Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Stereo Visual-Inertial Fusion for UAV State Estimation

Jinyao Zhu * **Chao Yao** * **Klaus Janschek** *

* *Institute of Automation, Technische Universität Dresden, Germany (e-mail: jinyao.zhu@mailbox.tu-dresden.de, {chao.yao, klaus.janschek}@tu-dresden.de)*

Abstract: Visual-inertial fusion is frequently used for state estimation in aerial robotic applications due to the low-cost, simple hardware setup as well as the high accuracy. This work proposes a stereo visual-inertial fusion system based on the monocular method VINS-Mono, which tightly combines the visual and inertial measurements. Timing statistics are provided for the system running on an Intel NUC Mini-PC. The system real-time capability fulfills the requirements of the closed-loop control for a UAV. The proposed fusion system is evaluated in the public EuRoC MAV dataset and compared with several representative state-of-the-art open-sourced state estimators. According to the results, our method achieves competitive performance with relative low estimation errors in a computationally efficient manner.

Keywords: State Estimation, Sensor Fusion, Visual Inertial Odometry, Aerial Robot

1. INTRODUCTION

State estimation plays a significant role in the navigation and control of aerial robotics. Because of the absence of absolute positioning measurements such as GPS or motion tracking system in some challenging environments, state estimation is a difficult task for unmanned aerial vehicles (UAVs). Dead reckoning based on the inertial measurement unit (IMU) can provide a high sample rate but suffers from accumulative integration error. On the other hand, visual odometry (VO) can estimate the camera motion with low drifts. Yet, only with low image update rate and limited robustness, VO is usually not sufficient for the low-level control of a UAV, which requires a high update rate and smooth state feedback. It comes naturally to combine both inertial and visual measurements in the state estimation problem, thus compensate for the drawback of each sensor. This type of system is named visual-inertial odometry (VIO).

Depending on the camera setup, VIO systems are categorized into monocular and stereo VIOs. In monocular case, additional inertial measurement eliminates the scale ambiguity of a pure mono VO system, hence forms a minimal suite for metric scale 6D pose estimation. A benchmark comparison of mono VIOs is provided by (Delmerico and Scaramuzza, 2018). In monocular system, degenerated motion modes of the UAV such as pure rotation motion, or hovering which causes the lack of visual cue parallax, can introduce substantial uncertainties in the visual landmark triangulation, thus easily lead to divergence of the system state. Moreover, due to the lack of direct distance measurement and prior scale information, mono system usually needs rigorous system bootstrap process which makes it more difficult to be applied to the UAV applications. Stereo VIO, on the other hand, overcomes above problems by providing redundant visual sensing, thus can triangulate 3D landmarks directly and is independent of the UAV flight modes, making it more suitable for UAV state estimation.

Several VIO frameworks with promising performance are proposed in recent years: The open keyframe-based visual-inertial SLAM (OKVIS) is an tightly-coupled optimization-based visual-SLAM framework from (Leutenegger et al., 2015), which is confirmed to have a relatively high computational cost both in (Delmerico and Scaramuzza, 2018) and our tests. VINS-Mono (Qin et al., 2018) is also a tightly-coupled optimization-based VIO framework. It provides a robust initialization procedure that is able to bootstrap the system from an unknown initial state. It includes re-localization and loop closure modules, which provides the possibility to achieve a drift-free pose estimation. Since it only supports monocular cameras, VINS-Mono could potentially suffer from the previously mentioned degenerated motion modes. VINS-Fusion (Qin et al., 2019) is an extension of VINS-Mono prepublished during the mid-term progress of this work. The authors aim at providing a general framework that supports multiple sensor combinations. Multi-State Constraint Kalman Filter (MSCKF) for vision-aided inertial navigation, which is tightly-coupled and filter-based, is proposed by (Mourikis and Roumeliotis, 2007). (Sun et al., 2018) presented a stereo implementation based on MSCKF called S-MSCKF with a noticeable small computational cost. However, according to our practical tests of the open-sourced implementation, the system states tend to become divergent in specific motion modes such as pure-rotation or fast motion, and reveals a relative inferior robustness in comparison to other optimization-based framework.

To this end, we summarize our contributions as follows:

(1) We propose a stereo visual-inertial fusion system based on the framework of VINS-Mono: We redesign its image processing system front-end which supports

stereo camera while taking advantage of inertial measurements. We introduce a stereo visual residual term into the local bundle adjustment problem. We simplify the initialization procedure by taking advantage of the stereo visual measurement.

(2) The real-time capability is proved by providing timing statistics of the proposed system recorded on an Intel® NUC computer.

The rest of the paper is structured as follows. Processing steps of the visual-inertial fusion are introduced in Section 2. In Section 3, we present the experiment results of timing statistics and accuracy evaluation. Finally, the paper is concluded in Section 4.

2. SYSTEM DESIGN

2.1 Image Processing

Visual features are detected and tracked in the system image processing front-end, then tracked stereo features are published to the system back-end.

