R^{2}LIVE: 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 measurements 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 measurements 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 environments of different scale (see attached video[1]). 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[2].**

***Index Terms*—SLAM, Mapping, Sensor Fusion.**

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]–[9] draw the attention of the community to this field of research. However, the accuracy of LiDAR-based SLAM methods would significantly degrade or even fail in those scenarios with few available geometry features, which is more critical for those LiDARs with small FoV [10]. In such scenarios, adding visual features could increase the system's robustness and accuracy. In this work, 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 develop a tightly-coupled LiDAR-inertial-visual system for real-time state estimation and mapping. Building on several key techniques from current state-of-the-art LiDAR-inertial and visual-inertial navigation systems, the system consists of a high-rate filter-based odometry and a low-rate 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.
- We conduct various experiments showing that the developed system is able to run 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. It achieves more accurate and robust results than the current existing baselines and is accurate enough to be used to reconstruct large-scale, indoor-outdoor dense 3D maps of building structures (see Fig. 1).
- We open-source the system, which could benefit the whole robotic community and serve as a baseline for comparison in this field of research.

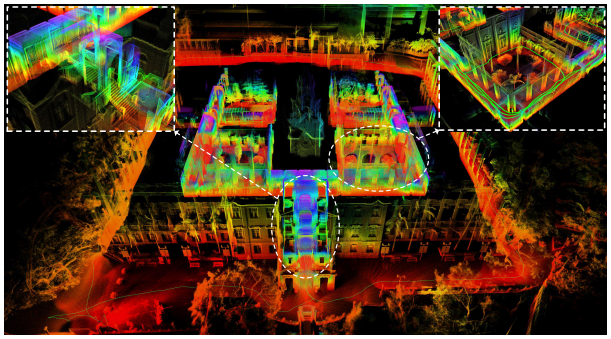


Fig. 1: We use our proposed system to reconstruct a high-precision, large-scale, indoor-outdoor, dense 3D map of the main building of the University of Hong Kong (HKU). The green path is the computed trajectory and the 3D points are colored by height.

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* [11] first proposed a LiDAR odometry and mapping framework, LOAM, that combines ICP method [12] 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* [13] propose a lightweight and ground-optimized LOAM (LeGO-LOAM), which discards unreliable features in the step of ground plane

Manuscript received February 24, 2021; Revised May 21, 2021; Accepted June 15, 2021.

This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers comments. This work is supported by DJI under the grant number 200009538.

J. Lin, C. Zheng, W. Xu and F. Zhang are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong SAR, China. {jiarong.lin, zhengcr, wuweii, fuzhang}@hku.hk

Digital Object Identifier (DOI): see top of this page.

[1]https://youtu.be/9lqRHmlN_MA

[2]https://github.com/hku-mars/r2live

segmentation. These works [11, 13] are mainly based on multi-line spinning LiDARs. Our previous works [10, 14] 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 are very vulnerable to degenerated scenarios with few geometric features.

B. LiDAR-Inertial Odometry

The existing works of LiDAR-Inertial fusion can be categorized into two classes: loosely-coupled and the tightly-coupled. Loosely-coupled methods deal with two sensors separately to infer their motion constraints while tightly-coupled approaches directly fuse LiDAR and inertial measurements through joint-optimization. In LOAM [11], IMU is used to de-skew the LiDAR scan and give a motion prior for scan-to-scan matching. It is a loosely-coupled method since the IMU is not engaged in the scan registration. Compared with loosely-coupled methods, tightly-coupled methods show higher robustness and accuracy, therefore drawing increasing research interests recently. For example, authors in [15] propose LIOM which uses a graph optimization based on the priors from LiDAR-Inertial odometry and a rotation-constrained refinement method. Compared with the former algorithm, LIO-SAM [16] optimizes a sliding window of keyframe poses in a factor graph to achieve higher accuracy. Similarly, Li *et al*. propose LiLi-OM [17] for both conventional and solid-state LiDARs based on sliding window optimization. LINS [18] is the first tightly-coupled LIO that solves the 6 DOF ego-motion via iterated Kalman filtering. To lower the high computation load in calculating the Kalman gain, our previous work FAST-LIO [5] proposes a new formula of the Kalman gain computation, the resultant computation complexity depends on the state dimension instead of measurement dimension. The work achieves up to 50 Hz odometry and mapping rate while running on embedded computers onboard a UAV.

C. LiDAR-Inertial-Visual Odometry

On the basis of LiDAR-Inertial methods, LiDAR-Inertial-Visual odometry incorporating measurements from visual sensors shows higher robustness and accuracy. Zhang and Singh in [19] propose a LiDAR-inertial-visual system that uses a loosely-coupled Visual-Inertial odometry (VIO) as the motion model for initializing the LiDAR mapping subsystem. Similarly, Shao *et al* in [20] propose a stereo visual-inertial LiDAR SLAM that incorporates the tightly-coupled stereo visual-inertial odometry with the LiDAR mapping and LiDAR enhanced visual loop closure. The overall system is still a loosely-coupled LiDAR-inertial-visual fusion since the LiDAR data are not jointly optimized together with the visual-inertial measurements. There are also some RGB-D-inertial SLAM systems, such as [20, 21]. Being designed for RGB-D cameras, their effectiveness for LiDARs are not verified due to the significant difference in measurement pattern, range and density between RGB-D cameras and LiDAR sensors. In the work of [22], the LiDAR measurements are used to provide depth information for camera images, forming a system similar to RGB-D camera that can leverage existing visual SLAM works such as ORB-SLAM2 [23]. This is also a loosely-coupled method as it ignores the direct constraints on state imposed by LiDAR measurements. Zuo *et al* [24] propose a LIC-fusion framework combining IMU measurements, sparse visual features, and LiDAR plane and edge features with online spatial and temporal calibration based on the MSCKF framework, which exhibits higher accuracy and robustness than other state-of-the-art methods in their experiment results. In quick succession, their further work termed LIC-Fusion 2.0 [25] refines a novel plane-feature tracking algorithm across multiple LiDAR scans within a sliding-window to make LiDAR scan matching more robust.

Our proposed system tightly fuses the data of different types of sensors, estimates the state within the framework of error-state iterated Kalman-filter achieving real-time performance, meanwhile improves the overall accuracy with a factor graph optimization. Various real-world dataset results show that our system is more accurate and robust than current state-of-the-art LiDAR-inertial (FAST-LIO [5]), visual-inertial (VINS-mono [26]), and LiDAR-visual systems (CamVox [22]). We also open source on our implementation on Github[3], which are not available for prior LiDAR-inertial-visual SLAM systems [19, 20, 24, 25].

III. The Overview of Our System

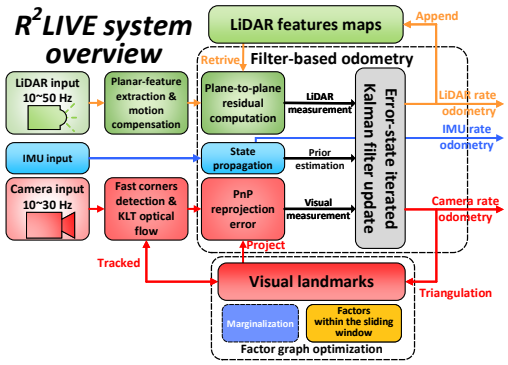


Fig. 2: The overview of our system.

The overview of our system is shown in Fig. 2. For each scan of LiDAR input, we extract planar feature points, compensate the motion distortion (i.e., de-skewing), and compute the point-to-plane residual. Similarly, for an incoming image, we detect fast corners, track them with a map of visual landmarks, and compute the re-projection error. Then, the LiDAR point-to-plane residual and the image re-projection errors are tightly fused with the IMU propagation in an error-state iterated Kalman filter (see Section IV). The fused pose is then used to append new points in current LiDAR scan to point-cloud map and triangulate new visual landmarks. To further improve the accuracy of visual measurements, we leverage the factor graph optimization to refine the visual landmarks within a local sliding window (see Section V).

[3]https://github.com/hku-mars/r2live

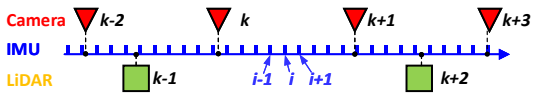


Fig. 3: Illustration of the input data sequences, where the frame rate of IMU, camera, and LiDAR is 200 Hz, 20 Hz and 10 Hz, respectively. The notation i denotes the index of IMU data while k denotes the index of LiDAR or camera measurements.

IV. Filter-based Odometry

A. The boxplus "$\boxplus$" and boxminus "$\boxminus$" operator

In this paper, we make use of the "$\boxplus$" and "$\boxminus$" 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, "$\boxplus$" and "$\boxminus$", establishing a bijective map between the local neighborhood of $\mathcal{M}$ and its tangent space $\mathbb{R}^n$ [27]:

$$\boxplus : \mathcal{M} \times \mathbb{R}^n \to \mathcal{M}, \quad \boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n \tag{1}$$

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

$$\begin{bmatrix}\mathbf{R}\\ \mathbf{a}_1\end{bmatrix} \boxplus \begin{bmatrix}\mathbf{r}\\ \mathbf{a}_2\end{bmatrix} \triangleq \begin{bmatrix}\mathbf{R}\cdot \mathtt{Exp}(\mathbf{r})\\ \mathbf{a}_1+\mathbf{a}_2\end{bmatrix}, \quad \begin{bmatrix}\mathbf{R}_1\\ \mathbf{a}_1\end{bmatrix} \boxminus \begin{bmatrix}\mathbf{R}_2\\ \mathbf{a}_2\end{bmatrix} \triangleq \begin{bmatrix}\mathtt{Log}(\mathbf{R}_2^T\mathbf{R}_1)\\ \mathbf{a}_1-\mathbf{a}_2\end{bmatrix} \tag{2}$$

where $\mathbf{r} \in \mathbb{R}^3$, $\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{R}^n$, $\mathtt{Exp}(\cdot)$ and $\mathtt{Log}(\cdot)$ denote the Rodrigues' transformation between the rotation matrix and rotation vector[4].

