本科毕业设计（论文）

外文翻译

题目：面向轻量化定位与地图构建方法研究

学　　院	信息科学与工程学院
专　　业	计算机科学与技术
班　　级	22 计算机一班
学　　号	226002618
学生姓名	李杰
指导教师	彭成斌
开题日期	2025 年 12 月 25 日

<文献翻译一：原文>

OpenVINS：A Research Platform for Visual-Inertial Estimation[1]

Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang

Abstract— In this paper, we present an open platform, termed OpenVINS, for visual-inertial estimation research for both the academic community and practitioners from industry. The open sourced codebase provides a foundation for researchers and engineers to quickly start developing new capabilities for their visual-inertial systems. This codebase has out of the box support for commonly desired visual-inertial estimation features, which include: (i) on-manifold sliding window Kalman filter, (ii) online camera intrinsic and extrinsic calibration, (iii) camera to inertial sensor time offset calibration, (iv) SLAM landmarks with different representations and consistent First-Estimates Jacobian (FEJ) treatments, (v) modular type system for state management, (vi) extendable visual-inertial system simulator, and (vii) extensive toolbox for algorithm evaluation. Moreover, we have also focused on detailed documentation and theoretical derivations to support rapid development and research, which are greatly lacked in the current open sourced algorithms. Finally, we perform comprehensive validation of the proposed OpenVINS against state-of-the-art open sourced algorithms, showing its competing estimation performance.

• Open source: https://github.com/rpng/open_vins

• Documentation: https://docs.openvins.com

I. INTRODUCTION

Autonomous robots and consumer-grade mobile devices such as drones and smartphones are becoming ubiquitous, in part due to a large increase in computing ability and a simultaneous reduction in power consumption and cost. To endow these robots and mobile devices with the ability to perceive and understand their contextual locations within local environments, which is desired in many different applications from mobile AR/VR to autonomous navigation, visual-inertial navigation systems (VINS) are often used to provide accurate motion estimates by fusing the data from on-board camera and inertial sensors [1].

[1] Geneva P, Eckenhoff K, Lee W, et al. OpenVINS: A Research Platform for Visual-Inertial Estimation[C]. 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020: 4666-4672.

Developing a working VINS algorithm from scratch has proven to be challenging, and in the robotics research community, this has shown to be a significant hurdle for researchers due to the lack of VINS codebases that have comprehensive documentation and detailed derivations for which even users with little background can learn and extend a current state-of-the-art work to address their problems at hand. While there are several open sourced visual-inertial codebases [2]–[8], they are not developed for extensibility and lack proper documentation and evaluation tools, which, in our experience, are crucial for rapid development and deep understanding, thus accelerating VINS research and development in the field. Moreover, these systems have many hard-coded assumptions or features that require an intricate understanding of the codebases in order to adapt them to the sensor systems at hand. This, along with inadequate documentation and support, limits their wide adoption in different applications.

To fill the aforementioned void in the community and to promote the VINS research in robotics and beyond, in this paper, we present an extendable, open sourced codebase that is particularly designed for researchers and practitioners with either limited or extensive background knowledge of state estimation. We provide the necessary documentation, tools, and theory for those who are even new to visual-inertial estimation, and term this collection of utilities as OpenVINS (OV). This codebase has been the foundation of many of the recent visual-inertial estimation projects in our group at the University of Delaware, which include multi-camera [9], multi-IMU [10], visual-inertial moving object tracking [11], [12], Schmidt-based visual-inertial SLAM [13], [14], point-plane and point-line visual-inertial navigation [15], [16], among others [17]–[19]. We summarize the key functionality of the different components in OpenVINS as follows:

• ov_core – Contains 2D image sparse visual feature tracking; linear and Gauss-Newton feature triangulation methods; visual-inertial simulator for arbitrary number of cameras and frequencies; and fundamental manifold math operations and utilities.

• ov_eval – Contains trajectory alignment; plotting utilities for trajectory accuracy and consistency evaluation; Monte-Carlo evaluation of different accuracy metrics; and utility for recording ROS topics to file.

• ov_msckf – Contains the extendable modular Extended Kalman Filter (EKF)-based sliding window visual-inertial estimator with on-manifold type system for flexible state representa-

tion. Features include: First-Estimates Jacobians (FEJ) [20]–[22], IMU-camera time offset calibration [23], camera intrinsics and extrinsic online calibration [24], standard MSCKF [25], and 3D SLAM landmarks of different representations.

In what follows we describe our generalized modular on-manifold EKF-based estimator which, in its simplest form, estimates the current state of a camera-IMU pair. We then introduce the implemented features that provide the foundation for researchers to quickly build and extend on. Note that what we present here is only a brief introduction to the feature set and readers are referred to our thorough documentation website. We also provide an evaluation of the proposed EKF-based solution in simulations and then on real-world datasets, clearly demonstrating its competing performance against other open sourced algorithms.

II. ON-MANIFOLD MODULAR EKF

The state vector of our visual-inertial system consists of the current inertial navigation state, a set of c historical IMU pose clones, a set of m environmental landmarks, and a set of w cameras' extrinsic and intrinsic parameters.

$$\mathbf{x}_k = [\mathbf{x}_I^\top \quad \mathbf{x}_C^\top \quad \mathbf{x}_M^\top \quad \mathbf{x}_W^\top \quad {}^{C}t_I]^\top \tag{1}$$

$$\mathbf{x}_I = \left[{}^{I_k}_{G}\bar{q}^\top \quad {}^{G}\mathbf{p}_{I_k}^\top \quad {}^{G}\mathbf{v}_{I_k}^\top \quad \mathbf{b}_{\omega_k}^\top \quad \mathbf{b}_{a_k}^\top \right]^\top \tag{2}$$

$$\mathbf{x}_C = \left[{}^{I_{k-1}}_{G}\bar{q}^\top \quad {}^{G}\mathbf{p}_{I_{k-1}}^\top \quad \cdots \quad {}^{I_{k-c}}_{G}\bar{q}^\top \quad {}^{G}\mathbf{p}_{I_{k-c}}^\top \right]^\top \tag{3}$$

$$\mathbf{x}_M = \left[{}^{G}\mathbf{p}_{f_1}^\top \quad \cdots \quad {}^{G}\mathbf{p}_{f_m}^\top \right]^\top \tag{4}$$

$$\mathbf{x}_W = \left[{}^{I}_{C_1}\bar{q}^\top \quad {}^{C_1}\mathbf{p}_I^\top \quad \boldsymbol{\zeta}_0^\top \cdots \quad {}^{I}_{C_w}\bar{q}^\top \quad {}^{C_w}\mathbf{p}_I^\top \quad \boldsymbol{\zeta}_w^\top \right]^\top \tag{5}$$

where ${}^{I_k}_{G}\bar{q}$ is the unit quaternion parameterizing the rotation $\mathbf{R}\left({}^{I_k}_{G}\bar{q}\right) = {}^{I_k}_{G}\mathbf{R}$ from the global frame of reference $\{G\}$ to the IMU local frame $\{I_k\}$ at time k[26], $\mathbf{b}_\omega$ and $\mathbf{b}_a$ are the gyroscope and accelerometer biases, and ${}^{G}\mathbf{v}_{I_k}$ and ${}^{G}\mathbf{p}_{I_k}$ are the velocity and position of the IMU expressed in the global frame, respectively. The inertial state $\mathbf{x}_I$ lies on the manifold defined by the product of the unit quaternions $\mathbb{H}$ with the vector space $\mathbb{R}^{12}$ (i.e. $\mathcal{M} = \mathbb{H} \times \mathbb{R}^{12}$) and has 15 total degrees of freedom (DOF).

For vector variables, the "boxplus" and "boxminus" operations, which map elements to and from a given manifold [27], equate to simple addition and subtraction of their vectors. For quaternions, we define the quaternion boxplus operation as:

$$\bar{q}_1 \boxplus \delta\boldsymbol{\theta} \triangleq \begin{bmatrix} \frac{\delta\boldsymbol{\theta}}{2} \\ 1 \end{bmatrix} \otimes \bar{q}_1 \simeq \bar{q}_2 \tag{6}$$

Note that although we have defined the orientations using the left quaternion error, it is not limited to this and any onmanifold representation in practice can be used (e.g., [28]).

The map of environmental landmarks $\mathbf{x}_M$ contains global 3D positions only for simplicity, while in practice we offer support for different representations (e.g. inverse MSCKF [25], full inverse depth [29], and anchored 3D position [30]).

The calibration vector $\mathbf{x}_W$ contains the camera intrinsics $\boldsymbol{\zeta}$, consisting of focal length, camera center, and distortion parameters, and the camera-IMU extrinsics, i.e., the spatial transformation (relative pose) from the IMU to each camera. Since we consider synchronized camera clocks, we include a single time offset ${}^{C}t_I$ between the IMU and the camera clock in the calibration vector.

A. Propagation

The inertial state $\mathbf{x}_I$ is propagated forward using incoming IMU measurements of linear accelerations ${}^{I}\mathbf{a}_m$ and angular velocities ${}^{I}\boldsymbol{\omega}_m$ based on the following generic nonlinear IMU kinematics propagating the state from timestep $k-1$ to k [31]:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, {}^{I}\mathbf{a}_m, {}^{I}\boldsymbol{\omega}_m, \mathbf{n}) \tag{7}$$

where $\mathbf{n}$ contains the zero-mean white Gaussian noise of the IMU measurements along with random walk bias noise. This state estimate is evaluated at the current estimate:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, {}^{I}\mathbf{a}_m, {}^{I}\boldsymbol{\omega}_m, \mathbf{0}) \tag{8}$$

where $\cdot$ denotes the estimated value and the subscript $k \mid k-1$ denotes the predicted estimate at time k given the measurements up to time $k-1$. The state covariance matrix is propagated typically by linearizing the nonlinear model at the current estimate:

$$\mathbf{P}_{k|k-1} = \boldsymbol{\Phi}_{k-1}\mathbf{P}_{k-1|k-1}\boldsymbol{\Phi}_{k-1}^{\top} + \mathbf{Q}_{k-1} \tag{9}$$

where $\boldsymbol{\Phi}_{k-1}$ and $\mathbf{Q}_{k-1}$ are respectively the system Jacobian and discrete noise covariance matrices [25]. The clones $\mathbf{x}_C$, environmental features $\mathbf{x}_M$, and calibration $\mathbf{x}_W$ states do not evolve with time and thus the corresponding state Jacobian entries are identity with zero propagation noise and allow for exploitation of the sparsity for computational savings.

B. On-Manifold Update

Consider the following nonlinear measurement function:

$$\mathbf{z}_{m,k} = h(\mathbf{x}_k) + \mathbf{n}_{m,k} \tag{10}$$

where we have the measurement noise $\mathbf{n}_{m,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{m,k})$. For the standard EKF update, one linearizes the above equation at the current state estimate. In our case, as in the indirect EKF [26], we linearize (10) with respect to the current zero-mean error state (i.e. $\tilde{\mathbf{x}} = \mathbf{x} \boxminus \hat{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$):

$$\mathbf{z}_{m,k} = h(\hat{\mathbf{x}}_{k|k-1} \boxplus \tilde{\mathbf{x}}_{k|k-1}) + \mathbf{n}_{m,k} \tag{11}$$
$$= h(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}_k \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_{m,k} \tag{12}$$
$$\Rightarrow \tilde{\mathbf{z}}_{m,k} = \mathbf{H}_k \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_{m,k} \tag{13}$$

where $\mathbf{H}_k$ is the measurement Jacobian computed as follows:

$$\mathbf{H}_k = \left. \frac{\partial h(\hat{\mathbf{x}}_{k|k-1} \boxplus \tilde{\mathbf{x}}_{k|k-1})}{\partial \tilde{\mathbf{x}}_{k|k-1}} \right|_{\tilde{\mathbf{x}}_{k|k-1}=\mathbf{0}} \tag{14}$$

Using this linearized measurement model, we can now perform the following standard EKF update to ensure the updated states remain on-manifold:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} \boxplus \mathbf{K}_k \left(\mathbf{z}_{m,k} - h(\hat{\mathbf{x}}_{k|k-1}) \right) \tag{15}$$
$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1} \tag{16}$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \left(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_{m,k} \right)^{-1} \tag{17}$$

III. OPENVINS RESEARCH PLATFORM

A. Type-based Index System

At the core of the OpenVINS library is the type-based index system. Inspired by graph-based optimization frameworks such as GTSAM [32], we abstract away from the user the need to directly manipulate the covariance and instead provide the tools to automatically manage the state and its covariance. This offers many benefits such as reduced implementation time and being less prone to development errors due to explicit state and covariance access.

