R²LIVE: A Robust, Real-time, LiDAR-Inertial-Visual tightly-coupled state Estimator and mapping

Jiarong Lin, Chunran Zheng, Wei Xu, and Fu Zhang

Abstract—In this letter, we propose a robust, real-time tightly-coupled multi-sensor fusion framework, which fuses measurement from LiDAR, inertial sensor, and visual camera to achieve robust and accurate state estimation. Our proposed framework is composed of two parts: the filter-based odometry and factor graph optimization. To guarantee real-time performance, we estimate the state within the framework of error-state iterated Kalman-filter, and further improve the overall precision with our factor graph optimization. Taking advantage of measurement from all individual sensors, our algorithm is robust enough to various visual failure, LiDAR-degenerated scenarios, and is able to run in real-time on an on-board computation platform, as shown by extensive experiments conducted in indoor, outdoor, and mixed environment of different scale. Moreover, the results show that our proposed framework can improve the accuracy of state-of-the-art LiDAR-Inertial or visual-inertial odometry. To share our findings and to make contributions to the community, we open source our codes on our Github.

I. INTRODUCTION

With the capacity of estimating ego-motion in six degrees of freedom (DOF) and simultaneously building dense and high precision maps of surrounding environments, LiDAR-based SLAM has been widely applied in the field of autonomous driving vehicles [1], drones [2, 3], and etc. With the development of LiDAR technologies, the emergence of low-cost LiDARs (e.g., Livox LiDAR [4]) makes LiDAR more accessible. Following this trend, a number of related works [5]–[8] draw the attention of the community to this field of research. However, LiDAR-based SLAM methods easily fail (i.e., degenerate) in those scenarios with few available geometry features, which is more critical for those LiDARs with small FoV [9]. In this work, to address the degeneration problems of LiDAR-based odometry, we propose a LiDAR-inertial-visual fusion framework to obtain the state estimation of higher robustness and accuracy. The main contributions of our work are:

- We take advantage of measurements from LiDAR, inertial, and camera sensors and fuse them in a tight-coupled way. Experiments show that our method is robust enough in various challenging scenarios with aggressive motion, sensor failure, and even in narrow tunnel-like environments with a large number of moving objects and small LiDAR field of view.
- We propose a framework with a high-rate filter-based odometry and a low-rarte factor graph optimization. The filter-based odometry fuses the measurements of LiDAR, inertial, and camera sensors within an error-state iterated Kalman filter to achieve real-time performance. The factor graph optimization refines a local map of keyframe poses and visual landmark positions.
- By tightly fusing different types of sensors, we achieve high-accuracy state estimation. Experiment results show that our system is accurate enough to be used to reconstruct large-scale, indoor-outdoor dense 3D maps of building structures (see Fig. 1).
- Our system is carefully engineered and open sourced to benefit the whole robotics community.

II. RELATED WORK

In this section, we review existing works closely related to our work, including LiDAR-only odometry and mapping, LiDAR-Inertial fusion and LiDAR-Inertial-Visual methods.

A. LiDAR Odometry and Mapping

Zhang et al [10] first proposed a LiDAR odometry and mapping framework, LOAM, that combines ICP method [11] with point-to-plane and point-to-edge distance. It achieves good odometry and mapping performance by running the two modules at different rates. To make the algorithm run in real time at a computation limited platform, Shan et al [12] propose a lightweight and ground-optimized LOAM (LeGO-LOAM), which discards unreliable features in the step of ground plane segmentation. These works are mainly based on multi-line spinning LiDARs. Our previous work [9, 13] develop an accurate and robust algorithm by considering the low-level physical properties of solid-state LiDARs with small FOV. However, these methods solely based on LiDAR measurements and are very vulnerable to featureless environments or other degenerated scenarios.
III. THE OVERVIEW OF OUR SYSTEM

The overview of our system is shown in Fig. 2, where the filter-based odometry taking advantage of the measurements from LiDAR, camera and inertial sensor, estimates the state within the framework of error-state iterated Kalman filter as detailed in Section IV. To further improve the visual measurements, we leverage the factor graph optimization to refine the visual landmarks within a local sliding window as detailed in Section V.

IV. FILTER-BASED ODOMETRY

A. The boxplus “⊕” and boxminus “⊖” operator

In this paper, we make use of the “⊕” and “⊖” operations encapsulated on a manifold \( \mathcal{M} \) to simplify the notations and derivations. Let \( \mathcal{M} \) be the manifold on which the system state lies. Since a manifold is locally homeomorphic to \( \mathbb{R}^n \), where \( n \) is the dimension of \( \mathcal{M} \), we can use two operators, “⊕” and “⊖”, establishing a bijective map between the local neighborhood of \( \mathcal{M} \) and its tangent space \( \mathbb{R}^n \) [23]:

\[
\oplus: \mathcal{M} \times \mathbb{R}^n \to \mathcal{M}, \quad \ominus: \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n
\]  

(1)

For the compound manifold \( \mathcal{M} = SO(3) \times \mathbb{R}^n \), we have:

\[
\begin{align*}
\mathbb{R}^3 & \oplus \mathbb{R}^{a_1} \oplus \mathbb{R}^{a_2} = \begin{bmatrix} \mathbb{R} \cdot \exp(\mathbb{r}) \end{bmatrix}, \\
\mathbb{R}^{a_1} & \ominus \mathbb{R}^{a_2} = \begin{bmatrix} \log(\mathbb{R}_{a_2}^{-1} \mathbb{R}_{a_1}) \end{bmatrix}
\end{align*}
\]

where \( \mathbb{r} \in \mathbb{R}^3 \), \( a_1, a_2 \in \mathbb{R}^n \), \( \exp(\cdot) \) and \( \log(\cdot) \) denote the Rodrigues’ transformation between the rotation matrix and rotation vector\(^2\).