B. Continuous-time kinematic model

In our current work, we assume that the time offsets among all the three sensors, LiDAR, camera, and IMU, are pre-calibrated. Furthmore, 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.

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

$$\begin{aligned} {}^G\dot{\mathbf{p}}_I &= {}^G\mathbf{v}_I, {}^G\dot{\mathbf{v}}_I = {}^G\mathbf{R}_I(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) + {}^G\mathbf{g},\\ {}^G\dot{\mathbf{R}}_I &= {}^G\mathbf{R}_I\left[\boldsymbol{\omega}_m - \mathbf{b}_\mathbf{g} - \mathbf{n}_\mathbf{g}\right]_\times, \ \dot{\mathbf{b}}_\mathbf{g} = \mathbf{n}_\mathbf{bg}, \dot{\mathbf{b}} = \mathbf{n}_\mathbf{ba} \end{aligned} \tag{3}$$

where ${}^G(\cdot)$ denotes a vector represented in the global frame (i.e. the first gravitational aligned IMU frame [26]), ${}^G\mathbf{R}_I$ and ${}^G\mathbf{p}_I$ are the attitude and position of the IMU relative to the global frame, ${}^G\mathbf{g}$ is the gravitational acceleration[5], $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$ are the raw gyroscope and accelerometer readings, $\mathbf{n}_\mathbf{a}$ and $\mathbf{n}_\mathbf{g}$ are the white noise of IMU measurement, $\mathbf{b}_\mathbf{a}$ and $\mathbf{b}_\mathbf{g}$ are the bias of gyroscope and accelerometer, which are modelled as random walk driven by Gaussian noise $\mathbf{n}_\mathbf{bg}$ and $\mathbf{n}_\mathbf{ba}$.

[4]https://en.wikipedia.org/wiki/Rodrigues'_rotation_formula

[5] ${}^G\mathbf{g} = [0, 0, 9.805]$ at Hong Kong SAR, China.

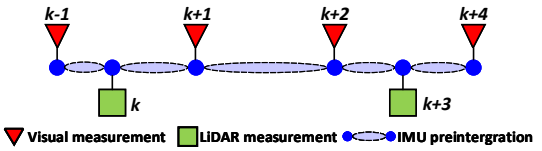


Fig. 4: The illustration of the update of our error-state iterated Kalman filter.

C. Discrete IMU model

We discretize the continuous model (3) at the IMU rate. Let $\mathbf{x}_i$ be the state vector at the i-th IMU measurement:

$$\mathbf{x}_i = \begin{bmatrix} {}^G\mathbf{R}_{I_i}^T & {}^G\mathbf{p}_{I_i}^T & {}^I\mathbf{R}_{C_i}^T & {}^I\mathbf{p}_{C_i}^T & {}^G\mathbf{v}_i^T & \mathbf{b}_{\mathbf{g}_i}^T & \mathbf{b}_{\mathbf{a}_i}^T \end{bmatrix}^T \tag{4}$$

Discretizing (3) by zero-order holder (i.e., IMU measurements over one sampling time period Δt are constant), we obtain

$$\mathbf{x}_{i+1} = \mathbf{x}_i \boxplus (\Delta t \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_i)) \tag{5}$$

where

$$\mathbf{u}_i = \begin{bmatrix} \boldsymbol{\omega}_{m_i}^T & \mathbf{a}_{m_i}^T \end{bmatrix}^T, \quad \mathbf{w}_i = \begin{bmatrix} \mathbf{n}_{\mathbf{g}_i}^T & \mathbf{n}_{\mathbf{a}_i}^T & \mathbf{n}_{\mathbf{bg}_i}^T & \mathbf{n}_{\mathbf{ba}_i}^T \end{bmatrix}^T \tag{6}$$

$$\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_i) = \begin{bmatrix} \boldsymbol{\omega}_{m_i} - \mathbf{b}_{\mathbf{g}_i} - \mathbf{n}_{\mathbf{g}_i} \\ {}^G\mathbf{v}_i \\ \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \\ {}^G\mathbf{R}_{I_i}(\mathbf{a}_{m_i} - \mathbf{b}_{\mathbf{a}_i} - \mathbf{n}_{\mathbf{a}_i}) + {}^G\mathbf{g} \\ \mathbf{n}_{\mathbf{bg}_i} \\ \mathbf{n}_{\mathbf{ba}_i} \end{bmatrix} \tag{7}$$

D. Propagation

In our work, we leverage an on-manifold iterated error state Kalman filter [28] to estimate the state vector $\mathbf{x}_i$, in which the state estimation error $\delta\hat{\mathbf{x}}_i$ is characterized in the tangent space of the state estimate $\hat{\mathbf{x}}_i$:

$$\begin{aligned} \delta\hat{\mathbf{x}}_i &\triangleq \mathbf{x}_i \boxminus \hat{\mathbf{x}}_i \\ &= \begin{bmatrix} {}^G\delta\hat{\mathbf{r}}_{I_i}^T & {}^G\delta\hat{\mathbf{p}}_{I_i}^T & {}^I\delta\hat{\mathbf{r}}_{C_i}^T & {}^I\delta\hat{\mathbf{p}}_{C_i}^T & {}^G\delta\hat{\mathbf{v}}_i^T & \delta\hat{\mathbf{b}}_{\mathbf{g}_i}^T & \delta\hat{\mathbf{b}}_{\mathbf{a}_i}^T \end{bmatrix}^T \\ &\sim \mathcal{N}(\mathbf{0}_{21\times 1}, \boldsymbol{\Sigma}_{\delta\hat{\mathbf{x}}_i}) \end{aligned} \tag{8}$$

Note that $\delta\hat{\mathbf{x}}_i \in \mathbb{R}^{21}$ is in minimum dimension (the system dimension 21) and is a random vector with covariance $\boldsymbol{\Sigma}_{\delta\hat{\mathbf{x}}_i}$. ${}^G\delta\hat{\mathbf{r}}_{I_i}^T$ and ${}^I\delta\hat{\mathbf{r}}_{C_i}^T$ are:

$${}^G\delta\hat{\mathbf{r}}_{I_i} = \mathtt{Log}({}^G\hat{\mathbf{R}}_{I_i}^T {}^G\mathbf{R}_{I_i}), \quad {}^I\delta\hat{\mathbf{r}}_{C_i} = \mathtt{Log}({}^I\hat{\mathbf{R}}_{C_i}^T {}^I\mathbf{R}_{C_i}) \tag{9}$$

Once receiving a new IMU measurement, the state estimate is propagated by setting the process noise in (5) to zero:

$$\hat{\mathbf{x}}_{i+1} = \hat{\mathbf{x}}_i \boxplus (\Delta t \cdot \mathbf{f}(\hat{\mathbf{x}}_i, \mathbf{u}_i, \mathbf{0})). \tag{10}$$

The associated estimation error is propagated in the linearized error space as follows (see [28] for more details):

$$\begin{aligned} \delta\hat{\mathbf{x}}_{i+1} &= \mathbf{x}_{i+1} \boxminus \hat{\mathbf{x}}_{i+1} \\ &= (\mathbf{x}_i \boxplus (\Delta t \cdot \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_i))) \boxminus (\hat{\mathbf{x}}_i \boxplus (\Delta t \cdot \mathbf{f}(\hat{\mathbf{x}}_i, \mathbf{u}_i, \mathbf{0}))) \\ &\approx \mathbf{F}_{\delta\hat{\mathbf{x}}}\delta\hat{\mathbf{x}} + \mathbf{F}_\mathbf{w}\mathbf{w}_i \\ &\sim \mathcal{N}(\mathbf{0}_{21\times 1}, \boldsymbol{\Sigma}_{\delta\hat{\mathbf{x}}_{i+1}}) \end{aligned} \tag{11}$$

where:

$$\boldsymbol{\Sigma}_{\delta\hat{\mathbf{x}}_{i+1}} = \mathbf{F}_{\delta\hat{\mathbf{x}}}\boldsymbol{\Sigma}_{\delta\hat{\mathbf{x}}_i}\mathbf{F}_{\delta\hat{\mathbf{x}}}^T + \mathbf{F}_\mathbf{w}\mathbf{Q}\mathbf{F}_\mathbf{w}^T \tag{12}$$

$$\mathbf{F}_{\delta\hat{\mathbf{x}}} = \left.\frac{\partial(\delta\hat{\mathbf{x}}_{i+1})}{\partial\delta\hat{\mathbf{x}}_i}\right|_{\delta\hat{\mathbf{x}}_i=\mathbf{0},\mathbf{w}_i=\mathbf{0}}, \quad \mathbf{F}_\mathbf{w} = \left.\frac{\partial(\delta\hat{\mathbf{x}}_{i+1})}{\partial\mathbf{w}_i}\right|_{\delta\hat{\mathbf{x}}_i=\mathbf{0},\mathbf{w}_i=\mathbf{0}}$$

with $\mathbf{Q}$ is the diagonal covariance matrix of noise, and the exact values of $\mathbf{F}_{\delta\hat{\mathbf{x}}}$ and $\mathbf{F}_{\mathbf{w}}$ are computed in Section B of our Supplementary Material.[6]

The two propagations in (10) and (12) start from the optimal state and covariance estimate after fusing the most recent LiDAR/camera measurement (e.g., the k-th measurement, see Section IV-I), and repeat until receiving the next LiDAR/camera measurement (e.g., the $(k+1)$-th measurement). The relation between time index i and k is shown in Fig. 3.