Each state variable "type" has internally the location of where it is in the error state which is automatically updated during initialization, cloning, or marginalization operations which affect variable ordering. A type is defined by its covariance location, its current estimate and its error state size. The current value does not have to be a vector, but could be a matrix in the case of an $\mathbb{SO}(3)$ rotation representation. The error state for all types is a vector and thus a type will need to define the boxplus mapping between its error state and its manifold representation (i.e. the update function).

```
class Type {
protected:
    // Current best estimate
    Eigen :: MatrixXd _value;
    // Index of error state in covariance
    int _id = -1;
    // Dimension of error state
    int _size = -1;
    // Vector correction, how to update
    void update (const Eigen ::VectorXd dx);
};
```

One of the main advantages of this type system is that it reduces the complexity of adding new features by allowing the user to construct sparse Jacobians. Instead of constructing a Jacobian for all state elements, the "sparse" Jacobian needs to only include the state elements that the measurement is a function of. This both saves computation in the cases where a measurement is a function of only a few state elements and allows for measurement functions to be state agnostic as long as their involved state variables are present.

B. State Variable Initialization

Based on a set of linearized measurement equations (13), we aim to optimally compute the initial estimate of a new state variable and its covariance and correlations with the existing state variables. As a motivating example, we here describe how to initialize a new SLAM landmark ${}^{G}\mathbf{p}_f$, whose key logic can be used for any new state variable and is generalized to any type within the codebase. As in [33] we first perform QR decomposition (e.g., using computationally efficient in-place Givens rotations) to separate the linear system (13) into two subsystems: (i) one that depends on the new state (i.e., ${}^{G}\mathbf{p}_f$), and (ii) the other that does not.

$$\tilde{\mathbf{z}}_{m,k} = [\mathbf{H}_x \quad \mathbf{H}_f] \begin{bmatrix} \tilde{\mathbf{x}}_k \\ {}^{G}\tilde{\mathbf{p}}_f \end{bmatrix} + \mathbf{n}_{m,k} \tag{18}$$

$$\Rightarrow \begin{bmatrix} \tilde{\mathbf{z}}_{m1,k} \\ \tilde{\mathbf{z}}_{m2,k} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{x1} & \mathbf{H}_{f1} \\ \mathbf{H}_{x2} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_k \\ {}^{G}\tilde{\mathbf{p}}_f \end{bmatrix} + \begin{bmatrix} \mathbf{n}_{f1} \\ \mathbf{n}_{f2} \end{bmatrix} \tag{19}$$

where $\mathbf{n}_{fi} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{fi}), i \in \{1,2\}$. Note that in the above expression $\tilde{\mathbf{z}}_{m1,k}$ and $\tilde{\mathbf{z}}_{m2,k}$ are orthonormally transformed measurement residuals, not the direct partitions of $\tilde{\mathbf{z}}_{m,k}$. With the top transformed linearized measurement residual $\tilde{\mathbf{z}}_{m1,k}$ in (19), we now perform efficient EKF update to initialize the state estimate of ${}^{G}\hat{\mathbf{p}}_{f}$ and its covariance and correlations to $\mathbf{x}_k$ [see (15)], which will then be augmented to the current state and covariance matrix.

$$
\begin{aligned}
{}^{G}\hat{\mathbf{p}}_{f} &= {}^{G}\hat{\mathbf{p}}_{f} \boxplus \mathbf{H}_{f1}^{-1}\tilde{\mathbf{z}}_{m1,k} && (20)\\
\mathbf{P}_{xf} &= -\mathbf{P}_{k}\mathbf{H}_{x1}^{\top}\mathbf{H}_{f1}^{-\top} && (21)\\
\mathbf{P}_{ff} &= \mathbf{H}_{f1}^{-1}\left(\mathbf{H}_{x1}\mathbf{P}_{k}\mathbf{H}_{x1}^{\top} + \mathbf{R}_{f1}\right)\mathbf{H}_{f1}^{-\top} && (22)
\end{aligned}
$$

It should be noted that a full-rank $\mathbf{H}_{f1}$ is needed to perform the above initialization, which normally is the case if enough measurements are collected (i.e., delayed initialization). Note also that to utilize all available measurement information, we also perform EKF update using the bottom measurement residual $\tilde{\mathbf{z}}_{m2,k}$ in (19), which essentially is equivalent to the Multi-State Constraint Kalman Filter (MSCKF) [25] update with nullspace projection [34].

C. Landmark Update

We generalize the landmark measurement model as a series of nested functions to encompass different feature parameterizations such as 3D position and inverse depth and so on. Assuming a visual feature that has been tracked over the sliding window of stochastic clones [35], we can write the visual-bearing measurements (i.e., pixel coordinates) as the following series of nested functions:

$$
\begin{aligned}
\mathbf{z}_{m,k} &= h(\mathbf{x}_k) + \mathbf{n}_{m,k} && (23)\\
&= h_d\left(\mathbf{z}_{n,k}, \boldsymbol{\zeta}\right) + \mathbf{n}_{m,k} && (24)\\
&= h_d\left(h_p\left({}^{C_k}\mathbf{p}_f\right), \boldsymbol{\zeta}\right) + \mathbf{n}_{m,k} && (25)\\
&= h_d\left(h_p\left(h_t\left({}^{G}\mathbf{p}_f, {}^{C_k}_{G}\mathbf{R}, {}^{G}\mathbf{p}_{C_k}\right)\right), \boldsymbol{\zeta}\right) + \mathbf{n}_{m,k} && (26)
\end{aligned}
$$

where $\mathbf{z}_{m,k}$ is the raw uv pixel coordinate; $\mathbf{n}_{m,k}$ the raw pixel noise and typically assumed to be zero-mean white Gaussian; $\mathbf{z}_{n,k}$ is the normalized undistorted uv measurement; ${}^{C_k}\mathbf{p}_f$ is the landmark position in the current camera frame; ${}^{G}\mathbf{p}_f$ is the landmark position in the global frame and depending on its representation may also be a function of state elements; and $\left\{{}^{C_k}_{G}\mathbf{R}, {}^{G}\mathbf{p}_{C_k}\right\}$ denotes the current camera pose (position and orientation) in the global frame.

The measurement functions h_d, h_p, and h_t correspond to the intrinsic distortion, projection, and transformation functions and the corresponding measurement Jacobians can be computed through a simple chain rule. Note that we compute the errors on the raw uv pixels to allow for calibration of the camera intrinsics ζ and that the function h_d can be changed to support any camera model (e.g., radial-tangential and equidistant). We refer readers to the documentation website for the details of these measurement functions.

D. Online Calibration

We perform online spatiotemporal calibration of the camera-IMU time offset and extrinsic transformation, and camera intrinsics. Looking at the landmark measurement (26), one can simply take the derivative with respect to the desired variables that they wish to calibrate online. In this case we will have additional Jacobians for the intrinsic ζ in function h_d and $\{{}^{C}_{I}\mathbf{R},\ {}^{C}\mathbf{p}_I\}$ extrinsics that the global pose $\left\{{}^{C_k}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{C_k}\right\}$ is a function of. For derivations and Jacobian results, we refer the reader to our documentation.

We also co-estimate the time offset between the camera and IMU, which can commonly exist in low-cost devices due to sensor latency, clock skew, or data transmission delays. Consider the time ${}^{C}t$ as expressed in the camera clock is related to the same instant represented in the IMU clock, ${}^{I}t$, by a time offset ${}^{C}t_I$:

$$ {}^{I}t = {}^{C}t + {}^{C}t_I \tag{27}$$

This offset is unknown and estimated online. We refer the reader to [23] for further details.

E. Codebase Documentation

It is our belief that the documentation of this work in itself is one of the main contributions to the research community. Both researchers and practitioners with little background in estimation may struggle to grasp the core theoretical concepts and important implementation details when it comes to

visual-inertial estimation algorithms. To bridge this gap the documentation of this codebase takes as much of a priority as new features that could improve the estimation performance. As compared to existing open sourced systems with limited documentation, we focus on providing

additional dedicated derivation pages on how different parts of the code are derived and interact. The in-code and page documentation is automatically generated from the codebase using Doxygen [36] which is then post-processed using m.css [37] to provide high quality search functionality and mobile friendly layout. This tight-coupling of our documentation and derivations within the codebase also ensures that the documentation is up to date and that developers can easily find answers.

IV. VISUAL-INERTIAL SIMULATOR

We now detail how our simulator generates visual-inertial measurements. We note that this simulator can be easily extended to include other measurements besides the inertial and visual-bearing measurements presented below.

A. B-Spline Interpolation

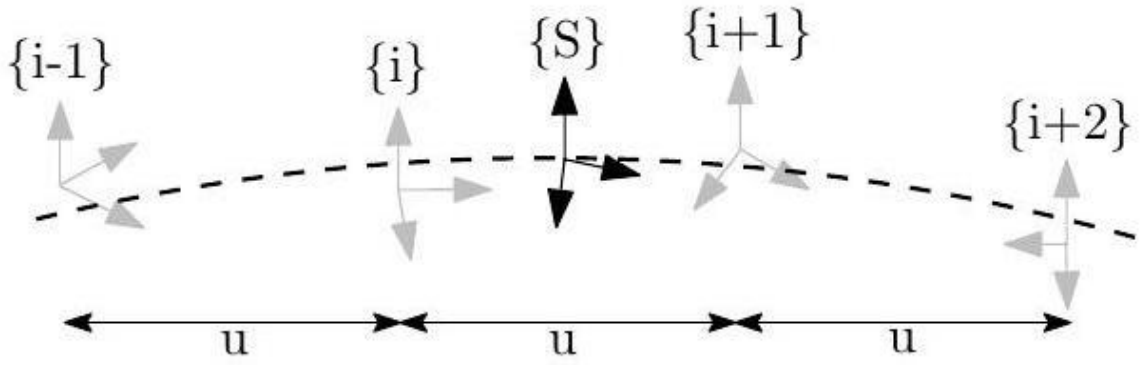

Fig. 1: Illustrate the B-spline interpolation to a pose ${}^{G}_{S}\mathbf{T}$ which is bounded by four control poses.

At the center of the simulator is an $\mathbb{SE}(3)$ B-spline which allows for the calculation of the pose, velocity, and accelerations at any given timestep along a given trajectory. We follow the work of Patron-Perez et al. [38] and Mueggler et al. [39] in which given a series of temporally uniformly distributed "control point" poses, the pose $\{S\}$ at a given timestep t_s can be interpolated by:

$$ {}^{G}_{S}\mathbf{T}\big(u(t_s)\big) = {}^{G}_{i-1}\mathbf{T}\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2 \tag{28} $$

$$ \mathbf{A}_j = \exp\left(B_j\big(u(t)\big)\,{}^{i-1+j}_{i+j}\mathbf{\Omega}\right) \tag{29} $$

$$ {}^{i-1}_{i}\mathbf{\Omega} = \log\left({}^{G}_{i-1}\mathbf{T}^{-1}\,{}^{G}_{i}\mathbf{T}\right) \tag{30} $$

where $B_j(u(t))$ are our spline interpolation constants, $\exp(\cdot), \log(\cdot)$ are the $\mathbb{SE}(3)$ matrix exponential and $\log$ arithm, and the frame notations are shown in Figure 1. Equation (28) can be interpreted as compounding the fraction portions of the bounding poses to the first pose ${}^{G}_{i-1}\mathbf{T}$. It is then simple to take the time derivative to allow the computation of the velocity and acceleration at any point. The only needed input into the simulator is a pose trajectory which we uniformly sample to construct control points for the B-spline. This B-spline is then used to both generate the inertial measurements while also providing the pose information needed to generate visual-bearing measurements.

B. Inertial Measurements

To incorporate inertial measurements from an IMU sensor, we can leverage the continuous nature and C^2-continuity of

our cubic B-spline. To obtain the true measurements from our $\mathrm{SE}(3)$ B-spline we can do the following:

$$ {}^{I}\boldsymbol{\omega}(t) = \mathrm{vee}({}^{G}_{I}\mathbf{R}(u(t))^{\top}\,{}^{G}_{I}\dot{\mathbf{R}}(u(t))) \tag{31} $$

$$ {}^{I}\mathbf{a}(t) = {}^{G}_{I}\mathbf{R}(u(t))^{\top G}\ddot{\mathbf{p}}_{I}(u(t)) \tag{32} $$

where vee($\cdot$) returns the vector portion of the skewsymmetric matrix. These are then corrupted using the random walk biases and corresponding white noises.

