

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

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

Abstract—We present R-LINS, a lightweight robocentric lidar-inertial state estimator, which estimates robot ego-motion using a 6-axis IMU and a 3D lidar in a tightly-coupled scheme. To achieve robustness and computational efficiency even in challenging environments, an iterated error-state Kalman filter (ESKF) is designed, which recursively corrects the state via repeatedly generating new corresponding feature pairs. Moreover, a novel robocentric formulation is adopted in which we reformulate the state estimator concerning a moving local frame, rather than a fixed global frame as in the standard world-centric lidar-inertial odometry(LIO), in order to prevent filter divergence and lower computational cost. To validate generalizability and long-time practicability, extensive experiments are performed in indoor and outdoor scenarios. The results indicate that R-LINS outperforms lidar-only and loosely-coupled algorithms, and achieve competitive performance as the state-of-the-art LIO with close to an order-of-magnitude improvement in terms of speed.

I. INTRODUCTION

High-precision, energy-efficient, and robust navigation is a fundamental prerequisite to enable the application of autonomous driving: slowness 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 comes into the well-known lidar odometry and mapping (LOAM) system [1]. The key advantages of a typical 3D lidar include (i) it has wide horizontal field of view (FOV) [2] and (ii) is invariant to ambient lighting conditions [3]. However, lidar-based navigation systems are sensitive to surroundings, and the motion distortion [4] and sparse nature of point clouds [5] compound this problem, leading to erroneous results in specific scenarios (e.g., a wide and open square). Additionally, its output frequency (usually 10 Hz) does not satisfy demands for the feedback controller (at least 50 Hz empirically).

A combination with another passive sensor - an inertial measurement unit (IMU) could be a good solution. An IMU provides accurate short-term motion constraints and, unlike lidar, is insensitive to surroundings. Moreover, IMU works with a high frequency (e.g., 100 Hz-500 Hz) that enables it to recover point clouds distorted by highly dynamic motion, while due to the sensor noise and bias, purely integrating IMU measurements may cause accumulated errors in ease. Fortunately, extensive research in visual-inertial odometry has demonstrated that tightly coupling an IMU with a aiding sensor, e.g., a camera, can effectively reduce the accumulated drifts and yield consistent motion estimates [6–8]. This idea

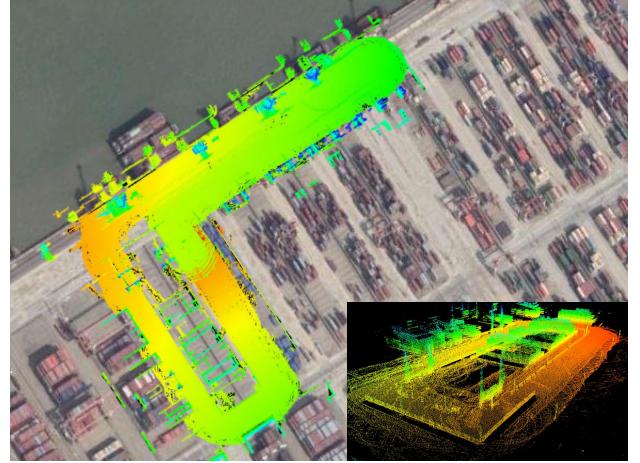


Fig. 1. Map built by our approach shows good superposition with Google Map.

can be elegantly generalized in the case of lidar-inertial odometry (LIO) [9–12].

However, the state-of-the-art method [10] is based on graph optimization with world-centric formulation and is too time-consuming in practical tests. For instance, it requires more than 100 ms in processing a single scan in the LIO algorithm, let alone the mapping module.

In this paper, we propose R-LINS, a lightweight lidar-inertial state estimator for unmanned ground vehicle (UGV) navigation. An iterated error-state Kalman filter (ESKF) is proposed to fuse measurements from an IMU and a 3D lidar, which benefits from the tightly-coupled scheme while keeping the algorithm computationally efficient. We introduce a robocentric formulation of the LIO 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, in order to increase the system robustness in a long run. In summary, our contributions are:

- A tightly-coupled lidar-inertial odometry algorithm with a robocentric formulation, which can realize robust and efficient UGV navigation, is proposed.
- Extensive experiments in different challenging scenarios demonstrate robustness, efficiency, high-precision, and generalizability of our algorithm. Note that with competitive accuracy, R-LINS reduces the running time of the state-of-the-art algorithm by order of magnitude.
- To the best of our knowledge, R-LINS is the first LIO that fuses a 3D lidar and an IMU in the iterated Kalman

Chao Qin, Haoyang Ye, Christian E. Pranata, Shuyang Zhang, and Ming Liu are with the RAM lab, the Hong Kong University of Science and Technology, Kowloon, Hong Kong

filtering framework for 6 DOF ego-motion estimation.

