LINS: A Lidar-Inerital State Estimator for Robust and Fast Navigation

Chao Qin, Haoyang Ye, *Student Member, IEEE,* Christian E. Pranata, Jun Han, and Ming Liu, *Senior Member, IEEE*

***Abstract*—Robust and fast ego-motion estimation is a critical problem for autonomous robots. With high reliability and precision, 3D-lidar-based simultaneous localization and mapping (SLAM) has been widely used in the robotics community to solve this problem. However, the lidar alone is not enough to provide full autonomy to robot navigation in terms of robustness and operating scope, especially in feature-less scenes. In this paper, we present LINS: a lidar-inertial state estimator for robust and fast navigation. Our approach tightly couples the 3D lidar and the inertial measurement unit (IMU) by an iterative error-state Kalman filter (IESKF). To validate generalizability and long-time practicability, extensive experiments are performed in a variety of scenarios including the city, port, forest, and parking lot. The results indicate that LINS outperforms the lidar-only methods in terms of accuracy and it is faster than the state-of-the-art lidar-inertial fusion methods in nearly an order of magnitude.**

I. Introduction

Robust and fast navigation is a fundamental prerequisite to enable the application of autonomous robots: failures of the state estimator or slow response caused by long computation time can quickly lead to damage of the hardware and its surroundings. GPS is one of the most popular technology for positioning, but it has poor performance in the city area due to the multi-path effect [1]. Pose estimation by cameras has been widely discussed over the past decades [2]. However, the camera is vulnerable to illumination variation and not applicable during the night [3].

Recently 3D lidars play an important role in autonomous cars. As an active sensor, it can sense the surroundings with a horizontal field of view (FOV) of 360° by precise range measurements. In addition, it is insensitive to ambient lighting and optical texture in the scene, which makes it applicable in the dark [4]. However, lidar-based methods are challenged by two problems. The first problem is feature insufficiency in the outdoor environment. When the objects are far from the lidar, the received point clouds become discrete and the number of stable features is reduced dramatically. The second problem is the moving objects. Conventional approaches assume all things in the surroundings are static, whereas this assumption does not hold in real-world application and system stability will be severely undermined [5].

这里说了单独的雷达的挑战
1. 室外环境特征不充分
2. 往往假设目标都是静止的，没有考虑移动目标的问题

Research in visual-inertial odometry has proved that fusing the inertial measurement unit (IMU) can effectively handle the texture-less scenes and increase system stability [6].

Chao Qin, Haoyang Ye, Christian E. Pranata, and Ming Liu are with the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, Kowloon, Hong Kong

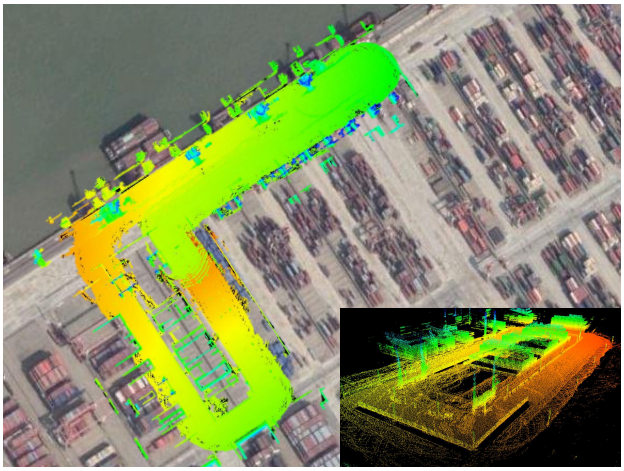


Fig. 1. Global 3D Map built by LINS with a Velodyne VLP-16 and an Xsens MTi-G-710 IMU in a port. We can see the good superposition of the map with Google Map.

IESKF框架是本来就有的，这里只是直接拿来用了

In this paper, we introduce **LINS**, a robust and fast lidar-inertial state estimator that based on an iterative error-state Kalman filter (IESKF) [7]. It fuses a 3D lidar and an IMU in a tightly-coupled scheme, where their measurements are jointly optimized from the raw measurement level. The main contributions of our work are as follows:

- We propose a lidar-inertial state estimator for robust and fast ground-vechicle navigation. It realizes a similar performance to the state-of-the-art methods but with much higher computational efficiency. 性能差不多，但是计算的速度更快
- Experiments in indoor, outdoor, and dynamic environments are performed to demonstrate generalizability and robustness.
- To the best of our knowledge, this is the first IESKF-based ego-motion estimator for lidar-inertial odometry and mapping.

II. Related Work

There are hundreds of works on lidar-based odometry and mapping algorithms in the literature. We restrict our attention to related work on the 6-DOF ego-motion estimator and the sensor fusion algorithms.

Most approaches are variations of the well-known iterative closest point (ICP) scan matching method which is based on scan-to-scan registration. [8, 9] had surveyed efficient variants of ICP. For real-time application, J. Zhang and S. Singh [10] established LOAM which sequentially registers extracted edge

and planar features to an incrementally built global map. T. Shan et al. [11] made further adaptations based on the original LOAM which include the ground plane extraction and point cloud segmentation algorithms and proposed LeGO for the unmanned-ground-vehicle (UGV) navigation.

