Dense Visual-Inertial Odometry for Tracking of Aggressive Motions

Yonggen Ling and Shaojie Shen

***Abstract*— We propose a sliding window-based dense visual-inertial fusion method for real-time tracking of challenging aggressive motions. Our method combines recent advances in direct dense visual odometry, inertial measurement unit (IMU) preintegration, and graph-based optimization. At the front-end, direct dense visual odometry provides camera pose tracking that is resistant to motion blur. At the back-end, a sliding window optimization-based fusion framework with efficient IMU preintegration generates smooth and high-accuracy state estimates, even with occasional visual tracking failures. A local loop closure that is integrated into the back-end further eliminates drift after extremely aggressive motions. Our system runs real-time at 25 Hz on an off-the-shelf laptop. Experimental results show that our method is able to accurately track motions with angular velocities up to 1000 degrees/s and velocities up to 4 m/s. We also compare our method with state-of-the-art systems, such as Google Tango, and show superior performance during challenging motions. We show that our method achieves reliable tracking results, even if we throw the sensor suite during experiments.**

I. Introduction

Accurate state estimation is essential for many robotic and intelligent applications, such as aerial vehicles, humanoid robots, and augmented reality. In practice, aggressive motions with large angular velocities and linear accelerations are commonly observed on such platforms, making state estimation extremely difficult. The visual-inertial system (VINS), which consists of off-the-shelf cameras and a MEMS inertial measurement unit (IMU), forms the ideal sensor suite for fast motion estimation due to the complementary nature of the sensors. Low-cost MEMS IMUs generate outlier-free but noisy measurements that are ideal for short-term tracking of fast motions. However, the IMUs also suffer from long-term drift. On the other hand, the cameras are great for drift-free tracking of slow movements. Nevertheless, tracking failure is ultimately unavoidable due to the downgraded image quality during aggressive motions. Fusing the complementary sensing modalities of the cameras and IMU is required for reliable motion estimation [1]–[3].

Recent advances in high-performance mobile computing have given rise to direct dense visual odometry methods [4, 5] that are resistant to motion blur and more adaptable in featureless environments. Dense methods directly optimize the camera motion by minimizing the difference in image intensity. The elimination of feature extraction and matching makes dense methods more robust in featureless environments that are known to be unfavorable for corner [6] or blob [7] detection. Dense methods are even able to track camera poses during fast movements where motion blur patterns are similar between images. However, dense methods are subject to failure during aggressive motions during which camera pose tracks are broken into multiple segments.

In this work, we address the problem of reliable tracking of aggressive motions in challenging environments by proposing a system for real-time fusion of dense visual odometry and IMUs. The front-end of our system is a state-of-the-art direct dense visual odometry module [5]. At the back-end, we utilize our IMU preintegration and two-way marginalization techniques proposed recently in [3] to form a sliding window estimator to connect and optimize motion tracks from the front-end. A local dense loop closure module further eliminates drift that may occur after very aggressive motions. Our system can run real-time at 25 Hz on an off-the-shelf laptop. Experimental results show that our method is able to accurately track motions with angular velocities up to 1000 degrees/s and velocities up to 4 m/s. To the best of our knowledge, we are the first to demonstrate reliable state estimation and trajectory recovery in extremely aggressive motions. Our method achieves reliable tracking results, even if we throw the sensor suite during experiments. [1]

Next, in Section II, we review the state-of-the-art scholarly work. A system overview of the proposed method is presented in Section III, and then Section IV provides the details. Section V shows the implementation details and experimental evaluation of our approach. Finally, Section VI draws the conclusions and discusses possible future extensions.