C. Visual-Bearing Measurement

After creating the B-spline trajectory we generate environmental landmarks that can be later projected into the synthetic camera frames. To generate these landmarks, we increment along the spline at a fixed interval and ensure that all cameras see enough landmarks in the map. If there are not enough landmarks in the given camera frame, we generate new landmarks by sending out random rays from the camera and assigning a random depth. Landmarks are then added to the map so that they can be projected into future frames. We generate landmarks' visual measurements by projecting them into the current frame. Projected landmarks are limited to being within the field of view, in front, and close in distance to the camera. Pixel noise can be directly added to the true pixel values.

V. BENCHMARKS

A. Simulation Results

With the proposed visual-inertial simulator, we evaluate the proposed online calibration and the consistency of our MSCKF estimator, which is implemented based on the First Estimate Jacobians (FEJ)-EKF [21], [22]. In particular, the system is run with a monocular camera, a window size of 11, a maximum of 100 feature tracks per frame, and a maximum of 50 SLAM landmarks kept in the state, [1] along with VIO feature tracks that are processed by the MSCKF update. The camera is simulated at 10 Hz while the IMU is simulated at 400 Hz . We inject one pixel noise and the IMU noise characteristics of an ADIS16448 MEMS IMU. To simulate bad initial calibration values, we randomly initialize the calibration values using the prior distribution values of the estimator. This ensures that during Monte-Carlo simulation we have both different measurement noises and initial calibration values for each run.

As summarized in Table I, the average Absolute Trajectory Error (ATE) and Normalized Estimation Error Squared (NEES) for each different scenario shows that when performing online calibration, estimation accuracy does not degrade if we are given the true calibration; while in the case that we have bad initial guesses, the estimator remains consistent and is able to estimate with reasonable accuracy. A representative run with uncertainty bounds is shown in Figure 3. When calibration is disabled and a bad initial guess is used, the NEES becomes large due to not modeling the uncertainty that these calibration parameters have, and in many cases the estimate diverges. We also plot the first ten and sixty

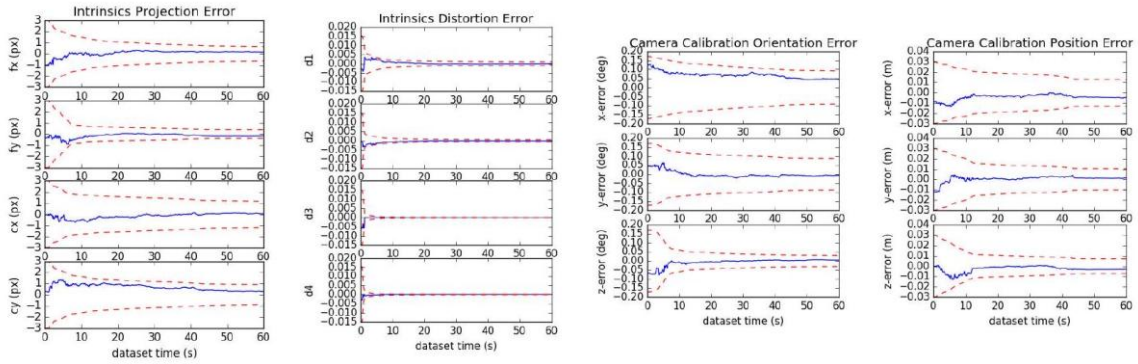

Fig. 2: Camera intrinsic projection and distortion along with extrinsic orientation and positions parameters error (blue-solid) and 3σ bounds (red-dashed) for a representative run. Note that we only plot the first sixty seconds of the dataset.

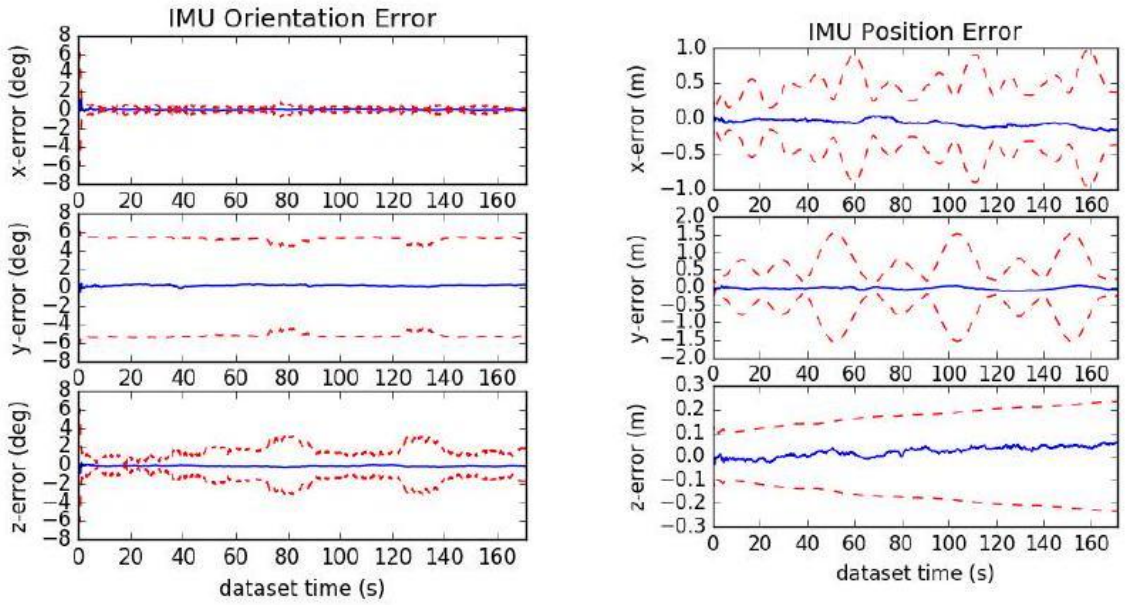

Fig. 3: IMU pose errors (blue-solid) and 3σ bounds (reddashed) for a representative run of the proposed method with SLAM landmarks and online calibration.

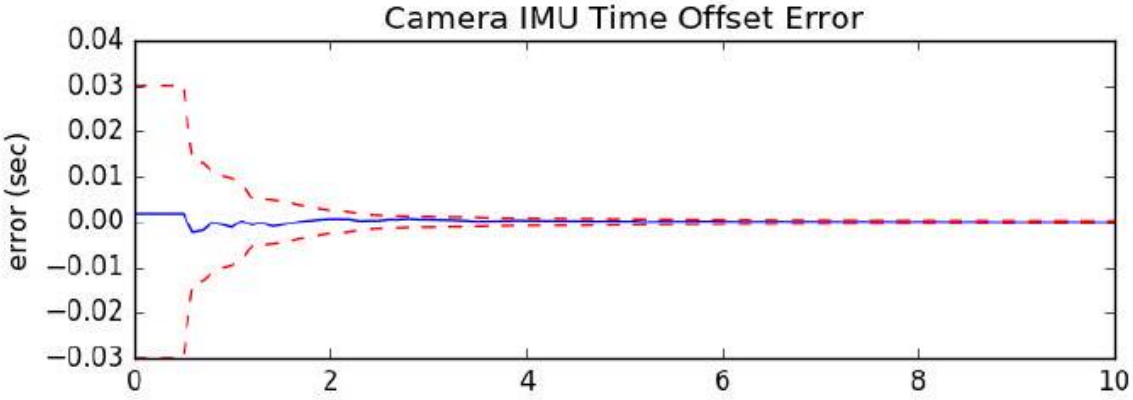

Fig. 4: Camera to IMU time offset error (blue-solid) and 3σ bounds (red-dashed) for a representative run.

seconds of all calibration parameters of a representative run in Figures 2 and 4, showing that these parameters rapidly converge from their initially poor guesses.

B. Real-World Comparison

We evaluate the proposed visual-inertial FEJ-MSCKF estimator with and without SLAM landmarks on the Vicon room scenarios from the EurocMav dataset [41] which provides both 20 Hz stereo images, 200 Hz ADIS16448 MEMS IMU measurements, and optimized groundtruth trajectories. It should be noted that we have recalculated the V1_01_easy groundtruth due to the original having incorrect orientation values and have provided this corrected groundtruth

trajectory to the community on our documentation website. All methods were run with the configuration files from their open sourced repositories with each algorithm being run ten times on each dataset to compensate for some randomness inherent to the visual front-ends. In this benchmarking test, we eval-

TABLE I: Average ATE and NEES over twenty runs with true or bad calibration, with and without online calibration.

	ATE (deg)	ATE (m)	Ori. NEES	Pos. NEES
true w/ calib	0.212	0.134	2.203	1.880
true w/o calib	0.200	0.128	2.265	1.909
bad w/ calib	0.218	0.139	2.235	2.007
bad w/o calib	5.432	508.719	9.159	1045.174

uate the following state-of-the-art visual-inertial estimation algorithms:

OKVIS [2] - Keyframe-based fixed-lag smoother which optimizes arbitrarily spaced keyframe poses connected with inertial measurement factors and environmental landmarks. A fixed window size was enforced to ensure computational feasibility with the focus on selective marginalization to allow for problem sparsity.

VINS-Fusion VIO [3] - Extension of the original VINSMono [42] sliding optimization-based method that leverages IMU preintegration which is then loosely coupled with a secondary pose-graph optimization. VINS-Fusion extends the original codebase to support stereo cameras.

Basalt VIO [4] - Stereo keyframe-based fix-lag smoother with custom feature tracking frontend with focus on extracting relevant information from the VIO for later offline visual-inertial mapping.

R-VIO [5] - Robocentric MSCKF-based algorithm which estimates in a local frame and updates the global frame through a composition step. The direction of gravity is also estimated within the filter.

ROVIO [6] - We use the ROVIO implementation within maplab [43], which is a monocular iterative EKF-based approach that performs minimization on the direct image intensity patches allowing for tracking of non-corner features such as high gradient lines.

ICE-BA [7] - Stereo incremental bundle adjustment (BA) method which optimizes both a local siding window and global optimization problem in parallel. They exploited the sparseness of their formulation and introduced a relative marginalization procedure.

S-MSCKF [8] - An open sourced implementation of original

TABLE II: Ten runs mean absolute trajectory error (ATE) for each algorithm in units of degree/meters. Note that V2_03 dataset is excluded due the inability for some algorithms to run on it. Green denotes the best, while blue is second best.

	V1_01_easy	V1_02_medium	V1_03_difficult	V2_01_easy	V2_02_medium	Average
mono_ov_slam	0.699 / 0.058	1.675 / 0.076	2.542 / 0.063	0.773 / 0.124	1.538 / 0.074	1.445 / 0.079
mono_ov_vio	0.642 / 0.076	1.766 / 0.096	2.391 / 0.344	1.164 / 0.121	1.248 / 0.106	1.442 / 0.148

mono_okvis	0.823 / 0.090	2.082 / 0.146	4.122 / 0.222	0.826 / 0.117	1.704 / 0.197	1.911 / 0.154
mono_rovi-oli	2.249 / 0.153	1.635 / 0.131	3.253 / 0.158	1.455 / 0.106	1.678 / 0.153	2.054 / 0.140
mono_rvio	0.994 / 0.094	2.288 / 0.129	1.757 / 0.147	1.735 / 0.144	1.690 / 0.233	1.693 / 0.149
mono_vinsfusion_vio	1.199 / 0.064	3.542 / 0.103	5.934 / 0.202	1.585 / 0.073	2.370 / 0.079	2.926 / 0.104
ste-reo_ov_slam	0.856 / 0.061	1.813 / 0.047	2.764 / 0.059	1.037 / 0.056	1.292 / 0.047	1.552 / 0.054
ste-reo_ov_vio	0.905 / 0.061	1.767 / 0.056	2.339 / 0.057	1.106 / 0.053	1.151 / 0.048	1.454 / 0.055
stereo_basalt	0.654 / 0.035	2.067 / 0.059	2.017 / 0.085	0.981 / 0.046	0.888 / 0.059	1.321 / 0.057

stereo_iceba	0.909 / 0.059	2.574 / 0.120	3.206 / 0.137	1.819 / 0.128	1.212 / 0.116	1.944 / 0.112
stereo_okvis	0.603 / 0.039	1.963 / 0.079	4.117 / 0.122	0.834 / 0.075	1.201 / 0.092	1.744 / 0.081
stereo_smsckf	1.108 / 0.086	2.147 / 0.121	3.918 / 0.198	1.181 / 0.083	2.142 / 0.164	2.099 / 0.130
stereo_vinsfusion_vio	1.073 / 0.054	2.695 / 0.089	3.643 / 0.132	2.499 / 0.071	2.006 / 0.074	2.383 / 0.084

TABLE III: Relative pose error (RPE) for different segment lengths for each algorithm variation over all datasets in units of degree/meters. Note that V2_03 dataset is excluded due the inability for some algorithms to run on it.