There are several methods trying to combine the lidar with the IMU. The simplest way to deal with lidar and inertial measurements is loosely-coupled sensor fusion. In this scheme, IMU information generally serves as priors for the states in lidar-based methods. IMU-aided LOAM [10] took the orientation and translation calculated by the IMU as priors to facilitate the convergence of optimization. If a pre-built map is available, [12] combined the IMU measurements with pose estimates from a Gaussian particle filter via an error-state Kalman filter. Loosely-coupled fusion is computationally efficient and easy to implement but is less accurate than tightly-coupled methods [13] since it takes odometry as a black box and does not update it with measurements from the IMU [14]. Tightly-coupled fusions are either based on filtering methods or graph optimization methods [15]. Hesch et al. [16] introduced a lidar-aided inertial Kalman filter based on a 2D lidar for indoor mapping. The IMU measurements were integrated to obtain pose estimates, which would be further corrected through line-to-plane correspondences. However, this work was not suitable for outdoor navigation because it assumed all measured planes are in orthogonal structure. LIPS [17] leveraged the graph optimization method to fuse the inertial pre-integration measurements and the plane primitive measurements from a 3D lidar[18], but it was mainly tested indoors. H. Ye et al. [14] established LIOM which a tightly-coupled 3D lidar-inertial odometry and mapping system based on graph optimization. A rotation-constrained mapping was applied to align the lidar with the global map and gravity to further improve the accuracy,

The core of filtering based approaches is the Kalman filter and the core of optimization-based approaches is the Gauss-Newton method [19]. The link between them is the iterative extented kalman filter (IEKF), which is a filter yet preserving some characteristics of the Gauss-Newton method. IEKF reaches a balance between computational efficiency and accuracy [20], making it a suitable state estimator for real-time robot navigation. To our best knowledge, no work has been done that leverages the IEKF or IESKF to fuse a 3D lidar and an IMU in a tightly-coupled scheme, which is the focus of this work.

Our work is similar to LeGO [11] which is a lightweight and ground-optimized lidar odometry and mapping system. The difference is that we fuse IMU in the lidar odoemtry module and increase the odometry output frequency from 10 Hz to 400 Hz.

III. Lidar-Inertial Odometry and Mapping

A. System Overview

Our goal is to estimate the 6-DOF ego-motion and establish a global map by measurements obtained from a 3D lidar and an IMU. An overview of the proposed framework is shown in Fig. 2. The overall system consists of four modules: the feature extraction, state propagation, lidar-inertial odometry, and mapping modules. The feature extraction module aims at extracting stable features from the point cloud. The state propagation module performs state and covariance matrix predictions once a new IMU measurement is received. Their results are sent to the lidar-inertial odometry where the states are updated iteratively by an IESKF. Finally, the mapping module refines the estimated pose by a global map and integrate the new point cloud into the map. In the rest of this section, we start at introducing the feature extraction module in Sect. III-B. The IESKF-based lidar-inertial odometry is illustrated in Sect. III-C, followed by the description of the lidar mapping method in Sect. III-D.

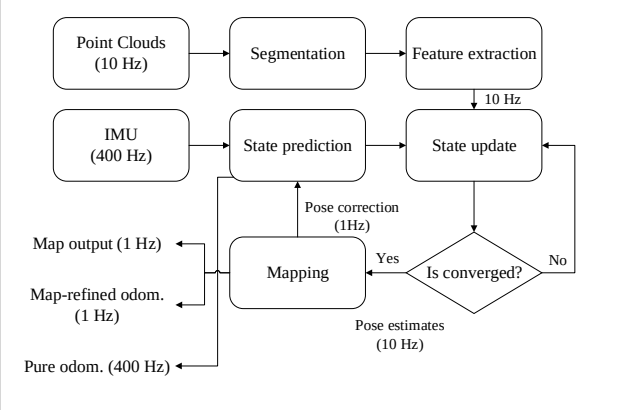


Fig. 2. Pipeline of the LINS system.

B. Feature Extraction

We apply a similar feature extraction algorithm proposed in [11]. At first, the ground plane is detected and removed from the original point cloud using the algorithm presented in [21]. Then an image-based segmentation method is utilized to cluster the points. In the next step, we remove the clusters with their size less than 30, since those whose sizes less than 30 are very possible to correspond to unstable or dynamic objects. Following the definition of the smoothness c in [10], we calculate the smoothness of all segmented points and sort them by row. A threshold is set to split those points into two categories, edge and planar features, according to their c. We then select n edge features with the highest c and m planar features with lowest c from each row. Note that the ground points are treated as the planar features and will be selected in this step. Finally, we perform scan-to-scan matching by the same algorithm used in [10].

C. Lidar-Inertial Odometry with IESKF

The lidar-inertial odometry aims at estimating the pose of the IMU-affixed frame b with respect to global frame of reference w. However, instead of directly estimating the pose with respect to w, we use an IESKF to solve the scan-to-scan transformation and then integrate it to the historical results to obtain new pose estimates represented in w. There are two reasons for doing that: (1) scan-to-scan matching only provides

local measurements instead of global measurements and (2) without transforming the point cloud from the local frame to the global frame, it is more time-saving.

Even though the pose and its covariance matrix will be reset after each scan-to-scan estimation, some states and their covariance matrices remain in the filter. In our case, these states are the acceleration and gyroscope biases.

Following the Kalman filtering traditions, there are two steps, *propagation* and *update*, which is the same as the extended Kalman filter (EKF). Except for the difference in the state definition, the major difference between EKF and IESKF is that there is an iterative loop in the update step of IESKF while a single loop in the update step of EKF [19]. The details of these steps are introduced below.