II. RELATED WORK

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

A. Lidar-only Methods

Most lidar-only approaches are variations of the well-known iterative closest point (ICP) scan matching method which is based on scan-to-scan registration. [14, 15] had surveyed efficient variants of ICP. For real-time application, [1] established LOAM which sequentially registers extracted edge and planar features to an incrementally built global map. [16] proposed LeGO which made further adaptations based on the original LOAM including the ground plane extraction and point cloud segmentation.

B. Loosely-coupled Lidar-IMU Fusion

There are different schemes available for combining the lidar and inertial measurements which can be broadly categorized into the loosely-coupled and the tightly-coupled. Loosely-coupled deal with two sensors separately to infer their own motion constraints which are fused later (e.g., [17–19]). IMU-aided LOAM [1] took the orientation and translation calculated by the IMU as priors to facilitate the convergence. Given a pre-built map, [19] combined the IMU measurements with pose estimates from a lidar-based Gaussian particle filter. In general, loosely-coupled fusion is computationally efficient and easy to implement [20], but the decoupling of lidar and inertial constraints results in information loss [21].

C. Tightly-coupled Lidar-IMU Fusion

Tightly-coupled approach directly fuses the lidar and inertial measurements through joint-optimization and achieves higher accuracy. It can be categorized into the optimization-based [22, 23] and the extended Kalman filter (EKF)-based [21, 24]. [12] performed local trajectory optimization via minimizing constraints from the IMU and lidar together. [11] presented LIPS which leveraged the graph optimization to fuse the inertial pre-integration measurements [25] and plane constraints from a lidar. And [10] established LIO-mapping (for brevity, termed LIOM in the followings), a tightly-coupled 3D LIO based on graph optimization as well. As the state-of-the-art algorithm, LIOM was verified with extensive indoor and outdoor tests and outperformed the state-of-the-art lidar-only or loosely coupled lidar-inertial algorithms. However, practical experiments showed that it is too time-consuming for real-time application. [26] introduced a lidar-aided inertial EKF based on a 2D lidar, but it was not suitable for outdoor navigation because it assumed all planes are in the orthogonal structure. We find few works that utilize EKF to combine a 3D lidar and an IMU in a tightly-coupled scheme. One possible reason could be that EKF is vulnerable to linearization errors, which can cause poor performance or even divergence [27, 28]. This

consequence is even worse when it comes to ICP-based lidar-observed constraint, which is deemed highly nonlinear.

To reduce the estimation errors due to the EKFs linearization, iterated Kalman filtering [29, 30] is proposed, which is the focus of this work.

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 the 6 DOF ego-motion and establish a global map simultaneously (as shown in Fig. 1). An overview of the proposed framework is shown in Fig. 2. The overall system consists of three chief 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 sub-modules, carries out iterated Kalman filtering and outputs a so-called pure odometry (PO) result¹, along with undistorted features. (iii) The mapping module refines the PO result by a global map and outputs a map-refined odometry (MRO) result, followed by updating the global map by new undistorted features. In the rest of this section, we will introduce them in order.

B. Feature Extraction

Our feature extraction algorithm is referred to [16]. Due to the space issue, readers can see [1, 16] for the detailed procedures. All in all, 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 .

C. Lidar-Inertial Odometry with Iterated ESKF

The goal of LIO is estimating a coarse pose of the UGV using IMU measurements and features extracted in Sect. III-B.

In contrast to the standard world-centric formulation using a fixed world frame, w , in the proposed robocentric formulation, the IMU-affixed frame, b , is set to be the local frame of reference for navigation. As illustrated in [13, 31], the key advantages of robocentric formulation include: (i) preventing large linearization errors caused by ever-growing uncertainty, and (ii) reducing computational cost by avoiding coordinate transformation of the point cloud. The iterated ESKF framework is detailed as follows.

1) *State Definitions:* Let $\mathbf{x}_w^{b_k}$ denote the location of frame w with respect to the IMU-affixed frame at k lidar time-step, b_k , and let $\mathbf{x}_{b_{k+1}}^{b_k}$ denote the local state between two IMU-affixed frames corresponding to two consecutive lidar time-steps:

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

$$\mathbf{x}_{b_{k+1}}^{b_k} := [\mathbf{p}_{b_{k+1}}^{b_k}, \mathbf{v}_{b_{k+1}}^{b_k}, \mathbf{q}_{b_{k+1}}^{b_k}, \mathbf{b}_{a_{k+1}}, \mathbf{b}_{g_{k+1}}], \quad (2)$$

where $\mathbf{p}_w^{b_k}$ is the position of frame w with respect to frame b_k and $\mathbf{q}_w^{b_k}$ is the unit quaternion describing the rotation from frame w to frame b_k . $\mathbf{p}_{b_{k+1}}^{b_k}$ and $\mathbf{q}_{b_{k+1}}^{b_k}$ represent the translation

¹The odometry result includes a 6 DOF pose.