	8m	16m	24m	**32m**	40m	48m
mono_ov_slam	0.661 / 0.074	0.802 / 0.086	0.979 / 0.097	1.061 / 0.105	1.145 / 0.120	1.289 / 0.122
mono_ov_vio	0.826 / 0.094	1.039 / 0.106	1.215 / 0.111	1.283 / 0.132	1.342 / 0.151	1.425 / 0.184
mono_okvis	0.662 / 0.107	0.870 / 0.161	1.031 / 0.190	1.225 / 0.213	1.384 / 0.240	1.603 / 0.251

mono_rovioli	1.136 / 0.095	1.585 / 0.135	1.847 / 0.184	2.078 / 0.226	2.218 / 0.263	2.402 / 0.295
mono_rvio	0.705 / 0.130	0.902 / 0.160	1.029 / 0.183	1.074 / 0.213	0.991 / 0.227	1.077 / 0.232
mono_vinsfu-sion_vio	0.940 / 0.070	1.298 / 0.103	1.680 / 0.118	1.822 / 0.146	1.833 / 0.153	1.860 / 0.171
stereo_ov_slam	0.685 / 0.069	0.876 / 0.080	1.064 / 0.087	1.169 / 0.087	1.275 / 0.098	1.488 / 0.105
stereo_ov_vio	0.722 / 0.068	0.892 / 0.077	1.089 / 0.087	1.218 / 0.088	1.342 / 0.101	1.489 / 0.106
stereo_basalt	0.538 / 0.063	0.576 / 0.070	0.649 / 0.078	0.715 / 0.086	0.647 / 0.097	0.758 / 0.111
stereo_iceba	0.955 / 0.096	1.227 / 0.114	1.415 / 0.120	1.658 / 0.152	1.856 / 0.173	1.803 / 0.180
stereo_okvis	0.611 / 0.066	0.772 / 0.089	0.916 / 0.103	1.089 / 0.119	1.173 / 0.136	1.404 / 0.141
stereo_smsckf	1.084 / 0.098	1.462 / 0.136	1.578 / 0.159	1.667 / 0.187	1.901 / 0.200	2.134 / 0.217
stereo_vinsfu-sion_vio	0.946 / 0.057	1.357 / 0.079	1.721 / 0.097	1.928 / 0.111	1.935 / 0.125	1.805 / 0.132

MSCKF [25] paper with stereo feature tracking and a focus on high-speed motion scenarios.

Note that we evaluate only the VIO portion of these codebases (i.e., not the non-realtime backend pose graph thread output of VINS-Fusion [3] and visual-inertial mapping of Basalt [4]), as one could simply append a pose graph optimizer after any of these odometry methods to improve long-term accuracy.

Table II shows the average ATE of all methods for each dataset. It is clear that the addition of SLAM landmarks in our OpenVINS greatly reduces the drift in the monocular case, while it has a smaller impact on the stereo performance; and more importantly, OpenVINS is able to perform competitively to other methods. We additionally compared the Relative Pose Error (RPE) of all methods. Shown in Table III, our monocular system clearly outperforms the current open sourced codebases, with our stereo system being able to perform second to Basalt. While we did not evaluate perframe timing rigorously, we found that Basalt outperformed all other algorithms, with our proposed method being limited by the visual-frontend implementation from OpenCV [44] and SLAM feature update equally. On the first EurocMav dataset we could process at $\frac{2.7x}{4.3x}$ and $\frac{1.2x}{1.9x}$ realtime

for our monocular SLAM/VIO, and stereo SLAM/VIO, respectively, on an Intel® Xeon(R) CPU E3-1505M v6 @ 3.00 GHz processor in single threaded execution.

VI. CONCLUSION AND FUTURE WORK

In this paper we have presented our OpenVINS (OV) system as a platform for the research community. At the core we provide the visual processing frontend, full visual-inertial simulator, and modular on-manifold EKF. In particular, we have implemented the FEJ-based MSCKF with and without SLAM landmarks and demonstrated the competing performance of our estimator. We have heavily documented the project to allow for researchers and practitioners to quickly build on top of this work with minimal estimation theory background. In the future we plan to expand our system to provide a sliding window optimization-based estimator leveraging our closed-form preintegration [45]. We are also interested in integrating visual-inertial mapping and perception capabilities into OpenVINS.

References

[1] G. Huang, "Visual-inertial navigation: A concise review," in Proc. International Conference on Robotics and Automation, Montreal, Canada, May 2019.

[2] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," International Journal of Robotics Research, vol. 34, no. 3, pp. 314-334, 2015.

[3] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," CoRR, vol. abs/1901.03638, 2019.

[4] V. C. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," CoRR, vol. abs/1904.06504, 2019.

[5] Z. Huai and G. Huang, "Robocentric visual-inertial odometry," International Journal of Robotics Research, Apr. 2019, (to appear).

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

[7] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1974-1982.

[8] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 965-972, April 2018.

[9] K. Eckenhoff, P. Geneva, J. Bloecker, and G. Huang, "Multi-camera visual-inertial navigation with online intrinsic and extrinsic calibration," in Proc. International Conference on Robotics and Automation, Montreal, Canada, May 2019.

[10] K. Eckenhoff, P. Geneva, and G. Huang, "Sensor-failure-resilient multi-imu visual-inertial navigation," in Proc. International Conference on Robotics and Automation, Montreal, Canada, May 2019.

[11] K. Eckenhoff, Y. Yang, P. Geneva, and G. Huang, "Tightly-coupled visual-inertial localization and 3D rigid-body target tracking," IEEE Robotics and Automation Letters (RA-L), vol. 4, no. 2, pp. 1541-1548, 2019.

[12] K. Eckenhoff, P. Geneva, N. Merrill, and G. Huang, "Schmidt-ekfbased visual-inertial moving object tracking," in Proc. of the IEEE International Conference on Robotics and Automation, Paris, France, 2020.

[13] P. Geneva, K. Eckenhoff, and G. Huang, "A linear-complexity EKF for visual-inertial navigation with loop closures," in Proc. International Conference on Robotics and Automation, Montreal, Canada, May 2019.

[14] P. Geneva, J. Maley, and G. Huang, "An efficient schmidt-ekf for 3D visual-inertial SLAM," in Proc. Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, June 2019, (accepted).

[15] Y. Yang, P. Geneva, X. Zuo, K. Eckenhoff, Y. Liu, and G. Huang, "Tightly-coupled aided inertial navigation with point and plane features," in Proc. International Conference on Robotics and Automation, Montreal, Canada, May 2019.

[16] Y. Yang, P. Geneva, K. Eckenhoff, and G. Huang, "Visual-inertial navigation with point and line features," Macau, China, Nov. 2019, (accepted).

[17] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "LIC-Fusion: Lidar-inertial-camera odometry," Macau, China, Nov. 2019, (accepted).

[18] X. Zuo, P. Geneva, Y. Yang, W. Ye, Y. Liu, and G. Huang, "Visualinertial localization with prior lidar map constraints," IEEE Robotics and Automation Letters (RA-L), 2019, (to appear).

[19] Y. Yang, P. Geneva, K. Eckenhoff, and G. Huang, "Degenerate motion analysis for aided INS with online spatial and temporal calibration," IEEE Robotics and Automation Letters (RA-L), vol. 4, no. 2, pp. 20702077, 2019.

[20] G. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and improvement of the consistency of extended Kalman filter-based SLAM," in Proc. of the IEEE International Conference on Robotics and Automation, Pasadena, CA, May 19-23 2008, pp. 473-479.

[21] -, "A first-estimates Jacobian EKF for improving SLAM consistency," in Proc. of the 11th International Symposium on Experimental Robotics, Athens, Greece, July 14-17, 2008.

[22] -, "Observability-based rules for designing consistent EKF SLAM estimators," International Journal of Robotics Research, vol. 29, no. 5, pp. 502-528, Apr. 2010.

[23] M. Li and A. I. Mourikis, "Online temporal calibration for CameraIMU systems: Theory and algorithms," International Journal of Robotics Research, vol. 33, no. 7, pp. 947-964, June 2014.

[24] M. Li, H. Yu, X. Zheng, and A. I. Mourikis, "High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation," in IEEE International Conference on Robotics and Automation (ICRA), May 2014, pp. 409-416.

[25] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in Proceedings of the IEEE International Conference on Robotics and Automation, Rome, Italy, Apr. 10-14, 2007, pp. 3565-3572.

[26] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep., Mar. 2005.

[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] K. Wu, T. Zhang, D. Su, S. Huang, and G. Dissanayake, "An invariant-ekf vins algorithm for improving consistency," in Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Sept 2017, pp. 1578-1585.

[29] J. Civera, A. Davison, and J. Montiel, "Inverse depth parametrization for monocular SLAM," IEEE Transactions on Robotics, vol. 24, no. 5, pp. 932-945, Oct. 2008.

[30] M. K. Paul, K. Wu, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS," in Proc. of the IEEE International Conference on Robotics and Automation, Singapore, July 2017, pp. 165-172.

[31] A. B. Chatfield, Fundamentals of High Accuracy Inertial Navigation. AIAA, 1997.

[32] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[33] M. Li, "Visual-inertial odometry on resource-constrained systems," Ph.D. dissertation, UC Riverside, 2014.

[34] Y. Yang, J. Maley, and G. Huang, "Null-space-based marginalization: Analysis and algorithm," in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, Canada, Sept. 24-28, 2017, pp. 6749-6755.

[35] S. I. Roumeliotis and J. W. Burdick, "Stochastic cloning: A generalized framework for processing relative state measurements," in Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, May 11-15, 2002, pp. 1788-1795.

[36] D. Van Heesch, "Doxygen: Source code documentation generator tool," URL: http://www.doxygen.org, 2008.

[37] V. Vondruš, "m.css: A no-nonsense, no-javascript css framework and pelican theme for content-oriented websites," URL: https://mcss. mosra.cz/, 2018.

[38] A. Patron-Perez, S. Lovegrove, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," International Journal of Computer Vision, vol. 113, no. 3, pp. 208219, 2015.

[39] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-time visual-inertial odometry for event cameras," IEEE Transactions on Robotics, pp. 1-16, 2018.

[40] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation," in Robotics: Science and Systems, Berlin, Germany, June 2013, pp. 241-248.

[41] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," The International Journal of Robotics Research, vol. 35, no. 10, pp. 1157-1163, 2016.

[42] 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.

[43] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "Maplab: An open framework for research in visual-inertial mapping and localization," IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1418-1425, July 2018.

[44] OpenCV Developers Team, "Open source computer vision (OpenCV) library," Available: http://opencv.org.

[45] K. Eckenhoff, P. Geneva, and G. Huang, "Closed-form preintegration methods for graph-based visual-inertial navigation," International Journal of Robotics Research, vol. 38, no. 5, pp. 563-586, 2019.

<文献翻译二：原文>

A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors[2]

Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen

Abstract

Nowadays, more and more sensors are equipped on robots to increase robustness and autonomous ability. We have seen various sensor suites equipped on different platforms, such as stereo cameras on ground vehicles, a monocular camera with an IMU (Inertial Measurement Unit) on mobile phones, and stereo cameras with an IMU on aerial robots. Although many algorithms for state estimation have been proposed in the past, they are usually applied to a single sensor or a specific sensor suite. Few of them can be employed with multiple sensor choices. In this paper, we proposed a general optimization-based framework for odometry estimation, which supports multiple sensor sets. Every sensor is treated as a general factor in our framework. Factors which share common state variables are summed together to build the optimization problem. We further demonstrate the generality with visual and inertial sensors, which form three sensor suites (stereo cameras, a monocular camera with an IMU, and stereo cameras with an IMU). We validate the performance of our system on public datasets and through real-world experiments with multiple sensors. Results are compared against other state-of-the-art algorithms. We highlight that our system is a general framework, which can easily fuse various sensors in a pose graph optimization. Our implementations are open source [1].

I. Introduction

[2] Qin T, Pan J, Cao S, et al. A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors[R]. arXiv preprint arXiv:1901.03638, 2019.