1) State Definitions: The global IMU state vector is described by the vector:

$$\mathbf{x}_b^w := [\mathbf{p}_b^w, \mathbf{v}_b^w, \mathbf{q}_b^w, \mathbf{b}_a, \mathbf{b}_g] \tag{1}$$

where $\mathbf{p}_b^w$ and $\mathbf{v}_b^w$ express the position and velocity of frame b with respect to frame w, respectively. $\mathbf{q}_b^w$ denotes the unit quaternion describing the rotation from frame b to frame w. $\mathbf{b}_a$ is the acceleration bias and $\mathbf{b}_g$ is the gyroscope bias [22].

The state vector for scan-to-scan estimation is defined in two IMU-affixed frames, b_k and b_{k+1}, as:

$$\mathbf{x}_{b_{k+1}}^{b_k} := \left[\mathbf{p}_{b_{k+1}}^{b_k}, \mathbf{v}_{b_{k+1}}^{b_k}, \mathbf{q}_{b_{k+1}}^{b_k}, \mathbf{b}_{a_{k+1}}, \mathbf{b}_{g_{k+1}}\right]. \tag{2}$$

where $\mathbf{p}_{b_{k+1}}^{b_k}$ and $\mathbf{q}_{b_{k+1}}^{b_k}$ denote the translation and rotation from frame b_{k+1} to frame b_k, respectively. $\mathbf{v}_{b_{k+1}}^{b_k}$ is the velocity of frame b_{k+1} represented in frame b_k. $\mathbf{b}_{a_{k+1}}$ and $\mathbf{b}_{g_{k+1}}$ are current acceleration and gyroscope biases, respectively.

Using equations below, we can integrate the $\mathbf{x}_{b_{k+1}}^{b_k}$ to $\mathbf{x}_{b_k}^w$ to obtain $\mathbf{x}_{b_{k+1}}^w$:

$$\mathbf{x}_{b_{k+1}}^w = \begin{bmatrix} \mathbf{p}_{b_{k+1}}^w \\ \mathbf{v}_{b_{k+1}}^w \\ \mathbf{q}_{b_{k+1}}^w \\ \mathbf{b}_{a_{k+1}} \\ \mathbf{b}_{g_{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{b_k}^w + \mathbf{R}_{b_k}^w \mathbf{p}_{b_{k+1}}^{b_k} \\ \mathbf{R}_{b_k}^w \mathbf{v}_{b_{k+1}}^{b_k} \\ \mathbf{q}_{b_k}^w \otimes \mathbf{q}_{b_{k+1}}^{b_k} \\ \mathbf{b}_{a_{k+1}} \\ \mathbf{b}_{g_{k+1}} \end{bmatrix} \tag{3}$$

where $\mathbf{R}_{b_k}^w$ is the equivalent matrix representation of quaternion $\mathbf{q}_{b_k}^w$ [23] and $\otimes$ denotes the quaternion product [24].

Here we introduce an error-state representation, which has several benefits [25] compared to a nominal state representation. One of the key advantages is that the rotation error is expressed as a 3D vector which is the minimal representation for the rotation and it is also free of singularities because the error-state is always close to zero. We define $\delta\mathbf{x}$ the error state vector of $\mathbf{x}_{b_{k+1}}^{b_k}$. As detailed in [24], it is expressed as:

$$\delta\mathbf{x} := [\delta\mathbf{p}, \delta\mathbf{v}, \delta\boldsymbol{\theta}, \delta\mathbf{b}_a, \delta\mathbf{b}_g] \tag{4}$$

where $\delta\boldsymbol{\theta} \in \mathbb{R}^3$ denotes the small-angle perturbation vector. A boxplus operator $\boxplus$ is introduced to uptate $\mathbf{x}_{b_{k+1}}^{b_k}$ by $\delta\mathbf{x}$ as:

$$\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta\mathbf{x} = \begin{bmatrix} \mathbf{p}_{b_{k+1}}^{b_k} + \delta\mathbf{p} \\ \mathbf{v}_{b_{k+1}}^{b_k} + \delta\mathbf{v} \\ \mathbf{q}_{b_{k+1}}^{b_k} \otimes \exp(\delta\boldsymbol{\theta}/2) \\ \mathbf{b}_{a_{k+1}} + \delta\mathbf{b}_a \\ \mathbf{b}_{g_{k+1}} + \delta\mathbf{b}_g \end{bmatrix} \tag{5}$$

where $\exp(\cdot)$ maps the angle vector to its quaternion representation [24].

2) Propagation: According to [24], the raw accelerometer and gyroscope measurements from IMU at time t, $\mathbf{a}_{m_t}$ and $\boldsymbol{\omega}_{m_t}$, are given by:

$$\mathbf{a}_{m_t} = \mathbf{R}_t^T(\mathbf{a}_t - \mathbf{g}) + \mathbf{b}_{a_t} + \mathbf{n}_a \tag{6}$$

$$\boldsymbol{\omega}_{m_t} = \boldsymbol{\omega}_t + \mathbf{b}_{\omega_t} + \mathbf{n}_\omega \tag{7}$$

where $\mathbf{g}$ is the gravity vector in the local navifation frame, $\mathbf{a}_t$ is the true acceleration, and $\boldsymbol{\omega}_t$ is the true angular rate. $\mathbf{n}_a$ and $\mathbf{n}_g$ are additive noises in $\mathbf{a}_{m_t}$ and $\boldsymbol{\omega}_{m_t}$, respectively, which are assumed as Gaussian, $\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_a^2)$, $\mathbf{n}_g \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_g^2)$. We model the $\mathbf{b}_{a_t}$ and $\mathbf{b}_{\omega_t}$ as random walk, whose derivatives are Gaussian, $\mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b_a}^2)$, $\mathbf{n}_{b_g} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b_g}^2)$:

$$\dot{\mathbf{b}}_{a_t} = \mathbf{n}_{b_a}, \dot{\mathbf{b}}_{g_t} = \mathbf{n}_{b_g} \tag{8}$$