For feature detection, features in left images are detected using the Shi-Tomasi Corner detector. The number of features in each frame is limited by a maximum feature number (typ. 100 – 200). The features are enforced to obey a uniform distribution by setting circle mask around corner with a minimum pixel distance. For feature tracking, the pixel coordinates of each 2D feature on the latest left image will firstly be predicted, using the IMU integration during the update interval of consecutive image frames, to give a good initial guess. The rotation of the body(IMU) frame between two camera frames is calculated as

$$ {}^{B_k}_{B_{k+1}}\boldsymbol{R} = \int_{t_k}^{t_{k+1}} {}^{B_k}_{t}\boldsymbol{R}\,[\tilde{\boldsymbol{\omega}}_b - \boldsymbol{b}_{\omega_b}]_\times \quad dt \tag{1} $$

with ${}^{B_k}_{t}\boldsymbol{R} \in \mathrm{SO}(3)$ being the rotation of the body frame $\{\mathrm{B}\}$ at time t w.r.t. the body frame $\{\mathrm{B_k}\}$ which is at time t_k. $\tilde{\boldsymbol{\omega}}_b$ is the body angular velocity measured by IMU, and $\boldsymbol{b}_{\omega_b}$ is gyroscope bias. $[\boldsymbol{v}]_\times$ denotes the 3×3 screw-symmetric matrix from the 3D vector $\boldsymbol{v}$. Under the assumption that the angular velocity remains constant during the IMU sampling interval $[t_i, t_{i+1}]$, the rotation between two IMU measurements can be modeled discretely as (Forster et al., 2015):

$$ {}^{B_k}_{t_{i+1}}\boldsymbol{R} \approx {}^{B_k}_{t_i}\boldsymbol{R}\left(\boldsymbol{I} + [\Delta t\,(\tilde{\boldsymbol{\omega}}_{b_i} - \boldsymbol{b}_{\omega_b})]_\times\right) \tag{2} $$

where t_i is the time index of the IMU measurement within the camera sampling interval $[t_k, t_{k+1}]$, and $\Delta t = t_{i+1} - t_i$ is the IMU sampling time interval. Assuming the IMU is synchronized with the camera, i.e. ${}^{B_k}_{t_0}\boldsymbol{R} = \boldsymbol{I}$, then (1) can be approximated using discrete inertial measurements:

$$ {}^{B_k}_{B_{k+1}}\boldsymbol{R} \approx \prod_{i=0}^{i=N-1} \left(\boldsymbol{I} + [\Delta t\,(\tilde{\boldsymbol{\omega}}_{b_i} - \boldsymbol{b}_{\omega_b})]_\times\right) \tag{3} $$

where N is the number of IMU measurements in $[t_k, t_{k+1}]$. The rotation of the camera is computed with

$$ {}^{C_{k+1}}_{C_k}\boldsymbol{R} = {}^{C_l}_{B}\boldsymbol{R}\,{}^{B_{k+1}}_{B_k}\boldsymbol{R}\,{}^{B}_{C_l}\boldsymbol{R} = {}^{B}_{C_l}\boldsymbol{R}^\top\,{}^{B_k}_{B_{k+1}}\boldsymbol{R}^\top\,{}^{B}_{C_l}\boldsymbol{R} \tag{4} $$

with ${}^{B}_{C_l}\boldsymbol{R}$ being the relative rotation between the body frame $\{\mathrm{B}\}$ and the left camera frame $\{\mathrm{C_l}\}$ w.r.t $\{\mathrm{B}\}$.

Features from last image are projected onto the latest image with the following equation (without considering the relative translation):

$$ \begin{bmatrix} \hat{u}_i^{C_{k+1}} \\ \hat{v}_i^{C_{k+1}} \end{bmatrix} = \pi_{C_l}\left({}^{C_{k+1}}_{C_k}\boldsymbol{R}\,\pi_{C_l}^{-1}\left(\begin{bmatrix} u_i^{C_k} \\ v_i^{C_k} \end{bmatrix}\right)\right) \tag{5} $$

where $\left[u_i^{C_k}, v_i^{C_k}\right]^\top$ is the pixel coordinate of the i-th feature in image frame C_k, and $\left[\hat{u}_i^{C_{k+1}}, \hat{v}_i^{C_{k+1}}\right]^\top$ is the corresponding predicted location in image frame C_{k+1}. $\pi_{C_l}(\cdot)$ is the projection function of left camera which maps a point from normalized image plane onto image plane.

Having prediction from IMU measurements, KLT algorithm is then applied to refine the position of each feature on the latest image: Two time-successive left images and the pixel coordinates of each feature initialized with the IMU prediction (5) are input to the KLT algorithm.

Subsequently, stereo matching is performed on the latest stereo image pair to find the correspondences of the current tracked features: Firstly, the features from the left image are projected (ignoring the relative translation) onto the right image with:

$$ \begin{bmatrix} \hat{u}_i^{C_r} \\ \hat{v}_i^{C_r} \end{bmatrix} = \pi_{C_r}\left({}^{C_r}_{C_l}\boldsymbol{R}\,\pi_{C_l}^{-1}\left(\begin{bmatrix} u_i^{C_l} \\ v_i^{C_l} \end{bmatrix}\right)\right) \tag{6} $$

where $\left[u_i^{C_l}, v_i^{C_l}\right]^\top$ and $\left[\hat{u}_i^{C_r}, \hat{v}_i^{C_r}\right]^\top$ are pixel coordinates of the i-th feature on the left image and its prediction on the right image. ${}^{C_r}_{C_l}\boldsymbol{R}$ is the rotation matrix between the left $\{\mathrm{C_l}\}$ and right camera $\{\mathrm{C_r}\}$ w.r.t. $\{\mathrm{C_r}\}$.

Since prediction from (6) only compensates for the rotation between the cameras, a relative large prediction error may still exist, especially for the features in the near view. In order to achieve a more precise prediction, an offset in pixel coordinate is heuristically added to each predicted position:

$$ \begin{bmatrix} u_\Delta \\ v_\Delta \end{bmatrix} = K_s \bar{d}\left(\frac{{}^{C_r}\boldsymbol{t}_{C_l}}{\|{}^{C_r}\boldsymbol{t}_{C_l}\|}\right)_{xy} \tag{7} $$

where K_s is a heuristically selected proportional coefficient, $\bar{d}$ is the average disparity computed from the tracked features from the last stereo image pairs, and ${}^{C_r}\boldsymbol{t}_{C_l}$ is the translation between the left and right camera. $(\cdot)_{xy}$ indicates that only x, y components of the vector are used. KLT algorithm is performed to refine the feature positions on the right image, similar to that for the feature tracking on the left image, while treating the predicted right position using (6) and (7) as the initial guess.