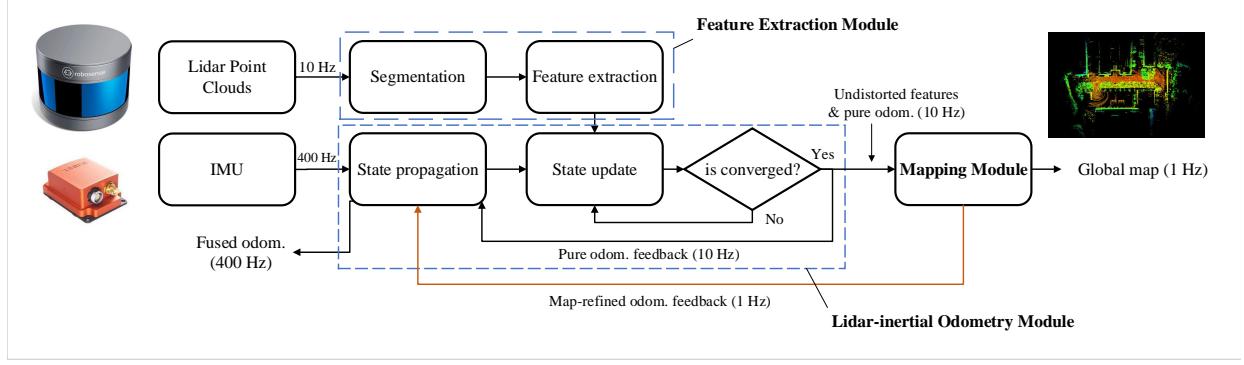


Fig. 2. Pipeline of the R-LINS system. The LIO, which consists of state propagation and update R-LINSs, performs iterated Kalman filtering using IMU measurements and point cloud features extracted from the feature extraction module. The mapping module receives a coarse pose estimate and undistorted features from the LIO and output a refined pose estimate along with a global map, while its roll and pitch estimates will be used to correct the global pose in the LIO.

and rotation from frame b_{k+1} to frame b_k , respectively. $\mathbf{v}_{b_{k+1}}^{b_k}$ is the robocentric velocity with respect to frame b_k , while $\mathbf{b}_{a_{k+1}}$ is the acceleration bias and $\mathbf{b}_{g_{k+1}}$ the gyroscope bias.

An *error-state* representation of states is introduced for its good properties [32]. We denote an error with δ and define $\delta\mathbf{x}$ the error vector of $\mathbf{x}_{b_{k+1}}^{b_k}$ as

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

where $\delta\theta$ is the 3 DOF error angle. According to ESKF traditions, after estimating the observed errors, $\mathbf{x}_{b_{k+1}}^{b_k}$, we will inject it into the state prior, $-\mathbf{x}_{b_{k+1}}^{b_k}$, through a boxplus operator \boxplus defined in Appendix A,

$$\mathbf{x}_{b_{k+1}}^{b_k} = -\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta\mathbf{x}, \quad (4)$$

to yield the final estimation result.

Note that, different from [21] which formulates the local gravity in the state vector, we acquire it via transforming the gravity from the world frame to the local frame of reference using roll and pitch estimated in the mapping module. This is reasonable because the roll and pitch angles of a car running on the road will not sharply change within 1 second. And it is also advantageous since roll and pitch results are almost drift-free in the mapping module after loop closures [16].

2) *Propagation*: The state propagation submodule is in charge of propagating the error state, $\delta\mathbf{x}$, with its covariance matrix, \mathbf{P}_k , and calculating the state prior, $-\mathbf{x}_{b_{k+1}}^{b_k}$, using IMU measurements.

The linearized continuous-time model [34] for the IMU error state is:

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

where $\mathbf{w} = [\mathbf{n}_a^T, \mathbf{n}_g^T, \mathbf{n}_{b_a}^T, \mathbf{n}_{b_g}^T]^T$ is the system noise vector (detailed in Appendix B). \mathbf{F}_t is the error-state transition matrix and \mathbf{G}_t is the noise Jacobian at time t :