The linearized continuous-time model for the IMU error state [22] can be expressed by:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{F}_t\delta\dot{\mathbf{x}}(t) + \mathbf{G}_t\mathbf{w} \tag{9}$$

where $\mathbf{w} = [\mathbf{n}_a, \mathbf{n}_g, \mathbf{n}_{b_a}, \mathbf{n}_{b_g}]^T$ is the system noise vector. Similar to [15], the matrices $\mathbf{F}_t$ and $\mathbf{G}_t$ are derived as follows:

$$\mathbf{F}_t = \begin{bmatrix} 0 & \mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{R}_t^{b_k}[\mathbf{a}_{m_t} - \mathbf{b}_{a_t}]_\times & -\mathbf{R}_t^{b_k} & 0 \\ 0 & 0 & -[\boldsymbol{\omega}_{m_t} - \mathbf{b}_{\omega_t}]_\times & 0 & -\mathbf{I} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

$$\mathbf{G}_t = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\mathbf{R}_t^{b_k} & 0 & 0 & 0 \\ 0 & -\mathbf{I}_{3\times3} & 0 & 0 \\ 0 & 0 & \mathbf{I}_{3\times3} & 0 \\ 0 & 0 & 0 & \mathbf{I}_{3\times3} \end{bmatrix} \tag{11}$$

where $[\mathbf{v}]_\times \in \mathbb{R}^{3\times3}$ denotes the skew symmetric matrix of a coordinate vector $\mathbf{v}$, $\mathbf{I} \in \mathbb{R}^{3\times3}$ the identity matrix, and $\mathbf{R}_t^{b_k}$ the rotation matrix from frame b_k to the current frame t. Transforming the continuous-time system equations to the discrete-time form yields the state and covariance propagation formulas:

$$\delta\mathbf{x}_k = (\mathbf{I} + \mathbf{F}_t\Delta t)\delta\mathbf{x}_{k-1} \tag{12}$$

$$\mathbf{P}_k = (\mathbf{I}+\mathbf{F}_t\Delta t)\mathbf{P}_{k-1}(\mathbf{I}+\mathbf{F}_t\Delta t)^T + (\mathbf{G}_t\Delta t)\mathbf{Q}(\mathbf{G}_t\Delta t)^T \tag{13}$$

where Δt is the IMU sample period and $\mathbf{Q}$ is the covariance matrix of $\mathbf{w}$.

3) Update: The update step of the IESKF can be linked to an optimization problem considering the deviation from the state prior $\mathbf{x}_{b_{k+1}}^{\bar{b_k}}$ and the residual functions from the measurement model.

The measurement model of the lidar scan-to-scan matching is described as follows. Let $h_e(\cdot)$ and $h_p(\cdot)$ be the residual functions for point-edge and point-plane pairs between two consecutive scans. Following [10], they can be expressed as:

$$h_e(\mathbf{x}) = \frac{|(\hat{\mathbf{P}}_i^{l_k} - \mathbf{P}_j^{l_k}) \times (\hat{\mathbf{P}}_i^{l_k} - \mathbf{P}_m^{l_k})|}{|\mathbf{P}_j^{l_k} - \mathbf{P}_m^{l_k}|} \tag{14}$$

$$h_p(\mathbf{x}) = \frac{|(\hat{\mathbf{P}}_i^{l_k} - \mathbf{P}_j^{l_k})^T((\mathbf{P}_j^{l_k} - \mathbf{P}_m^{l_k}) \times (\mathbf{P}_j^{l_k} - \mathbf{P}_n^{l_k}))|}{|(\mathbf{P}_j^{l_k} - \mathbf{P}_m^{l_k}) \times (\mathbf{P}_j^{l_k} - \mathbf{P}_n^{l_k})|} \tag{15}$$

, where

$$\hat{\mathbf{P}}_i^{l_k} = \mathbf{R}_l^{b^T}(\mathbf{R}_{b_{k+1}}^{b_k}(\mathbf{R}_l^b \hat{\mathbf{P}}_i^{l_{k+1}} + \mathbf{p}_l^b) + \mathbf{p}_{b_{k+1}}^{b_k} - \mathbf{p}_l^b) \quad (16)$$

is the estimated point projected from $(k+1)\ th$ scan to $k\ th$ scan. In essence, $h_e(\cdot)$ describes the distance between point $\hat{\mathbf{P}}_i^{l_k}$ and its corresponding edge $\overline{\mathbf{P}_j^{l_k}\mathbf{P}_m^{l_k}}$, while $h_p(\cdot)$ is the distance between point $\hat{\mathbf{P}}_i^{l_k}$ and its matched plane which is represented by three points, $\mathbf{P}_j^{l_k}$, $\mathbf{P}_m^{l_k}$, and $\mathbf{P}_n^{l_k}$. Reader can refer to [10] for how these matchinges are acquired. $\mathbf{R}_l^b$ and $\mathbf{p}_l^b$ together denote the relative transformation between the lidar and IMU which are calibrated offline.