In addition, thresholding based on epipolar constraint is performed between the left and right images for preliminary outlier rejection.

All features on the left and right images are undistorted and projected onto the normalized image plane with

$$ \mathbf{p}_i^{C_l} = \pi_{C_l}\left(\begin{bmatrix} u_i^{C_l} \\ v_i^{C_l} \end{bmatrix}\right) \quad \mathbf{p}_i^{C_r} = \pi_{C_r}\left(\begin{bmatrix} u_i^{C_r} \\ v_i^{C_r} \end{bmatrix}\right) \tag{8} $$

where $\mathbf{p}_i^{C_l}, \mathbf{p}_i^{C_r} \in \left\{\mathbb{R}^3 | z = 1\right\}$ are coordinates on the normalized image plane of the left and right camera. The essential matrix $\boldsymbol{E}$ between left and right camera is computed with the camera extrinsic parameters:

$$\boldsymbol{E} = \left[{}^{C_r}\boldsymbol{t}_{C_l}\right]_\times {}^{C_r}_{C_l}\boldsymbol{R} \ . \tag{9}$$

The epipolar line $\boldsymbol{l}_i^{C_r} \in \mathbb{P}^2$ (Hartley and Zisserman, 2004, p.26) corresponding to the i-th feature in the right camera is

$$\boldsymbol{l}_i^{C_r} = \boldsymbol{E}\mathbf{p}_i^{C_l} \ . \tag{10}$$

In ideal case, the corresponding feature seen by the right camera will lie exactly on the epipolar line, such that $\mathbf{p}_i^{C_r\top}\boldsymbol{l}_i^{C_r} = 0$. In practice, due to noise or incorrect matching, $\mathbf{p}_i^{C_r}$ probably locates outside of the epipolar line. To quantify the error, the distance between point $\mathbf{p}_i^{C_r}$ and line $\boldsymbol{l}_i^{C_r}$ is computed:

$$d_i = \mathbf{p}_i^{C_r\top}\tilde{\boldsymbol{l}}_i^{C_r} \tag{11}$$

where $\tilde{\boldsymbol{l}}_i^{C_r}$ is the epipolar line in normalized homogeneous coordinate and can be computed with:

$$\tilde{\boldsymbol{l}}_i^{C_r} = \frac{\boldsymbol{l}_i^{C_r}}{\sqrt{a^2+b^2}}, \quad \text{with} \quad \boldsymbol{l}_i^{C_r} = [a \ b \ c]^\top . \tag{12}$$

Then d_i is converted into the pixel unit:

$$d_{i,pixel} = d_i \cdot f_{xy}^{C_r} \tag{13}$$

where $f_{xy}^{C_r} = \frac{f_x^{C_r}+f_y^{C_r}}{2}$ is the averaged focal length factor of the right camera.

Finally, each feature from the stereo matching is thresholded with a given maximum pixel error:

$$d_{i,pixel} < d_{max,pixel} \ . \tag{14}$$

Features that do not satisfy (14) will be marked as outliers.

To further remove outliers from the matched feature pairs, fundamental matrix based RANSAC algorithm is performed between the latest two time-successive left/right image frames and the latest stereo pairs.

2.2 Optimization

Optimization system back-end subscribes stereo features (expressed in normalized image plane) from system front-end and performs joint optimization in sliding window fashion.

Triangulation Tracked 2D features are triangulated with Direct Linear Transformation (DLT) method to obtain their corresponding 3D points. Let $\mathbf{p}, \mathbf{p'}$ be two 2D measurements on the normalized image planes of two cameras from the projection of a single 3D point $\boldsymbol{p}$ in a global frame, which have $\mathbf{p} = \boldsymbol{M}\boldsymbol{p}$ and $\mathbf{p'} = \boldsymbol{M'}\boldsymbol{p}$, $\boldsymbol{M}, \boldsymbol{M'}$ are the 3×4 projection matrices of the respective left/right camera without considering the camera intrinsic, and all points are expressed in homogeneous coordinates. These two equations are combined into a form $\boldsymbol{A}\boldsymbol{p} = \mathbf{0}$, and $\boldsymbol{A} \in \mathbb{R}^{4\times4}$, such that:

$$\boldsymbol{A} = \begin{bmatrix} x\boldsymbol{m}^{3\top} - \boldsymbol{m}^{1\top} \\ y\boldsymbol{m}^{3\top} - \boldsymbol{m}^{2\top} \\ x'\boldsymbol{m}'^{3\top} - \boldsymbol{m}'^{1\top} \\ y'\boldsymbol{m}'^{3\top} - \boldsymbol{m}'^{2\top} \end{bmatrix} \tag{15}$$

where $\boldsymbol{m}^{i\top}, \boldsymbol{m}'^{i\top}$ is the i-th row of $\boldsymbol{M}, \boldsymbol{M'}$, and $\mathbf{p} = [x, y, 1]^\top$, $\mathbf{p'} = [x', y', 1]^\top$. In case of multiple (n) observations of a feature, we combine all measurements and solving the null space for $\left[\boldsymbol{A}_1^\top, \cdots, \boldsymbol{A}_n^\top\right]^\top \boldsymbol{p} = \mathbf{0}$ using SVD which can achieve a more accurate triangulation result.