Real-time 6-DoF (Degrees of Freedom) state estimation is a fundamental technology for robotics. Accurate state estimation plays an important role in various intelligent applications, such as robot exploration, autonomous driving, VR (Virtual Reality) and AR (Augmented Reality). The most common sensors we use in these applications are cameras. A large number of impressive vision-based algorithms for pose estimation has been proposed over the last decades, such as [1]-[5]. Besides cameras, the IMU is another popular option for state estimation. The IMU can measure acceleration and angular velocity at a high frequency, which is necessary for low-latency pose feedback in real-time applications. Hence, there are numerous research works fusing vision and IMU together, such as [6]-[12]. Another popular sensor used in state estimation is LiDAR. LiDAR-based approaches [13] achieve accurate pose estimation in a confined local environment. Although a lot of algorithms have been proposed in the past, they are usually applied to a single input sensor or a specific sensor suite.

Recently, we have seen platforms equipped with various sensor sets, such as stereo cameras on ground vehicles, a monocular camera with an IMU on mobile phones, stereo

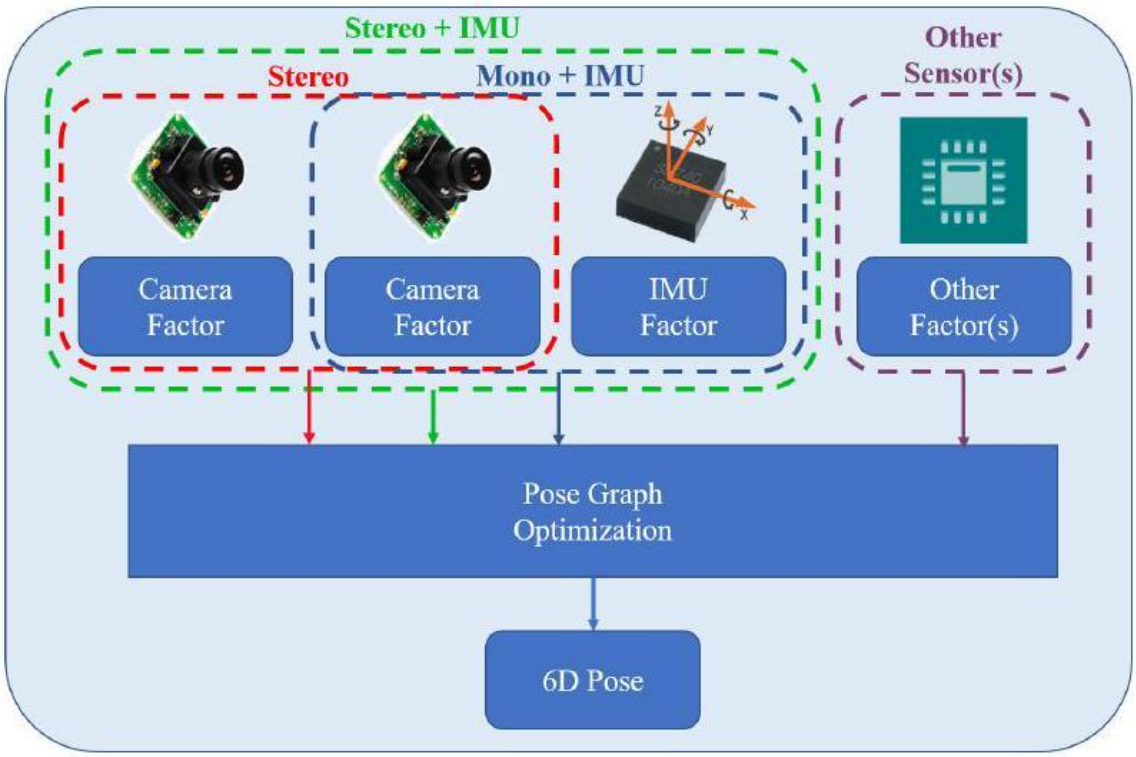

Fig. 1. An illustration of the proposed framework for state estimation, which supports multiple sensor choices, such as stereo cameras, a monocular camera with an IMU, and stereo cameras

with an IMU. Each sensor is treated as a general factor. Factors which share common state variables are summed together to build the optimization problem.
cameras with an IMU on aerial robots. However, as most traditional algorithms were designed for a single sensor or a specific sensor set, they cannot be ported to different platforms. Even for one platform, we need to choose different sensor combinations in different scenarios. Therefore, a general algorithm which supports different sensor suites is required. Another practical requirement is that in case of sensor failure, an inactive sensor should be removed and an alternative sensor should be added into the system quickly. Hence, a general algorithm which is compatible with multiple sensors is in need.

In this paper, we propose a general optimization-based framework for pose estimation, which supports multiple sensor combinations. We further demonstrate it with visual and inertial sensors, which form three sensor suites (stereo cameras, a monocular camera with an IMU, and stereo cameras with an IMU). We can easily switch between different sensor combinations. We highlight the contribution of this paper as follows:

- a general optimization-based framework for state estimation, which supports multiple sensors.
- a detailed demonstration of state estimation with visual and inertial sensors, which form different sensor suites (stereo cameras, a monocular camera + an IMU, and stereo cameras + an IMU).
- an evaluation of the proposed system on both public datasets and real experiments.
- open-source code for the community.

II. Related Work

State estimation has been a popular research topic over the last decades. A large number of algorithms focus on accurate 6-DoF pose estimation. We have seen many impressive approaches that work with one kind of sensor, such as visual-based methods [1]-[5], LiDAR-based methods [13], RGB-D based methods [14]. and event-based methods [15]. Approaches work with a

monocular camera is hard to achieve 6-DoF pose estimation, since absolute scale cannot be recovered from a single camera. To increase the observability and robustness, multiple sensors which have complementary properties are fused together.

There are two trends of approaches for multi-sensor fusion. One is filter-based methods, the other is optimizationbased methods. Filter-based methods are usually achieved by EKF (Extended Kalman Filter). Visual and inertial measurements are usually filtered together for 6-DoF state estimation. A high-rate inertial sensor is used for state propagation and visual measurements are used for the update in [9, 16]. MSCKF [6, 7] was a popular EKF-based VIO (Visual Inertial Odometry), which maintained several camera poses and leveraged multiple camera views to form the multi-constraint update. Filter-based methods usually linearize states earlier and suffer from error induced by inaccurate linear points. To overcome the inconsistency caused by linearized error, observability constrained EKF [17] was proposed to improve accuracy and consistency. An UKF (Unscented Kalman Filter) algorithm was proposed in [18], where visual, Li DAR and GPS measurements were fused together. UKF is an externsion of EKF without analytic Jacobians. Filterbased methods are sensitive to time synchronization. Any late-coming measurements will cause trouble since states cannot be propagated back in filter procedure. Hence, special ordering mechanism is required to make sure that all measurements from multiple sensors are in order.

Optimization-based methods maintain a lot of measurements and optimize multiple variables at once, which is also known as Bundle Adjustment (BA). Compared with filterbased method, optimization-based method have advantage in time synchronization. Because the big bundle serves as a nature buffer, it can easily handle the case when measurments from multiple sensors come in disorder. Optimization-based algorithms also outperform the filter-based algorithms in term of accuracy at the cost of computational complexity. Early optimization solvers, such as G2O [19], leveraged the Gauss-Newton and Levenberg-Marquardt approaches to solve the problem. Although the sparse structure was employed in optimization solvers, the complexity grown quadratically with the number of states and measurements. In order to achieve real-time performance, some algorithms have explored incremental solvers, while others bounded the

size of the pose graph. iSAM2 [20] was an efficient incremental solver, which reused the previous optimization result to reduce computation when new measurements came. The optimization iteration only updated a small part of states

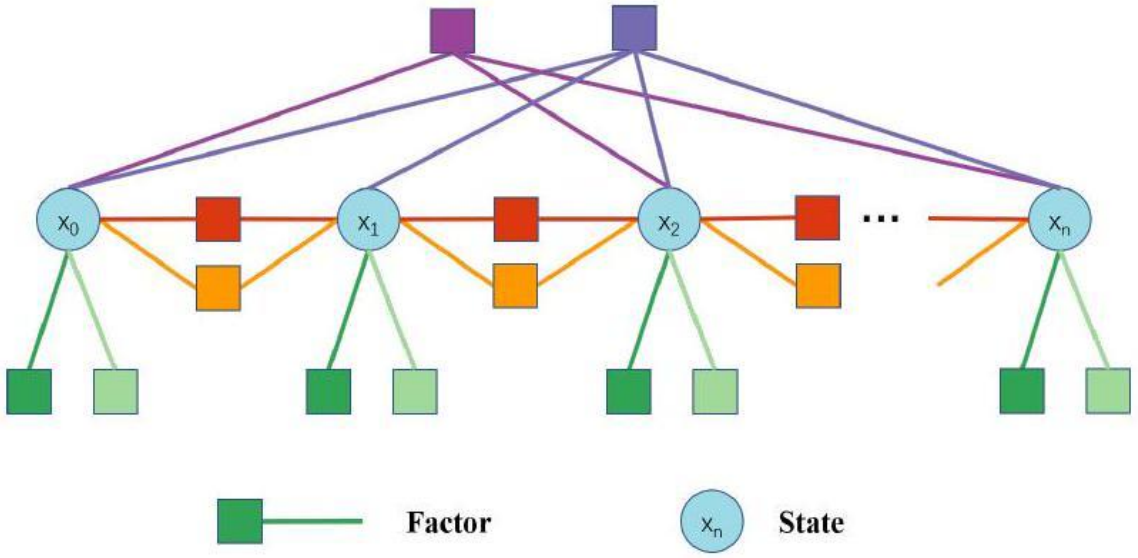

Fig. 2. A graphic illustration of the pose graph. Each node represents states (position, orientation, velocity and so on) at one moment. Each edge represents a factor, which is derived by one measurement. Edges constrain one state, two states or multiple states.

instead of the whole pose graph. Afterward, an accelerated solver was proposed in [21], which improved efficiency by reconstructing dense structure into sparse blocks. Methods, that keep a fixed sized of pose graph, are called slidingwindow approaches. Impressive optimization-based VIO approaches, such as [8,10,12], optimized variables over a bounded-size sliding window. The previous states were marginalized into a prior factor without loss of information in [8, 12]. In this paper, we adopt a sliding-window optimization-based framework for state estimation.

III. System Overview

The structure of proposed framework is shown in Fig. 1. Multiple kinds of sensors can be freely combined. The mesurement of each sensor is treated as a general factor. Factors and their related states form the pose graph. An illustration of pose graph is shown in Fig. 2. Each node represents states (position, orientation, velocity and so on) at one moment. Each edge represents a factor, which is derived by one measurement. Factors constrain one state, two states or multiple states. For IMU factor, it constrains two consecutive states by continuous motion restriction. For a visual landmark, its factor constrains multiple states since it is observed on

multiple frames. Once the graph is built, optimizing it equals to finding the configuration of nodes that match all edges as much as possible.

In this paper, we specifically demonstrate the system with visual and inertial sensors. Visual and inertial sensors can form three combinations for 6-DoF state estimation, which are stereo cameras, a monocular camera with an IMU, and stereo cameras with an IMU. A graphic illustration of the proposed framework with visual and inertial sensors is shown in Fig. 3. Several camera poses, IMU measurements and visual measurements exist in the pose graph. The IMU and one of cameras are optional.

