

LINS: A Lidar-Inertial State Estimator for Robust and Efficient Navigation

Chao Qin¹, Haoyang Ye¹, Christian E. Pranata¹, Jun Han, Shuyang Zhang¹, and Ming Liu¹

Abstract—We present LINS, a lightweight lidar-inertial state estimator, for real-time ego-motion estimation. The proposed method enables robust and efficient navigation for ground vehicles in challenging environments, such as feature-less scenes, via fusing a 6-axis IMU and a 3D lidar in a tightly-coupled scheme. An iterated error-state Kalman filter (ESKF) is designed to correct the estimated state recursively by generating new feature correspondences in each iteration, and to keep the system computationally tractable. Moreover, we use a robocentric formulation that represents the state in a moving local frame in order to prevent filter divergence in a long run. To validate robustness and generalizability, extensive experiments are performed in various scenarios. Experimental results indicate that LINS offers comparable performance with the state-of-the-art lidar-inertial odometry in terms of stability and accuracy and has order-of-magnitude improvement in speed.

I. INTRODUCTION

Ego-motion estimation is a fundamental prerequisite to enable most mobile robotic applications—poor real-time capability and failure of the algorithm can quickly lead to damage of the hardware and its surroundings. To this end, active sensors, such as lidars, are proposed to fulfill this task, which widely known as simultaneous localization and mapping (SLAM). Some of the key advantages of a typical 3D lidar include (i) wide horizontal field-of-view (FOV) [1] and (ii) invariance to ambient lighting conditions [2]. However, the lidar-based navigation system is sensitive to surroundings. Furthermore, the motion distortion [3] and sparse nature of point clouds [4] make it even worse in some challenging scenarios (e.g. a wide and open area).

Recent research has demonstrated that deficiencies of a stand-alone lidar can be compensated by fusing an IMU. An IMU, unlike lidar, is insensitive to surroundings. It provides accurate short-term motion constraints and generally works at a high frequency (e.g., 100 Hz–500 Hz). These features can help the lidar navigation system to recover point clouds from highly dynamic motion distortion and thus increase accuracy. However, the state-of-the-art lidar-inertial odometry (LIO) [5] which is based on graph optimization cannot be directly applied to real-time navigation due to high computational expense; for a single scan, it takes more than 100 millisecond to compute the lidar-inertial odometry and even more time to maintain a map.

In this paper, we propose LINS, a lightweight lidar-inertial state estimator for real-time navigation of unmanned-

¹The authors are with the RAM lab, the Hong Kong University of Science and Technology, Kowloon, Hong Kong, cscharlesqin@gmail.com, hy.ye@connect.ust.hk, shuyang.zhang1995@gmail.com, christianedwimp@yahoo.com, eelium@ust.hk

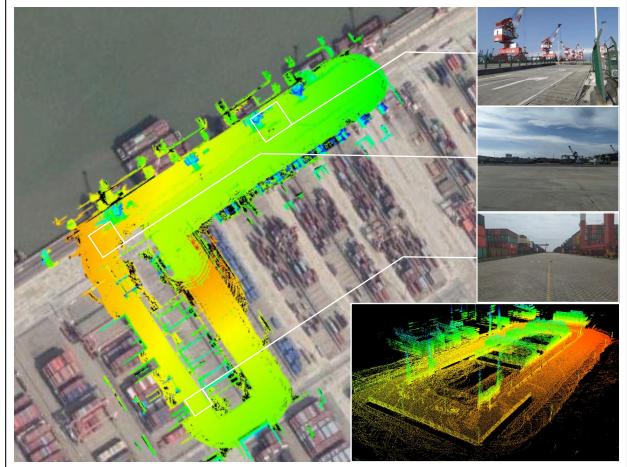


Fig. 1. 3D Map built by LINS with a Velodyne VLP-16 and an Xsens MTi-G-710 IMU in a port in Guangdong. We can observe the good alignment of the resulting map with Google Map, even in some feature-less environments as shown in images on the right.

ground-vehicles (UGVs). An iterated error-state Kalman filter (ESKF) is designed to ensure both accuracy and efficiency. To achieve long-term stability, we introduce a robocentric formulation of the state in which the local frame of reference is shifted at every lidar time-step, and the relative pose estimate between two consecutive local frames is used for updating the global pose estimate. The main contributions of our work are as follows:

- A tightly-coupled lidar-inertial odometry algorithm, which is faster than our previous work [5] by an order of magnitude, is proposed.
- We present a robocentric iterated ESKF, which is verified in various challenging scenarios and shows superior performance over the state of the art.
- The source code is available online¹. To the best of our knowledge, LINS is the first tightly-coupled LIO that solves the 6 DOF ego-motion via iterated Kalman filtering.

The remaining paper is organized as follows. In Sect. II, we discuss relevant literature. We give an overview of the complete system pipeline in Sect. III. The experimental results are illustrated in Sect. IV, followed by a conclusion in Sect. V.