With the state prior $\mathbf{x}_{b_{k+1}}^{\bar{b_k}}$, propagated covariance matrix $\mathbf{P}_k$, and the measurement model, we obtain the equivalent cost function of the IESKF as:

$$\min_{\delta\mathbf{x}_k} \|\delta\mathbf{x}_k\|_{(\mathbf{P}_k)^{-1}} + \|h(\mathbf{x}_{b_{k+1}}^{b_k^+})\|_{(\mathbf{J}_k\mathbf{M}_k\mathbf{J}_k^T)^{-1}} \quad (17)$$

where $\|\cdot\|_{\mathbf{W}}$ denotes the L_2 norm weighted by matrix $\mathbf{W}$. $\mathbf{x}_{b_{k+1}}^{b_k^+} = \mathbf{x}_{b_{k+1}}^{\bar{b_k}} \boxplus \delta\mathbf{x}_k$ is a-posteriori estimate, $h(\cdot)$ is the augmented vector of $h_e(\cdot)$ and $h_p(\cdot)$, $\mathbf{J}_k$ is the jacobian of $h(\mathbf{x}_{b_{k+1}}^{b_k^+})$ with respect to the measurement noise, and $\mathbf{M}_k$ is the covariance matrix of the measurements.

However, in contrast to the EKF, an iterative scheme is employed where the problem is linearized around continuously refined linearization points and Eq. (17) can be rewritten as:

$$\min_{\Delta\mathbf{x}_k} \|\delta\mathbf{x}_{k,j} \boxplus \Delta\mathbf{x}_k\|_{(\mathbf{P}_k)^{-1}} + \|h(\mathbf{x}_{b_{k+1}}^{b_k^+}) + \mathbf{H}_{k,j}\Delta\mathbf{x}_k\|_{(\mathbf{J}_k\mathbf{R}_k\mathbf{J}_k^T)^{-1}} \quad (18)$$

where $\mathbf{H}_{k,j}$ is the jacobian of $h(\mathbf{x}_{b_{k+1}}^{b_k^+})$ with respect to the error-state and it is updated in every iteration with index j:

$$\mathbf{H}_{k,j} = \frac{\partial h}{\partial \delta\mathbf{x}_k}(\mathbf{x}_{b_{k+1}}^{b_k^+}) = \frac{\partial h(\mathbf{x}_{b_{k+1}}^{b_k^+})}{\partial \mathbf{x}_{b_{k+1}}^{b_k^+}} \frac{\partial \mathbf{x}_{b_{k+1}}^{\bar{b_k}} \boxplus \delta\mathbf{x}_k}{\partial \delta\mathbf{x}_k} \quad (19)$$

Setting the derivative of the cost function w.r.t. the incremental update $\Delta\mathbf{x}_k$ to zero and employing some matrix operation yields the following recursive solution:

$$\mathbf{K}_{k,j} = \mathbf{P}_k\mathbf{H}_{k,j}^T(\mathbf{H}_{k,j}\mathbf{P}_k\mathbf{H}_{k,j}^T + \mathbf{J}_{k,j}\mathbf{R}_k\mathbf{J}_{k,j}^T)^{-1} \quad (20)$$

$$\Delta\mathbf{x}_{k,j} = \mathbf{K}_{k,j}(\mathbf{H}_{k,j}\delta\mathbf{x}_{k,j} - h(\mathbf{x}_{b_{k+1},j}^{b_k^+})) \quad (21)$$

$$\delta\mathbf{x}_{k,j+1} = \delta\mathbf{x}_{k,j} \boxplus \Delta\mathbf{x}_k \quad (22)$$

whereby the iteration is terminated when the update $\Delta\mathbf{x}_{k,j}$ is below a certain threshold. Note that the new closest points will be searched before every iteration to further minimize the error metric. The method to find the corresponding edges and planes is introduced in detail in [10]. Finally, the error-state covariance matrix $\mathbf{P}_k$ is only updated once the process has converged after n iterations:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_{k,n}\mathbf{H}_{k,n})\mathbf{P}_k(\mathbf{I} - \mathbf{K}_{k,n}\mathbf{H}_{k,n})^T + \mathbf{K}_{k,n}\mathbf{R}_k\mathbf{K}_{k,n}^T \quad (23)$$

Once the state has properly converged, we update the global state using Eq. (3) and reset the temporary state to wait for the next scan.

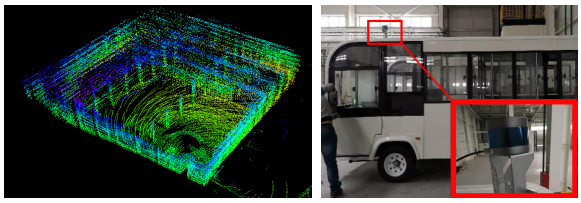
D. Lidar Mapping

The lidar mapping module matches features to a surrounding point cloud from the global map to further refine the pose transformation and expand the global map by the new point cloud. We use the mapping module proposed in [11]. Due to the space issue, we refer the reader to the description from [10, 11] for the detailed matching and optimization procedure.

IV. Experiments

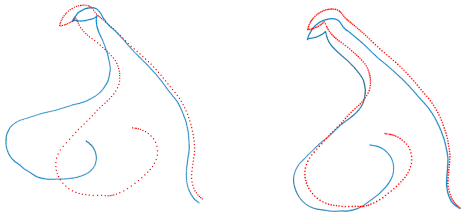
We now describe a series of experiments to qualitatively and quantitatively analyze the performance of the proposed LINS system and compare it with other state-of-the-art methods on a laptop computer with 2.4GHz quad cores and 8Gib memory. All algorithms are implemented in C++ and executed using the robot operating system (ROS) [26] in Ubuntu Linux.