B. Continuous-time kinematic model

In our current work, we assume that the time offset among all the three sensors, LiDAR, camera, and IMU, are pre-calibrated. Furthermore, we assume the extrinsic between LiDAR and IMU are known as they are usually integrated and calibrated in factory, but estimate the camera IMU extrinsic online. Moreover, a LiDAR typically scans points sequentially, and points in a LiDAR frame could be measured at different body pose. This motion is compensated by IMU back propagation as shown in [5], hence points in a LiDAR frame are assumed to be measured at the same time. With these, the input data sequences of our system can be simplified into Fig. 3.

\(^2\)https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula
Camera \(\downarrow\) IMU \(k-2\) \(\downarrow\) IMU \(k\) \(\downarrow\) IMU \(k+1\) \(\downarrow\) IMU \(k+3\) \(\downarrow\)

**LIDAR** \(k-2\) \(i\) \(+1\) \(k\) \(k+2\)

Fig. 3: Illustration of the input data sequences, where the frame rate of IMU, camera, and LiDAR is 200 Hz, 20 Hz and 10 Hz, respectively. The notation \(i\) denotes the index of IMU data while \(k\) denotes the index of LiDAR or camera measurements.

We assume that the IMU, LiDAR, and camera sensors are rigidly attached together with the extrinsic LiDAR and IMU (LiDAR frame w.r.t. IMU frame) as \(I^T C = (\hat{R}_C, \hat{p}_C)\), and the extrinsic between camera and IMU (camera frame w.r.t. IMU frame) is \(I^T C = (\hat{R}_C, \hat{p}_C)\). For the sake of convenience, we take IMU as the body frame, which leads to the following continuous kinematic model:

\[
\begin{align*}
G \hat{p}_I &= G \dot{v}_I, \quad G \dot{v}_I = G \dot{R}_I (a_m - b_n - n_n) + G g, \\
G \hat{R}_I &= G R_I [\omega_m - b_g - n_g], \quad b_g = b_{bg} \quad b = b_{ba}
\end{align*}
\]

where \(G(\cdot)\) denotes a vector represented in the global frame (i.e. the first gravitational aligned IMU frame [22]), \(G R_I\) and \(G p_I\) are the attitude and position of the IMU relative to the global frame, \(G g\) is the gravitational acceleration, \(\omega_m\) and \(a_m\) are the raw gyroscopes and accelerometer readings, \(n_n\) and \(n_g\) are the white noise of IMU measurement, \(b_{bg}\) and \(b_{ba}\) are the bias of gyroscopes and accelerometer, which are modelled as random walk driven by Gaussian noise \(n_{bg}\) and \(n_{ba}\).

### C. Discrete IMU model

We discretize the continuous model (2) at the IMU rate. Let \(x_i\) be the state vector at the \(i\)-th IMU measurement:

\[
x_i = \begin{bmatrix} G \hat{R}_I, G \hat{p}_I, I \delta \hat{R}_C, I \delta \hat{P}_C, G \hat{v}_I, b_{bg}, b_{ba} \end{bmatrix}^T
\]

Discretizing (2) by zero-order holder (i.e., IMU measurements over one sampling time period \(\Delta t\) are constant), we obtain

\[
x_{i+1} = x_i \oplus (\Delta t \cdot f(x_i, u_i, w_i))
\]

where

\[
\begin{align*}
&u_i = \begin{bmatrix} \omega_{mI}, a_{mI} \end{bmatrix}^T, \quad w_i = \begin{bmatrix} n_{gI}, n_{nI}, n_{bgi}, n_{ba} \end{bmatrix}^T, \\
f(x_i, u_i, w_i) = &\begin{bmatrix} \omega_m - b_{mg} - n_{mg} \\
0_{3 \times 1} \\
0_{3 \times 1} \\
G R_I (a_m - b_n - n_n) - G g \\
b_g \\
b_a
\end{bmatrix}
\end{align*}
\]

### D. Propagation

In our work, we leverage an on-manifold iterated error state Kalman filter [24] to estimate the state vector \(x_i\), in which the state estimation error \(\delta x_i\) is characterized in the tangent space of the state estimate \(\hat{x}_i\):

\[
\delta x_i \doteq x_i \oplus \hat{x}_i
\]

\[
= \begin{bmatrix} G \delta \hat{R}_I, G \delta \hat{p}_I, I \delta \hat{R}_C, I \delta \hat{P}_C, G \delta \hat{v}_I, \delta b_{bg}, \delta b_{ba} \end{bmatrix}^T
\]

\[
\sim \mathcal{N}(0_{21 \times 1}, \Sigma_{\delta x_i})
\]

\[
(4)
\]