IV. Methodology

A. Problem Definition

1)States: Main states that we need to estimate includes 3D position and orientation of robot's center. In addition, we

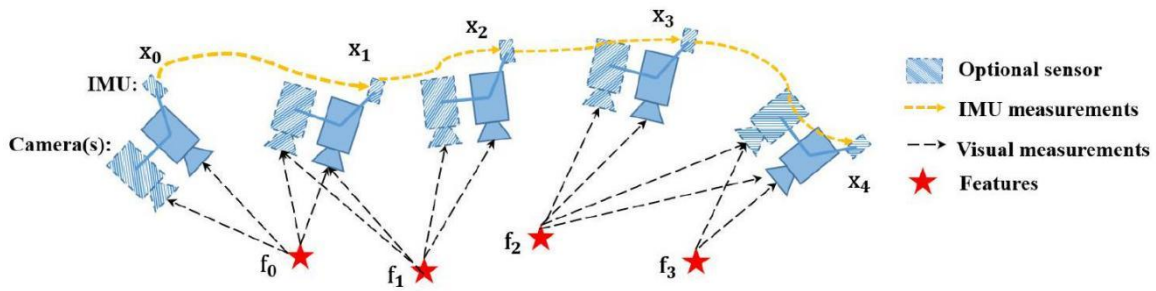

Fig. 3. A graphic illustration of the proposed framework with visual and inertial sensors. The IMU and one of cameras are optional. Therefore, it forms three types (stereo cameras, a monocular camera with an IMU, and stereo cameras with an IMU). Several camera poses, IMU measurements and visual measurements exist in the pose graph.

have other optional states, which are related to sensors. For cameras, depths or 3D locations of visual landmarks need to be estimated. For IMU, it produces another motion variable, velocity, Also, time-variant acceleration bias and gyroscope bias of the IMU are needed to be estimated. Hence, for visual and inertial sensors, whole states we need to estimate are defined as follows:

$$\begin{aligned} \mathcal{X} &= [\mathbf{p}_0, \mathbf{R}_0, \mathbf{p}_1, \mathbf{R}_1, \dots, \mathbf{p}_n, \mathbf{R}_n, \mathbf{x}_{cam}, \mathbf{x}_{imu}] \\ \mathbf{x}_{cam} &= [\lambda_0, \lambda_1, \dots, \lambda_l] \\ \mathbf{x}_{imu} &= [\mathbf{v}_0, \mathbf{b}_{a_0}, \mathbf{b}_{g_0}, \mathbf{v}_1, \mathbf{b}_{a_1}, \mathbf{b}_{g_1}, \dots, \mathbf{v}_n, \mathbf{b}_{a_n}, \mathbf{b}_{g_n}] \end{aligned} \tag{1}$$

where $\mathbf{p}$ and $\mathbf{R}$ are basic system states, which correspond to position and orientation of body expressed in world frame. $\mathbf{x}_{\text{cam}}$ is camera-ralated state, which includes depth λ of each feature observed in the first frame. $\mathbf{x}_{imu}$ is IMU-ralated variable, which is composed of velocity $\mathbf{v}$, acceleration bias $\mathbf{b}_a$ and gyroscope bias $\mathbf{b}_g$. $\mathbf{x}_{imu}$ can be omitted if we only use stereo camera without an IMU. The translation from sensors' center to body's center are assumed to be known, which are calibrated offline. In order to simplify the notation, we denote the IMU as body's center (If the IMU is not used, we denote left camera as body's center).

2) Cost Function: The nature of state estimation is an MLE (Maximum Likelihood Estimation) problem. The MLE consists of the joint probability distribution of robot poses over a period of time. Under the assumption that all measurements are independent, the problem is typically derived as,

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \prod_{t=0}^{n} \prod_{k\in\mathbf{S}} p\left(\mathbf{z}_t^k \mid \mathcal{X}\right), \tag{2}$$

where $\mathbf{S}$ is the set of measurements, which come from cameras, IMU and other sensors. We assume the uncertainty of measurements is Gaussian distributed, $p\left(\mathbf{z}_t^k \mid \mathcal{X}\right) \sim \mathcal{N}\left(\overline{\mathbf{z}}_t^k, \Omega_t^k\right)$. Therefore, the negative log-likelihood of abovementioned equation is written as,

$$\begin{aligned} X^* &= \arg\max_{X} \prod_{t=0}^{n} \prod_{k\in\mathbf{S}} \exp\left(-\frac{1}{2}\left\|\mathbf{z}_t^k - \mathrm{h}_t^k(X)\right\|_{\mathbf{\Omega}_t^k}^2\right) \\ &= \arg\min_{X} \sum_{t=0}^{n} \sum_{k\in\mathbf{S}} \left\|\mathbf{z}_t^k - \mathrm{h}_t^k(X)\right\|_{\mathbf{\Omega}_t^k}^2 \end{aligned} \tag{3}$$

The Mahalanobis norm is defined as $\|\mathbf{r}\|_{\mathbf{\Omega}}^2 = \mathbf{r}^T\mathbf{\Omega}^{-1}\mathbf{r}$. $h(\cdot)$ is the sensor model, which is detailed in the following section. Then the state estimation is converted to a nonlinear least squares problem, which is also known as Bundle Adjustment (BA).

B. Sensor Factors

1. Camera Factor: The framework supports both monocular and stereo cameras. The intrinsic parameters of every camera and the extrinsic transformation between cameras are supposed to be known, which can be easily calibrated offline. For each camera

frame, corner features [22] are detected. These features are tracked in previous frame by KLT tracker [23]. For the stereo setting, the tracker also matches features between the left image and right image. According to the feature associations, we construct the camera factor with per feature in each frame. The camera factor is the reprojection process, which projects a feature from its first observation into following frames.

Considering the feature l that is first observed in the image i, the residual for the observation in the following image t is defined as:

$$\begin{aligned}\mathbf{z}_t^l\text{-}\mathrm{h}_t^l(X)&=\mathbf{z}_t^l\text{-}\mathrm{h}_t^l\left(\mathbf{R}_i,\mathbf{p}_i,\mathbf{R}_t,\mathbf{p}_t,\lambda_l\right)\\&=\begin{bmatrix}u_t^l\\v_t^l\end{bmatrix}\text{-}\pi_c\left(\mathbf{T}_c^{b\text{-}1}\mathbf{T}_t^{-1}\mathbf{T}_i\mathbf{T}_c^b\pi_c^{-1}\left(\lambda_l,\begin{bmatrix}u_i^l\\v_i^l\end{bmatrix}\right)\right)\end{aligned}\tag{4}$$

where $\left[u_i^l, v_i^l\right]$ is the first observation of the l feature that appears in the i image. [u_t^l, v_t^l] is the observation of the same feature in the t image. π_c and π_c^{-1} are the projection and back-projection functions which depend on camera model (pinhole, omnidirectional or other models). $\mathbf{T}$ is the 4×4 homogeneous transformation, which is $\begin{bmatrix}\mathbf{R} & \mathbf{p}\\\mathbf{0} & 1\end{bmatrix}$. We omit some homogeneous terms for concise expression. $\mathbf{T}_b^c$ is the extrinsic transformation from body center to camera center, which is calibrated offline. The covariance matrix $\mathbf{\Omega}_t^l$ of reprojection error is a constant value in pixel coordinate, which comes from the camera's intrinsic calibration results.

This factor is universal for both left camera and right camera. We can project a feature from the left image to the left image in temporal space, also we can project a feature from the left image to the right image in spatial space.

For different cameras, a different extrinsic transformation $\mathbf{T}_b^c$ should be used.

2) IMU Factor: We use the well-known IMU preintegration algorithm [11, 12] to construct the IMU factor. We assume that the additive noise in acceleration and gyroscope measurements are Gaussian white noise. The time-varying acceleration and gyroscope bias are modeled as a random walk process, whose derivative is Gaussian white noise. Since the IMU acquires data at a higher frequency than other sensors, there are usually multiple IMU measurements existing be-

tween two frames. Therefore, we pre-integrate IMU measurements on the manifold with covariance propagation. The detailed preintegration can be found at [12]. Within two time instants, $t-1$ and t, the preintegration produces relative position $\boldsymbol{\alpha}_t^{t-1}$, velocity $\boldsymbol{\beta}_t^{t-1}$ and rotation $\boldsymbol{\gamma}_t^{t-1}$. Also, the preintegration propagates the covariance of relative position, velocity, and rotation, as well as the covariance of bias. The IMU residual can be defined as:

$$\mathbf{z}_t^{imu}\text{-}\mathrm{h}_t^{imu}(X)=\begin{bmatrix}\boldsymbol{\alpha}_t^{t-1}\\ \boldsymbol{\beta}_t^{t-1}\\ \boldsymbol{\gamma}_t^{t-1}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\ominus\begin{bmatrix}\mathbf{R}_{t-1}{}^{-1}\left(\mathbf{p}_t\text{-}\mathbf{p}_{t-1}+\frac{1}{2}\mathbf{g}dt^2\text{-}\mathbf{v}_{t-1}dt\right)\\ \mathbf{R}_{t-1}{}^{-1}(\mathbf{v}_t\text{-}\mathbf{v}_{t-1}+\mathbf{g}dt)\\ \mathbf{R}_{t-1}{}^{-1}\mathbf{R}_t\\ \mathbf{b}_{a_t}\text{-}\mathbf{b}_{a_{t-1}}\\ \mathbf{b}_{g_t}\text{-}\mathbf{b}_{g_{t-1}}\end{bmatrix} \tag{5}$$

where $\ominus$ is the minus operation on manifold, which is specially used for non-linear rotation. dt is the time interval between two time instants. g is the known gravity vector, whose norm is around 9.81. Every two adjacent frames construct one IMU factor in the cost function.

3) Other Factors: Though we only specify camera and IMU factors, our system is not limited to these two sensors. Other sensors, such as wheel speedometer, LiDAR and Radar, can be added into our system without much effort. The key is to model these measurements as general residual factors and add these residual factors into cost function.

C. Optimization

In traditional, the nonlinear least square problem of eq. 3 is solved by Newton-Gaussian or Levenberg-Marquardt approaches. The cost function is linearized with respect to an initial guess of states, $\hat{\mathcal{X}}$. Then, the cost function is equals to:

$$\arg\min_{\delta\mathcal{X}}\sum_{t=0}^{n}\sum_{k\in\mathbf{S}}\left\|\mathbf{e}_t^k+\mathbf{J}_t^k\delta\mathcal{X}\right\|_{\boldsymbol{\Omega}_t^k}^2, \tag{6}$$

where $\mathbf{J}$ is the Jacobian matrix of each factor with respect to current states $\hat{\mathcal{X}}$. After linearization approximation, this cost function has closed-form solution of $\delta\mathcal{X}$. We take NewtonGaussian as example, the solution is derived as follows,

$$\underbrace{\sum \sum \mathbf{J}_t^{k^T} \mathbf{\Omega}_t^{k^{-1}} \mathbf{J}_t^k}_{\mathbf{H}} \delta \mathcal{X} = \underbrace{-\sum \sum \mathbf{J}_t^{k^T} \mathbf{\Omega}_t^{k^{-1}} \mathbf{e}_t^k}_{\mathbf{b}}. \tag{7}$$

Finally, current state $\hat{\mathcal{X}}$ is updated with $\hat{\mathcal{X}} \oplus \delta \mathcal{X}$, where $\oplus$ is the plus operation on manifold for rotation. This procedure iterates several times until convergence. We adopt Ceres solver [24] to solve this problem, which utilizes advanced mathematical tools to get stable and optimal results efficiently.

D. Marginalization

Since the number of states increases along with time, the computational complexity will increase quadratically accordingly. In order to bound the computational complexity, marginalization is incorporated without loss of useful information. Marginalization procedure converts previous measurements into a prior term, which reserves past information. The set of states to be marginalized out is denoted as $\mathcal{X}_m$, and the set of remaining states is denoted as $\mathcal{X}_r$. By summing all marginalized factors (eq.7), we get a new $\mathbf{H}$ and $\mathbf{b}$. After rearrange states' order, we get the following relationship:

$$\begin{bmatrix} \mathbf{H}_{mm} & \mathbf{H}_{mr} \\ \mathbf{H}_{rm} & \mathbf{H}_{rr} \end{bmatrix} \begin{bmatrix} \delta \mathcal{X}_m \\ \delta \mathcal{X}_r \end{bmatrix} = \begin{bmatrix} \mathbf{b}_m \\ \mathbf{b}_r \end{bmatrix} \tag{8}$$

The marginalization is carried out using the Schur complement [25] as follows:

$$\underbrace{(\mathbf{H}_{rr} - \mathbf{H}_{rm} \mathbf{H}_{mm}{}^{-1} \mathbf{H}_{mr})}_{\mathbf{H}_p} \delta \mathcal{X}_r = \underbrace{\mathbf{b}_r - \mathbf{H}_{rm} \mathbf{H}_{mm}{}^{-1} \mathbf{b}_m}_{\mathbf{b}_p} \tag{9}$$

We get a new prior $\mathbf{H}_p, \mathbf{b}_p$ for the remaining states. The information about marginalized states is converted into prior term without any loss. To be specific, we keep ten spacial camera frames in our system. When a new keyframe comes, we marginalize out the visual and inertial factors, which are related with states of the first frame.