¹<https://github.com/ChaoqinRobotics>

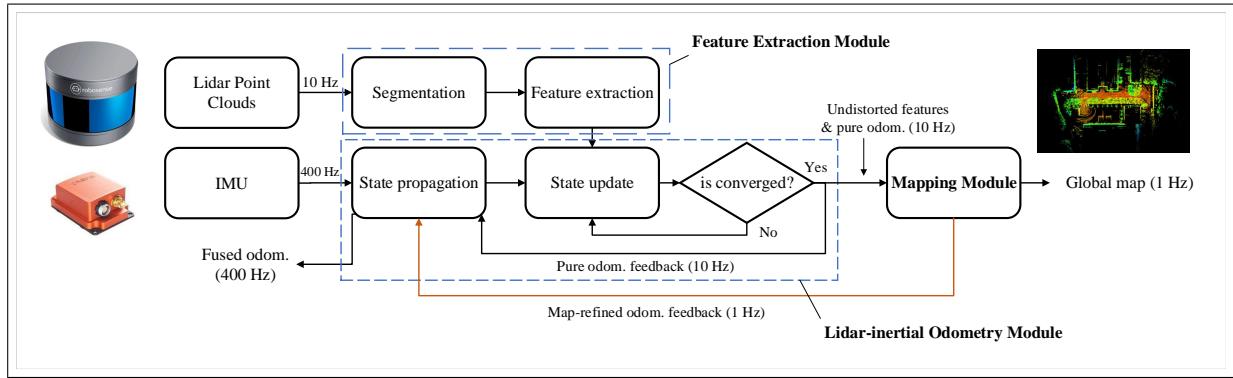


Fig. 2. Pipeline of the LINS system. The lidar-inertial odometry module, which consists of state propagation and update submodule, performs iterated Kalman filtering using IMU measurements and point cloud features extracted from the feature extraction module. The mapping module outputs a refined pose estimate along with a global 3D map. The refined pose estimate is combined with IMU measurements to generate high-output-rate results. Note that the focus of this work is the lidar-inertial odometry module.

II. RELATED WORK

There are hundreds of works on lidar-related odometry in the literature. We restrict our attention to related work on the 6 DOF ego-motion estimators and relevant fusion algorithms, which are categorized into the loosely-coupled and the tightly-coupled.

A. Lidar-only Methods

Many lidar-only approaches are variations of the well-known iterative closest point (ICP) scan matching method which is based on scan-to-scan registration. [6], [7] had surveyed efficient variants of ICP. For real-time application, [8] established LOAM which sequentially registers extracted edge and planar features to an incrementally built global map. [9] proposed LeGO which adapts the original LOAM to UGV application. By applying ground plane extraction and point cloud segmentation, LeGO filtered out unreliable features and showed great stability in areas covered with noisy objects, i.e., grass and trees. [10] provided an efficient loop closure mechanism to enable large-scale mapping in real time.

B. Loosely-coupled Lidar-IMU Fusion

Loosely-coupled methods deal with two sensors separately to infer their motion constraints which are fused later (e.g., [11], [12]). IMU-aided LOAM [8] took the orientation and translation calculated by the IMU as priors for optimization. [13] combined the IMU measurements with pose estimates obtained from a lidar-based Gaussian particle filter and a pre-built map. In general, loosely-coupled fusion is computationally efficient [14], but the decoupling of lidar and inertial constraints results in information loss [15].

C. Tightly-coupled Lidar-IMU Fusion

Tightly-coupled approaches directly fuse lidar and inertial measurements through joint-optimization, which can be categorized into optimization based [16], [17] and extended Kalman filter (EKF) based [18], [15]. [19] performed local trajectory optimization via minimizing constraints from the IMU and lidar together. [20] presented LIPS which

leveraged graph optimization over the inertial pre-integration constraints [21] and plane constraints from a lidar. [5] established LIO-mapping (for brevity, termed LIOM in the followings) which is also based on graph optimization but with a novel rotation-constrained mapping method to optimize the final poses and maps. However, constraint construction and batch optimization in a local map window are too time-consuming for real-time application. [22] introduced a lidar-aided inertial EKF based on a 2D lidar. But its application scenarios were limited to indoor environments because it required that all surrounding planes are in an orthogonal structure.

It is well known that EKF is vulnerable to linearization errors which may cause poor performance and even lead to divergence [23], [24]. This shortcoming becomes salient when it involves lidar-observed scan-to-scan constraints, which is deemed highly nonlinear if the initial pose is incorrect and causes wrong feature-matching results. To eliminate errors caused by wrong matching, we presented an iterated Kalman filtering [26] which can repeatedly find better matchings in each iteration. Besides, we adopted an error-state representation to guarantee linearization validity [27]. This characteristic distinguishes our method from the iterated extented Kalman filter [25].

III. LIDAR-INERTIAL ODOMETRY AND MAPPING

A. System Overview

Consider an UGV equipped with an IMU and a 3D lidar. Our goal is to estimate its 6 DOF ego-motion and to establish a global map simultaneously, as shown in Fig. 1. An overview of the system framework is depicted in Fig. 2. The overall system consists of three major modules: feature extraction, LIO, and mapping. (i) The feature extraction module aims at extracting stable features from raw point clouds. (ii) The LIO module, which consists of propagation and update submodules, carries out iterated Kalman filtering and outputs an initial odometry along with undistorted features. (iii) The mapping module refines the initial odometry by the global map and outputs a new odometry, followed by updating the map using new features. Due to the space issue, we only

focus on the odometry module. We refer the reader to [8], [9] for detailed procedures of feature extraction and mapping.