Different from previous works that only consider the performance of the map-refined odometry[1] (MRO), we attach importance to the pure odometry (PO) performance, since: (1) the robot controller requires high-frequency pose feedback and only the PO (400 Hz) can meet this demand, (2) the PO is very useful in dynamic scenes where the MRO may fail due to the changing map, and (3) the performance of the MRO is deeply influenced by the PO because it serves as the initial pose estimates for the mapping module.



(a) Map of the parking lot built by LINS. (b) Bus and lidar installation.

Fig. 3. The sensor configuration for indoor tests.



(a) Trajectories of LeGO-PO (blue line) and LeGO-MRO (red dot). (b) Trajectories of LINS-PO (blue line) and LINS-MRO (red dot).

Fig. 4. Results of the indoor experiment with comparison against LeGO.

A. Indoor Experiment

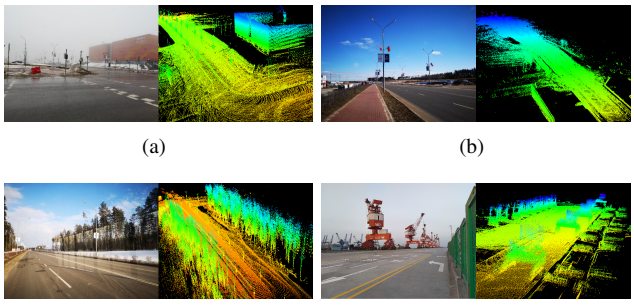
In the indoor tests, an indoor parking lot is chosen as the experiment area as shown in Fig. 3(a). We installed our sensor

[1]The map-refined odometry refers to the odometry after map-correction by the mapping module as in [10]. It is more accurate than the pure odometry yet with a much lower output frequency (1 Hz)

TABLE I
REATIVE ERRORS FOR MOTION ESTIMATION DRIFT (%)

		Num.of Features		Map-Refined Odometry				Pure Odometry			
Dataset	Dist. (m)	Edge	Planar	LOAM [10]	LeGO [11]	LIOM [14]	LINS	LOAM	LeGO	LIOM	LINS
Urban	1100	85	2552	72.91	10.17	**1.76**	2.98	76.84	30.13	4.44	**3.23**
Port	1264	103	2487	2.16	3.35	1.40	**1.25**	4.64	8.70	**1.72**	2.12
Park	117	420	3598	19.35	1.97	2.61	**1.28**	26.50	26.08	13.60	**5.77**
Forest	371	99	2633	5.59	3.66	9.58	**3.16**	10.60	18.93	12.96	**6.09**
Parking Lot	144	512	5555	1.21	1.12	**1.03**	1.08	5.38	6.62	2.17	**1.57**

Note: the bold style denotes the smallest drift.



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

Fig. 5. Photos and maps of (a) a building in the urban area, (b) a wide and open highway, (c) a road through a forest, and (d) a dock next to the ocean. All maps are produced by LINS.

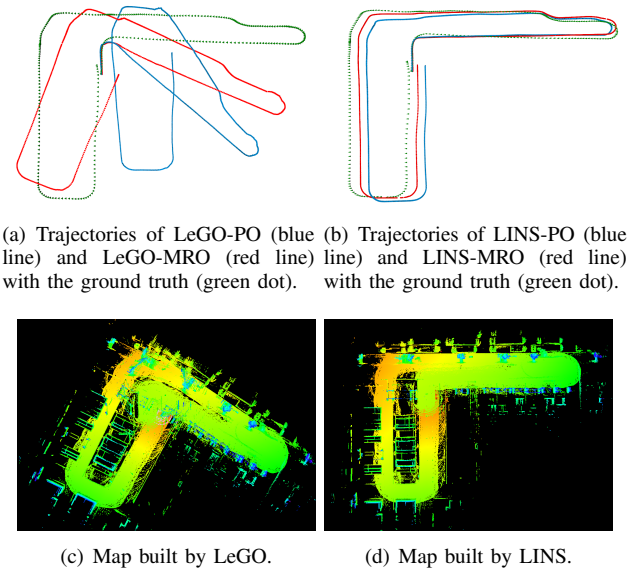
suite on a bus as shown in Fig. 3(b), where a RS-LiDAR-16 lidar (10 Hz) was mounted on the top and an IMU (400 Hz) was placed inside the bus.

To show how fusion algorithms improve accuracy, we compared our result with LeGO, which has a similar feature extraction and mapping strategies. Fig. 4(a) is the output from LeGO and Fig. 4(b) is the result from LINS. We see noticeable drifts occurred in the x, y, z-axis, and yaw angle in the LeGO-PO trajectory. In contrast, LINS efficiently eliminates these drifts by tightly-coupling IMU and shows a trajectory close to the MRO results.

B. Large-scale Environment

To verify the long-time practicability and generalizability, extensive experiments are carried out in different scenarios popular for unmanned-vehicle application. Fig. 5 shows some photos of the scenes and maps generated by LINS. Following [10], the measured drifts are compared to the distance traveled as the relative accuracy, and listed in Table I. The results show that LINS has great long-term stability and robustness in all scenarios. Due to the space issue, we only present analyses of two important dataset here

1) Port Datasets: We evaluate LINS in a port in Guangdong. The sensor suite we use consists of a Velodyne VLP-16 lidar (10Hz) and an Xsens MTi-G-710 IMU (400Hz) fixed on the top of a car. The ground-truth trajectories are obtained from an accurate GPS module.