After we get the prior information about current states, with Bayes' rule, we could calculate the posterior as a product of likelihood and prior: $p(\mathcal{X} \mid \mathbf{z}) \propto p(\mathbf{z} \mid \mathcal{X}) p(\mathcal{X})$. The state estimation then becomes a MAP (Maximum A Posteriori) problem. Denote that we keep states from instant m to instant n in the sliding window. The states before m are marginalized out and converted to a prior term. Therefore, the MAP problem is written as:

$$
\begin{aligned}
X^{*}_{m:n} &= \arg\max_{X_{m:n}} \prod_{t=m}^{n} \prod_{k\in\mathbf{S}} p\left(\mathbf{z}_t^k | X_{m:n}\right) p\left(X_{m:n}\right) \\
&= \arg\min_{X_{m:n}} \sum_{t=m}^{n} \sum_{k\in\mathbf{S}} \left\| \mathbf{z}_t^k - \mathrm{h}_t^k\left(X_{m:n}\right) \right\|^2_{\Omega_t^k} \\
&\quad + \left(\mathbf{H}_p \delta X_{m:n} - \mathbf{b}_p\right)
\end{aligned}
\tag{10}
$$

Compared with eq.3, the above-mentioned equation only adds a prior term. It is solved as same as eq. 3 by Ceres solver [24].

E. Discussion

The proposed system is a general framework. Various sensors can be easily added into our system, as long as it can be derived as a general residual factor. Since our system is not specially designed for a certain sensor, it is capable to handle sensor failure case. When sensor failure occurs, we just remove factors of the inactive sensor and add new factors from other alternative sensors.

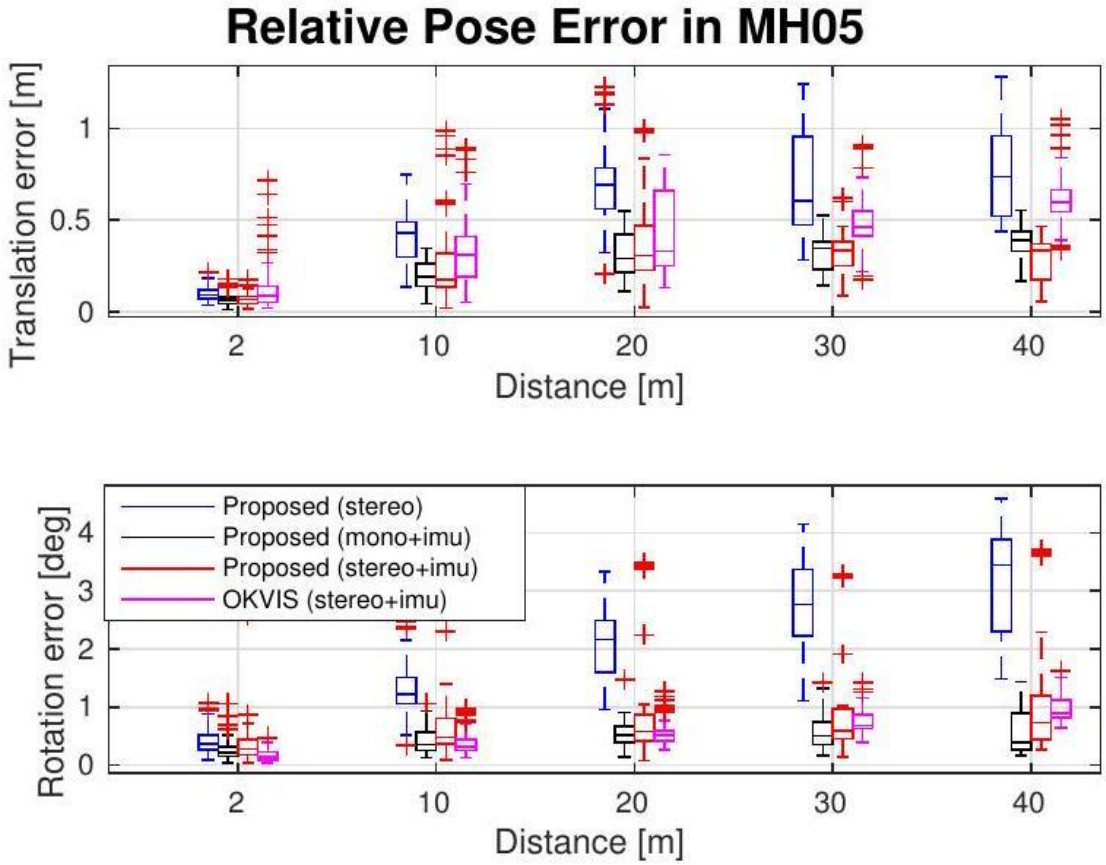

Fig. 4. Relative pose error [26] in MH_05_difficult. Two plots are relative errors in translation and rotation respectively.

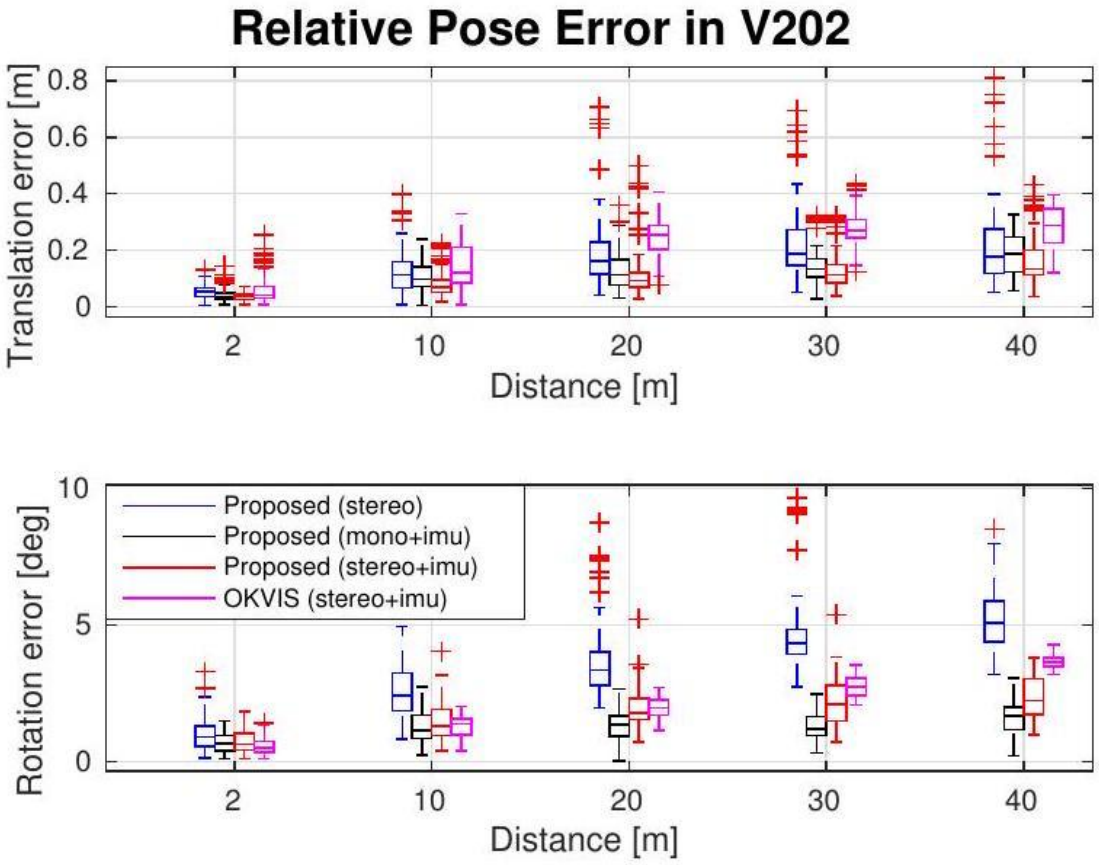

Fig. 5. Relative pose error [26] in V2_02_medium. Two plots are relative errors in translation and rotation respectively.

V. Experimental Results

We evaluate the proposed system with visual and inertial sensors both on datasets and with real-world experiments. In the first experiment, we compare the proposed algorithm with another state-of-the-art algorithm on public datasets. We then test our system in the large-scale outdoor environment. The numerical analysis is generated to show the accuracy of our system in detail.

A. Datasets

We evaluate our proposed system using the EuRoC MAV Visual-Inertial Datasets [27]. This datasets are collected onboard a micro aerial vehicle, which contain stereo images (Aptina MT9V034 global shutter, 752×480 monochrome, 20

TABLE I

RMSE[m] in EuRoC dataset.

Sequence	Length	Proposed RMSE			OKVIS RMSE
		stereo	mono+imu	stereo+imu	
MH_01	79.84	0.54	0.18	0.24	0.16
MH_02	72.75	0.46	0.09	0.18	0.22
MH_03	130.58	0.33	0.17	0.23	0.24
MH_04	91.55	0.78	0.21	0.39	0.34
MH_05	97.32	0.50	0.25	0.19	0.47
V1_01	58.51	0.55	0.06	0.10	0.09
V1_02	75.72	0.23	0.09	0.10	0.20
V1_03	78.77	x	0.18	0.11	0.24
V2_01	36.34	0.23	0.06	0.12	0.13
V2_02	83.01	0.20	0.11	0.10	0.16
V2_03	85.23	x	0.26	0.27	0.29

FPS), synchronized IMU measurements (ADIS16448, 200 Hz), Also, the ground truth states are provided by VICON and Leica MS50. We run datasets with three different combinations of sensors, which are stereo cameras, a monocular camera with an IMU, stereo cameras with an IMU separately.

In this experiment, we compare our results with OKVIS [8], a state-of-the-art VIO that works with stereo cameras and an IMU. OKVIS is another optimization-based sliding-window algorithm. OKVIS is specially designed for visual-inertial sensors, while our system is a more general framework, which supports multiple sensors combinations. We tested the proposed framework and OKVIS with all sequences in EuRoC datasets. We evaluated accuracy by RPE (Rela-

tive Pose Errors) and ATE (Absolute Trajectory Errors). The RPE is calculated by tools proposed in [26]. The RPE (Relative Pose Errors) plot of two sequences, MH_05_difficult and V2_02_medium, are shown in Fig. 4 and Fig. 5 respectively.

The RMSE (Root Mean Square Errors) of ATE for all sequences in EuRoC datasets is shown in Table. I. Estimated trajectories are aligned with the ground truth by Horn's method [28]. The stereo-only case fails in V1_03_difficult and V2_03_difficult sequences, where the movement is too aggressive for visual tracking to survive. Methods which involves the IMU work successfully in all sequences. It is a good case to show that the IMU can dramatically improve motion tracking performance by bridging the gap when visual tracks fail due to illumination change, textureless area, or motion blur.

From the relative pose error and absolute trajectory error, we can see that the stereo-only method performed worst in most sequences. Position and rotation drift obviously grown along with distance in stereo-only case. In other words, the IMU significantly benefited vision in states estimation. Since the IMU measures gravity vector, it can effectively suppress drifts in roll and pitch angles. Stereo cameras with an IMU didn't always perform best, because it requires more accurate calibration than the case of a monocular camera with an IMU. Inaccurate intrinsic and extrinsic calibration will introduce more noise into the system. In general, multiple sensor fusion increase the robustness of the system. Our results outperforms OKVIS in most sequences.

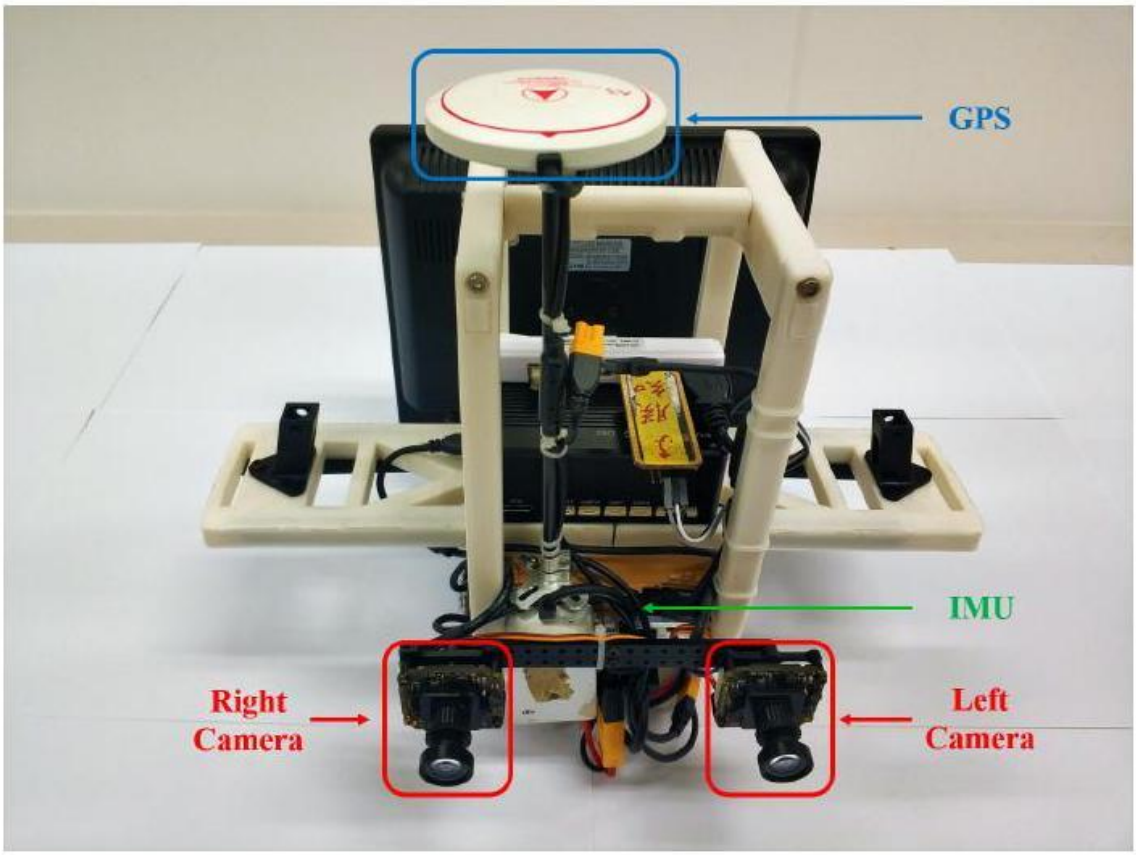

Fig. 6. The self-developed sensor suite used in the outdoor environment. It contains stereo cameras (mvBlueFOX-MLC200w, 20 Hz) and DJI A3 controller, which include inbuilt IMU (200 Hz) and GPS receiver.