B. Feature Extraction

This module inputs the raw point cloud and outputs a group of edge features, \mathbb{F}_e , and a group of planar features, \mathbb{F}_p . Readers can see [9], [8] for detailed implementations.

C. Lidar-Inertial Odometry with Iterated ESKF

The LIO module uses IMU measurements and features extracted in two consecutive scans to estimate the relative transformation of the vehicle. We use a robocentric formulation to build the iterated ESKF because it prevents large linearization errors caused by ever-growing uncertainty [28], [29]. Let \mathcal{F}_w represent the fixed world frame, \mathcal{F}_{b_k} represent the IMU-affixed frame at k lidar time-step, and \mathcal{F}_{l_k} represent the lidar frame at k lidar time-step. Note that, in our work, the local frame is always set as the IMU-affixed frame at previous lidar time-step.

1) *State Definitions:* Let $\mathbf{x}_w^{b_k}$ denote the location of \mathcal{F}_w w.r.t. \mathcal{F}_{b_k} . Let $\mathbf{x}_{b_{k+1}}^{b_k}$ denote the local state that describes the relative transformation from $\mathcal{F}_{b_{k+1}}$ to \mathcal{F}_{b_k} :

$$\mathbf{x}_w^{b_k} := \left[\mathbf{p}_w^{b_k}, \mathbf{q}_w^{b_k} \right], \quad (1)$$

$$\mathbf{x}_{b_{k+1}}^{b_k} := \left[\mathbf{p}_{b_{k+1}}^{b_k}, \mathbf{v}_{b_{k+1}}^{b_k}, \mathbf{q}_{b_{k+1}}^{b_k}, \mathbf{b}_a, \mathbf{b}_g, \mathbf{g}^{b_k} \right], \quad (2)$$

where $\mathbf{p}_w^{b_k}$ is the position of \mathcal{F}_w w.r.t. \mathcal{F}_{b_k} and $\mathbf{q}_w^{b_k}$ is the unit quaternion describing the rotation from \mathcal{F}_w to \mathcal{F}_{b_k} . $\mathbf{p}_{b_{k+1}}^{b_k}$ and $\mathbf{q}_{b_{k+1}}^{b_k}$ represent the translation and rotation from $\mathcal{F}_{b_{k+1}}$ to \mathcal{F}_{b_k} . $\mathbf{v}_{b_{k+1}}^{b_k}$ is the velocity w.r.t. \mathcal{F}_{b_k} . \mathbf{b}_a is the acceleration bias and \mathbf{b}_g the gyroscope bias. It is important to note that the local gravity, \mathbf{g}^{b_k} (represented in \mathcal{F}_{b_k}), is also part of the local state.

For having good properties in state estimation [30], an error-state representation is used to solve $\mathbf{x}_{b_{k+1}}^{b_k}$. We denote an error term with δ and define the error vector of $\mathbf{x}_{b_{k+1}}^{b_k}$ as

$$\delta\mathbf{x} := [\delta\mathbf{p}, \delta\mathbf{v}, \delta\theta, \delta\mathbf{b}_a, \delta\mathbf{b}_g, \delta\mathbf{g}], \quad (3)$$

where $\delta\theta$ is a 3 DOF error angle.

According to the ESKF traditions, once $\delta\mathbf{x}$ is solved, we can obtain the final $\mathbf{x}_{b_{k+1}}^{b_k}$ by injecting $\delta\mathbf{x}$ into the state prior of $\mathbf{x}_{b_{k+1}}^{b_k}$, $-\mathbf{x}_{b_{k+1}}^{b_k}$. This is conducted via a boxplus operator \boxplus which is defined as:

$$\mathbf{x}_{b_{k+1}}^{b_k} = -\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta\mathbf{x} = \begin{bmatrix} -\mathbf{p}_{b_{k+1}}^{b_k} + \delta\mathbf{p} \\ -\mathbf{v}_{b_{k+1}}^{b_k} + \delta\mathbf{v} \\ -\mathbf{q}_{b_{k+1}}^{b_k} \otimes \exp(\delta\theta) \\ -\mathbf{b}_a + \delta\mathbf{b}_a \\ -\mathbf{b}_g + \delta\mathbf{b}_g \\ -\mathbf{g}^{b_k} + \delta\mathbf{g} \end{bmatrix}, \quad (4)$$

where \otimes denotes the quaternion product and $\exp : \mathbb{R}^3 \rightarrow SO(3)$ maps the angle vector to quaternion rotation [31].