### E. The prior distribution

Let the two propagation in (5) and (7) stop at the \((k+1)\)-th LiDAR/camera measurement (see Fig. 4), and the propagated state estimate and covariance are \(\hat{x}_{k+1}\) and \(\Sigma_{\delta x_{k+1}}\), respectively. They essentially impose a prior distribution for the state \(x_{k+1}\) before fusing the \((k+1)\)-th measurement as below:

\[
x_{k+1} \oplus \hat{x}_{k+1} \sim \mathcal{N}(0, \Sigma_{\delta x_{k+1}}).
\]

\[
(8)
\]

### F. Initialization of iterated update

The prior distribution in (8) will be fused with the LiDAR or camera measurements to produce a maximum a-posteriori (MAP) estimate (denoted as \(x_{k+1}\)) of \(x_{k+1}\). The MAP estimate \(x_{k+1}\) is initialized as the prior estimate \(\hat{x}_{k+1}\) and is refined iteratively due to the nonlinear nature of the problem. In each iteration, the error \(\delta x_{k+1}\) between the true state \(x_{k+1}\) and the current estimate \(\hat{x}_{k+1}\), defined as

\[
\delta x_{k+1} \doteq x_{k+1} \oplus \hat{x}_{k+1},
\]

\[
(9)
\]
will be solved by minimizing the posterior distribution considering the prior in (8) and LiDAR/visual measurements. Therefore, the prior distribution in terms of $x_{k+1}$ represented by (8) should be transformed to an equivalent prior distribution in terms of $\delta x_{k+1}$:

$$x_{k+1} \square \delta x_{k+1} = (x_{k+1} \square \delta x_{k+1}) \square \delta x_{k+1}$$

$$\approx x_{k+1} \square \delta x_{k+1} + \mathcal{H} \delta x_{k+1}$$

$$\sim \mathcal{N}(0, \Sigma_{\delta x_{k+1}})$$

(10)

where

$$\mathcal{H} = \left( \frac{\delta \delta x_{k+1}}{\delta x_{k+1}} \right) \left( \frac{\delta x_{k+1}}{\delta x_{k+1}} \right)^{-1}$$

is computed in detail in Appendix B. (10) essentially imposes a prior distribution to $\delta x_{k+1}$ as below:

$$\delta x_{k+1} \sim \mathcal{N}(-H^{-1}(\delta x_{k+1}), H^{-1} \Sigma_{\delta x_{k+1}} H^{-T})$$

(11)

### G. LiDAR measurement

If the $(k+1)$-th measurement is a LiDAR frame, we extract planar feature points from the raw 3D points as in [5] and compensate the in-frame motion as in Section IV-B. Denote $\mathcal{L}_{k+1}$ the set of feature points after motion compensation, we compute the residual of each feature point $L_{pj} \in \mathcal{L}_{k+1}$ where $j$ is the index of feature point and the superscript $L$ denotes that the point is represented in the LiDAR-reference frame.

With $\hat{x}_{k+1}$ being the current estimate of $x_{k+1}$, we can transform $L_{pj}$ from LiDAR frame to the global frame $C_{pj} = G_{R_{Lk+1}}(L_{pj} + L_{pL}) + G_{I_{Lk+1}}$. As the previous LOAM pipeline does in [5] and [9], we search for the nearest planar feature points in the map and use them to fit a plane with normal $u_j$ and an in-plane point $q_j$, the measurement residual is:

$$r_i(\hat{x}_{k+1}, r_{pj}) = u_j^T (G_{pj} - q_j)$$

(12)

Let $n_j$ be the measurement noise of the point $L_{pj}$, we can obtain the true point location $L_{pj}^{sT}$ by compensating the noise from $L_{pj}$:

$$L_{pj} = t_p^{sT} + n_j, n_j \sim \mathcal{N}(0, \Sigma_{n_j})$$

(13)

This true point location together with the true state $x_{k+1}$ should lead to zero residual in (12), i.e.,

$$0 = r_i(x_{k+1}, L_{pj}^{sT}) = r_i(\hat{x}_{k+1}, L_{pj}) + H^i \delta x_{k+1} + \alpha_j$$

(14)

which constitutes a posteriori distribution for $\delta x_{k+1}$. In (14), $x_{k+1}$ is parameterized by its error $\delta x_{k+1}$ defined in (9) and $\alpha_j \sim \mathcal{N}(0, \Sigma_{\alpha_j})$:

$$H^i_j = \frac{\partial r_i(\hat{x}_{k+1}, L_{pj})}{\delta x_{k+1}}$$

$$\Sigma_{\alpha_j} = F_{pj} \Sigma_{n_{pj}} F_{pj}^T$$

$$F_{pj} = \left( \frac{\delta r_i(\hat{x}_{k+1}, L_{pj})}{\delta p_j} \right) = G_{R_{Lk+1}}(L_{pj})$$

(15)

The detailed computation of $H^i_j$ can be found in Appendix C.