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

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

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 frame b_k . $\hat{\mathbf{a}}_t$ and $\hat{\omega}_t$ are acceleration and angular rate estimates at time t in the IMU-affixed frame, respectively (detailed in Appendix B). Discretizing Eqs. (5) yields following propagation equations:

$$\delta\mathbf{x}_{t_\tau} = (\mathbf{I} + \mathbf{F}_{t_\tau} \Delta t) \delta\mathbf{x}_{t_{\tau-1}}, \quad (8)$$

$$\mathbf{P}_{t_\tau} = (\mathbf{I} + \mathbf{F}_{t_\tau} \Delta t) \mathbf{P}_{t_{\tau-1}} (\mathbf{I} + \mathbf{F}_{t_\tau} \Delta t)^T + (\mathbf{G}_{t_\tau} \Delta t) \mathbf{Q} (\mathbf{G}_{t_\tau} \Delta t)^T, \quad (9)$$

where $\Delta t = t_\tau - t_{\tau-1}$ and $t_\tau, t_{\tau-1}$ is the IMU time-step. \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 [21, 33] for details in integrating IMU measurements.

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

In iterated Kalman filtering, the state update can be linked to an optimization problem [15, 31] considering the deviation from the state prior $-\mathbf{x}_{b_{k+1}}^{b_k}$ and the residual functions from the measurement model, which can be written as

$$\min_{\delta\mathbf{x}} \|\delta\mathbf{x}\|_{(\mathbf{P}_k)^{-1}} + \|f(-\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta\mathbf{x})\|_{(\mathbf{J}_k \mathbf{M}_k \mathbf{J}_k^T)^{-1}}, \quad (10)$$

where $\|\cdot\|$ denotes the Mahalanobis norm. \mathbf{J}_k is the Jacobian of $f(-\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta\mathbf{x})$ w.r.t. the measurement noise and \mathbf{M}_k is the covariance matrix of the measurement noise. $f(\cdot)$ is the ICP-based residual function, which outputs the stacked error terms arose from point cloud matching. Let l_k represent the lidar frame of reference at k time-step. Then given a current

estimates $\mathbf{x}_{b_{k+1}}^{b_k}$, the error term of a feature point, $\mathbf{p}_i^{l_{k+1}}$ can be described as:

$$f_i(\mathbf{x}_{b_{k+1}}^{b_k}) = \begin{cases} \frac{|(\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_a^{l_k}) \times (\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_b^{l_k})|}{|\mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k}|} & \text{if } \mathbf{p}_i^{l_{k+1}} \in \mathbb{F}_e \\ \frac{|(\hat{\mathbf{p}}_i^{l_k} - \mathbf{p}_a^{l_k})^T((\mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k}) \times (\mathbf{p}_a^{l_k} - \mathbf{p}_c^{l_k}))|}{|(\mathbf{p}_a^{l_k} - \mathbf{p}_b^{l_k}) \times (\mathbf{p}_a^{l_k} - \mathbf{p}_c^{l_k})|} & \text{if } \mathbf{p}_i^{l_{k+1}} \in \mathbb{F}_p \end{cases} \quad (11)$$

$$\hat{\mathbf{p}}_i^{l_k} = \mathbf{R}_l^{b^T} (\mathbf{R}_{b_{k+1}}^{b_k} (\mathbf{R}_l \mathbf{p}_i^{l_{k+1}} + \mathbf{p}_l^b) + \mathbf{p}_{b_{k+1}}^{b_k} - \mathbf{p}_l^b), \quad (12)$$

where $f_i(\cdot)$ is the corresponding residual to $\mathbf{p}_i^{l_{k+1}}$ in the output of $f(\cdot)$. $\hat{\mathbf{p}}_i^{l_k}$ is the transformed point of $\mathbf{p}_i^{l_{k+1}}$ from frame l_{k+1} to the previous frame l_k . \mathbf{R}_l^b and \mathbf{p}_l^b together denote the extrinsic parameters between the lidar and IMU.

An explanation of the physical meanings of $f_i(\mathbf{x}_{b_{k+1}}^{b_k})$ is provided here. For an edge point, it describes the distance between $\hat{\mathbf{p}}_i^{l_k}$ and its matching edge $\mathbf{p}_a^{l_k} \mathbf{p}_b^{l_k}$. If it is a planar point, it describes the distance between $\hat{\mathbf{p}}_i^{l_k}$ and its matching plane which is formed by three points, $\mathbf{p}_a^{l_k}$, $\mathbf{p}_b^{l_k}$, and $\mathbf{p}_c^{l_k}$. Details in how to find their matchings can be found in [1].