2) *Propagation:* In this step, we propagate the error state, $\delta\mathbf{x}$, the error-state covariance matrix, \mathbf{P}_k , and the state prior, $-\mathbf{x}_{b_{k+1}}^{b_k}$, if a new IMU measurement arrives. The linearized continuous-time model [32] for the IMU error state is written as

$$\dot{\delta\mathbf{x}}(t) = \mathbf{F}_t \delta\mathbf{x}(t) + \mathbf{G}_t \mathbf{w}, \quad (5)$$

where $\mathbf{w} = [\mathbf{n}_a^T, \mathbf{n}_g^T, \mathbf{n}_{b_a}^T, \mathbf{n}_{b_g}^T]^T$ is the Gaussian noise vector (whose definitions are the same as [16]). \mathbf{F}_t is the error-state transition matrix and \mathbf{G}_t is the noise Jacobian at time t :

$$\mathbf{F}_t = \begin{bmatrix} 0 & \mathbf{I} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{R}_t^{b_k} [\hat{\mathbf{a}}_t]_\times & -\mathbf{R}_t^{b_k} & 0 & 0 \\ 0 & 0 & -[\hat{\omega}_t]_\times & 0 & -\mathbf{I}_3 & -\mathbf{I}_3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (6)$$

$$\mathbf{G}_t = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\mathbf{R}_t^{b_k} & 0 & 0 & 0 \\ 0 & -\mathbf{I}_3 & 0 & 0 \\ 0 & 0 & \mathbf{I}_3 & 0 \\ 0 & 0 & 0 & \mathbf{I}_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad (7)$$

where $[\cdot]_\times \in \mathbb{R}^{3 \times 3}$ transfers a 3D vector to its skew-symmetric matrix. $\mathbf{I}_3 \in \mathbb{R}^{3 \times 3}$ is the identity matrix and $\mathbf{R}_t^{b_k}$ is the rotation matrix from the IMU-affixed frame at time t to \mathcal{F}_{b_k} . $\hat{\mathbf{a}}_t$ and $\hat{\omega}_t$ are acceleration and angular rate at time t , respectively, and they are calculated by removing the biases and the gravity effect from the raw accelerometer measurement, \mathbf{a}_{m_t} , and gyroscope measurements, ω_{m_t} , as

$$\hat{\mathbf{a}}_t = \mathbf{a}_{m_t} - \mathbf{b}_a, \quad (8)$$

$$\hat{\omega}_t = \omega_{m_t} - \mathbf{b}_g. \quad (9)$$

Discretizing Equation (5) yields following propagation equations:

$$\delta\mathbf{x}_{t_\tau} = (\mathbf{I} + \mathbf{F}_{t_\tau} \Delta t) \delta\mathbf{x}_{t_{\tau-1}}, \quad (10)$$

$$\mathbf{P}_{t_\tau} = (\mathbf{I} + \mathbf{F}_{t_\tau} \Delta t) \mathbf{P}_{t_{\tau-1}} (\mathbf{I} + \mathbf{F}_{t_\tau} \Delta t)^T + (\mathbf{G}_{t_\tau} \Delta t) \mathbf{Q} (\mathbf{G}_{t_\tau} \Delta t)^T, \quad (11)$$

where $\Delta t = t_\tau - t_{\tau-1}$. t_τ and $t_{\tau-1}$ are consecutive IMU time-steps. \mathbf{Q} expresses the covariance matrix of \mathbf{w} , which is computed off-line during sensor calibration.

To predict $-\mathbf{x}_{b_{k+1}}^{b_k}$, the discrete-time propagation model for the robocentric state is required. Readers can refer to [15], [27] for details in integrating IMU measurements.

3) *Update:* We now present the iterated update scheme, which is the primary contribution of this work.

In iterated Kalman filtering, the state update can be linked to an optimization problem [7], [29] considering the deviation from the prior $-\mathbf{x}_{b_{k+1}}^{b_k}$ and the residual functions, $f(\cdot)$, (i.e., innovation²) derived from the measurement model:

$$\min_{\delta\mathbf{x}} \|\delta\mathbf{x}\|_{(\mathbf{P}_k)^{-1}} + \|f(-\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta\mathbf{x})\|_{(\mathbf{J}_k \mathbf{M}_k \mathbf{J}_k^T)^{-1}}, \quad (12)$$

²Note that innovation is the difference between the actual an expected measurements [25]

where $\|\cdot\|$ denotes the Mahalanobis norm. \mathbf{J}_k is the Jacobian of $f(\cdot)$ w.r.t. the measurement noise and \mathbf{M}_k is the covariance matrix of the measurement noise. The output of $f(\cdot)$ is actually a stacked residual vector calculated from point-edge or point-plane pairs. Given $\mathbf{x}_{b_{k+1}}^{b_k}$, the error term in $f(\cdot)$ that corresponds to $\mathbf{p}_i^{l_{k+1}}$, the i th feature point which is represented in $\mathcal{F}_{l_{k+1}}$, can be described as:

$$f_i(\mathbf{x}_{b_{k+1}}^{b_k}) = \begin{cases} \frac{|(\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_a^{l_k}) \times (\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_b^{l_k})|}{|\mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k}|} & \text{if } \mathbf{p}_i^{l_{k+1}} \in \mathbb{F}_e \\ \frac{|(\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_a^{l_k})^T((\mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k}) \times (\mathbf{p}_a^{l_k} - \mathbf{p}_c^{l_k}))|}{|(\mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k}) \times (\mathbf{p}_a^{l_k} - \mathbf{p}_c^{l_k})|} & \text{if } \mathbf{p}_i^{l_{k+1}} \in \mathbb{F}_p \end{cases}, \quad (13)$$

and we have

$$\hat{\mathbf{p}}_i^{l_k} = \mathbf{R}_l^{b^T} (\mathbf{R}_{b_{k+1}}^{b_k} (\mathbf{R}_l^b \mathbf{p}_i^{l_{k+1}} + \mathbf{p}_l^b) + \mathbf{p}_{b_{k+1}}^{b_k} - \mathbf{p}_l^b), \quad (14)$$

where $\hat{\mathbf{p}}_i^{l_k}$ is the transformed point of $\mathbf{p}_i^{l_{k+1}}$ from $\mathcal{F}_{l_{k+1}}$ to \mathcal{F}_{l_k} . \mathbf{R}_l^b and \mathbf{p}_l^b together denote the extrinsic parameters between the lidar and IMU (calculated in off-line calibration).

