate Professor, and a Professor at the National University of Defense Technology. He is currently a Professor with the National Innovation Institute of Defense Technology, Academy of Military Sciences China, Beijing, China. His research interests include human–machine interaction and mixed reality.