E. The prior distribution

Let the two propagations in (10) and (12) stop at the $(k+1)$-th LiDAR/camera measurement (see Fig. 4), and the propagated state estimate and covariance are $\hat{\mathbf{x}}_{k+1}$ and $\mathbf{\Sigma}_{\delta\hat{\mathbf{x}}_{k+1}}$, respectively. They essentially impose a prior distribution for the state $\mathbf{x}_{k+1}$ before fusing the $(k+1)$-th measurement as below:

$$\mathbf{x}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\delta\hat{\mathbf{x}}_{k+1}}). \tag{13}$$

F. Initialization of iterated update

The prior distribution in (13) will be fused with the LiDAR or camera measurements to produce a maximum a-posteriori (MAP) estimate (denoted as $\check{\mathbf{x}}_{k+1}$) of $\mathbf{x}_{k+1}$. The MAP estimate $\check{\mathbf{x}}_{k+1}$ is initialized as the prior estimate $\hat{\mathbf{x}}_{k+1}$ and is refined iteratively due to the nonlinear nature of the problem. In each iteration, the error $\delta\check{\mathbf{x}}_{k+1}$ between the true state $\mathbf{x}_{k+1}$ and the current estimate $\check{\mathbf{x}}_{k+1}$, defined as

$$\delta\check{\mathbf{x}}_{k+1} \triangleq \mathbf{x}_{k+1} \boxminus \check{\mathbf{x}}_{k+1}, \tag{14}$$

will be solved by minimizing the posterior distribution considering the prior in (13) and LiDAR/visual measurements. Therefore, the prior distribution in terms of $\mathbf{x}_{k+1}$ represented by (13) should be transformed to an equivalent prior distribution in terms of $\delta\check{\mathbf{x}}_{k+1}$:

$$\begin{aligned} \mathbf{x}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1} &= (\check{\mathbf{x}}_{k+1} \boxplus \delta\check{\mathbf{x}}_{k+1}) \boxminus \hat{\mathbf{x}}_{k+1} \\ &\approx \check{\mathbf{x}}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1} + \boldsymbol{\mathcal{H}}\delta\check{\mathbf{x}}_{k+1} \\ &\sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\delta\hat{\mathbf{x}}_{k+1}}), \end{aligned} \tag{15}$$

where

$$\boldsymbol{\mathcal{H}} = \left.\frac{(\check{\mathbf{x}}_{k+1} \boxplus \delta\check{\mathbf{x}}_{k+1}) \boxminus \hat{\mathbf{x}}_{k+1}}{\partial\delta\check{\mathbf{x}}_{k+1}}\right|_{\delta\check{\mathbf{x}}_{k+1}=\mathbf{0}}$$

is computed in detail in Section C of our Supplementary Material. essentially imposes a prior distribution to $\delta\check{\mathbf{x}}_{k+1}$ as below:

$$\delta\check{\mathbf{x}}_{k+1} \sim \mathcal{N}(-\boldsymbol{\mathcal{H}}^{-1}\left(\check{\mathbf{x}}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1}\right), \boldsymbol{\mathcal{H}}^{-1}\mathbf{\Sigma}_{\delta\hat{\mathbf{x}}_{k+1}}\boldsymbol{\mathcal{H}}^{-T}) \tag{16}$$

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 $\boldsymbol{\mathcal{L}}_{k+1}$ the set of feature points after motion compensation, we compute the residual of each feature point ${}^{L}\mathbf{p}_j \in \boldsymbol{\mathcal{L}}_{k+1}$ where j is the index of feature point and the superscript L denotes that the point is represented in the LiDAR-reference frame.

With $\check{\mathbf{x}}_{k+1}$ being the current estimate of $\mathbf{x}_{k+1}$, we can transform ${}^{L}\mathbf{p}_j$ from LiDAR frame to the global frame ${}^{G}\mathbf{p}_j = {}^{G}\check{\mathbf{R}}_{I_{k+1}}({}^{I}\mathbf{R}_L\,{}^{L}\mathbf{p}_j + {}^{I}\mathbf{p}_L) + {}^{G}\check{\mathbf{p}}_{I_{k+1}}$. As the previous LOAM pipeline does in [5, 10], we search for the nearest planar feature points in the map and use them to fit a plane with normal $\mathbf{u}_j$ and an in-plane point $\mathbf{q}_j$, the measurement residual is:

$$\mathbf{r}_l(\check{\mathbf{x}}_{k+1}, {}^{L}\mathbf{p}_j) = \mathbf{u}_j^T\left({}^{G}\mathbf{p}_j - \mathbf{q}_j\right) \tag{17}$$

Let $\mathbf{n}_j$ be the measurement noise of the point ${}^{L}\mathbf{p}_j$, we can obtain the true point location ${}^{L}\mathbf{p}_j^{\mathtt{gt}}$ by compensating the noise from ${}^{L}\mathbf{p}_j$:

$${}^{L}\mathbf{p}_j = {}^{L}\mathbf{p}_j^{\mathtt{gt}} + \mathbf{n}_j, \mathbf{n}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{n}_j}). \tag{18}$$

This true point location together with the true state $\mathbf{x}_{k+1}$ should lead to zero residual in (17), i.e.,

$$\mathbf{0} = \mathbf{r}_l(\mathbf{x}_{k+1}, {}^{L}\mathbf{p}_j^{\mathtt{gt}}) \approx \mathbf{r}_l(\check{\mathbf{x}}_{k+1}, {}^{L}\mathbf{p}_j) + \mathbf{H}_j^l\delta\check{\mathbf{x}}_{k+1} + \boldsymbol{\alpha}_j, \tag{19}$$

which constitutes a posterior distribution for $\delta\check{\mathbf{x}}_{k+1}$. In (19), $\mathbf{x}_{k+1}$ is parameterized by its error $\delta\check{\mathbf{x}}_{k+1}$ defined in (14) and $\boldsymbol{\alpha}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\boldsymbol{\alpha}_j})$:

$$\mathbf{H}_j^l = \left.\frac{\partial\mathbf{r}_l(\check{\mathbf{x}}_{k+1} \boxplus \delta\check{\mathbf{x}}_{k+1}, {}^{L}\mathbf{p}_j)}{\partial\delta\check{\mathbf{x}}_{k+1}}\right|_{\delta\check{\mathbf{x}}_{k+1}=\mathbf{0}} \tag{20}$$

$$\mathbf{\Sigma}_{\boldsymbol{\alpha}_j} = \mathbf{F}_{\mathbf{p}_j}\mathbf{\Sigma}_{\mathbf{n}_j}\mathbf{F}_{\mathbf{p}_j}^T \tag{21}$$

$$\mathbf{F}_{\mathbf{p}_j} = \left(\frac{\partial\mathbf{r}_l(\check{\mathbf{x}}_{k+1}, {}^{L}\mathbf{p}_j)}{\partial\,{}^{L}\mathbf{p}_j}\right) = {}^{G}\check{\mathbf{R}}_{I_{k+1}}\,{}^{I}\mathbf{R}_L \tag{22}$$

The detailed computation of $\mathbf{H}_j^l$ can be found in our Section D of our Supplementary Material.

H. Visual measurement

If the $(k+1)$-th frame is a camera image, we extract the FAST corner feature points $\boldsymbol{\mathcal{C}}_{k+1}$ from the undistorted image and use KLT optical flow to track feature points in $\boldsymbol{\mathcal{C}}_{k+1}$ seen by keyframes in the current sliding window (Section V). If a feature point in $\boldsymbol{\mathcal{C}}_{k+1}$ 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 $\check{\mathbf{x}}_{k+1}$. For an extracted corner point ${}^{C}\mathbf{p}_s = \begin{bmatrix} u_s & v_s \end{bmatrix}^T \in \boldsymbol{\mathcal{C}}_{k+1}$ where s is the index of corner point, its correspondence landmark in 3D space is denoted as ${}^{G}\mathbf{P}_s$, then the measurement residual of ${}^{C}\mathbf{p}_s$ is:

$$\begin{aligned} {}^{C}\mathbf{P}_s &= \left({}^{G}\check{\mathbf{R}}_{I_{k+1}}\,{}^{I}\check{\mathbf{R}}_{C_{k+1}}\right)^T{}^{G}\mathbf{P}_s - {}^{I}\check{\mathbf{R}}_{C_{k+1}}^T\,{}^{I}\check{\mathbf{p}}_{C_{k+1}} \\ &\quad - \left({}^{G}\check{\mathbf{R}}_{I_{k+1}}\,{}^{I}\check{\mathbf{R}}_{C_{k+1}}\right)^T{}^{G}\check{\mathbf{p}}_{I_{k+1}} \\ \mathbf{r}_c\left(\check{\mathbf{x}}_{k+1}, {}^{C}\mathbf{p}_s, {}^{G}\mathbf{P}_s\right) &= {}^{C}\mathbf{p}_s - \boldsymbol{\pi}({}^{C}\mathbf{P}_s) \end{aligned} \tag{23}$$

where $\boldsymbol{\pi}(\cdot)$ is the pin-hole projection model.