II. Related Work

The scholarly work on visual-inertial fusion is extensive. Different sensing modalities, including monocular [1]–[3, 8, 9], stereo [10] and RGB-D cameras [11], have been considered. At the back-end, fusion methods can mainly be divided into two classes. In loosely-coupled fusion [8, 9, 11], visual measurements are first processed independently to obtain high-level pose information and then fused with inertial measurements, usually using a filtering framework (such as KF, EKF [8, 11] or UKF [9]). Effectively, two sub-problems are solved separately in loosely-coupled fusion, resulting in a lower computational cost, but results are suboptimal. In tightly-coupled fusion [1]–[3, 10], both visual and inertial measurements are fused and optimized in a single framework. It considers the coupling between two

All authors are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. ylingaa@connect.ust.hk,eeshaojie@ust.hk

[1] http://1drv.ms/1Hv2l8d

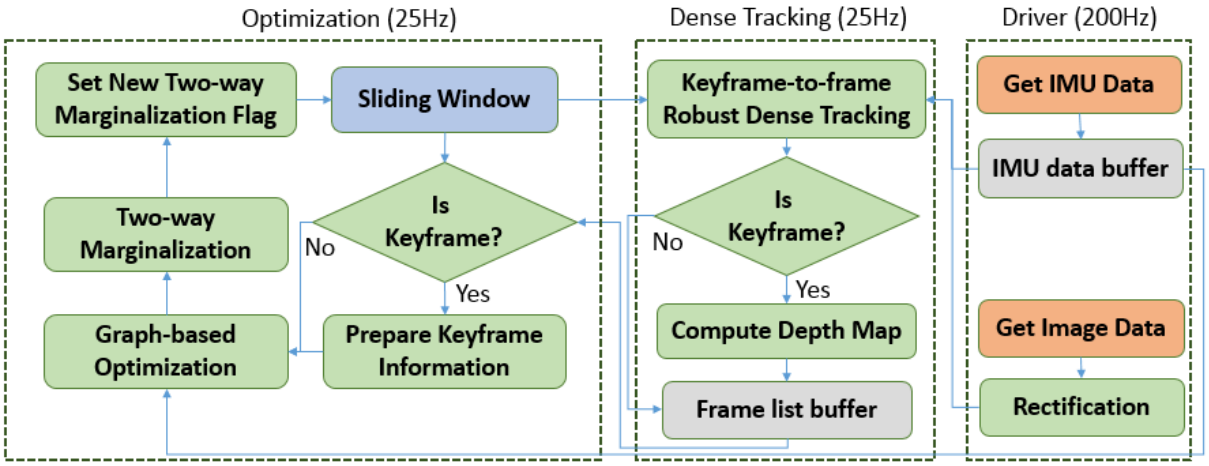


Fig. 1. The pipeline of our proposed method, which consists of the optimization thread, the dense tracking thread and the driver thread.

types of measurement and allows the adoption of a graph optimization-based framework with iterative re-linearization to achieve better performance. Tightly-coupled methods usually come with a higher computational cost.

Regardless of the use of loosely- or tightly-coupled methods, visual measurements are traditionally represented as features [1]–[3, 8]–[11]. Sparse image features, such as blobs and corners, are detected and tracked. Motion estimation can then be performed from multiple observations of features. However, the quality of features is largely determined by image quality. When the camera undergoes aggressive motions, motion blur can seriously affect feature detection and tracking performance, causing failure of the visual estimator.

To overcome the drawbacks of feature-based methods, dense tracking methods that directly operate on image intensities have become popular. Dense methods make full use of all the available information in an image, which gives robustness and higher accuracy against motion blur [5, 12, 13]. While direct dense tracking is well-established in vision-only settings (RGB-D [14], stereo [15] and monocular [5, 13] cameras), few visual-inertial fusion methods that fully utilize dense tracking exist. Most similar to our proposed approach is the method proposed in [16], in which dense tracking results are loosely fused with inertial measurements using an extended Kalman filter. However, in our work, dense tracking is fused with inertial measurements in a graph-based optimization framework, where the tracking results from multiple keyframes and local loop closure can be incorporated seamlessly. Our formulation allows drift correction even after aggressive motions. In addition, inertial measurements provide angular initialization for dense tracking, which improves the convergence property during fast rotation and helps with failure detection for dense tracking. We also demonstrate reliable tracking in significantly more aggressive motions than those presented in [16] with the same sensor suite (Fig. 3).