### H. Visual measurement

If the $(k+1)$-th frame is a camera image, we extract the FAST corner feature points $C_{k+1}$ from the undistorted image and use KLT optical flow to track feature points in $C_{k+1}$ seen by keyframes in the current sliding window (Section V). If a feature point in $C_{k+1}$ is lost or has not been yet tracked before, we triangulate the new feature point in 3D space (visual landmarks) with the optimal estimated camera poses.

The reprojection errors between visual landmarks and its tracked feature points in the $(k+1)$-th frame are used for updating the current state estimate $\hat{x}_{k+1}$. For an extracted corner point $C_{ps} = [u_s, v_s] \in C_{k+1}$ where $s$ is the index of corner point, its correspondence landmark in 3D space is denoted as $G_{ps}$, then the measurement residual of $G_{ps}$ is:

$$C_{ps} = (G_{R_{Lk+1}}^T \hat{R}_{C_{k+1}})^T G_{ps}$$

(16)

where $C_{ps}$ is the pin-hole projection model.

Now considering the measurement noise, we have:

$$G_{ps} = G_{ps}^{st} + n_{ps}, \ n_{ps} \sim \mathcal{N}(0, \Sigma_{n_{ps}})$$

(17)

$$C_{ps} = C_{ps}^{st} + n_{ps}, \ n_{ps} \sim \mathcal{N}(0, \Sigma_{n_{ps}})$$

(18)

where $G_{ps}^{st}$ and $C_{ps}^{st}$ are the true value of $G_{ps}$ and $C_{ps}$, respectively. With these, we obtain the first order Taylor expansion of the true zero residual $r_c(\hat{x}_{k+1}, C_{ps}^{st})$ as:

$$0 = r_c(x_{k+1}, C_{ps}^{st})$$

(19)

which constitutes another posteriori distribution for $\delta x_{k+1}$. In (19), $\beta_s \sim \mathcal{N}(0, \Sigma_{\beta_s})$ and:

$$H_s^{\beta_s} = \frac{\partial r_c(\hat{x}_{k+1}, C_{ps}^{st}, G_{ps})}{\delta x_{k+1}}$$

$$\Sigma_{\beta_s} = \Sigma_{n_{ps}} + F_{ps} \Sigma_{n_{ps}} F_{ps}^T$$

(20)

The detailed computation of $H_s^{\beta_s}$ and $F_{ps}$ is given in appendix D.

### I. Update of error-state iterated Kalman filter

Combining the prior distribution (11), the posterior distribution due to LiDAR measurement (14) and the posterior distribution due to visual measurement (19), we obtain the maximum a posteriori (MAP) estimation of $\delta x_{k+1}$:

$$\min_{\delta x_{k+1}} \left( \left\| \delta x_{k+1} \right\| = \delta x_{k+1} + \mathcal{H} \delta x_{k+1} \right)$$

$$\left\| \delta x_{k+1} \right\| = \left\| \delta x_{k+1} \right\|$$

(21)

$$\left\| \delta x_{k+1} \right\|^2 = x_{k+1} \Sigma_{x_{k+1}} x_{k+1}$$

Notice that the measurements of LiDAR and camera may not appear at the same time instant.
Our factor graph optimization is similar to VINS-Mono [22], but further incorporates pose constraints due to LiDAR measurements as shown in Fig. 5. Constraints due to IMU preintegration are also included to connect the LiDAR factor with camera factor. To keep the back-end optimization lightweight, the LiDAR poses in the pose graph are fixed and the LiDAR raw point measurements are not engaged in the pose graph optimization.

VI. EXPERIMENTS AND RESULTS

A. Our device for data sampling

Our handheld device for data sampling is shown in Fig. 6 (a), which includes the power supply unit, the onboard DJI manifold-2c\(^3\) computation platform (equipped with an Intel i7-8550u CPU and 8 GB RAM), a global shutter camera, and a LiVOX AVIA\(^4\) LiDAR. The FoV of the camera is 82.9° × 66.5° while the FoV of LiDAR is 70.4° × 77.2°. For quantitatively evaluating the precision of our algorithm (our experiment in Section VI-E), we install a differential-GPS (D-GPS) real-time kinematic (RTK) system\(^3\) on our device, shown in Fig. 6 (b).

![Fig. 6: Our handheld device for data sampling. (a) shows our minimum system, with a total weight 2.09 Kg; (b) a D-GPS RTK mobile station is used to evaluate the accuracy.](image)

![Fig. 7: We evaluate the robustness of our algorithm under scenarios with the aggressive motion (sequence in a–d) and sensor failure by intentionally blocking the camera (e) and LiDAR sensor (f).](image)

![Fig. 8: Our estimated pose and the raw gyroscope reading of Experiment-1. The shaded area in blue, yellow and red represent different phases of aggressive motion, camera-failure and LiDAR-failure, respectively.](image)

\(^3\)https://www.dji.com/manifold-2

\(^4\)https://www.livoxtech.com/avia

\(^5\)https://www.dji.com/d-rtk
Fig. 9: We evaluate our algorithm in a Hong Kong MTR station consisting of cluttered lobby and very long narrow tunnels, as shown in (a). The tunnel is up to 190 meters long and is filled with moving pedestrians, making it extremely challenging for both LiDAR-based and camera-based SLAM methods. (b): the map built by our system is well aligned with the street map of MTR station. (c) Trajectory comparison among our system “R2LIVE”, the LiDAR-inertial system “Fast-LIO”, and visual-inertial system “VINS-Mono” and (our). The starting point is marked with ◆ while the ending point of each trajectory is marked with ●. “VINS-Mono” stopped at middle way due to the failure of feature tracking.