A physical explanation of Equation (13) is provided in the followings. For an edge point, it describes the distance between $\hat{\mathbf{p}}_i^{l_k}$ and its corresponding edge $\mathbf{p}_a^{l_k} \mathbf{p}_b^{l_k}$. For a planar point, it describes the distance between $\hat{\mathbf{p}}_i^{l_k}$ and its corresponding plane which is formed by three points, $\mathbf{p}_a^{l_k}$, $\mathbf{p}_b^{l_k}$, and $\mathbf{p}_c^{l_k}$. Details of how to choose $\mathbf{p}_a^{l_k}$, $\mathbf{p}_b^{l_k}$, and $\mathbf{p}_c^{l_k}$ can be found in [8].

We solve Equation (12) using following iterated update equations:

$$\mathbf{K}_{k,j} = \mathbf{P}_k \mathbf{H}_{k,j}^T (\mathbf{H}_{k,j} \mathbf{P}_k \mathbf{H}_{k,j}^T + \mathbf{J}_{k,j} \mathbf{M}_k \mathbf{J}_{k,j}^T)^{-1}, \quad (15)$$

$$\Delta \mathbf{x}_j = \mathbf{K}_{k,j} (\mathbf{H}_{k,j} \delta \mathbf{x}_j - f(-\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta \mathbf{x}_j)), \quad (16)$$

$$\delta \mathbf{x}_{j+1} = \delta \mathbf{x}_j + \Delta \mathbf{x}_j, \quad (17)$$

where $\Delta \mathbf{x}_j$ denotes the correction vector at j th iteration. $\mathbf{H}_{k,j}$ is the jacobian of $f(-\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta \mathbf{x}_j)$ w.r.t. $\delta \mathbf{x}_j$. Note that, in every iteration, we will find new matched edges and planes to further minimize the error metric, followed by computing new $\mathbf{H}_{k,j}$, $\mathbf{J}_{k,j}$, and $\mathbf{K}_{k,j}$. When $f(\mathbf{x}_{b_{k+1}}^{b_k})$ is below a certain threshold, say at the n th iteration, we update \mathbf{P}_k by

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k,n} \mathbf{H}_{k,n}) \mathbf{P}_k (\mathbf{I} - \mathbf{K}_{k,n} \mathbf{H}_{k,n})^T + \mathbf{K}_{k,n} \mathbf{M}_k \mathbf{K}_{k,n}^T. \quad (18)$$

Using Equation (4), we are able to obtain the final $\mathbf{x}_{b_{k+1}}^{b_k}$. The raw distorted features can now be undistorted using the estimated relative transformation.

Finally, we initialize the next state, $\mathbf{x}_{b_{k+2}}^{b_{k+1}}$, with

$$[\mathbf{0}_3, \mathbf{v}_{b_{k+1}}^{b_{k+1}}, \mathbf{q}_0, \mathbf{b}_a, \mathbf{b}_g, \mathbf{g}^{b_{k+1}}], \quad (19)$$

where \mathbf{q}_0 denotes identity quaternion. $\mathbf{v}_{b_{k+1}}^{b_{k+1}}$ and $\mathbf{g}^{b_{k+1}}$ can be computed by $\mathbf{v}_{b_{k+1}}^{b_{k+1}} = \mathbf{R}_{b_k}^{b_{k+1}} \mathbf{v}_{b_{k+1}}^{b_k}$ and $\mathbf{g}^{b_{k+1}} = \mathbf{R}_{b_k}^{b_{k+1}} \mathbf{g}^{b_k}$, respectively. Note that, the covariances regarding the velocity, biases, and local gravity remain in the covariance matrix, while the covariance corresponding to the relative pose is set to zero, i.e., no uncertainty for the robocentric frame of reference itself.

4) State Composition: In the robocentric formulation, every time when the update is finished, we need to update the global pose, $\mathbf{x}_w^{b_k}$, through a composition step as

$$\mathbf{x}_w^{b_{k+1}} = \begin{bmatrix} \mathbf{p}_w^{b_{k+1}} \\ \mathbf{q}_w^{b_{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{b_k}^{b_{k+1}} (\mathbf{p}_w^{b_k} - \mathbf{p}_{b_{k+1}}^{b_k}) \\ \mathbf{q}_{b_k}^{b_{k+1}} \otimes \mathbf{q}_w^{b_k} \end{bmatrix}. \quad (20)$$

5) Initialization: As described in Sect. III-C, the robocentric formulation can facilitate the initialization of the filter state. Regarding the initial parameter settings, in our implementation, (i) the initial acceleration bias and lidar-IMU extrinsic parameters are obtained via off-line calibration, while the initial gyroscope bias is the average of the corresponding stationary measurements, (ii) the initial roll and pitch are obtained from the unbiased acceleration measurements before moving, and (iii) the initial local gravity is acquired via transforming the gravity vector represented in navigation frame to current local frame using initial roll and pitch from (ii).