Now considering the measurement noise, we have:

$${}^{G}\mathbf{P}_s = {}^{G}\mathbf{P}_s^{\mathtt{gt}} + \mathbf{n}_{\mathbf{P}_s},\ \mathbf{n}_{\mathbf{P}_s} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{n}_{\mathbf{P}_s}}) \tag{24}$$

$${}^{C}\mathbf{p}_s = {}^{C}\mathbf{p}_s^{\mathtt{gt}} + \mathbf{n}_{\mathbf{p}_s},\ \mathbf{n}_{\mathbf{p}_s} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{n}_{\mathbf{p}_s}}) \tag{25}$$

[6] https://github.com/hku-mars/r2live/blob/master/supply/r2live_Supplementary_material.pdf

where ${}^{G}\mathbf{P}_s^{\mathtt{gt}}$ and ${}^{C}\mathbf{p}_s^{\mathtt{gt}}$ are the true value of ${}^{G}\mathbf{P}_s$ and ${}^{C}\mathbf{p}_s$, respectively. With these, we obtain the first order Taylor expansion of the true zero residual $\mathbf{r}_c(\mathbf{x}_{k+1}, {}^{C}\mathbf{p}_s^{\mathtt{gt}})$ as:

$$\begin{aligned}\mathbf{0} &= \mathbf{r}_c(\mathbf{x}_{k+1}, {}^{C}\mathbf{p}_s^{\mathtt{gt}}, {}^{G}\mathbf{P}_s^{\mathtt{gt}}) \\ &\approx \mathbf{r}_c\left(\check{\mathbf{x}}_{k+1}, {}^{C}\mathbf{p}_s, {}^{G}\mathbf{P}_s\right) + \mathbf{H}_s^c \delta\check{\mathbf{x}}_{k+1} + \boldsymbol{\beta}_s,\end{aligned} \tag{26}$$

which constitutes another posterior distribution for $\delta\check{\mathbf{x}}_{k+1}$. In (26), $\boldsymbol{\beta}_s \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}_s})$ and:

$$\mathbf{H}_s^c = \left.\frac{\partial \mathbf{r}_c(\check{\mathbf{x}}_{k+1} \boxplus \delta\check{\mathbf{x}}_{k+1}, {}^{C}\mathbf{p}_s, {}^{G}\mathbf{P}_s)}{\partial \delta\check{\mathbf{x}}_{k+1}}\right|_{\delta\check{\mathbf{x}}_{k+1}=\mathbf{0}} \tag{27}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\beta}_s} = \boldsymbol{\Sigma}_{\mathbf{n}_{\mathbf{p}_s}} + \mathbf{F}_{\mathbf{P}_s} \boldsymbol{\Sigma}_{\mathbf{n}_{\mathbf{P}_s}} {\mathbf{F}_{\mathbf{P}_s}}^T \tag{28}$$

$$\mathbf{F}_{\mathbf{P}_s} = \frac{\partial \mathbf{r}_c(\check{\mathbf{x}}_{k+1}, {}^{C}\mathbf{p}_s, {}^{G}\mathbf{P}_s)}{\partial {}^{G}\mathbf{P}_s} \tag{29}$$

The detailed computation of $\mathbf{H}_s^c$ and $\mathbf{F}_{\mathbf{P}_s}$ is given in Section E of our Supplementary Material.

I. Update of error-state iterated Kalman filter

Combining the prior distribution (16), the posterior distribution due to LiDAR measurement (19) and the posterior distribution due to visual measurement (26), we obtain the maximum a posteriori (MAP) estimation of $\delta\check{\mathbf{x}}_{k+1}$:

$$\begin{aligned}\min_{\delta\check{\mathbf{x}}_{k+1}} \Bigg(&\left\|\check{\mathbf{x}}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1} + \boldsymbol{\mathcal{H}}\delta\check{\mathbf{x}}_{k+1}\right\|^2_{\boldsymbol{\Sigma}_{\delta\hat{\mathbf{x}}_{k+1}}} \\ &+ \sum_{j=1}^{m_l} \left\|\mathbf{r}_l(\check{\mathbf{x}}_{k+1}, {}^{L}\mathbf{p}_j) + \mathbf{H}_j^l \delta\check{\mathbf{x}}_{k+1}\right\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_j}} \\ &+ \sum_{s=1}^{m_c} \left\|\mathbf{r}_c(\check{\mathbf{x}}_{k+1}, {}^{C}\mathbf{p}_s, {}^{G}\mathbf{P}_s) + \mathbf{H}_s^c \delta\check{\mathbf{x}}_{k+1}\right\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{\beta}_s}} \Bigg)\end{aligned} \tag{30}$$

where $\|\mathbf{x}\|^2_{\boldsymbol{\Sigma}} = \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$ is the squared Mahalanobis distance with covariance $\boldsymbol{\Sigma}$.

Notice that the measurements of LiDAR and camera may not appear at the same time instant (see Fig. 4), therefore m_l (or m_c) could be zero in the above optimization. Denote

$$\mathbf{H} = \left[{\mathbf{H}_1^l}^T, \ldots, {\mathbf{H}_{m_l}^l}^T, {\mathbf{H}_1^c}^T, \ldots, {\mathbf{H}_{m_c}^c}^T\right]^T \tag{31}$$

$$\mathbf{R} = \mathrm{diag}(\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_1}, \ldots, \boldsymbol{\Sigma}_{\boldsymbol{\alpha}_{m_l}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}_1}, \ldots, \boldsymbol{\Sigma}_{\boldsymbol{\beta}_{m_c}}) \tag{32}$$

$$\check{\mathbf{z}}_{k+1} = \left[\mathbf{r}_l(\check{\mathbf{x}}_{k+1}, {}^{L}\mathbf{p}_1)^T, \ldots, \mathbf{r}_l(\check{\mathbf{x}}_{k+1}, {}^{L}\mathbf{p}_{m_l})^T, \right. \tag{33}$$

$$\left. \mathbf{r}_c(\check{\mathbf{x}}_{k+1}, {}^{C}\mathbf{p}_1, {}^{G}\mathbf{P}_1)^T, \ldots, \mathbf{r}_c(\check{\mathbf{x}}_{k+1}, {}^{C}\mathbf{p}_{m_c}, {}^{G}\mathbf{P}_{m_c})^T\right]^T \tag{34}$$

$$\mathbf{P} = (\boldsymbol{\mathcal{H}})^{-1} \boldsymbol{\Sigma}_{\delta\hat{\mathbf{x}}_{k+1}} (\boldsymbol{\mathcal{H}})^{-T} \tag{35}$$

Following [5], we have the Kalman gain computed as:

$$\mathbf{K} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} + \mathbf{P}^{-1}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1} \tag{36}$$

Then we can update the state estimate as:

$$\check{\mathbf{x}}_{k+1} = \check{\mathbf{x}}_{k+1} \boxplus \left(-\mathbf{K}\check{\mathbf{z}}_{k+1} - (\mathbf{I} - \mathbf{K}\mathbf{H})(\boldsymbol{\mathcal{H}})^{-1}(\check{\mathbf{x}}_{k+1} \boxminus \hat{\mathbf{x}}_{k+1})\right) \tag{37}$$

The above process (Section IV-G to Section IV-I) is iterated until convergence (i.e., the update is smaller than a given threshold). Such an iterated Kalman Filter is known to be equivalent to a Gauss-Newton optimization [29]. The converged state estimate is then used to (1) project points in the the new LiDAR frame to the world frame and append them to the existing point cloud map; (2) triangulate new visual landmarks of the current frame if it is a keyframe; (3) serve as the starting point of the propagation in Section IV-D for the next cycle:

$$\hat{\mathbf{x}}_{k+1} = \check{\mathbf{x}}_{k+1}, \quad \hat{\boldsymbol{\Sigma}}_{\delta\hat{\mathbf{x}}_{k+1}} = (\mathbf{I} - \mathbf{K}\mathbf{H})\check{\boldsymbol{\Sigma}}_{\delta\mathbf{x}_{k+1}} \tag{38}$$

V. Factor Graph Optimization

As mentioned in Section IV-I, untracked visual landmarks in the newly added keyframe are triangulated to create new visual landmarks. This triangulation is usually of low precision due to keyframe pose estimation error. To further improve the quality of visual landmarks, keyframe poses, and simultaneously calibrate the time offset between the camera and LiDAR-IMU subsystem, we leverage a factor graph optimization for optimizing the camera-poses and the visual landmarks within a sliding window of image keyframes.

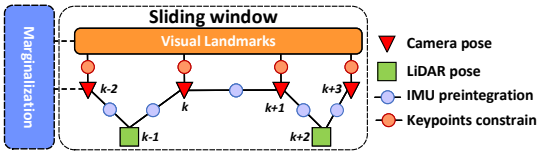


Fig. 5: Our factor graph optimization is a windowed visual bundle adjustment with LiDAR poses constraints.

Our factor graph optimization is similar to VINS-Mono [26], but further incorporates pose constraints from the LiDAR pose estimated in the ESIKF (see Fig. 5). Constraints due to IMU pre-integration are also included to connect the LiDAR factor with camera factor. To keep the back-end optimization light-weight, the LiDAR poses in the pose graph are fixed and the LiDAR raw point measurements are not engaged in the pose graph optimization. The complete factor graph optimization is implemented based on the sliding-window optimization of VINS-Mono [26].

VI. Experiments and Results

In this section, we present experimental results to verify our developed system. Due to the unavailability of open-source implementation of prior LiDAR-inertial-visual SLAM systems [19, 20, 24, 25], we perform comparison with current state-of-the-art LiDAR-inertial (FAST-LIO [5]), visual-inertial (VINS-mono [26]), and LiDAR-visual systems (CamVox [22]) whenever possible.