Bundle Adjustment Once a new frame of visual measurement (tracked 3D features) is available, local bundle adjustment (BA) is performed in sliding window fashion.

The system state vector is defined as

$$\boldsymbol{s} = \left[\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N, {}^{B}_{C_l}\boldsymbol{x}, {}^{B}_{C_r}\boldsymbol{x}, \lambda_0, \lambda_1, \cdots, \lambda_{m-1}\right] \tag{16}$$

$$\boldsymbol{x}_k = \left[{}^{W}\boldsymbol{t}_{B_k}, {}^{W}\boldsymbol{v}_{B_k}, {}^{W}_{B_k}\boldsymbol{q}, \boldsymbol{b}_{a_k}, \boldsymbol{b}_{\omega_k}\right], \ k = 0, \ldots, N \tag{17}$$

with ${}^{B}_{C_l}\boldsymbol{x} = \left[{}^{B}\boldsymbol{t}_{C_l}, {}^{B}_{C_l}\boldsymbol{q}\right]$, ${}^{B}_{C_r}\boldsymbol{x} = \left[{}^{B}\boldsymbol{t}_{C_r}, {}^{B}_{C_r}\boldsymbol{q}\right]$ are extrinsic parameter of the stereo camera. ${}^{A}\boldsymbol{t}_B$, ${}^{A}\boldsymbol{v}_B$ denote the translation and velocity vectors of the origin of frame {A} w.r.t. frame {B}. ${}^{A}_{B}\boldsymbol{q}$ is the unit quaternion corresponding to the rotation matrix ${}^{A}_{B}\boldsymbol{R}$. $\boldsymbol{b}_{a_i}, \boldsymbol{b}_{\omega_i}$ are accelerometer and gyroscope bias. λ_i is the inverse depth of the i-th feature w.r.t. the coordinate frame of the left camera from its first observation. N is the number of keyframes in the sliding window, and m is the total number of features that can be observed from the current sliding window.

The cost function for the local bundle adjustment is defined similar to (Qin et al., 2018):

$$\begin{aligned} \min_{\boldsymbol{s}} \Bigg\{ & \left\|\boldsymbol{r}_{\mathcal{M}}(\boldsymbol{s})\right\|^2 + \sum_{k\in\mathcal{B}} \left\|\boldsymbol{r}_{\mathcal{B}}\left(\hat{\boldsymbol{z}}_{B_{k+1}}^{B_k}, \boldsymbol{s}\right)\right\|^2_{\boldsymbol{\Omega}_{imu}} \\ & + \sum_{(i,j)\in\mathcal{C},k\in\mathcal{F}} \rho\left(\left\|\boldsymbol{r}_{\mathcal{C}}\left(\hat{\boldsymbol{z}}_k^{C_{li}}, \hat{\boldsymbol{z}}_k^{C_{rj}}, \boldsymbol{s}\right)\right\|^2_{\boldsymbol{\Omega}_{cam}}\right) \Bigg\} \end{aligned} \tag{18}$$

where $\boldsymbol{r}_{\mathcal{M}}(\cdot)$ is the prior information from marginalization based on Schur complement and $\boldsymbol{r}_{\mathcal{B}}(\cdot)$ is the IMU residual from the set of all inertial measurements $\mathcal{B}$ in $[t_k, t_{k+1}]$, these terms are same as in VINS-Mono. $\boldsymbol{r}_{\mathcal{C}}(\cdot)$ is the visual measurement residual from set of all features $\mathcal{F}$ and cameras $\mathcal{C}$ in current sliding window. And $\|\cdot\|_{\boldsymbol{\Omega}}$ denotes the mahalanobis norm with the corresponding covariance matrix $\boldsymbol{\Omega}$, $\rho(\cdot)$ denotes the Cauchy loss function which is defined as

$$\rho(s) = \log(1+s) \ . \tag{19}$$

IMU measurement residual is defined as

$$\boldsymbol{r}_{\mathcal{B}} = \begin{bmatrix} {}^{B_k}_{W}\boldsymbol{R}\left(\Delta^W\boldsymbol{t} + \frac{1}{2}\boldsymbol{g}_W\Delta t_k^2 - {}^{W}\boldsymbol{v}_{B_k}\Delta t_k\right) - {}^{B_k}\hat{\boldsymbol{\alpha}}_{B_{k+1}} \\ {}^{B_k}_{W}\boldsymbol{R}\left({}^{W}\boldsymbol{v}_{B_{k+1}} + \boldsymbol{g}_W\Delta t_k - {}^{W}\boldsymbol{v}_{B_k}\right) - {}^{B_k}\hat{\boldsymbol{\beta}}_{B_{k+1}} \\ 2\left({}^{W}_{B_k}\boldsymbol{q}^{-1} \otimes {}^{W}_{B_{k+1}}\boldsymbol{q} \otimes {}^{B_k}_{B_{k+1}}\hat{\boldsymbol{\gamma}}^{-1}\right)_{xyz} \\ \Delta\boldsymbol{b}_a \\ \Delta\boldsymbol{b}_\omega \end{bmatrix} \tag{20}$$

where $\Delta x = x_{k+1} - x_k$. $(\cdot)_{xyz}$ denotes the vector part of a unit quaternion, and $\otimes$ the quaternion multiplication. ${}^{B_k}\hat{\boldsymbol{\alpha}}_{B_{k+1}}$, ${}^{B_k}\hat{\boldsymbol{\beta}}_{B_{k+1}}$, and ${}^{B_k}_{B_{k+1}}\hat{\boldsymbol{\gamma}}$ are preintegration terms that directly computed from the IMU measurements following (Shen et al., 2015) to avoid the computationally intensive re-propagation:

$${}^{B_k}\hat{\boldsymbol{\alpha}}_{i+1} = {}^{B_k}\hat{\boldsymbol{\alpha}}_i + {}^{B_k}\hat{\boldsymbol{\beta}}_i\Delta t_i + \frac{1}{2}\mathbf{R}\left({}^{B_k}_{i}\hat{\boldsymbol{\gamma}}\right)\left(\tilde{\boldsymbol{a}}_{b_i} - \boldsymbol{b}_{a_k}\right)\Delta t_i^2 \tag{21}$$

$${}^{B_k}\hat{\boldsymbol{\beta}}_{i+1} = {}^{B_k}\hat{\boldsymbol{\beta}}_i + \mathbf{R}\left({}^{B_k}_{i}\hat{\boldsymbol{\gamma}}\right)\left(\tilde{\boldsymbol{a}}_{b_i} - \boldsymbol{b}_{a_k}\right)\Delta t_i \tag{22}$$

$${}^{B_k}_{i+1}\hat{\boldsymbol{\gamma}} = {}^{B_k}_{i}\hat{\boldsymbol{\gamma}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\left(\tilde{\boldsymbol{\omega}}_{b_i} - \boldsymbol{b}_{\omega_k}\right)\Delta t_i \end{bmatrix} \tag{23}$$

where i is the discrete time index of the IMU measurements in the visual measurements update time interval $[t_k, t_{k+1}]$, $\mathbf{R}(\cdot)$ denotes the operation that converts a unit

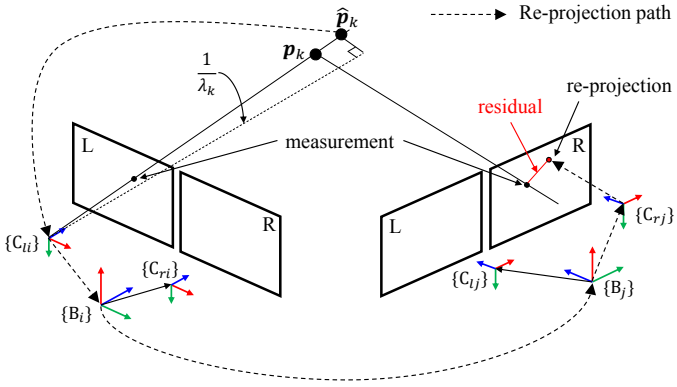

Fig. 1. Re-projection error of the visual measurements.

quaternion into corresponding rotation matrix. $\tilde{\boldsymbol{a}}_{b_i}$ and $\tilde{\boldsymbol{\omega}}_{b_i}$ are acceleration and angular velocity directly from the IMU outputs at time t_i.

Visual measurement residual is calculated with 2D measurements from both the left and right cameras. Features observed by i-th left camera are projected into the j-th right camera. Figure 1 describes the re-projection process. Assuming that the k-th feature is observed in i-th camera frame, the residual of the feature in the j-th camera frame is defined as:

$$\boldsymbol{r}_{\mathcal{C}}\left(\hat{\boldsymbol{z}}_k^{C_{li}}, \hat{\boldsymbol{z}}_k^{C_{rj}}, \boldsymbol{s}\right) = \left(\frac{\hat{\mathbf{p}}_k^{C_{rj}}}{\hat{\mathbf{p}}_{k,z}^{C_{rj}}} - \tilde{\mathbf{p}}_k^{C_{rj}}\right)_{xy} \tag{24}$$

$$\tilde{\mathbf{p}}_k^{C_{li}} = \pi_{C_l}^{-1}\left(\begin{bmatrix} u_k^{C_{li}} \\ v_k^{C_{li}} \end{bmatrix}\right) \tag{25}$$

$$\tilde{\mathbf{p}}_k^{C_{rj}} = \pi_{C_r}^{-1}\left(\begin{bmatrix} u_k^{C_{rj}} \\ v_k^{C_{rj}} \end{bmatrix}\right) \tag{26}$$

$$\hat{\mathbf{p}}_k^{C_{rj}} = {}_{C_r}^{B}\boldsymbol{R}^{\top}\left({}_{B_j}^{W}\boldsymbol{R}^{\top}\left({}_{B_i}^{W}\boldsymbol{R}\left({}_{C_l}^{B}\boldsymbol{R}\frac{1}{\lambda_k}\tilde{\mathbf{p}}_k^{C_{li}} + {}^{B}\boldsymbol{t}_{C_l}\right) + {}^{W}\boldsymbol{t}_{B_i} - {}^{W}\boldsymbol{t}_{B_j}\right) - {}^{B}\boldsymbol{t}_{C_r}\right) \tag{27}$$

where $\tilde{\mathbf{p}}_k^{C_{li}} \in \mathbb{R}^3$ is the coordinates of the k-th feature on the normalized image plane of the i-th left camera directly from the visual measurement. Similarly, $\tilde{\mathbf{p}}_k^{C_{rj}}$ is the feature observed by j-th frame of right camera. $\hat{\mathbf{p}}_k^{C_{rj}}$ is the predicted feature position in j-th right image based on the visual measurement from i-th left image and the system state.