B. Experiment-1: Robustness evaluation with aggressive motion and sensor failure

In this experiment, we evaluate the robustness of our algorithm under the scenario with aggressive motion (see Fig. 7 (a~d)), in which the maximum angular speed reaches up to $300^\circ/s$ (see the gyroscope readings in Fig. 8). In addition, we also simulate drastic sensor failures by intentionally blocking the camera (see Fig. 7(e)) and LiDAR (see Fig. 7(f)).

The results of our estimated attitude and position are shown in Fig. 8, where we use shade the different testing phases with different colors. As shown in Fig. 8, we can see that our estimated trajectory can tightly track the actual motion even in severe rotation and translation. Even when the camera or LiDAR provides no measurements, the estimated trajectory is still very smooth and exhibits no noticeable degradation. These results show that our algorithm is robust enough to endure with the case with aggressive motion or even with the failure of sensors. We refer readers to the accompanying video on XXX showing details of the experiment in practice.

C. Experiment-2: Robustness evaluation in a narrow tunnel-like environments with moving objects

In this experiment, we challenge one of the most difficult scenarios in the scope of camera-based and LiDAR-based SLAM, a MTR station, the HKU station\(^6\). It is a typical narrow tunnel-like environment (see the left figure of Fig. 9(a)), with the maximum reaches 190 meters. Moreover, there are many moving pedestrians frequently showing in the sensor views. All these factors make the SLAM extremely challenging: 1) the long tunnel structure significantly reduces the geometric features for LiDAR-based SLAM, especially when walking along the tunnel or turning the LiDAR around at one end of the tunnel; 2) The highly repeated visual feature (pattern on the floor) causes the error in matching and tracking the visual features in visual SLAM; 3) The many moving pedestrians could cause outliers to the already few point- and visual-features.

Despite these challenges, our system is robust enough to survive in this scene. In Fig. 9 (b), we align our point cloud data with the HKU station street map\(^7\), and find them match tightly. This demonstrates that our localization is of high robustness and accuracy. Moreover, we plot our estimated trajectory together with that of Vins-Mono\(^8\) and Fast-LIO\(^9\) in Fig. 9 (c), where we can see that our method achieves the best overall performance in this experiment.

![Relative pose error of Experiment-4](https://github.com/HKUST-Aerial-Robotics/VINS-Mono)

Fig. 10: The upper figure plots the different trajectories in experiments-4, while the lower figure shows the raw gyroscope reading in the experiment.

![The comparison of trajectories in Experiment-4](https://github.com/HKUST-Aerial-Robotics/VINS-Mono)

Fig. 11: The relative pose error in experiments-4, for the sequence of 300 meters, the median of pose error of “Vins-Mono”, “Fast-LIO” and “R2LIVE” are $0.35\%$,$3.84\%$, $0.41\%$,$0.34\%$ and $(0.27\%$,$0.21\%$) respectively.

\(\text{https://en.wikipedia.org/wiki/HKU\_station}\)
\(\text{https://www.mtr.com.hk/archive/en/services/maps/hku.pdf}\)
\(\text{https://github.com/HKUST-Aerial-Robotics/VINS-Mono}\)
\(\text{https://github.com/hku-mars/FAST\_LIO/}\)
D. Experiment-3: High precision maps building in large-scale indoor & outdoor urban environment

In this experiment, we show that our proposed method is accurate enough to reconstruct a dense 3D, high precision, large-scale indoor-outdoor map of urban environments. We collect data of the HKU main building exterior and interior. The real-time reconstructed 3D maps is shown in Fig. 12, in which the clear and high-quality point cloud demonstrates that our proposed method is of high accuracy. What worth to be mentioned is, without any additional processing (i.e. loop closure), our algorithm can still close the loop itself (see Fig. 12 (c)) after traversing 876 meters, which demonstrates that our proposed method is extremely low drift. Finally, we merge our maps together with the satellite image and find them match tightly (see Fig. 12 (e)).

E. Experiment-4: Quantitative evaluation of precision using D-GPS RTK

In this experiment, we quantitatively evaluate the precision of Vins-Mono (IMU+Camera), Fast-LIO (IMU+LiDAR) and our algorithm by comparing their estimated trajectory with the ground truth trajectory (see the upper figure of Fig. 10) obtained from a real-time differential-GPS (D-GPS) Kinematic (RTK). The data has the maximum angular velocity reaching $130^\circ/s$ (see the gyroscope reading in Fig. 10). We calculate translational and rotational errors for all possible subsequences of length (50,...,300) meters, with the relative pose error (RPE) among these methods shown in Fig. 11.