We solve Eqs. (10) using following iterated update equations

$$\mathbf{K}_{k,j} = \mathbf{P}_k \mathbf{H}_{k,j}^T (\mathbf{H}_{k,j} \mathbf{P}_k \mathbf{H}_{k,j}^T + \mathbf{J}_{k,j} \mathbf{M}_k \mathbf{J}_{k,j}^T)^{-1}, \quad (13)$$

$$\Delta \mathbf{x}_j = \mathbf{K}_{k,j} (\mathbf{H}_{k,j} \delta \mathbf{x}_j - f(-\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta \mathbf{x}_j)), \quad (14)$$

$$\delta \mathbf{x}_{j+1} = \delta \mathbf{x}_j + \Delta \mathbf{x}_j, \quad (15)$$

where $\Delta \mathbf{x}_j$ is the correction vector at j iteration and $\mathbf{H}_{k,j}$ is the jacobian of $f(-\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta \mathbf{x}_j)$ with respect to $\delta \mathbf{x}_j$. Note that in every iteration, we find new matching edges and planes to further minimize the error metric and then calculate new $\mathbf{H}_{k,j}$, $\mathbf{J}_{k,j}$, and $\mathbf{K}_{k,j}$.

When the iteration is terminated, say at iteration n , \mathbf{P}_k will be updated as

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k,n} \mathbf{H}_{k,n}) \mathbf{P}_k (\mathbf{I} - \mathbf{K}_{k,n} \mathbf{H}_{k,n})^T + \mathbf{K}_{k,n} \mathbf{M}_k \mathbf{K}_{k,n}^T. \quad (16)$$

Now we can obtain an accurate $\mathbf{x}_{b_{k+1}}^{b_k}$ using Eqs. (4) and raw features will be undistorted using the relative transformation.

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

$$[\mathbf{0}_3, \mathbf{v}_{b_{k+1}}^{b_{k+1}}, \mathbf{q}_0, \mathbf{b}_{a_{k+1}}, \mathbf{b}_{g_{k+1}}], \quad (17)$$

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

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

$$\mathbf{x}_w^{b_{k+1}} = \begin{bmatrix} \mathbf{p}_w^{b_{k+1}} \\ \mathbf{q}_w^{b_{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{b_k}^{b_{k+1}} (\mathbf{p}_w^{b_k} - \mathbf{p}_{b_{k+1}}^{b_k}) \\ \mathbf{q}_{b_k}^{b_{k+1}} \otimes \mathbf{q}_w^{b_k} \end{bmatrix}, \quad (18)$$

where \otimes denotes the quaternion product.

5) Initialization: The proposed robocentric formulation facilitates initialization of the filter state as described in Sect. III-C. 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, and (ii) the initial roll and pitch for the global pose are calculated from the unbiased acceleration measurements before moving.

D. Lidar Mapping

We apply mapping algorithms proposed in [16]. Due to the space issue, we refer the reader to the description from [1, 16] for the detailed matching and optimization procedure.

The difference between [16, 21] and our work is that we use the resulting roll and pitch in this module as a feedback to the LIO module, which proved effective to increase the system robustness.

IV. EXPERIMENTS

We now describe a series of experiments to qualitatively and quantitatively analyze the performance of R-LINS and compare it with other methods, such as LeGO (a lidar-only algorithm), LOAM (a loosely-coupled algorithm), and LIOM (a tightly-coupled algorithm), 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) [35] in Ubuntu Linux.

Different from previous works which only consider the performance of map-refined odometry (MRO), we also attach importance to the pure odometry (PO) performance as well, because we find that the robot has to rely on PO in some dynamic scenes (e.g., a port) where maps are always changing. Moreover, many experiments show that PO has great impact to the MRO performance.

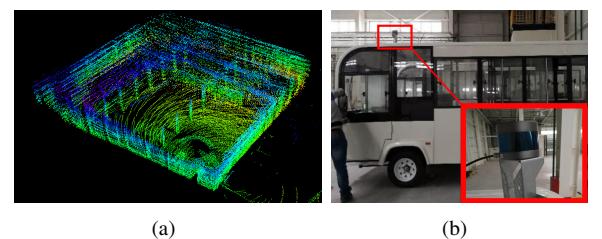


Fig. 3. The sensor configuration for indoor tests. (a) Map of an indoor parking lot built by R-LINS. (b) Bus, lidar, and IMU installation. IMU is placed inside the bus.

A. Indoor Experiment

In indoor tests, a parking lot is chosen as the experiment area as shown in Fig. 3(a). We installed our sensor suite on a bus as shown in Fig. 3(b), where a RS-LiDAR-16 was mounted on the top and an IMU was placed inside the bus.

Fig. 4(a) is the output from LeGO and Fig. 4(b) the output from R-LINS. Because MROs from both methods are very accurate in indoor environment, we use them as baselines to

TABLE I
RELATIVE ERRORS FOR MOTION ESTIMATION DRIFT (%)

Scenario	Dist. (m)	Num. of Features		Map-Refined Odometry (MRO)				Pure Odometry (PO)			
		Edge	Planar	LOAM [1]	LeGO [16]	LIOM [10]	R-LINS	LOAM	LeGO	LIOM	R-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 number denotes the smallest value in this block.