IV. EXPERIMENTS

We now evaluate the performance of LINS in different scenarios and compare it with LeGO [9], LOAM [8], and LIOM [5], on a laptop computer with 2.4GHz quad cores and 8Gib memory. All methods are implemented in C++ and executed using the robot operating system (ROS) [33] in Ubuntu Linux. In the following experiments, the mapping module of LINS is implemented by the mapping algorithm proposed in LeGO [9]. Most of the previous works only analyzed the performance of the final trajectory, i.e. the odometry already refined by the map. However, we found that the initial odometry, i.e. the odometry purely produced by the odometry module, has great impact to the overall performance. Therefore, we took both of them into account. To distinguish these two odometries, we termed the odometry refined by the map as map-refined odometry (MRO), and the initial odometry as pure odometry (PO).

A. Indoor Experiment

In the indoor test, a parking lot was chosen as the experiment area as shown in Fig. 4(a). We installed our sensor suite on a bus as shown in Fig. 4(b), where a RS-LiDAR-16 was mounted on the top and an IMU was placed inside the bus. Fig. 5(a), 5(b), and 5(c) provide the results from LINS, LeGO, and LOAM, respectively. Although we do not have ground truth, we can still visually inspect that LINS-PO's trajectory can be precisely aligned with the MRO trajectory (generally, MRO is almost drift-free indoors and more accurate than PO), while the LeGO-PO and LIOM-PO both have noticeable drifts in the yaw angle.

B. Large-scale Outdoor Environment

To verify generalizability and stability, experiments were carried out in four outdoor application scenarios: city, port, industrial park, and forest. Fig. 3 showcases some photos of the environments and corresponding maps generated by LINS. We measured the gap between the ground truth produced by a GPS receiver and the estimated position provided

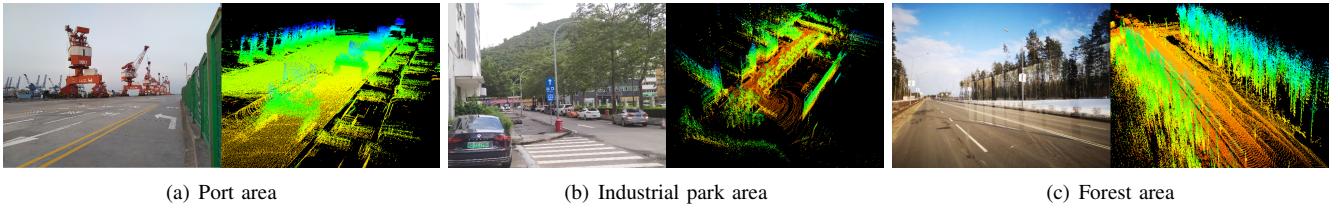


Fig. 3. Photos and corresponding maps (produced by LINS) of (a) a wide and open port area, (c) an industrial park with numerous buildings, trees, and cars, and (d) a clean road through a forest area. LINS performs well in all tested environments.

TABLE I
RELATIVE ERRORS FOR MOTION ESTIMATION DRIFT

Scenario	Dist. (m)	Num. of Features		Drifts of Map-Refined Odometry (%)				Drifts of Pure Odometry (%)			
		Edge	Planar	LOAM [8]	LeGO [9]	LIOM [5]	LINS	LOAM	LeGO	LIOM	LINS
City	1100	85	2552	72.91	10.17	1.76	1.79	76.84	30.13	4.44	4.42
Port	1264	103	2487	2.16	3.35	1.40	1.56	4.64	8.70	1.72	2.75
Park	117	420	3598	19.35	1.97	2.61	1.32	26.50	26.08	13.60	7.69
Forest	371	99	2633	5.59	3.66	9.58	3.31	10.60	18.93	12.96	7.27
Parking Lot	144	512	5555	1.21	1.12	1.05	1.08	5.38	6.62	2.17	1.72

by each method, which indicates the amount of drift, and then compare it to the distance traveled to yield a relative drift. The experimental results are listed in Table I.

In summary, LINS performed well in all tested scenarios. The detailed analysis for specific environments is conducted below.

1) Port Experiment: We evaluated LINS in a port in Guangdong. The sensor suite consisted of a Velodyne VLP-16 lidar and an Xsens MTi-G-710 IMU fixed on the top of a car. The ground-truth trajectory was offered by a GPS module. We started recording data from a path surrounded by containers. The car headed to a dock and then returned to the original spot after traveling a distance of 1264 meters. It is worth to mention that the containers would go in and out incessantly changing the global map, which may undermine the performance of MRO.

According to Table I, we find that LINS and LIOM present the lowest drifts. The relative drift of LINS-MRO is 1.56%, slightly higher than that of LIOM which is 1.40%, while the relative drift of LINS-PO is just 2.75%. The results indicate that combining IMU and lidar can effectively improve accuracy. Even though the relative drifts of LOAM and LeGO

seems to be small, they may suffer from huge orientation errors. Fig. 6(a) and 6(b) provide detailed trajectories and maps from LeGO and LINS. Compared to the ground truths (green lines), we find trajectories from LeGO (including MRO and PO) turned to an erroneous direction around the first turn. We can also visually inspect the deformation of the map built by LeGO in Fig. 6(c). In contrast, LINS exhibited good alignment with the ground-truth trajectory and the resulting map presented high fidelity to the real-world environment. Even in the first turn where features were insufficient (only about 30 edge features available per scan), LINS performed very well, which shows that our algorithm is more robust to feature-less scenes.