F. Run time analysis

The average time consumption in experiment 1~4 are listed in Table I, which demonstrates that R2LIVE can achieve real-time on both the desktop PC and embedded computation platform. Noticed that the factor graph optimization is run on a separate thread and therefore is allowed to run at a lower rate.

VII. CONCLUSION

In this letter, we propose a robust, real-time LiDAR-inertial-visual fusion framework based on a high-rate filter-based odometry and factor graph optimization. We fuse the

| Exp-1 (ms) | Exp-2 (ms) | Exp-3 (ms) | Exp-4 (ms) |
|-----------|-----------|-----------|-----------|
| LI-Odom   | 8.95 / 15.98 | 11.38 / 25.30 | 14.92 / 20.34 | 10.92 / 24.37 |
| VI-Odom   | 7.84 / 13.92 | 8.84 / 19.10 | 8.57 / 19.56 | 8.99 / 20.16 |
| FG-OPM    | 26.10 / 43.35 | 30.20 / 65.25 | 29.25 / 68.08 | 27.98 / 60.43 |

### REFERENCES

[1] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, et al., “Towards fully autonomous driving: Systems and algorithms,” in 2011 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2011, pp. 163–168.

[2] A. Bry, A. Bachrach, and N. Roy, “State estimation for aggressive flight in gps-denied environments using onboard sensing,” in 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012, pp. 1–8.

[3] F. Gao, W. Wu, W. Gao, and S. Shen, “Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments,” Journal of Field Robotics, vol. 36, no. 4, pp. 710–733, 2019.

[4] Z. Liu, F. Zhang, and X. Hong, “Low-cost retina-like robotic lidars based on incommensurable scanning,” IEEE Transactions on Mechatronics, 2021, in press. [Online]. Available: https://arxiv.org/pdf/2006.11034.pdf

[5] W. Xu and F. Zhang, “Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter,” IEEE Robotics and Automation Letters, 2021, in press. [Online]. Available: https://arxiv.org/pdf/2010.08196.pdf

[6] J. Lin, X. Liu, and F. Zhang, “A decentralized framework for simultaneous calibration, localization and mapping with multiple lidars,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 4870–4877.

[7] Z. Liu and F. Zhang, “Balm: Bundle adjustment for lidar mapping,” IEEE Robotics and Automation Letters, 2021, in press. [Online]. Available: https://arxiv.org/pdf/2010.08215.pdf

[8] X. Liu and F. Zhang, “Extrinsic calibration of multiple lidars of small fov in targetless environments,” IEEE Robotics and Automation Letters, 2021, in press.
where $\dot{\omega}_i = \omega_{i,m} - b_{ik} \cdot \dot{\omega}_{k,m} - b_{ka} \cdot \dot{\omega}_{a,m}$, and $J_r(\cdot)$ are called the right Jacobian matrix of $SO(3)$:

$$J_r(r) = I - \frac{1 - \cos \|r\|}{\|r\|^2} [r]_x + \frac{|r| - \sin(|r|)}{\|r\|^3} [r]_x [r]_x^T, r \in \mathbb{R}^3$$

For the detailed derivation of $F_{s,k}$ and $F_w$, please refer to the Section B of our supplementary material.

B. The computation of $H$

$$H = \left[ \delta x_{k+1} \otimes \delta x_{k+1} \right]_{\delta \delta x_{k+1}}$$

The $3 \times 3$ matrix $A = J_r^{-1}(\log(C_{R_{k+1}} C_{R_{k+1}}^T))$ and $B = J_r^{-1}(\log(C_{R_{k+1}} C_{R_{k+1}}^T))$. $J_r^{-1}(\cdot)$ are called the inverse right Jacobian matrix of $SO(3)$:

$$J_r^{-1}(\cdot) = I - \frac{1}{2} [r]_x + \frac{1 + \cos(|r|)}{2 |r|^2} [r]_x [r]_x^T$$

C. The computation of $H^j$

$$H^j = u_j^T \left[ -C_{R_{k+1}} [P_A]_x I_{3 \times 3} 0_{3 \times 15} \right]$$

For the detailed derivation of $H^j$, please refer to the Section D of our supplementary material.

D. The computation of $H_s$ and $F_P$

$$H_s = -F_A \cdot F_B$$

$$F_{P_s} = -F_A \cdot F_C$$

with:

$$F_A = \begin{bmatrix} f_x & 0 & -f_z C_{P_{sz}} \\ 0 & f_y & f_z C_{P_{sy}} \\ 0 & -f_y C_{P_{sz}} & C_{P_{sz}} \end{bmatrix}$$

$$F_B = \begin{bmatrix} M_A & M_B & M_C \end{bmatrix}$$

$$F_C = (C^T_{R_{k+1}}) R_{C_k}$$

where $f_x$ and $f_y$ are the focal length, $c_x$ and $c_y$ are the principal point offsets in image plane, and the $3 \times 3$ matrix $M_A$, $M_B$ and $M_C$ are:

$$M_A = (i R_{k+1})^T \left[ (C^T_{R_{k+1}}) P_s \right]$$

$$M_B = - (i R_{k+1})^T$$

$$M_C = \left[ (C^T_{R_{k+1}}) P_s \right] - \left[ (i R_{k+1})^T C_{P_{sz}} \right]$$