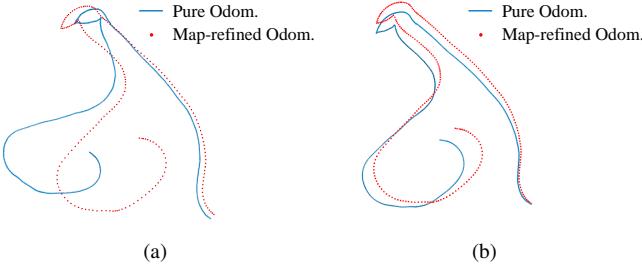


Fig. 4. Results of the indoor experiment of (a) LeGO and (b) R-LINS. Blue lines are the PO trajectories while red dots denote the MRO results.

compare with POs. For the LeGO-PO trajectory (blue line), we see noticeable drifts occurred in the yaw estimate. By contrast, the trajectory of R-LINS-PO is very close to its MRO, showing that fusing IMU can correct the yaw angle effectively, leading to lower drifts.

B. Large-scale Environment

To verify long-time practicability and generalizability, experiments in five typical outdoor scenarios are carried out. Fig. 5 showcases some photos of the real scenes and maps generated by R-LINS. Following [1], we measured the relative drift which are compared to the distance traveled, and listed the results in Table I.

In summary, R-LINS performs well in all tested scenarios. The detailed analysis for some experiments is given below.

1) *Port Experiment*: We evaluate R-LINS in a port in Guangdong, China. The sensor suite consists of a Velodyne VLP-16 lidar and an Xsens MTi-G-710 IMU fixed on the top of a car. The ground truths are obtained from an accurate GPS module.

We started recording data from a path surrounded by containers. The car head to a dock and then returned to the original spot after traveling a distance of 1264 meters. By the way, we observed that the containers went in and out all the times, changing the global map incessantly, which may make MRO unreliable in a long run.

According to Table I, tightly-coupled algorithms, R-LINS and LIOM, present the lowest drifts. The relative drift of R-LINS-MRO is only 1.25%, slightly lower than that of LIOM

TABLE II
RUNTIME OF THE LIO FOR PROCESSING ONE SCAN

Method	Urban	Port	Park	Forest	Parking Lot
LIOM	143.12 ms	185.96 ms	201.04 ms	173.64 ms	223.77 ms
R-LINS	18.74 ms	19.13 ms	21.26 ms	20.54 ms	25.39 ms

which is 1.28%, while the relative drift of R-LINS-PO is 2.12% and very close to the MRO result. LOAM-PO is less accurate than the tightly-coupled approaches, but its drift is still much lower than LeGO-PO, showing that fusing inertial information can effectively reduce the PO drift.

To present improvement by R-LINS visually, we display the trajectories of LeGO and R-LINS in Fig. 6(a) and 6(b), respectively. Compared to the ground truths, trajectories of LeGO-MRO and LeGO-PO are severely tilted around the first turn. One possible reason is lacking features, because this is the place where the car would face directly to the ocean and few edge features were extracted (e.g., only about 30 edge features available in one scan). In contrast, R-LINS's trajectories are aligned well with the ground-truth trajectory (green triangle). Furthermore, we can visually inspect that the map built by R-LINS, as shown in Fig. 6(d), has higher fidelity than that of LeGO, as shown in Fig. 6(c).

2) *Urban Experiment*: To evaluate robustness in the feature-less scenes, a 1100-meter-long urban dataset was collected with the same sensor suite in the indoor experiment. Notably, the average number of the edge features is only 56, the lowest among all tested scenarios. For space issue, only the MRO results are analysed here.

We first take a look at the output of R-LINS-MRO as shown in Fig. 7. It can be seen that its trajectory shows good superposition to the real road in the Google Map. At the same time, LIOM-MRO offers good motion tracking result as well, as shown in Fig. 8(a), but in the beginning of the trajectory, small errors occur due to inferior initialization, which is mainly caused by lacking stable feature points. The quantitative comparison of the translation error of R-LINS and LIOM is provided in Figs. 9(a) and 9(b). The results indicate that R-LINS can achieve competitive accuracy as LIOM.