2.3 Initialization

At system start up, the optimization framework needs an initial guess of the state vector letting the system converge into a correct state. When the number of sampled frames reaches the sliding window's size, all observed features are triangulated using the methodology described in Section 2.2, using all stereo frames. Recovered 3D features which can be seen by all camera frames in the sliding window are projected onto each left camera's normalized image plane, the corresponding camera pose is then recovered using the Perspective-n-Point (PnP) algorithm (Lepetit et al., 2009) with RANSAC. With the recovered left camera poses, the pose of IMU frame $\{\mathrm{B_k}\}$ are computed with:

$${}_{B_k}^{C_{l0}}\boldsymbol{q} = {}_{C_{lk}}^{C_{l0}}\boldsymbol{q} \otimes {}_{C_l}^{B}\boldsymbol{q}^{-1} \tag{28}$$

$${}^{C_{l0}}\boldsymbol{t}_{B_k} = {}^{C_{l0}}\boldsymbol{t}_{C_{lk}} - {}_{B_k}^{C_{l0}}\boldsymbol{R}\,{}^{B}\boldsymbol{t}_{C_l}, \quad k = 0, 1 \ldots N. \tag{29}$$

After recovering the coarse poses. We further estimate the gyroscope bias using same formulations described in (Qin et al., 2018).

In the next step, we estimate the velocity vectors in the sliding window without accounting the metric scale, since the scale information is already embedded in the stereo visual measurements. It also implies that our system is able to be initialized from a static state without any specific sensor excitation movement which is required by monocular systems. Moreover, the gravity vector is also taken into account to give an initialization for the world frame $\{\mathrm{W}\}$. The parameter vector is defined as

$$\boldsymbol{s}_I = \left[\boldsymbol{v}_{B_0}^{\top}, \boldsymbol{v}_{B_1}^{\top}, \cdots, \boldsymbol{v}_{B_N}^{\top}, {}^{C_0}\boldsymbol{g}^{\top}\right]^{\top} \tag{30}$$

where ${}^{C_0}\boldsymbol{g} = {}^{C_{l0}}\boldsymbol{g}$ represents the gravity vector in $\{\mathrm{C_{l0}}\}$. $\boldsymbol{v}_{B_i}$ are body velocities expressed in body frame $\{\mathrm{B_i}\}$. The measurement model is given by

$$\begin{aligned}\boldsymbol{\alpha}_{B_{k+1}} &= {}_{C_0}^{B_k}\boldsymbol{R}\left({}^{C_0}\Delta\boldsymbol{t}_B + \frac{1}{2}{}^{C_0}\boldsymbol{g}\Delta t_k^2 - {}_{B_k}^{C_0}\boldsymbol{R}\,\boldsymbol{v}_{B_k}\Delta t_k\right) \\ &= {}_{C_0}^{B_k}\boldsymbol{R}\,{}^{C_0}\Delta\boldsymbol{t}_B - \boldsymbol{v}_{B_k}\Delta t_k + \frac{1}{2}{}_{C_0}^{B_k}\boldsymbol{R}\,\Delta t_k^2{}^{C_0}\boldsymbol{g}\end{aligned} \tag{31}$$

$$\begin{aligned}\boldsymbol{\beta}_{B_{k+1}} &= {}_{C_0}^{B_k}\boldsymbol{R}\left({}_{B_{k+1}}^{C_0}\boldsymbol{R}\boldsymbol{v}_{B_{k+1}} - {}_{B_k}^{C_0}\boldsymbol{R}\boldsymbol{v}_{B_k} + {}^{C_0}\boldsymbol{g}\Delta t_k\right) \\ &= -\boldsymbol{v}_{B_k} + {}_{C_0}^{B_k}\boldsymbol{R}\,{}_{B_{k+1}}^{C_0}\boldsymbol{R}\boldsymbol{v}_{B_{k+1}} + {}_{C_0}^{B_k}\boldsymbol{R}\,\Delta t_k{}^{C_0}\boldsymbol{g}\end{aligned} \tag{32}$$

with ${}^{C_0}\Delta\boldsymbol{t}_B = {}^{C_0}\boldsymbol{t}_{B_{k+1}} - {}^{C_0}\boldsymbol{t}_{B_k}$. The left side of the equations are the IMU preintegration terms that can be computed directly with (21) and (22). Rewrite (31) and (32) into the following form:

$$\hat{\boldsymbol{z}}_{B_{k+1}}^{B_k} = \begin{bmatrix}\hat{\boldsymbol{\alpha}}_{B_{k+1}} - {}_{C_0}^{B_k}\boldsymbol{R}\,{}^{C_0}\Delta\boldsymbol{t}_B \\ \hat{\boldsymbol{\beta}}_{B_{k+1}}\end{bmatrix} = \boldsymbol{H}_{B_{k+1}}^{B_k}\boldsymbol{s}_{I_k} \tag{33}$$

where

$$\boldsymbol{H}_{B_{k+1}}^{B_k} = \begin{bmatrix}-\boldsymbol{I}\Delta t_k & \boldsymbol{0} & \frac{1}{2}{}_{C_0}^{B_k}\boldsymbol{R}\,\Delta t_k^2 \\ -\boldsymbol{I} & {}_{C_0}^{B_k}\boldsymbol{R}\,{}_{B_{k+1}}^{C_0}\boldsymbol{R} & {}_{C_0}^{B_k}\boldsymbol{R}\,\Delta t_k\end{bmatrix} \in \mathbb{R}^{6\times 9} \tag{34}$$

$$\boldsymbol{s}_{I_k} = \begin{bmatrix}\boldsymbol{v}_{B_k} \\ \boldsymbol{v}_{B_{k+1}} \\ {}^{C_0}\boldsymbol{g}\end{bmatrix} \in \mathbb{R}^{9\times 1}. \tag{35}$$