For the detailed derivation of $H^j$ and $F_{P_s}$, please refer to Section E of our supplementary material.
Supplementary Material: R²LIVE: A Robust, Real-time, LiDAR-Inertial-Visual tightly-coupled state Estimator and mapping

A. Perturbation on SO(3)

In this appendix, we will use the following approximation of perturbation \( \delta r \to 0 \) on \( SO(3) \) [25, 26]:

\[
\begin{align*}
\text{Exp}(r + \delta r) & \approx \text{Exp}(r)\text{Exp}(J_r(r)\delta r) \\
\text{Exp}(r)\text{Exp}(\delta r) & \approx \text{Exp}(r + J_r^{-1}(r)\delta r) \\
R \cdot \text{Exp}(\delta r) \cdot u & = R (I + [\delta r]_x) u = Ru - R[u]_x \delta r
\end{align*}
\]

where \( u \in \mathbb{R}^3 \) and we use \( [\cdot]_x \) denote the skew-symmetric matrix of vector \( \cdot \); \( J_r(r) \) and \( J_r^{-1}(r) \) are called the right Jacobian and the inverse right Jacobian of \( SO(3) \), respectively.

\[
\begin{align*}
J_r(r) = & I - \frac{1}{||r||^2} ||r|| \cdot [r]_x + \frac{||r|| - \sin(||r||)}{||r||^3} [r]_x^2 \\
J_r^{-1}(r) = & I + \frac{1}{||r||^2} [r]_x + \left( \frac{1}{2||r||^2} - \frac{1 + \cos(||r||)}{2||r||} \right) [r]_x^2
\end{align*}
\]

B. Computation of \( F_{\delta x} \) and \( F_w \)

Combining (4) and (6), we have:

\[
\delta \bar{x}_{i+1} = \bar{x}_{i+1} \begin{bmatrix} \delta \bar{x}_{i+1} \\ \delta \bar{v}_{i+1} \end{bmatrix} = (X_i \otimes (\Delta t \cdot f(\bar{x}_i, u_i, w_i))) \otimes (X_i \otimes (\Delta t \cdot f(\bar{x}_i, u_i, w_i)))
\]

\[
\begin{align*}
\text{Log} & \left( \begin{bmatrix} G R_j \exp (\bar{\omega}_i t) \end{bmatrix} \text{Exp}(G \delta r_i) \text{Exp}(\bar{\omega}_i t) \right) \\
= & \text{Log} \left( \begin{bmatrix} G R_j \exp (\bar{\omega}_i t) \end{bmatrix} \text{Exp}(G \delta r_i) \text{Exp}(\bar{\omega}_i t) \right) \\
= & \log \left( \begin{bmatrix} G R_j \exp (\bar{\omega}_i t) \end{bmatrix} \text{Exp}(G \delta r_i) \text{Exp}(\bar{\omega}_i t) \right)
\end{align*}
\]

\[
\begin{align*}
\delta \bar{x}_i & = \omega_{m_i} - b_{g_i}, \quad \delta \bar{v}_i = \hat{\omega}_i - \delta b_{g_i} - n_{g_i}, \\
\bar{a}_i & = \bar{a}_m_i - b_{a_i}, \quad \bar{a}_i = \bar{a}_i - \delta b_{a_i} - n_{a_i},
\end{align*}
\]

C. The computation of \( H \)

Recalling (15), we have:

\[
\begin{align*}
H = & (\delta \bar{x}_{k+1} \otimes \delta \bar{v}_{k+1}) \otimes (\delta \bar{x}_{k+1} \otimes \delta \bar{v}_{k+1}) \\
= & \text{J}_r^{-1} \left( \text{Log} \left( \text{C}_h \text{R}_{k+1} \right) \right) \text{C}_h \text{R}_{k+1} + \text{I}_L \text{p}_j + \text{I}_P L
\end{align*}
\]

with the \( 3 \times 3 \) matrix \( A = \text{J}_r^{-1} \left( \text{Log} \left( \text{C}_h \text{R}_{k+1} \right) \right) \) and \( B = \text{J}_r^{-1} \left( \text{Log} \left( \text{I}_L \text{R}_{k+1} \right) \right) \).

D. The computation of \( H_j^1 \)

Recalling (12) and (15), we have:

\[
\delta \bar{r}_i (\delta \bar{x}_{k+1} \otimes \delta \bar{v}_{k+1}, \text{P}_j) = \text{u}_j^T (\text{C}_h \text{P}_{i+1} + \delta \bar{p}_{i+1} - \text{q}_j + \text{G} \text{R}_{i+1} \text{Exp}(G \delta r_{i+1}^j + \text{I}_L \text{p}_j + \text{I}_P L))
\]

And with the small perturbation approximation, we get:

\[
\begin{align*}
\text{G} \text{R}_{i+1} \text{Exp}(G \delta r_{i+1}^j) \text{P}_a & = \text{G} \text{R}_{i+1} \text{P}_a \\
& \approx \text{I}_L \text{R}_{i+1} \text{P}_a \text{C}_h \text{R}_{i+1} \text{P}_a \text{C}_h \text{R}_{i+1}
\end{align*}
\]