TABLE II
RUNTIME OF THE LIO MODULE PER SCAN

Method	City	Port	Park	Forest	Parking Lot
LIOM	143 ms	185 ms	201 ms	173 ms	223 ms
LINS	18 ms	19 ms	21 ms	20 ms	25 ms

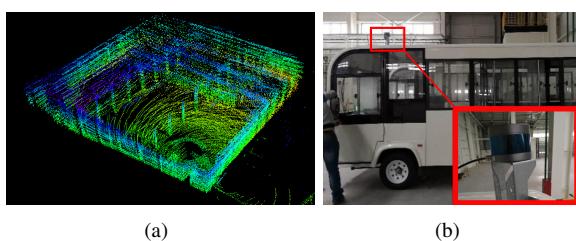


Fig. 4. The sensor configuration for indoor tests. (a) Map of the parking lot built by LINS. (b) Lidar installation. An IMU is stuck to the bus.

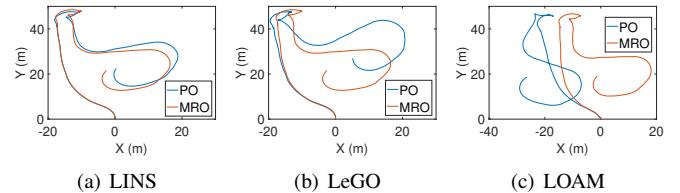


Fig. 5. Resulting trajectories in the indoor experiment of (a) LINS, (b) LeGO, and (c) LOAM. We observe that MRO trajectories from all methods look very similar, but their PO trajectories are totally different. The PO trajectory from LINS aligns much better with its MRO trajectory than other methods.

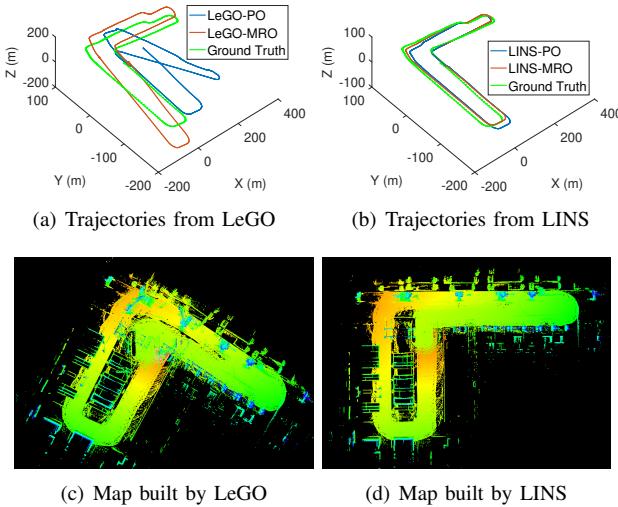


Fig. 6. Estimated trajectories and maps from LeGO and LINS. Note that the PO trajectories are drawn in blue lines, the MRO trajectories are drawn by red lines, and the GPS ground truths are drawn in green lines. We see that the trajectories from LINS is close to the ground truths, and the resulting map has higher fidelity than that of LeGO.

2) *Urban Experiment*: We carried out the urban experiment with the same sensor suite in Fig. 4(b). Positions produced by a GPS receiver were used as ground truths. It is worth to mention that in this scene, the average edge feature number is only 56 per scan, which is the lowest among all tested scenarios.

We first take a glance at the output of LeGO and LOAM, which can be seen in Fig. 7(b) and 7(c), respectively. We observed that huge orientation errors occur in almost every turns. Fig. 7(a) showcases the result of LINS when running on the same dataset. The resulting trajectories show good superposition with the real-world road, which confirms that LINS can stably run even in the feature-less scenes. The final MRO and PO drifts of LINS are 1.79% and 4.42%, respectively, very close to that of LIOM which are 1.76% and 4.44%, respectively. Moreover, in comparison to the trajectory from LIOM as shown in Fig. 7(d) and absolute trajectory error (ATE) as shown in Fig. 8(a) and 8(b), we can see the performance of LINS is close to that of LIOM in terms of accuracy. We analyze that LIOM benefits from rotation-constrained refinement in the mapping step, which results in higher accuracy in the MRO results.

C. Runtime Comparison

Table II compares the mean runtimes of the lidar-inertial odometry module in LINS and LIOM. We see that LINS is much faster than LIOM—LINS requires less than 30 millisecond in processing one scan, while LIOM always requires more than 100 millisecond. In some extreme case such as the parking lot where features abound, LIOM took up to 223 millisecond while LINS only needed 25 millisecond. The results demonstrate the real-time capacity of LINS is much better than that of LIOM.