Thus, the least-squares problem is formulated as

$$\min_{\boldsymbol{s}_I} \sum_{k\in[0,N-1]} \left\| \hat{\boldsymbol{z}}_{B_{k+1}}^{B_k} - \boldsymbol{H}_{B_{k+1}}^{B_k}\boldsymbol{s}_{I_k} \right\|^2. \tag{36}$$

The rotation of the first left camera frame w.r.t. the world frame ${}_{C_{l0}}^{W}\boldsymbol{R}$, is computed, using ${}^{C_{l0}}\boldsymbol{g}$ and ${}^{W}\boldsymbol{g} = [0\ 0\ g]^{\top}$, with g the gravitational acceleration, and enforcing the yaw angle to be zero. The poses and velocities of all body frame $\{\mathrm{B_k}\}$ w.r.t. the world frame $\{\mathrm{W}\}$ in the sliding windows can thus be recovered with ${}_{C_{l0}}^{W}\boldsymbol{R}$ and the results from (28) and (29).

3. EXPERIMENT RESULTS

In the following experiments, we evaluate the processing time of each module in our system on Intel® NUC7i7DNB computer with Linux Ubuntu and ROS. Futhermore, the accuracy of our system is compared with selected state-of-the-art VIOs. The configuration of the hardware platform is listed in Table 1.

Table 1. Configuration of the Intel® NUC computer used for the experiments.

Parameter	Value
CPU	Intel® Core i7-86500U Quad-core @ 1.90GHz
RAM	DDR4 SDRAM 16GB @ 2400MHz
Operating System	Ubuntu 18.04 Bionic Beaver
ROS Version	Melodic Morenia

3.1 Timing Statistics

The system front-end and back-end are separated in two CPU threads as in VINS-Mono. Table 2 shows the average time cost of each module in our proposed system running on the Intel® NUC computer. Camera update rate is set to 25 Hz and IMU update rate 200 Hz. The maximum feature number is 200. For local BA we use Gauss-Newton algorithm in combination with Powell's Dog-Leg method based on Ceres Solver, the maximum number of iteration is set to 5. Sliding window size for the state vector is set to 10.

From the results, it can be seen that the dominant time cost in the front-end is from the feature detection module which costs ∼14 ms per frame. Therefore, we limited the update rate of this module to 10 Hz to constraint the computational load of the system front-end without losing noticeable robustness and accuracy. The total front-end thread costs about 20 ms on average, thus the camera frame rate of 25 Hz can fulfill the real-time requirement of the system. From further experiments, it was shown that the feature tracking and matching modules can run at up to 60 Hz (feature detection module remains at 10 Hz) which can potentially increase the system robustness. However at this high update rate, we observed occasionally a relative large lag in system back-end due to the high CPU load, which, on the other hand, degrades the system stability.

The system back-end is running at 10 Hz. The major time costs are from the local BA and marginalization module (only perform when a new keyframe is inserted into the sliding window). The average time cost of the optimization is about 37 ms.

Table 2. Timing statistics of the proposed VIO.

Module	**Avg. time** [ms]	**Update rate** [Hz]
Feature Tracking	6.49	25
Stereo Matching	5.12	25
Feature Detection	14.32	∼10
Front-end Total	20	25
Triangulation	0.02	10
Local BA	26.81	10
Marginalization	21.74	-
Back-end Total	36.6	10

State vector estimated by the system back-end is updated at 10 Hz, which is typically not sufficient for most of the UAV control system. We, therefore, as in VINS-Mono, forward propagate the most recent IMU measurement with the latest estimated state to achieve a IMU-rate state update.

3.2 Dataset Comparison

Our stereo visual-inertial fusion concept named VINS-Stereo is evaluated using the public EuRoC MAV dataset [1] (Burri et al., 2016), and compared with other state-of-the-art open-sourced VIO systems including VINS-Mono [2], VINS-Fusion [3], S-MSCKF [4], and OKVIS [5]. Except for VINS-Mono which is our base system, all candidates are stereo VIOs. S-MSCKF and OKVIS failed due to the continuous inconsistency in brightness between stereo images in *V2_03_difficult*. Therefore, *V2_03_difficult* sequence is excluded here. For VINS-Mono, VINS-Fusion, S-MSCKF, and OKVIS, default configurations/parameters proposed by authors of the open-sourced implementations are used. All systems are evaluated without loop closure module.

CPU Load Figure 2 collects the CPU utilization of each algorithm throughout processing the *MH_01_easy* sequence. The usage is represented as a percentage of a single CPU core on the given platform (see Table 1). As supposed, the filter-based S-MSCKF had the lowest CPU utilization, while OKVIS was the most computational intensive candidate as mentioned before. Our method requires the lowest computing effort among the three optimization-based stereo VIOs.

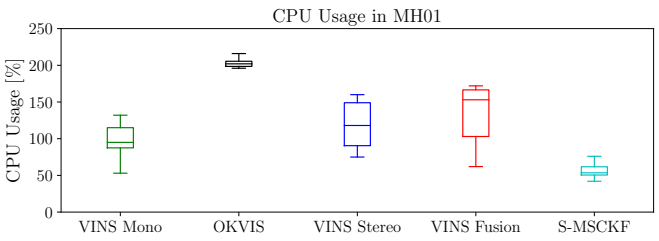

Fig. 2. CPU utilization statistics in *MH_01_easy*.