A. Our device for data sampling

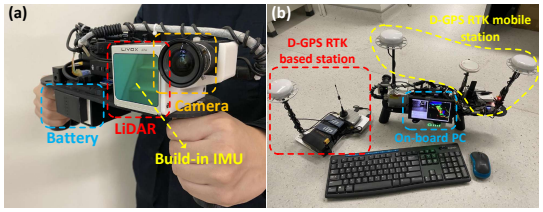
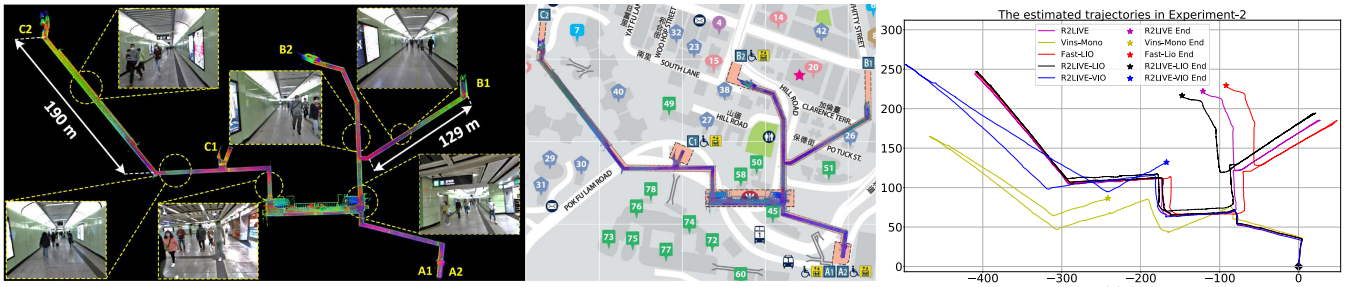


Fig. 6: Our handheld device for data collection, (a) shows our minimum system, with a total weight 2.09 Kg; (b) an additional D-GPS RTK system is used to evaluate the precision.



(a) (b) (c)

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 the MTR station. (c) Trajectory comparison among our system "R2LIVE", the LiDAR-inertial system "Fast-LIO" , and visual-inertial system "VINS-Mono". "R2LIVE-LIO" and "R2LIVE-VIO" are of the LiDAR-inertial and visual-inertial subsystem of our proposed system. 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.

Our handheld device for data collection is shown in Fig. 6 (a), which includes the power supply unit, the onboard DJI *manifold-2c*[7] computation platform (equipped with an *Intel i7-8550u* CPU and 8 GB RAM), a *FLIR Blackfly BFS-u3-13y3c* global shutter camera, and a *LiVOX AVIA*[8] LiDAR. The FoV of the camera is $82.9^\circ \times 66.5^\circ$ while the FoV of LiDAR is $70.4^\circ \times 77.2^\circ$. To quantitatively evaluate the precision of our algorithm (Section VI-E), we install a Differential-GPS (D-GPS) real-time kinematic (RTK) system[9] on our device, shown in Fig. 6 (b).

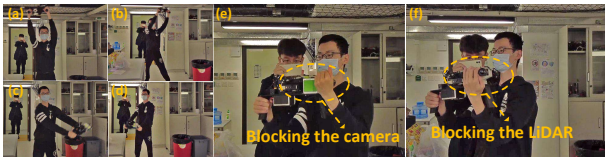


Fig. 7: We evaluate the robustness of our algorithm under aggressive motion (sequence in a∼d) and sensor failure by intentionally blocking the camera (e) and LiDAR sensor (f).

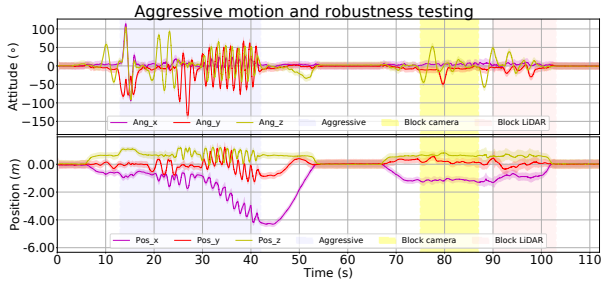


Fig. 8: Our estimated pose and its 3-σ bound with 5 times amplification for better visualization (the light-colored area around the trajectory) of Experiment-1. The shaded area in blue, yellow and red represent the phase of aggressive motion, camera-failure and LiDAR-failure, respectively. In all phases, the estimated trajectory can closely track the actual motion.

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

In this experiment, we evaluate the robustness of our algorithm in aggressive motion (see Fig. 7 (a∼d)), where 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 shade the different testing phases with different colors. It is seen that our estimated trajectory can tightly track the actual motion even in severe rotation and translation. Even when the camera or LiDAR is occasionally blocked, the estimated trajectory is sill very smooth and exhibits no noticeable degradation. These results show that our algorithm is robust to endure aggressive motion and sensor failures. We refer readers to the accompanying video on YouTube[10] showing details of the experiment.

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*[11]. It is a typical narrow tunnel-like environment (see the left figure of Fig. 9 (a)), with the maximum length reaching 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) easily fail the matching and tracking of 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. Due to the lack of GPS ground-truth, we align our point cloud data with the *HKU station street map*[12] to verify its long-term accuracy. As shown in Fig. 9 (b), our mapped point cloud matches the station street map tightly, demonstrating that our localization is of high robustness and accuracy. On the other hand, as shown in Fig. 9 (c), the

[7]https://www.dji.com/manifold-2
[8]https://www.livoxtech.com/avia
[9]https://www.dji.com/d-rtk
[10]https://youtu.be/9lqRHmlN_MA
[11]https://en.wikipedia.org/wiki/HKU_station
[12]https://www.mtr.com.hk/archive/en/services/maps/hku.pdf

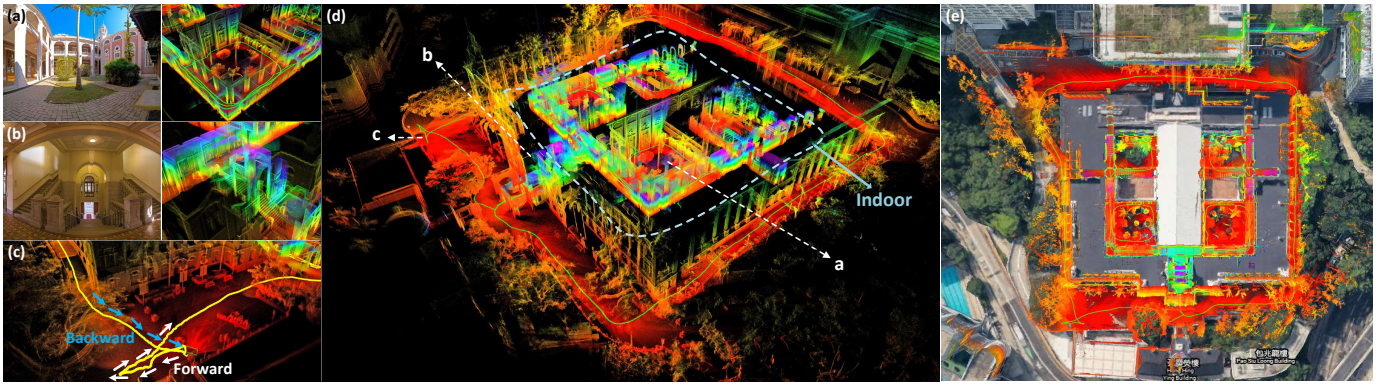


Fig. 10: The reconstructed 3D maps in Experiment-3 are shown in (d), and the detail point cloud with the correspondence panorama images are shown in (a) and (b). (c) shows that our algorithm can close the loop by itself (returning the starting point) without any additional processing (e.g. loop closure). In (e), we merge our map with the satellite image to further examine the accuracy of our system.

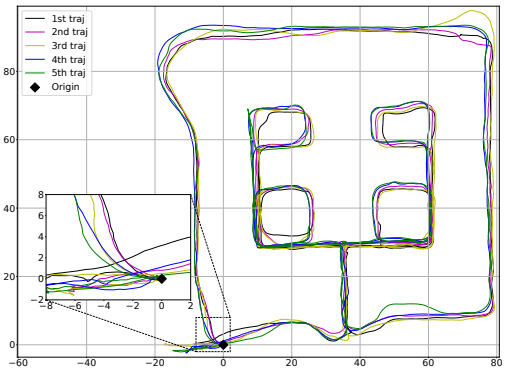


Fig. 11: Five different traveling trajectories in Experiments-3. Without any explicit loop closure, all of them can close the loop (i.e. return to zero) after traveling around 900 meters.

trajectory estimated by Vins-Mono[13], Fast-LIO[14], our LiDAR-inertial and visual-inertial subsystem significantly drift.

D. Experiment-3: High precision mapping large-scale indoor & outdoor urban environment

In this experiment, we show that our proposed method is able 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 5 times with slightly different traveling paths but all starting from and ending at the same location (see Fig. 11). What worth to be mentioned is, without any additional processing (i.e., loop detection), all of our 5 trajectories can still close the loop by themselves (see Fig. 10 (c) and Fig. 11) after traveling around 900 meters, which demonstrates that our proposed method is extremely low drift. The real-time reconstructed 3D maps along one path is shown in Fig. 10, in which the clear and high-quality point cloud demonstrates that our proposed method is of high accuracy. Finally, we overlay our map on the satellite image and find them match tightly (see Fig. 10 (e)).

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

In this experiment, we collect two fast-rotating (rotation speed reaching $130^\circ/s$ and $200^\circ/s$) sequences with a real-time Differential-GPS (D-GPS) Kinematic (RTK) system providing the ground-truth trajectory. We compare the trajectory estimated by VINS-Mono (IMU+Camera) [26], Fast-LIO (IMU+LiDAR) [5], Camvox (loosely-coupled LiDAR+Visual) [22] with the ground-truth in Fig. 12, it is seen that the trajectory estimated by our factor graph optimization agrees the best with the ground-truth trajectory in both sequences. To have more quantitative comparisons, we compute the relative rotation error (RPE) and relative translation error (RTE) [30] over all possible sub-sequences of length (50,100,150,...,300) meters are shown in Table. I.