(a) Trajectories of LeGO-PO (blue line) and LeGO-MRO (red line) with the ground truth (green dot).

(b) Trajectories of LINS-PO (blue line) and LINS-MRO (red line) with the ground truth (green dot).

(c) Map built by LeGO.

(d) Map built by LINS.

Fig. 6. Estimated trajectories and maps of LINS and LeGO.

The experiment area can be viewed in Fig. 1. We started from an open-air container depot. Then we head to a dock where one side is the ocean while the other side is filled with containers. After traveling the dock, we circled back to the original spot. The whole trajectory is 1264 meters and lasts approximately six minutes with 103 edge features observable in one scan (on average). We observe that the containers went in and out all the times, changing the map and making the MRO not reliable in the long run. Therefore we focus more on the PO performance in the analysis.

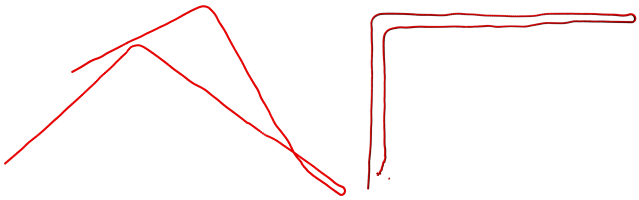
The trajectory of LeGO is shown in Fig. 6(a) and the trajectory of LINS is shown in Fig. 6(b). In Fig. 6(a), we see a serious drift occurred in the yaw angel when the car made its first turn. One of the major reason is lacking features. During the turn, the number of edge features in one scan is reduced to 33, much lower than the average feature number. The bad yaw estimation further leads to a deformed map as shown in Fig. 6(c). However, LINS and LIOM do not have this problem (for space issue, the LIOM results are not plotted here). The final drift of LINS-PO is [5.02, 12.79, 7.80] m in x, y and z-axis, which occupies 2.12% with respect

TABLE II
RUNTIME OF MODULES FOR PROCESSING ONE SCAN

Method	Odometry (ms)	Mapping (ms)
LIOM	276.40	170.97
LINS	29.26	54.98

to the total trajectory length, while the drift of the LIOM-PO is 1.72%, slightly smaller than LINS-PO. A possible reason is that LIOM-PO maintains a local map to estimate the scan-to-scan transformation instead of just using the points from the last scan. The denser point cloud helps LIOM to find correspondences that are more stable, resulting in better accuracy. Finally, we can visually inspect that the map built by our LINS can be precisely aligned with the satellite map as shown in Fig. 1 and Fig. 6(d).

Fig. 7. Estimated trajectory of the urban experiment aligned with Google Map. The red line is the final estimated trajectory from LINS-MRO.

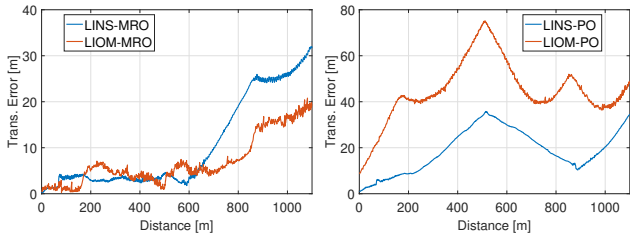


(a) (b)

Fig. 8. Estimated trajectories for the urban experiment. In Fig. 8(a), results from LeGO-MRO went bad during each turn in feature-less region (intersection). Fig. 8(b) shows results from LIOM. We can see that LIOM performs as well as LINS: no obvious drift shows up in feature-less region.

2) Urban Datasets: A large-scale urban dataset was collected with the same sensor suite in the indoor experiment. With a total length of 1.42 km, it involved a variety of environments, containing features such as buildings, streetlights, and sidewalks. Notably, parts of the dataset have very few features. For instance, the intersection area in the vicinity of open space only has 56 edge features in one scan. It is a very significant experiment to test the stability and durability of LINS in a complex environment.

The estimated trajectory of LINS-MRO is aligned with Google Map in Fig. 7. Compared with Google Map, we can see our results are almost drift-free during the test. Fig. 8(a) shows the result from LeGO-MRO. The drift in the yaw angle is accumulated when encountered a turn in feature-less places, whereas LINS achieves low drift in every turn.

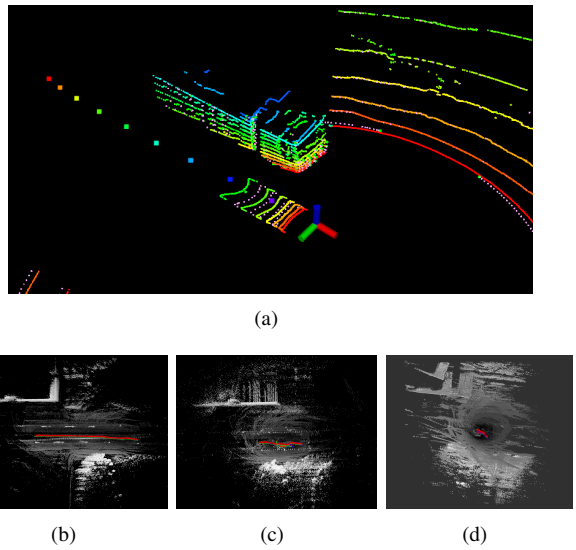


(a) (b)

Fig. 9. Translation errors of LINS and LIOM w.r.t (a) the PO and (b) the MRO

We then compare LINS with LIOM, another tighly-coupled lidar-inertial fusion algorithm. Fig. 8(b) is the trajectory of LIOM-MRO. Figs. 9(a) 9(b) compare the translation error of LINS and LIOM. In the error plot, we see two methods exhibit close performance in terms of the accuracy.