E. The computation of \( F_{\delta x} \) and \( F_w \)

Recalling (16), we have:

\[
\begin{align*}
\text{C}_h \text{P}_s & = \text{P}_C (\delta \bar{x}_{k+1}, \text{G} \text{P}_s) = \left[ \text{C} \text{P}_{s_x} \text{C} \text{P}_{s_y} \text{C} \text{P}_{s_z} \right]^T
\end{align*}
\]

where the function \( \text{P}_C (\delta \bar{x}_{k+1}, \text{G} \text{P}_s) \) is:

\[
\begin{align*}
\text{P}_C (\delta \bar{x}_{k+1}, \text{G} \text{P}_s) = & (\text{G} \text{R}_{i+1} \text{I}_L \text{C}_h \text{P}_s)^T \text{C} \text{P}_s \\
& - (\text{I}_L \text{R}_{i+1} \text{C}_h \text{P}_s)^T \text{P}_i + \text{P}_i \text{C}_h \text{P}_s
\end{align*}
\]

From (20), we have:

\[
\begin{align*}
\text{r}_c (\delta \bar{x}_{k+1}, \text{G} \text{P}_s) & = \text{C} \text{P}_s - \pi (\text{C} \text{P}_s) \\
\pi (\text{C} \text{P}_s) & = \left[ \text{f}_x \text{C} \text{P}_{s_x} + \text{c}_x \text{f}_y \text{C} \text{P}_{s_y} + \text{c}_y \right]^T
\end{align*}
\]

where \( f_x \) and \( f_y \) are the focal length, \( c_x \) and \( c_y \) are the principal point offsets in image plane.
For conveniently, we omit the \( \left( \cdot \right)_{\tilde{x}_{k+1} = 0} \) in the following derivation, and we have:

\[
\begin{align*}
H_s^c &= -\frac{\partial \pi^c(CP_s)}{\partial P_s} \cdot \frac{\partial P_C(\tilde{x}_{k+1} \oplus \delta \tilde{x}_{k+1}, GP_s)}{\delta \tilde{x}_{k+1}} \tag{S8} \\
F_p^c &= -\frac{\partial \pi^c(CP_s)}{\partial P_s} \cdot \frac{\partial P_C(\tilde{x}_{k+1}, GP_s)}{\delta P_s} \tag{S9}
\end{align*}
\]

where:

\[
\frac{\partial \pi^c(CP_s)}{\partial P_s} = \frac{1}{CP_s} \begin{bmatrix} f_s & 0 \\ 0 & f_y \end{bmatrix} \tag{S10}
\]

\[
\frac{\partial P_C(\tilde{x}_{k+1}, GP_s)}{\delta P_s} = \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T \tag{S11}
\]

According to Section A, we have the following approximation of \( P_C(\tilde{x}_{k+1} \oplus \delta \tilde{x}_{k+1}, GP_s) \):

\[
P_C(\tilde{x}_{k+1} \oplus \delta \tilde{x}_{k+1}, GP_s) \\
= \begin{bmatrix} C\tilde{R}_{k+1} \exp(C\delta \tilde{r}_{k+1}) \hat{\tilde{R}}_C \exp(I\delta \tilde{r}_{k+1}) \end{bmatrix}^T GP_s \hat{\tilde{p}}_C \\
- \begin{bmatrix} I \delta \tilde{p}_C \end{bmatrix} \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \hat{\tilde{p}}_C \\
\approx P_b(\tilde{x}_{k+1}, GP_s) + \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \hat{\tilde{p}}_C \\
+ \begin{bmatrix} I \delta \tilde{p}_C \end{bmatrix} \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \hat{\tilde{p}}_C \\
- \begin{bmatrix} I \delta \tilde{p}_C \end{bmatrix} \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \hat{\tilde{p}}_C \\
- \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \hat{\tilde{p}}_C \\
\begin{bmatrix} I \delta \tilde{p}_C \end{bmatrix} \begin{bmatrix} I \delta \tilde{p}_C \end{bmatrix} \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \hat{\tilde{p}}_C \\
\]

With this, we can derive:

\[
\frac{\partial P_C(\tilde{x}_{k+1} \oplus \delta \tilde{x}_{k+1}, GP_s)}{\delta \tilde{x}_{k+1}} = \begin{bmatrix} M_A & M_B & M_C - I \end{bmatrix} \begin{bmatrix} 0_{3 \times 1} \end{bmatrix} \tag{S12}
\]

\[
M_A = \begin{bmatrix} I \hat{\tilde{R}}_C \end{bmatrix} \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \\
M_B = -\begin{bmatrix} I \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \\
M_C = \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s - \begin{bmatrix} I \hat{\tilde{R}}_C \end{bmatrix} \begin{bmatrix} C\tilde{R}_{k+1} \hat{\tilde{R}}_C \end{bmatrix}^T GP_s \\
\]

Substituting (S10), (S11) and (S12) into (S8) and (S9), we finish the computation of \( H_s^c \) and \( F_p^c \).