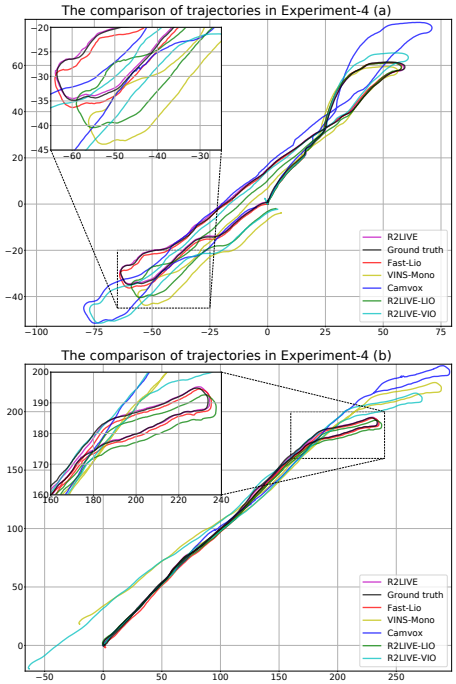


Fig. 12: Comparison results on two sequences with D-GPS groud-truth in Experiments-4.

F. Run time analysis

The average time consumption in experiment 1∼4 are listed in TABLE. II, which demonstrates that R^{2}LIVE can achieve real-time localization on both the desktop PC and embedded computation platform. Notice that the factor graph optimization runs on a separate thread and therefore is allowed to run at a lower rate.

[13]https://github.com/HKUST-Aerial-Robotics/VINS-Mono
[14]https://github.com/hku-mars/FAST_LIO/

TABLE I: The relative pose error in Experiment 4

Relative pose error in Experiments-4 (a)						
Sub-sequence RRE/RTE	50 m deg / %	100 m deg / %	150 m deg / %	200 m deg / %	250 m deg / %	300 m deg / %
VINS-Mono	**0.24** / 12.67	0.21 / 7.12	0.17 / 4.30	0.20 / 3.24	0.25 / 3.09	0.44 / 3.76
Fast-Lio	0.34 / 2.82	0.29 / 1.34	0.22 / 0.81	0.28 / 0.50	0.36 / 0.45	0.60 / 0.33
Camvox	0.67 / 27.53	0.57 / 14.42	0.31 / 5.67	0.28 / 4.05	0.83 / 3.74	0.71 / 3.26
R2LIVE-LIO	0.42 / 16.22	0.34 / 9.19	0.20 / 2.99	0.26 / 2.09	0.38 / 2.42	0.56 / 2.19
R2LIVE-VIO	0.39 / 11.28	0.32 / 6.10	0.22 / 2.56	0.29 / 2.45	0.37 / 2.12	0.56 / 2.02
R2LIVE-LC	0.45 / 24.73	0.54 / 23.05	0.68 / 11.54	0.56 / 11.72	0.50 / 10.62	0.27 / 7.89
R2LIVE	0.25 / **1.40**	**0.20 / 0.74**	**0.16 / 0.43**	**0.20 / 0.29**	**0.25 / 0.30**	**0.40 / 0.24**
Relative pose error in Experiments-4 (b)						
Sub-sequence RRE/RTE	50 m deg / %	100 m deg / %	150 m deg / %	200 m deg / %	250 m deg / %	300 m deg / %
VINS-Mono	0.39 / 13.19	0.50 / 12.33	0.55 / 3.61	0.34 / 4.49	0.50 / 1.92	0.47 / 2.77
Fast-Lio	0.09 / 1.39	0.08 / 1.00	0.08 / 0.56	0.08 / 0.26	0.09 / 0.25	0.09 / 0.37
Camvox	0.37 / 9.84	0.42 / 5.25	0.49 / 4.06	0.48 / 4.11	0.49 / 2.13	0.50 / 2.10
R2LIVE-LIO	0.13 / 2.11	0.12 / 1.58	0.12 / 1.13	0.13 / 0.67	0.15 / 0.39	0.16 / 0.14
R2LIVE-VIO	0.26 / 9.26	0.30 / 6.37	0.33 / 3.34	0.33 / 4.24	0.29 / 1.49	0.36 / 2.94
R2LIVE-LC	0.07 / 4.65	0.08 / 3.70	0.08 / 2.61	0.08 / 1.90	0.08 / 2.27	0.11 / 2.52
R2LIVE	**0.04 / 1.05**	**0.05 / 0.60**	**0.06 / 0.31**	**0.06 / 0.20**	**0.09 / 0.17**	**0.09 / 0.12**

TABLE II: The average running time of R2LIVE in Experiment-1∼4 (Exp-1∼4) on desktop PC (with Intel i7-9700K CPU and 32GB RAM) and on-board computer (with Intel i7-8550u CPU and 8GB RAM). The items "LI-Odom", "VI-Odom" and "FG-OPM" represent the average time for LiDAR-inertial fusion, Visual-Inertial fusion, and factor graph optimization, respectively, in our system.

PC/on-board	Exp-1 (ms)	Exp-2 (ms)	Exp-3 (ms)	Exp-4 (ms)
LI-Odom	8.81 / 15.98	11.54 / 25.30	14.91 / 30.64	10.92 / 24.37
VI-Odom	7.84 / 13.92	8.84 / 19.10	9.57 / 19.56	8.99 / 20.16
FG-OPM	26.10 / 45.35	30.20 / 65.25	29.25 / 58.08	27.98 / 60.43

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 measurements of LiDAR, inertial, and camera sensors within an error-state iterated Kalman filter and use the factor graph optimization to refine a sliding window of image keyframes and visual landmarks. Our system was tested in large-scale, indoor-outdoor, and narrow tunnel-like environments, challenging sensor failure and aggressive motion. In all the tests, our method achieves a high level of accuracy and robustness in localization and mapping. We also conducted various comparisons with other state-of-the-art visual-inertial, LiDAR-inertial, and LiDAR-visual SLAM whenever possible. Results show that our tightly fusion system achieves significantly higher accuracy and robustness.

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.

[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.

[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.

[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.

[9] C. Yuan, X. Liu, X. Hong, and F. Zhang, "Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments," *arXiv preprint arXiv:2103.01627*, 2021.

[10] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3126–3131.

[11] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9, 2014.

[12] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[13] 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)*. IEEE, 2018, pp. 4758–4765.

[14] J. Lin and F. Zhang, "A fast, complete, point cloud based loop closure for lidar odometry and mapping," *arXiv preprint arXiv:1909.11811*, 2019.

[15] 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.

[16] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," *arXiv preprint arXiv:2007.00258*, 2020.

[17] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-lidar-inertial odometry and mapping," *arXiv preprint arXiv:2010.13150*, 2020.

[18] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "Lins: A lidar-inertial state estimator for robust and efficient navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8899–8906.

[19] J. Zhang and S. Singh, "Laser–visual–inertial odometry and mapping with high robustness and low drift," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1242–1264, 2018.

[20] W. Shao, S. Vijayarangan, C. Li, and G. Kantor, "Stereo visual inertial lidar simultaneous localization and mapping," *arXiv preprint arXiv:1902.10741*, 2019.

[21] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense rgb-d-inertial slam with map deformations," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6741–6748.

[22] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "Camvox: A low-cost and accurate lidar-assisted visual slam system," *arXiv preprint arXiv:2011.11357*, 2020.

[23] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[24] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "Lic-fusion: Lidar-inertial-camera odometry," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5848–5854.

[25] X. Zuo, Y. Yang, J. Lv, Y. Liu, G. Huang, and M. Pollefeys, "Lic-fusion 2.0: Lidar-inertial-camera odometry with sliding-window plane-feature tracking," in *IROS 2020*, 2020.

[26] 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.

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

[28] D. He, W. Xu, and F. Zhang, "Embedding manifold structures into kalman filters," *arXiv preprint arXiv:2102.03804*, 2021.

[29] 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.

[30] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.

[31] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.

Supplementary Material: R^{2}LIVE: 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 $\delta\mathbf{r} \to \mathbf{0}$ on $SO(3)$ [31]:

$$\begin{aligned}
\mathtt{Exp}(\mathbf{r}+\delta\mathbf{r}) &\approx \mathtt{Exp}(\mathbf{r})\mathtt{Exp}(\mathbf{J}_r(\mathbf{r})\delta\mathbf{r}) \\
\mathtt{Exp}(\mathbf{r})\mathtt{Exp}(\delta\mathbf{r}) &\approx \mathtt{Exp}(\mathbf{r}+\mathbf{J}_r^{-1}(\mathbf{r})\delta\mathbf{r}) \\
\mathbf{R}\cdot\mathtt{Exp}(\delta\mathbf{r})\cdot\mathbf{u} &\approx \mathbf{R}\left(\mathbf{I}+[\delta\mathbf{r}]_\times\right)\mathbf{u} = \mathbf{R}\mathbf{u} - \mathbf{R}[\mathbf{u}]_\times\delta\mathbf{r}
\end{aligned}$$

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

$$\begin{aligned}
\mathbf{J}_r(\mathbf{r}) =& \mathbf{I} - \frac{1-\cos||\mathbf{r}||}{||\mathbf{r}||^2}[\mathbf{r}]_\times + \frac{||\mathbf{r}||-\sin(||\mathbf{r}||)}{||\mathbf{r}||^3}[\mathbf{r}]_\times^2 \\
\mathbf{J}_r^{-1}(\mathbf{r}) =& \mathbf{I} + \frac{1}{2}[\mathbf{r}]_\times + \left(\frac{1}{||\mathbf{r}||^2} - \frac{1+\cos(||\mathbf{r}||)}{2||\mathbf{r}||\sin(||\mathbf{r}||)}\right)[\mathbf{r}]_\times^2
\end{aligned}$$