Table II showcases the mean runtimes w.r.t each module of LINS and LIOM. The results show that LINS is more computationally efficient and suitable for real-time navigation. LINS takes 29.26 ms in solving the odometry for one scan, faster than LIOM in nearly an order of magnitude. Regarding the mapping time, LINS takes 54.98 ms which is twice faster than LIOM.



(a)

(b) (c) (d)

Fig. 10. (a) Point cloud of the truck, and results of the dynamic environment of (b) LINS with comparison against (c) LeGO and (d) LOAM.

C. Dynamic Environment

To verify the robustness to moving objects, we collected a dataset with the presence of a moving truck and see how LINS reacts. Fig. 10(a) shows the point cloud of the truck. Fig. 10(d) and Fig. 10(c) are the PO and mapping results from LOAM and LeGO, respectively. We see both algorithms failed and resulted in overlapped maps. However, LINS do not have this

problem. As shown in Fig. 10(b), LINS performs well during the whole test and generated a map with high fidelity. A major reason is that the inertial information attenuates the influence of the moving point clouds by providing extra translational and rotational constraints.

V. Conclusion

In this paper, we present a robust and fast tightly-coupled lidar-inertial state estimator for UGV. Extensive experiments are conducted in indoor, outdoor, and dynamic environments. The results show that LINS can provide accurate position service in complex scenarios and exhibits great long-time practicability, generalizability, robustness, and stability. Analysis and comparison to the state-of-the-art methods are performed based on each experiment. Furthermore, the proposed approach is faster than the state-of-the-art method in nearly an order of magnitude, showing good real-time performance while keeping high localization quality. In future research, we will integrate the IMU at a deeper level, e.g. aiding the ground plane extraction module.

References

[1] D. Dong, M. Wang, W. Chen, Z. Zeng, L. Song, Q. Zhang, M. Cai, Y. Cheng, and J. Lv, "Mitigation of multipath effect in gnss short baseline positioning by the multipath hemispherical map," *Journal of Geodesy*, vol. 90, no. 3, pp. 255–262, 2016.

[2] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2502–2509, IEEE, 2018.

[3] C. H. Tong and T. D. Barfoot, "Gaussian process gauss-newton for 3d laser-based visual odometry," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5204–5211, IEEE, 2013.

[4] T. D. Barfoot, C. McManus, S. Anderson, H. Dong, E. Beerepoot, C. H. Tong, P. Furgale, J. D. Gammell, and J. Enright, "Into darkness: Visual navigation based on a lidar-intensity-image pipeline," in *Robotics Research*, pp. 487–504, Springer, 2016.

[5] C. Zou, B. He, L. Zhang, and J. Zhang, "Dynamic objects detection and tracking for a laser scanner and camera system," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 350–354, IEEE, 2017.

[6] Y. Ling, T. Liu, and S. Shen, "Aggressive quadrotor flight using dense visual-inertial fusion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1499–1506, IEEE, 2016.

[7] B. M. Bell and F. W. Cathey, "The iterated kalman filter update as a gauss-newton method," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 294–297, 1993.

[8] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm.," in *3dim*, vol. 1, pp. 145–152, 2001.

[9] F. Pomerleau, F. Colas, R. Siegwart, *et al.*, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends® in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[10] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, vol. 2, p. 9, 2014.

[11] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, IEEE, 2018.

[12] W. Zhen, S. Zeng, and S. Soberer, "Robust localization and localizability estimation with a rotating laser scanner," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6240–6245, IEEE, 2017.

[13] M. Li, B. H. Kim, and A. I. Mourikis, "Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera," in *2013 IEEE International Conference on Robotics and Automation*, pp. 4712–4719, IEEE, 2013.

[14] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2019.

[15] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[16] J. A. Hesch, F. M. Mirzaei, G. L. Mariottini, and S. I. Roumeliotis, "A laser-aided inertial navigation system (l-ins) for human localization in unknown indoor environments," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5376–5382, IEEE, 2010.

[17] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "Lips: Lidar-inertial 3d plane slam," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 123–130, IEEE, 2018.

[18] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," Georgia Institute of Technology, 2015.

[19] J. Gui, D. Gu, S. Wang, and H. Hu, "A review of visual inertial odometry from filtering and optimisation perspectives," *Advanced Robotics*, vol. 29, no. 20, pp. 1289–1301, 2015.

[20] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[21] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *2010 IEEE Intelligent Vehicles Symposium*, pp. 560–565, IEEE, 2010.

[22] E.-H. Shin, "Estimation techniques for low-cost inertial navigation," *UCGE report*, vol. 20219, 2005.

[23] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[24] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.

[25] V. Madyastha, V. Ravindra, S. Mallikarjunan, and A. Goyal, "Extended kalman filter vs. error state kalman filter for aircraft attitude estimation," in *AIAA Guidance, Navigation, and Control Conference*, p. 6615, 2011.

[26] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.