III. System Overview

The pipeline of our proposed system is illustrated in Fig. 1. There are three threads in our algorithm. Multiple threads run simultaneously utilizing the multi-core architecture. The driver thread reads IMU data from the sensor, stores it into a buffer, rectifies images if new images come and sends them to the dense tracking thread. The dense tracking thread obtains incremental camera motion using a direct keyframe-to-frame dense tracking algorithm, assisted by angular velocity from the gyroscope. This thread also identifies instant tracking performance and determines whether to add a keyframe or report tracking failure. If a keyframe is added, a depth map will be calculated using a stereo block matching algorithm. The optimization thread periodically checks the frame list buffer. All frames (incremental camera motions) and IMU measurements are added to the optimization thread. If a keyframe is added, loop closure detection is performed to find possible connections between keyframes. Graph-optimization is then applied to find the maximum a posteriori estimate of all the states within the sliding window using connections from IMU preintegration, dense tracking, loop closure and the prior. A two-way marginalization scheme that selectively removes states is performed in order to both bound the computational complexity and maximize the information stored within the sliding window.

IV. Dense Visual-Inertial Fusion

Before the proposed algorithm is detailed, we first list assumptions that we adopt throughout this paper:

- Objects in the surroundings exhibit Lambertian reflection.
- The major part of the surroundings captured by the cameras is static.
- The camera-IMU sensor suite is rigidly mounted, with intrinsic and extrinsic parameters calibrated beforehand.

We consider $(\cdot)^k$ as the camera frame while taking the k^{th} image, and $(\cdot)^b$ as the instantaneous IMU body frame. Without loss of generality, we assume that the cameras and the IMU are aligned. $\mathbf{p}_Y^X$, $\mathbf{v}_Y^X$ and $\mathbf{R}_Y^X$ are the 3D position, velocity and rotation of camera frame Y with respect to frame X. We also have the corresponding quaternion ($\mathbf{q}_Y^X =$

$[q_x, q_y, q_z, q_w]$) representation. Hamilton notation is used for quaternions.

Given two time instants that correspond to two images, we can write the IMU propagation model for position, velocity and rotation with respect to the first state of the system as in [17],

$$\begin{aligned}\mathbf{p}^0_{k+1} &= \mathbf{p}^0_k + \mathbf{R}^0_k \mathbf{v}^k \Delta t - \mathbf{g}^0 \Delta t^2/2 + \mathbf{R}^0_k \boldsymbol{\alpha}^k_{k+1} \\ \mathbf{v}^{k+1} &= \mathbf{R}^{k+1}_k (\mathbf{v}^k + \boldsymbol{\beta}^k_{k+1} - \mathbf{R}^k_0 \mathbf{g}^0 \Delta t) \\ \mathbf{q}^0_{k+1} &= \mathbf{q}^0_k \otimes \mathbf{q}^k_{k+1},\end{aligned} \tag{1}$$

where Δt is the interval between the acquisition of two images, and $\mathbf{g}^0$ is the gravity vector expressed in the first state of the system. $\boldsymbol{\alpha}^k_{k+1}$ and $\boldsymbol{\beta}^k_{k+1}$ can be obtained by integrating the IMU measurements between time instants k and $k+1$, with the definitions detailed in Sect. IV-B.