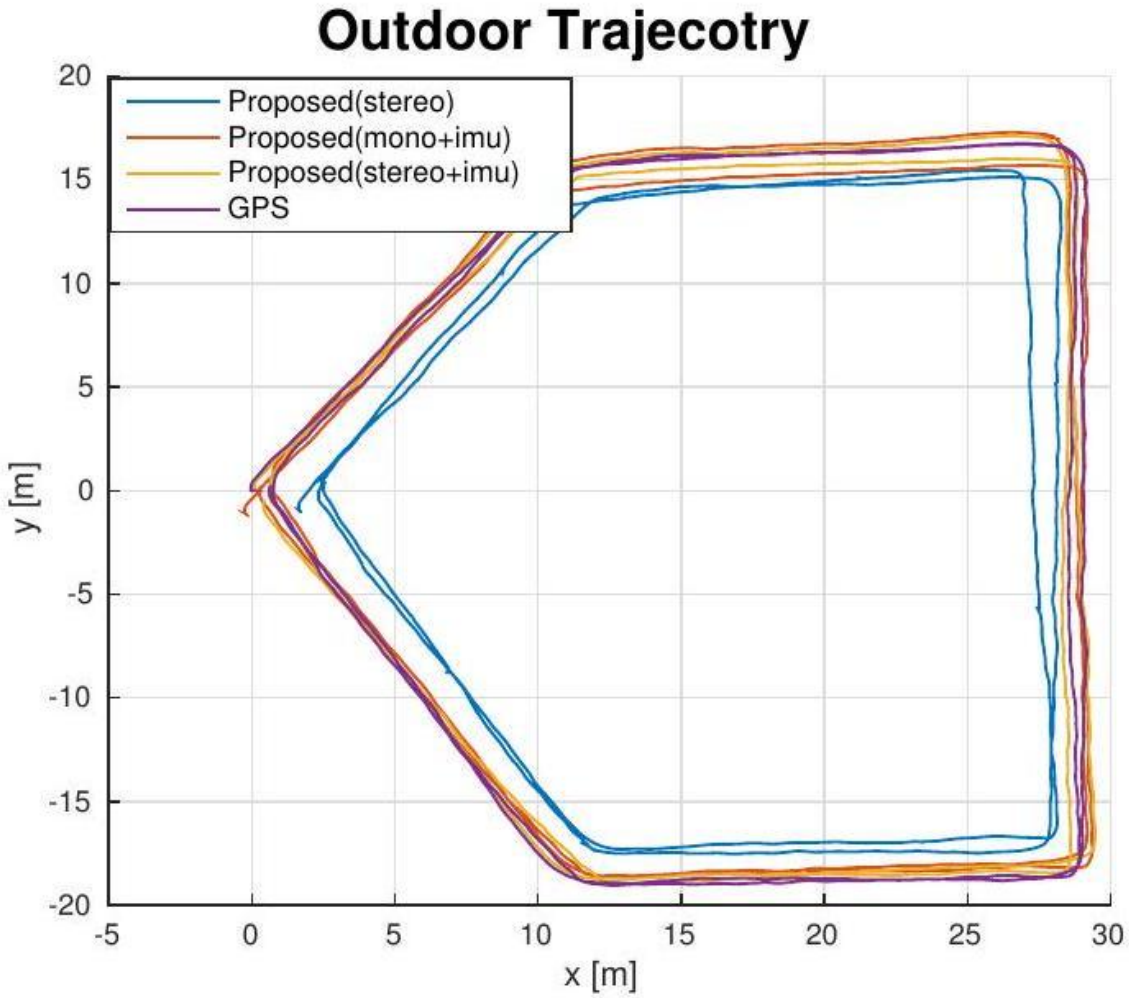

Fig. 7. Estimated trajectories in outdoor experiment.

B. Real-world experiment

In this experiment, we used a self-developed sensor suite to demonstrate our framework. The sensor suite is shown in Fig. 6. It contains stereo cameras (mvBlueFOX-MLC200w, 20 Hz) and DJI A3 controller [2], which inculdes inbuilt IMU（200 Hz）and GPS receiver. The GPS position is treated as ground truth. We hold the sensor suite by hand and walk

[2] http://www.dji.com/a3

Relative Pose Error in Outdoor Dataset

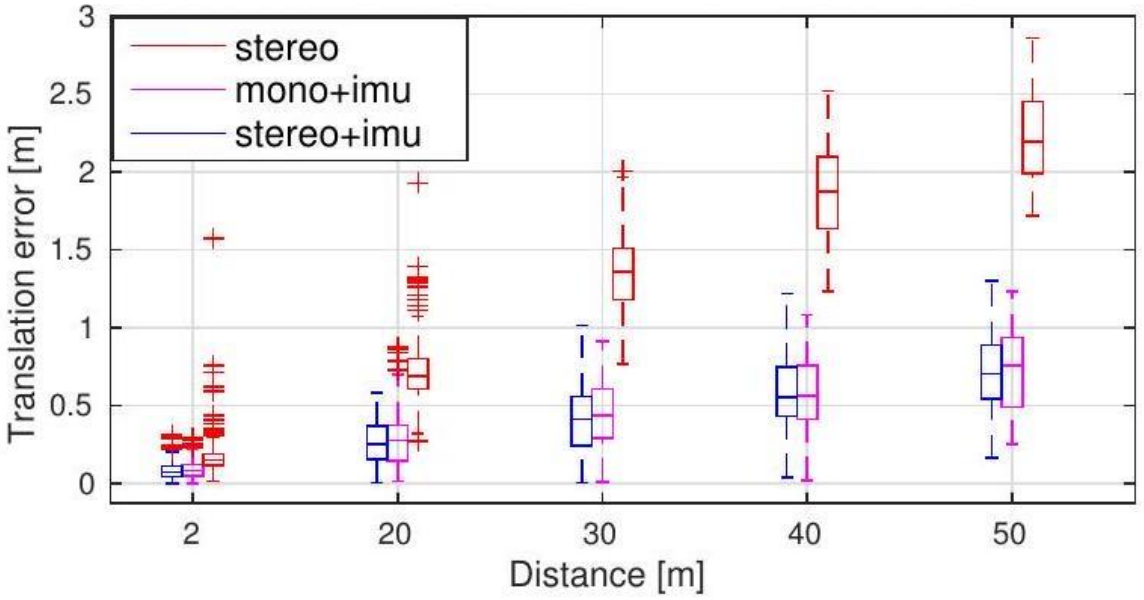

Fig. 8. Relative pose error [26] in outdoor experiment.

around on the outdoor ground. We run states estimation with three different combinations, which are stereo cameras, a monocular camera with an IMU, and stereo cameras with an IMU.

For accuracy comparison, we walked two circles on the ground and compared our estimation with GPS. The trajectory is shown in Fig. 7, and the RPE (Relative Pose Error) is shown in Fig. 8. As same as dataset experiment, noticeable position drifts occurred in the stereo-only scenario. With the assistance of the IMU, the accuracy improves a lot. The RMSE of more outdoor experiments is shown in Table. II. The method which involves the IMU always performs better than the stereo-only case.

VI. Conclusion

In this paper we have presented a general optimizationbased framework for local pose estimation. The proposed framework can support multiple sensor combinations, which is desirable in aspect of robustness and practicability. We further demonstrate it with visual and inertial sensors, which form three sensor suites (stereo cameras, a monocular camera with an IMU, and stereo cameras with an IMU). Note that although we only show the factor formulations for the camera and IMU, our framework can be generalized to other sensors as well. We validate the performance of our system with multiple sensors on both public datasets and real-world experiments. The numerical result indicates that our framework is able to fuse sensor data with different settings.

In future work, we will extend our framework with global sensors (e.g. GPS) to achieve locally accurate and globally aware pose estimation.

TABLE II
RMSE[M] IN OUTDOOR EXPERIMENT.

Sequence	Length	Proposed RMSE		
		stereo	mono+imu	stereo+imu
outdoor1	223.70	1.85	0.71	**0.52**
outdoor2	229.91	2.35	0.56	**0.43**
outdoor3	232.13	2.59	**0.65**	0.75

References

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in Mixed and Augmented Reality, 2007. IEEE and ACM International Symposium on, 2007, pp. 225-234.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in Proc. of the IEEE Int. Conf. on Robot. and Autom., Hong Kong, China, May 2014.

[3] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in European Conference on Computer Vision. Springer International Publishing, 2014, pp. 834-849.

[4] R. Mur-Artal, J. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," IEEE Trans. Robot., vol. 31, no. 5, pp. 1147-1163, 2015.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[6] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in Proc. of the IEEE Int. Conf. on Robot. and Autom., Roma, Italy, Apr. 2007, pp. 3565-3572.

[7] M. Li and A. Mourikis, "High-precision, consistent EKF-based visualinertial odometry," Int. J. Robot. Research, vol. 32, no. 6, pp. 690-711, May 2013.

[8] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," Int. J. Robot. Research, vol. 34, no. 3, pp. 314-334, Mar. 2014.

[9] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst. IEEE, 2015, pp. 298-304.

[10] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 796-803, 2017.

[11] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," IEEE Trans. Robot., vol. 33, no. 1, pp. 1-21, 2017.

[12] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," IEEE Trans. Robot., vol. 34, no. 4, pp. 1004-1020, 2018.

[13] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in realtime." in Robotics: Science and Systems, vol. 2, 2014, p. 9.

[14] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst.

[15] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza, "Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time," IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 593-600, 2017.

[16] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst. IEEE, 2013, pp. 3923-3929.

[17] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Observabilitybased rules for designing consistent ekf slam estimators," Int. J. Robot. Research, vol. 29, no. 5, pp. 502-528, 2010.

[18] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in Proc. of the IEEE Int. Conf. on Robot. and Autom., Hong Kong, China, May 2014, pp. 4974-4981.

[19] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in Proc. of the IEEE Int. Conf. on Robot. and Autom. IEEE, 2011, pp. 3607-3613.

[20] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," Int. J. Robot. Research, vol. 31, no. 2, pp. 216-235, 2012.

[21] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam," in Proc. of the IEEE Int. Conf. on Pattern Recognition, 2018, pp. 19741982.

[22] J. Shi and C. Tomasi, "Good features to track," in Computer Vision and Pattern Recognition, 1994. IEEE Computer Society Conference on, 1994, pp. 593-600.

[23] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in Proc. of the Intl. Joint Conf. on Artificial Intelligence, Vancouver, Canada, Aug. 1981, pp. 24-28.

[24] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver. org.

[25] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," J. Field Robot., vol. 27, no. 5, pp. 587-608, Sep. 2010.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in Proc. of the IEEE Int. Conf. on Pattern Recognition, 2012, pp. 3354-3361.

[27] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," Int. J. Robot. Research, 2016.

[28] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," JOSA A, vol. 4, no. 4, pp. 629-642, 1987.

All authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. {tong.qin, jie.pan, shaozu.cao}@connect.ust.hk, eeshaojie@ust.hk.

[1] https://github.com/HKUST-Aerial-Robotics/VINS-Fusion