Fig. 8(b) and 8(c) showcase the results from LeGO-MRO and LOAM-MRO, respectively. We see that trajectories ob-

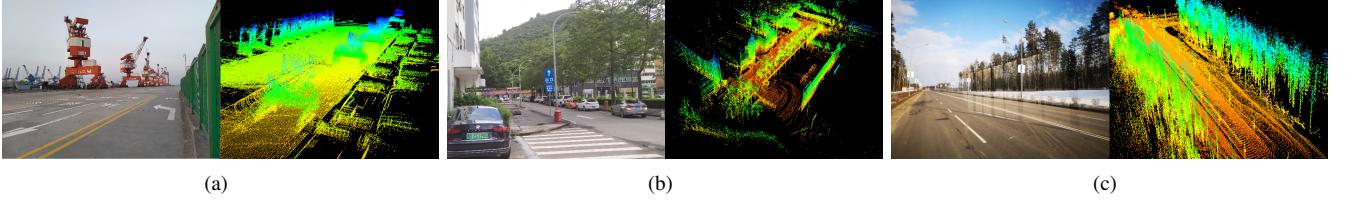


Fig. 5. Photos and maps of (a) a wide and open dock close to the ocean, (c) a industrial park with numerous trees and cars, and (d) a clean road through a forest. All 3D maps on the right side are produced by R-LINS.

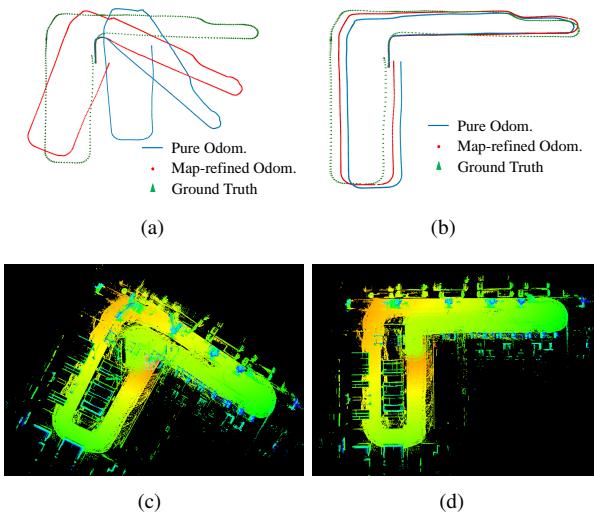


Fig. 6. Estimated trajectories of (a) LeGO and (b) R-LINS, and maps built by (c) LeGO and (d) R-LINS. Note that blue lines are the PO trajectories, red circles the MRO trajectories, and green triangles the GPS ground truths



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

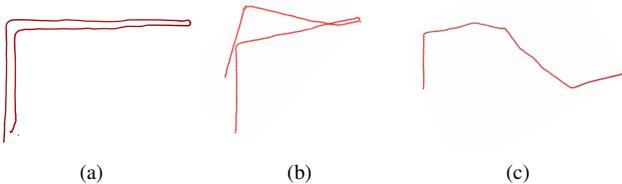


Fig. 8. Estimated trajectories of (a) LIOM-MRO, (b) LeGO-MRO, and LOAM-MRO in the urban experiment.

tained from these two methods are both twisted in different levels, which shows that the proposed LINS outperforms LeGO and LOAM in terms of robustness and accuracy, especially in the feature-less scene.

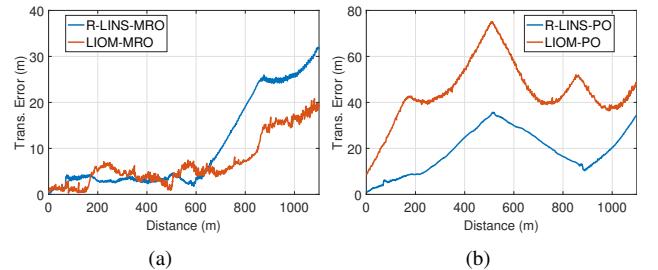


Fig. 9. Translation errors of R-LINS and LIOM w.r.t (a) PO and (b) MRO

C. Runtime Comparison

Table II offers the mean runtimes w.r.t the LIO module in R-LINS² and LIOM. We see that R-LINS is much faster than LIOM: in general, R-LINS take less than 30 ms to process a scan, while LIOM takes more than 100 ms, and in the park environment where features abound, R-LINS only requires a tenth of the time that LIOM needs.

There are multiple reasons to explain why R-LINS is faster than LIOM: (i) R-LINS only considers one state at a time, while LIOM estimates several states in the sliding window using bundle adjustment, and (ii) the local frame of R-LINS is the previous lidar frame, so the point cloud is naturally represented in the local frame and the time-consuming coordinate transformation of point clouds can be avoided.

V. CONCLUSION

In this paper, we developed a lightweight, high-precision, robocentric lidar-inertial state estimator, termed R-LINS. Using an iterated ESKF with robocentric formulation, our algorithm is capable of providing long-term, robust, and real-time navigation for UGVs even in challenging environments. Extensive experiments show that R-LINS outperforms the lidar-only and loosely-coupled methods, and reaches competitive performance as the state-of-the-art tightly-coupled algorithm with lower computational cost. In future research, we will integrate the inertial information further, e.g. aiding the ground plane extraction module.