B. Computation of $\mathbf{F}_{\delta\mathbf{x}}$ and $\mathbf{F}_{\mathbf{w}}$

Combing (8) and (11) , we have:

$$\begin{aligned}
&\delta\hat{\mathbf{x}}_{i+1} = \mathbf{x}_{i+1} \boxminus \hat{\mathbf{x}}_{i+1} \\
&= (\mathbf{x}_i \boxplus (\Delta t\cdot\mathbf{f}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{w}_i))) \boxminus (\hat{\mathbf{x}}_i \boxplus (\Delta t\cdot\mathbf{f}(\hat{\mathbf{x}}_i,\mathbf{u}_i,\mathbf{0}))) \\
&= \begin{bmatrix}
\mathbf{Log}\left(\left({}^G\hat{\mathbf{R}}_{I_i}\mathtt{Exp}(\hat{\boldsymbol{\omega}}_i\Delta t)\right)^T\cdot\left({}^G\hat{\mathbf{R}}_{I_i}\mathtt{Exp}\left({}^G\delta\hat{\mathbf{r}}_{I_i}\right)\mathtt{Exp}(\boldsymbol{\omega}_i\Delta t)\right)\right) \\
{}^G\delta\hat{\mathbf{p}}_{I_i} + {}^G\delta\hat{\mathbf{v}}_i\Delta t \\
{}^I\delta\hat{\mathbf{r}}_{C_i} \\
{}^I\delta\hat{\mathbf{p}}_{C_i} \\
{}^G\delta\hat{\mathbf{v}}_i + \left({}^G\hat{\mathbf{R}}_{I_i}\mathtt{Exp}\left({}^G\delta\hat{\mathbf{r}}_{I_i}\right)\right)\mathbf{a}_i\Delta t - {}^G\hat{\mathbf{R}}_{I_i}\hat{\mathbf{a}}_i\Delta t \\
\delta\hat{\mathbf{b}}_{\mathbf{g}_i} + \mathbf{n}_{\mathbf{b}_{\mathbf{g}_i}}\Delta t \\
\delta\hat{\mathbf{b}}_{\mathbf{a}_i} + \mathbf{n}_{\mathbf{b}_{\mathbf{a}_i}}\Delta t
\end{bmatrix}
\end{aligned}$$

with:

$$\hat{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_{m_i} - \hat{\mathbf{b}}_{\mathbf{g}_i},\ \boldsymbol{\omega}_i = \hat{\boldsymbol{\omega}}_i - \delta\hat{\mathbf{b}}_{\mathbf{g}_i} - \mathbf{n}_{\mathbf{g}_i} \tag{S1}$$

$$\hat{\mathbf{a}}_i = \mathbf{a}_{m_i} - \hat{\mathbf{b}}_{\mathbf{a}_i},\ \mathbf{a}_i = \hat{\mathbf{a}}_i - \delta\hat{\mathbf{b}}_{\mathbf{a}_i} - \mathbf{n}_{\mathbf{a}_i} \tag{S2}$$

And we have the following simplification and approximation from Section. A.

$$\begin{aligned}
&\mathtt{Log}\left(\left({}^G\hat{\mathbf{R}}_{I_i}\mathtt{Exp}(\hat{\boldsymbol{\omega}}_i\Delta t)\right)^T\cdot\left({}^G\hat{\mathbf{R}}_{I_i}\mathtt{Exp}\left({}^G\delta\hat{\mathbf{r}}_{I_i}\right)\mathtt{Exp}(\boldsymbol{\omega}_i\Delta t)\right)\right) \\
=&\mathtt{Log}\left(\mathtt{Exp}(\hat{\boldsymbol{\omega}}_i\Delta t)^T\cdot\left(\mathtt{Exp}\left({}^G\delta\hat{\mathbf{r}}_{I_i}\right)\cdot\mathtt{Exp}(\boldsymbol{\omega}_i\Delta t)\right)\right) \\
\approx&\mathtt{Log}\left(\mathtt{Exp}(\hat{\boldsymbol{\omega}}_i\Delta t)^T\,\mathtt{Exp}\left({}^G\delta\hat{\mathbf{r}}_{I_i}\right)\mathtt{Exp}(\hat{\boldsymbol{\omega}}_i\Delta t)\cdot\right. \\
&\qquad\left.\mathtt{Exp}\left(-\mathbf{J}_r(\hat{\boldsymbol{\omega}}_i\Delta t)\left(\delta\hat{\mathbf{b}}_{g_i}+\mathbf{n}_{\mathbf{g}_i}\right)\Delta t\right)\right) \\
\approx&\mathtt{Exp}(\hat{\boldsymbol{\omega}}_i\Delta t)\cdot{}^G\delta\hat{\mathbf{r}}_{I_i} - \mathbf{J}_r(\hat{\boldsymbol{\omega}}_i\Delta t)\delta\hat{\mathbf{b}}_{\mathbf{g}_i}\Delta t - \mathbf{J}_r(\hat{\boldsymbol{\omega}}_i\Delta t)\mathbf{n}_{\mathbf{g}_i}\Delta t \\
&\left({}^G\hat{\mathbf{R}}_{I_i}\mathtt{Exp}\left({}^G\delta\hat{\mathbf{r}}_{I_i}\right)\right)\mathbf{a}_i\Delta t \\
\approx&\left({}^G\hat{\mathbf{R}}_{I_i}\left(\mathbf{I}+[{}^G\delta\hat{\mathbf{r}}_{I_i}]_\times\right)\right)\left(\hat{\mathbf{a}}_i-\delta\hat{\mathbf{b}}_{\mathbf{a}_i}-\mathbf{n}_{\mathbf{a}_i}\right)\Delta t \\
\approx&{}^G\hat{\mathbf{R}}_{I_i}\hat{\mathbf{a}}_i\Delta t - {}^G\hat{\mathbf{R}}_{I_i}\delta\hat{\mathbf{b}}_{\mathbf{a}_i}\Delta t - {}^G\hat{\mathbf{R}}_{I_i}\mathbf{n}_{\mathbf{a}_i}\Delta t - {}^G\hat{\mathbf{R}}_{I_i}[\hat{\mathbf{a}}_i]_\times{}^G\delta\hat{\mathbf{r}}_{I_i}\Delta t
\end{aligned}$$

To conclude, we have the computation of $\mathbf{F}_{\delta\hat{\mathbf{x}}}$ and $\mathbf{F}_{\mathbf{w}}$ as follow:

$$\begin{aligned}
\mathbf{F}_{\delta\hat{\mathbf{x}}} =& \left.\frac{\partial(\delta\hat{\mathbf{x}}_{i+1})}{\partial\delta\hat{\mathbf{x}}_i}\right|_{\delta\hat{\mathbf{x}}_i=\mathbf{0},\mathbf{w}_i=\mathbf{0}} \\
=&\begin{bmatrix}
\mathtt{Exp}(-\hat{\boldsymbol{\omega}}_i\Delta t) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{J}_r(\hat{\boldsymbol{\omega}}_i\Delta t)\Delta t & \mathbf{0} \\
\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I}\Delta t & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
-{}^G\hat{\mathbf{R}}_{I_i}[\hat{\mathbf{a}}_i]_\times\Delta t & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & -{}^G\hat{\mathbf{R}}_{I_i}\Delta t \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}
\end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
\mathbf{F}_{\mathbf{w}} =& \left.\frac{\partial(\delta\hat{\mathbf{x}}_{i+1})}{\partial\mathbf{w}_i}\right|_{\delta\hat{\mathbf{x}}_i=\mathbf{0},\mathbf{w}_i=\mathbf{0}} \\
=&\begin{bmatrix}
-\mathbf{J}_r(\hat{\boldsymbol{\omega}}_i\Delta t)\Delta t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & -{}^G\hat{\mathbf{R}}_{I_i}\Delta t & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{I}\Delta t & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\Delta t
\end{bmatrix}
\end{aligned}$$

C. The computation of $\boldsymbol{\mathcal{H}}$

Recalling (21), we have:

$$\begin{aligned}
\boldsymbol{\mathcal{H}} =& \left.\frac{(\check{\mathbf{x}}_{k+1}\boxplus\delta\check{\mathbf{x}}_{k+1})\boxminus\hat{\mathbf{x}}_{k+1}}{\partial\delta\check{\mathbf{x}}_{k+1}}\right|_{\delta\check{\mathbf{x}}_{k+1}=\mathbf{0}} \\
=&\begin{bmatrix}
\mathbf{A} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}_{3\times9} \\
\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}_{3\times9} \\
\mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{0} & \mathbf{0}_{3\times9} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0}_{3\times9} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{9\times9}
\end{bmatrix}
\end{aligned}$$

with the 3×3 matrix $\mathbf{A} = \mathbf{J}_r^{-1}(\mathtt{Log}({}^G\hat{\mathbf{R}}_{I_{k+1}}{}^T{}^G\check{\mathbf{R}}_{I_{k+1}}))$ and $\mathbf{B} = \mathbf{J}_r^{-1}(\mathtt{Log}({}^I\hat{\mathbf{R}}_{C_{k+1}}{}^T{}^I\check{\mathbf{R}}_{C_{k+1}}))$.