(a) LINS's Trajectory overlaid with Google Map for visual comparison

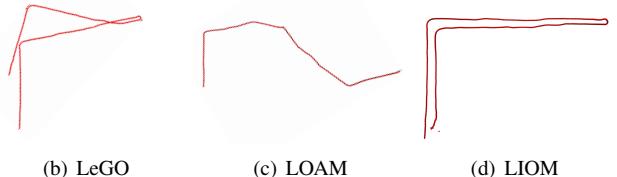


Fig. 7. MRO trajectories generated by different methods in the urban experiment, which are drawn in red lines.

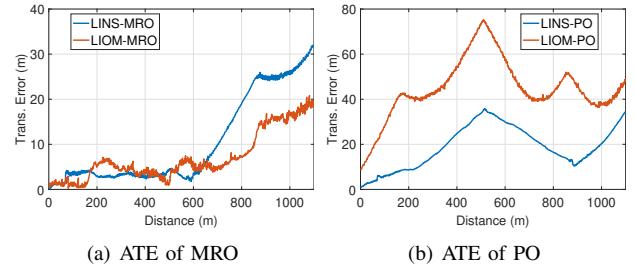


Fig. 8. Comparison of ATE bewteen LINS and LIOM w.r.t MRO and PO.

The most time-consuming parts in the LIO module of LIOM are local-map-constraint construction and batch optimization, in which it maintains a local map over multiple lidar scans and solves all relative states via MAP estimation. The main reason behind LINS's superior computational speed is that it uses Kalman filter rather than batch MAP, since Kalman filter implicitly reduces the dimension of the optimization problem by factoring the batch solution in time sequence and solving it in a recursive form [25]. The other reason is that we only use point clouds from the previous lidar scan for matching. In this way, although the used point cloud is sparser than the local map built in LIOM, we can still achieve accurate results with the aids of the IMU.

V. CONCLUSION

In this paper, we developed a lightweight lidar-inertial state estimator for robot navigation. Using an iterated ESKF with robocentric formulation, our algorithm is capable of providing real-time, long-term, robust, and high-precision ego-motion estimation under challenging environments. The proposed algorithm is verified in various scenarios including city, port, industrial park, forest, and indoor parking lot.

Experimental results demonstrate that LINS outperforms the lidar-only methods and reaches comparable performance with the state-of-the-art lidar-inertial odometry with a much lower computational cost.

REFERENCES

- [1] M. Velas, M. Spanel, and A. Herout, "Collar line segments for fast odometry estimation from velodyne point clouds," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4486–4495.
- [2] 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*. Springer, 2016, pp. 487–504.
- [3] S. Anderson and T. D. Barfoot, "Ransac for motion-distorted 3d visual sensors," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2093–2099.
- [4] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments," in *Robotics: Science and Systems*, 2018.
- [5] 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.
- [6] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3dim*, vol. 1, 2001, pp. 145–152.
- [7] F. C. Pomerleau, Franois and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.
- [8] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, vol. 2, 2014, p. 9.
- [9] T. Shan and B. Englöt, "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)*. IEEE, 2018, pp. 4758–4765.
- [10] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1271–1278.
- [11] J. Tang, Y. Chen, X. Niu, L. Wang, L. Chen, J. Liu, C. Shi, and J. Hyppä, "Lidar scan matching aided inertial navigation system in gnss-denied environments," *Sensors*, vol. 15, no. 7, pp. 16 710–16 728, 2015.
- [12] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 3923–3929.
- [13] 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)*. IEEE, 2017, pp. 6240–6245.
- [14] 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*. IEEE, 2013, pp. 4712–4719.
- [15] Z. Huai and G. Huang, "Robocentric visual-inertial odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6319–6326.
- [16] 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.
- [17] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial slam using nonlinear optimization," *Proceedings of Robotis Science and Systems (RSS) 2013*, 2013.
- [18] G. P. Huang, N. Trawny, A. I. Mourikis, and S. I. Roumeliotis, "Observability-based consistent ekf estimators for multi-robot cooperative localization," *Autonomous Robots*, vol. 30, no. 1, pp. 99–122, 2011.
- [19] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic lidar fusion: Dense map-centric continuous-time slam," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1206–1213.
- [20] 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)*. IEEE, 2018, pp. 123–130.
- [21] C. Forster, L. Carbone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," Georgia Institute of Technology, 2015.
- [22] 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)*. IEEE, 2010, pp. 5376–5382.
- [23] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A quadratic-complexity observability-constrained unscented kalman filter for slam," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1226–1243, 2013.
- [24] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," *IEEE Transactions on robotics*, vol. 23, no. 5, pp. 1036–1049, 2007.
- [25] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.
- [26] 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.
- [27] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.
- [28] J. Civera, O. G. Grasa, A. J. Davison, and J. Montiel, "1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, no. 5, pp. 609–631, 2010.
- [29] 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.
- [30] 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*, 2011, p. 6615.
- [31] M. Bloesch, H. Sommer, T. Laidlow, M. Burri, G. Nuetzi, P. Fankhauser, D. Bellicoso, C. Gehring, S. Leutenegger, M. Hutter, et al., "A primer on the differential calculus of 3d orientations," *arXiv preprint arXiv:1606.05285*, 2016.
- [32] E.-H. Shin, "Estimation techniques for low-cost inertial navigation," *UCGE report*, vol. 20219, 2005.
- [33] 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, no. 3.2. Kobe, Japan, 2009, p. 5.