²Note that the time cost of state propagation is also added into the result.

VI. APPENDIX A STATE INJECTION

We can inject an observed error into the nominal state by a boxplus operator \boxplus , which is expressed as

$$\mathbf{x}_{b_{k+1}}^{b_k} \boxplus \delta \mathbf{x} = \begin{bmatrix} \mathbf{p}_{b_{k+1}}^{b_k} + \delta \mathbf{p} \\ \mathbf{v}_{b_{k+1}}^{b_k} + \delta \mathbf{v} \\ \mathbf{q}_{b_{k+1}}^{b_k} \otimes \exp(\delta \theta / 2) \\ \mathbf{b}_{a_{k+1}} + \delta \mathbf{b}_a \\ \mathbf{b}_{g_{k+1}} + \delta \mathbf{b}_g \end{bmatrix}, \quad (19)$$

where \otimes denotes the quaternion product and $\exp(\cdot)$ maps the angle vector to its quaternion representation [33].

VII. APPENDIX B IMU MEASUREMENT MODEL

Let \mathbf{a}_{m_t} and ω_{m_t} the accelerometer and gyroscope measurements at time t , respectively. To obtain $\hat{\mathbf{a}}_t$ and $\hat{\omega}_t$, we need to remove the biases and the gravity effect by:

$$\hat{\mathbf{a}}_t = \mathbf{a}_{m_t} - \mathbf{b}_{a_k} - \mathbf{R}_n^{b_k} \mathbf{g}^n, \quad (20)$$

$$\hat{\omega}_t = \omega_{m_t} - \mathbf{b}_{\omega_k} \quad (21)$$

where \mathbf{g}^n expresses the gravity vector in the local navigation frame, such as the north-east-down (NED) frame [34]. $\mathbf{R}_n^{b_k}$ is the rotation matrix in terms of the roll and pitch estimates from the mapping module (we assume the map frame is well-aligned with the local navigation frame). \mathbf{n}_g are additive noises in \mathbf{a}_{m_t} and ω_{m_t} , respectively, which are assumed as Gaussian, $\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \sigma_a^2)$, $\mathbf{n}_g \sim \mathcal{N}(\mathbf{0}, \sigma_g^2)$. We model the \mathbf{b}_{a_t} and \mathbf{b}_{ω_t} as random walk as

$$\dot{\mathbf{b}}_{a_t} = \mathbf{n}_{b_a}, \dot{\mathbf{b}}_{g_t} = \mathbf{n}_{b_g}, \quad (22)$$

where we have $\mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \sigma_{b_a}^2)$ and $\mathbf{n}_{b_g} \sim \mathcal{N}(\mathbf{0}, \sigma_{b_g}^2)$.

REFERENCES

- [1] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.", in *Robotics: Science and Systems*, vol. 2, p. 9, 2014.
- [2] 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)*, pp. 4486–4495, IEEE, 2016.
- [3] 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.
- [4] S. Anderson and T. D. Barfoot, "Ransac for motion-distorted 3d visual sensors," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2093–2099, IEEE, 2013.
- [5] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments.", in *Robotics: Science and Systems*, 2018.
- [6] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3565–3572, IEEE, 2007.
- [7] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2510–2517, IEEE, 2018.
- [8] G. Huang, M. Kaess, and J. J. Leonard, "Towards consistent visual-inertial navigation," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4926–4933, IEEE, 2014.
- [9] A. Soloviev, D. Bates, and F. Van Graas, "Tight coupling of laser scanner and inertial measurements for a fully autonomous relative navigation solution," *Navigation*, vol. 54, no. 3, pp. 189–205, 2007.
- [10] 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.
- [11] 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.
- [12] 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)*, pp. 1206–1213, IEEE, 2018.
- [13] 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.
- [14] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm.", in *3dim*, vol. 1, pp. 145–152, 2001.
- [15] 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.
- [16] 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.
- [17] 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. 16710–16728, 2015.
- [18] 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*, pp. 3923–3929, IEEE, 2013.
- [19] 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.

- [20] 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.
- [21] Z. Huai and G. Huang, “Robocentric visual-inertial odometry,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6319–6326, IEEE, 2018.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] 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.
- [29] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.
- [30] 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.
- [31] 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.
- [32] 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.
- [33] J. Sola, “Quaternion kinematics for the error-state kalman filter,” *arXiv preprint arXiv:1711.02508*, 2017.
- [34] E.-H. Shin, “Estimation techniques for low-cost inertial navigation,” *UCGE report*, vol. 20219, 2005.
- [35] 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.