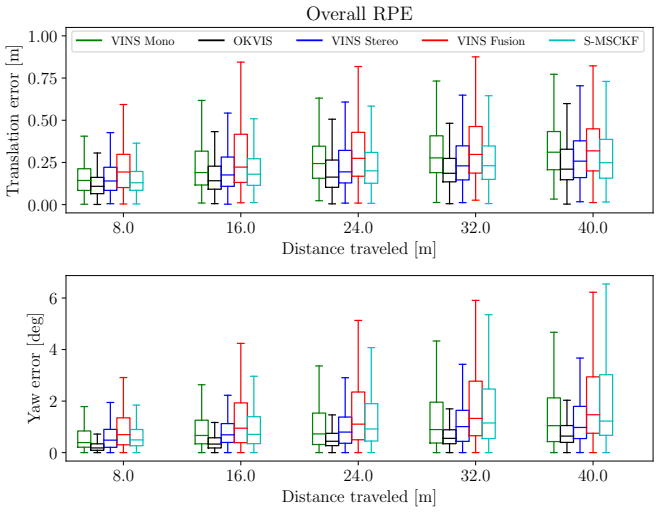

Fig. 3. Relative pose errors for the selected VIO pipelines over all successfully finished dataset sequences.

[1] http://robotics.ethz.ch/~asl-datasets
[2] https://github.com/HKUST-Aerial-Robotics/VINS-Mono
[3] https://github.com/HKUST-Aerial-Robotics/VINS-Fusion
[4] https://github.com/KumarRobotics/msckf_vio
[5] https://github.com/ethz-asl/okvis

Table 3. Absolute trajectory errors for the selected VIO pipelines in EuRoC dataset (RMSE [m]). **Blue bold** marks the lowest error, **black bold** marks the second-lowest.

Data sequence	VINS-Stereo	VINS-Mono	VINS-Fusion	S-MSCKF	OKVIS
MH01	**0.13**	**0.16**	0.26	0.22	0.26
MH02	**0.12**	0.18	0.22	0.18	**0.15**
MH03	**0.13**	0.20	0.29	**0.15**	**0.13**
MH04	0.28	0.35	0.43	**0.17**	**0.20**
MH05	**0.28**	0.30	0.31	0.30	**0.29**
V1_01	**0.05**	0.09	0.12	0.07	**0.04**
V1_02	**0.07**	0.11	0.11	0.10	**0.06**
V1_03	**0.13**	0.19	**0.13**	0.28	**0.11**
V2_01	0.11	0.09	0.14	**0.07**	**0.06**
V2_02	0.13	0.16	**0.12**	0.15	**0.08**
mean	**0.14**	0.18	0.21	**0.17**	**0.14**

Based on the tool from (Zhang and Scaramuzza, 2018) we evaluate our system in terms of Relative Pose Error and Absolute Trajectory Error.

Relative Pose Error Is a metric focus on the local/short-term accuracy, from all data sequences are summarized in Figure 3. It can be seen from the box plots, OKVIS provided the best results. However, the cost for the accurate estimations is the long per-frame processing time and high CPU load (see Figure 2). Following OKVIS, VINS-Stereo exhibited lower yaw error level than other candidates. In terms of the translational error, our method is similar to S-MSCKF and better than VINS-Mono and VINS-Fusion.

Absolute Trajectory Error Is usually used in VO/ VIO/ SLAM comparison to give an insight into the global consistency of the system and the accumulated drifts on the whole trajectory. We compared the absolute trajectory errors for the selected systems in different sequences of the EuRoC MAV dataset. Table 3 shows comprehensive test results of the algorithms in terms of the root-mean-square error (RMSE). The numbers in blue indicate the lowest values while the bold numbers are the second-lowest. According to the results, our system and the OKVIS performed better than others in most cases. The estimations using the proposed fusion system show the lowest or second-lowest RMSE except in the sequences *MH_04_difficult*, *V2_01_easy*, and *V2_02_medium*. In terms of the average score in all sequences, VINS-Stereo and OKVIS have same superior accuracy followed by S-MSCKF with a gap of 3 cm. Considering OKVIS requires much more computational resources, the proposed fusion system is more efficient.

4. CONCLUSION

In this work, we present a stereo visual-inertial fusion pipeline based on the framework of VINS-Mono, especially for UAV state estimation. Timing statistics of individual module is provided for the Intel® NUC computer installed on our drone. According to the experimental evaluations carried out with the EuRoC MAV dataset, the proposed approach shows competitive performance in most cases in comparison with other state-of-the-art VIO algorithms while keeping the computational cost at a relative low level.

In the future, the system will be integrated into the UAV control loop. Hardware acceleration for image processing could be investigated as well. Furthermore, line features are to be included to enhance the robustness in challenging scenarios.

REFERENCES

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M.W., and Siegwart, R. (2016). The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*. doi:10.1177/0278364915620033.

Delmerico, J. and Scaramuzza, D. (2018). A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. doi:10.1109/ICRA.2018.8460664.

Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2015). On-manifold preintegration theory for fast and accurate visual-inertial navigation. *CoRR*, abs/1512.02363.

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81. doi:10.1007/s11263-008-0152-6.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visualinertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3), 314–334. doi:10.1177/0278364914554813.

Mourikis, A.I. and Roumeliotis, S.I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 3565–3572.

Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4), 1004–1020. doi:10.1109/TRO.2018.2853729.

Qin, T., Cao, S., Pan, J., and Shen, S. (2019). A general optimization-based framework for global pose estimation with multiple sensors. *CoRR*, abs/1901.03642.

Shen, S., Michael, N., and Kumar, V. (2015). Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft mavs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 5303–5310. doi:10.1109/ICRA.2015.7139939.

Sun, K., Mohta, K., Pfrommer, B., Watterson, M., Liu, S., Mulgaonkar, Y., Taylor, C.J., and Kumar, V. (2018). Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3(2), 965–972. doi:10.1109/LRA.2018.2793349.

Zhang, Z. and Scaramuzza, D. (2018). A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*.