D. The computation of $\mathbf{H}_j^l$

Recalling (17) and (21), we have:

$$\begin{aligned}
&\mathbf{r}_l(\check{\mathbf{x}}_{k+1}\boxplus\delta\check{\mathbf{x}}_{k+1},{}^L\mathbf{p}_j) = \mathbf{u}_j^T\left({}^G\check{\mathbf{p}}_{I_{k+1}} + {}^G\delta\check{\mathbf{p}}_{I_{k+1}} - \right. \\
&\left.\mathbf{q}_j + {}^G\check{\mathbf{R}}_{I_{k+1}}\mathtt{Exp}({}^G\delta\check{\mathbf{r}}_{I_{k+1}})\left({}^I\mathbf{R}_L{}^L\mathbf{p}_j + {}^I\mathbf{p}_L\right)\right)
\end{aligned} \tag{S3}$$

And with the small perturbation approximation, we get:

$$\begin{aligned}
&{}^G\check{\mathbf{R}}_{I_{k+1}}\mathtt{Exp}({}^G\delta\check{\mathbf{r}}_{I_{k+1}})\mathbf{P}_{\mathbf{a}} \\
\approx&{}^G\check{\mathbf{R}}_{I_{k+1}}\left(\mathbf{I}+\left[{}^G\delta\check{\mathbf{r}}_{I_{k+1}}\right]_\times\right)\mathbf{P}_{\mathbf{a}} \\
=&{}^G\check{\mathbf{R}}_{I_{k+1}}\mathbf{P}_{\mathbf{a}} - {}^G\check{\mathbf{R}}_{I_{k+1}}[\mathbf{P}_{\mathbf{a}}]_\times{}^G\delta\check{\mathbf{r}}_{I_{k+1}}
\end{aligned} \tag{S4}$$

where $\mathbf{P}_{\mathbf{a}} = {}^I\mathbf{R}_L{}^L\mathbf{p}_j + {}^I\mathbf{p}_L$. Combining (S3) and (S4) together we can obtain:

$$\mathbf{H}_j^l = \mathbf{u}_j^T\begin{bmatrix}-{}^G\check{\mathbf{R}}_{I_{k+1}}[\mathbf{P}_{\mathbf{a}}]_\times & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times15}\end{bmatrix}$$

E. The computation of $\mathbf{H}_s^c$ and $\mathbf{F}_{\mathbf{P}_s}$

Recalling (23), we have:

$${}^C\mathbf{P}_s = \mathbf{P}_{\mathbf{C}}(\check{\mathbf{x}}_{k+1},{}^G\mathbf{P}_s) = \begin{bmatrix}{}^CP_{sx} & {}^CP_{sy} & {}^CP_{sz}\end{bmatrix}^T$$

where the function $\mathbf{P}_{\mathbf{C}}(\check{\mathbf{x}}_{k+1},{}^G\mathbf{P}_s)$ is:

$${}^C\mathbf{P}_s = \left({}^G\check{\mathbf{R}}_{I_{k+1}}{}^I\check{\mathbf{R}}_{C_{k+1}}\right)^T{}^G\mathbf{P}_s - {}^I\check{\mathbf{R}}_{C_{k+1}}^T{}^I\check{\mathbf{p}}_{C_{k+1}} \tag{S5}$$

$$-\left({}^G\check{\mathbf{R}}_{I_{k+1}}{}^I\check{\mathbf{R}}_{C_{k+1}}\right)^T{}^G\check{\mathbf{p}}_{I_{k+1}} \tag{S6}$$

From (28), we have:

$$\begin{aligned}
&\mathbf{r}_c\left(\check{\mathbf{x}}_{k+1},{}^C\mathbf{p}_s,{}^G\mathbf{P}_s\right) = {}^C\mathbf{p}_s - \boldsymbol{\pi}({}^C\mathbf{P}_s) \\
&\boldsymbol{\pi}({}^C\mathbf{P}_s) = \begin{bmatrix}f_x\dfrac{{}^CP_{sx}}{{}^CP_{sz}} + c_x & f_y\dfrac{{}^CP_{sy}}{{}^CP_{sz}} + c_y\end{bmatrix}^T
\end{aligned} \tag{S7}$$

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 $(\cdot)|_{\delta\check{\mathbf{x}}_{k+1}=\mathbf{0}}$ in the following derivation, and we have:

$$\mathbf{H}_s^c = -\frac{\partial \boldsymbol{\pi}({}^C\mathbf{P}_s)}{\partial {}^C\mathbf{P}_s} \cdot \frac{\partial \mathbf{P_C}(\check{\mathbf{x}}_{k+1} \boxplus \delta\check{\mathbf{x}}_{k+1}, {}^G\mathbf{P}_s)}{\partial \delta\check{\mathbf{x}}_{k+1}} \tag{S8}$$

$$\mathbf{F}_{\mathbf{P}_s} = -\frac{\partial \boldsymbol{\pi}({}^C\mathbf{P}_s)}{\partial {}^C\mathbf{P}_s} \cdot \frac{\partial \mathbf{P_C}(\check{\mathbf{x}}_{k+1}, {}^G\mathbf{P}_s)}{\partial {}^G\mathbf{P}_s} \tag{S9}$$

where:

$$\frac{\partial \boldsymbol{\pi}({}^C\mathbf{P}_s)}{\partial {}^C\mathbf{P}_s} = \frac{1}{{}^C P_{sz}} \begin{bmatrix} f_x & 0 & -f_x \dfrac{{}^C P_{sx}}{{}^C P_{sz}} \\ 0 & f_y & -f_y \dfrac{{}^C P_{sy}}{{}^C P_{sz}} \end{bmatrix} \tag{S10}$$

$$\frac{\partial \mathbf{P_b}(\check{\mathbf{x}}_{k+1}, {}^G\mathbf{P}_s)}{\partial {}^G\mathbf{P}_s} = \left({}^G\check{\mathbf{R}}_{I_{k+1}} {}^I\check{\mathbf{R}}_C\right)^T \tag{S11}$$

According to Section. A, we have the following approximation of $\mathbf{P_C}(\check{\mathbf{x}}_{k+1} \boxplus \delta\check{\mathbf{x}}_{k+1}, {}^G\mathbf{P}_s)$:

$$\begin{aligned}
&\mathbf{P_C}(\check{\mathbf{x}}_{k+1} \boxplus \delta\check{\mathbf{x}}_{k+1}, {}^G\mathbf{P}_s) \\
&= \left({}^G\check{\mathbf{R}}_{I_{k+1}} \mathtt{Exp}\left({}^G\delta\check{\mathbf{r}}_{I_{k+1}}\right) {}^I\check{\mathbf{R}}_{C_{k+1}} \mathtt{Exp}\left({}^I\delta\check{\mathbf{r}}_{C_{k+1}}\right)\right)^T {}^G\mathbf{P}_s \\
&- \left({}^I\check{\mathbf{R}}_{C_{k+1}} \mathtt{Exp}\left({}^I\delta\check{\mathbf{r}}_{C_{k+1}}\right)\right)^T \left({}^I\check{\mathbf{p}}_{C_{k+1}} + {}^I\delta\check{\mathbf{p}}_{C_{k+1}}\right) \\
&- \left({}^G\check{\mathbf{R}}_{I_{k+1}} \mathtt{Exp}\left({}^G\delta\check{\mathbf{r}}_{I_{k+1}}\right) {}^I\check{\mathbf{R}}_{C_{k+1}} \mathtt{Exp}\left({}^I\delta\check{\mathbf{r}}_{C_{k+1}}\right)\right)^T \left({}^G\check{\mathbf{p}}_{I_{k+1}} + {}^G\delta\check{\mathbf{p}}_{I_{k+1}}\right) \\
&\approx \mathbf{P_b}(\check{\mathbf{x}}_{k+1}, {}^G\mathbf{P}_s) - \left({}^I\check{\mathbf{R}}_{C_{k+1}}\right)^T \left({}^G\check{\mathbf{R}}_{I_{k+1}}\right)^T {}^G\delta\check{\mathbf{p}}_{I_{k+1}} \\
&+ \left({}^I\check{\mathbf{R}}_{C_{k+1}}\right)^T \left[{}^G\check{\mathbf{R}}_{I_{k+1}}^T ({}^G\mathbf{P}_s - {}^G\check{\mathbf{p}}_{I_{k+1}})\right]_\times {}^G\delta\check{\mathbf{r}}_{I_{k+1}} \\
&+ \left[\mathbf{P_b}(\check{\mathbf{x}}_{k+1}, {}^G\mathbf{P}_s)\right]_\times {}^I\delta\check{\mathbf{r}}_{C_{k+1}} - \left({}^I\check{\mathbf{R}}_C\right)^T {}^I\delta\check{\mathbf{p}}_{C_{k+1}}
\end{aligned} \tag{S12}$$

With this, we can derive:

$$\frac{\partial \mathbf{P_C}(\check{\mathbf{x}}_{k+1} \boxplus \delta\check{\mathbf{x}}_{k+1}, {}^G\mathbf{P}_s)}{\partial \delta\check{\mathbf{x}}_{k+1}} = \begin{bmatrix} \mathbf{M_A} & \mathbf{M_B} & \mathbf{M_C} & \mathbf{M_D} & \mathbf{0}_{3\times 12} \end{bmatrix} \tag{S13}$$

$$\begin{aligned}
\mathbf{M_A} &= \left({}^I\check{\mathbf{R}}_{C_{k+1}}\right)^T \left[{}^G\check{\mathbf{R}}_{I_{k+1}}^T ({}^G\mathbf{P}_s - {}^G\check{\mathbf{p}}_{I_{k+1}})\right]_\times \\
\mathbf{M_B} &= -\left({}^I\check{\mathbf{R}}_{C_{k+1}}\right)^T \left({}^G\check{\mathbf{R}}_{I_{k+1}}\right)^T \\
\mathbf{M_C} &= \left[{}^C\mathbf{P}_s\right]_\times \\
\mathbf{M_D} &= -\left({}^I\check{\mathbf{R}}_C\right)^T
\end{aligned} \tag{S14}$$

Substituting (S10), (S11) and (S13) into (S8) and (S9), we finish the computation of $\mathbf{H}_s^c$ and $\mathbf{F}_{\mathbf{P}_s}$.