A. Problem Formulation

We set the position and rotation of the first state to be zero. The initial velocity and gravity vector can be obtained using an online initialization method presented in [18]. The full state vector is defined as

$$\begin{aligned}\mathcal{X} &= [\mathbf{x}^0_0, \mathbf{x}^0_1, ..., \mathbf{x}^0_N] \\ \mathbf{x}^0_k &= [\mathbf{p}^0_k, \mathbf{v}^k, \mathbf{q}^0_k] \\ \mathbf{p}^0_0 &= [0,0,0],\ \mathbf{q}^0_k = [0,0,0,1].\end{aligned} \tag{2}$$

Our goal is to obtain a maximum a posteriori (MAP) estimate by minimizing the sum of the Mahalanobis norm of the visual measurement residuals, inertial measurement residuals and the prior:

$$\begin{aligned}\min_{\mathcal{X}} \ (\mathbf{b}_p - \boldsymbol{\Lambda}_p \mathcal{X}) + \sum_{k \in S_i} ||r_{S_i}(\hat{\mathbf{z}}^k_{k+1}, \mathcal{X})||^2_{\mathbf{P}^k_{k+1}} + \\ \sum_{(i,j) \in S_c} ||r_{S_c}(\hat{\mathbf{z}}^j_i, \mathcal{X})||^2_{\mathbf{P}^j_i}\end{aligned} \tag{3}$$

where $\boldsymbol{\Lambda}_p$ and $\mathbf{b}_p$ are the prior matrix and prior residual vector respectively, S_i and S_c are the set of inertial and visual measurements respectively. $r_{S_i}(\hat{\mathbf{z}}^k_{k+1}, \mathcal{X})$ is the residual function that measures the residual between the inertial measurements and $\mathcal{X}$ with covariance $\mathbf{P}^k_{k+1}$, and $r_{S_c}(\hat{\mathbf{z}}^j_i, \mathcal{X})$ is the residual function that measures the reprojection error between the visual measurements and $\mathcal{X}$ with covariance $\mathbf{P}^j_i$.

Inertial measurements are obtained by IMU preintegration (Sect. IV-B) and visual measurements are obtained by dense tracking (Sect. IV-C) and loop closure (Sect. IV-D).

B. IMU Preintegration

We adopt the pre-integration method proposed in our previous work [3]. The integration from the IMU measurement between time instant k and $k+1$ is

$$\hat{\mathbf{z}}^k_{k+1} = \begin{bmatrix} \hat{\boldsymbol{\alpha}}^k_{k+1} \\ \hat{\boldsymbol{\beta}}^k_{k+1} \\ \hat{\mathbf{q}}^k_{k+1} \end{bmatrix} = \begin{bmatrix} \iint_{t \in [k,k+1]} \mathbf{R}^k_t \mathbf{a}^b_t dt^2 \\ \int_{t \in [k,k+1]} \mathbf{R}^k_t \mathbf{a}^b_t dt \\ \int_{t \in [k,k+1]} \boldsymbol{\Omega}(\boldsymbol{\omega}^{b_t}) \mathbf{q}^k_t dt \end{bmatrix}, \tag{4}$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}^{b_t}) = \frac{1}{2} \begin{bmatrix} -\lfloor \boldsymbol{\omega}^{b_t} \times \rfloor & \boldsymbol{\omega}^{b_t} \\ -\boldsymbol{\omega}^{b_t}{}^T & \mathbf{0} \end{bmatrix}, \tag{5}$$

and $\mathbf{a}^b_t$ and $\boldsymbol{\omega}^{b_t}$ are the instantaneous linear acceleration and angular velocity in the IMU body frame at time instant t respectively. The residual function between the states and the IMU measurement is defined as

$$\begin{aligned} r_{S_i}(\hat{\mathbf{z}}^k_{k+1}, \mathcal{X}) &= \begin{bmatrix} \delta\boldsymbol{\alpha}^k_{k+1} \\ \delta\boldsymbol{\beta}^k_{k+1} \\ \delta\boldsymbol{\theta}^k_{k+1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}^k_0 (\mathbf{p}^0_{k+1} - \mathbf{p}^0_k + \mathbf{g}^0 \frac{\Delta t^2}{2}) - \mathbf{v}^k \Delta t - \hat{\boldsymbol{\alpha}}^k_{k+1} \\ \mathbf{R}^k_0 (\mathbf{R}^0_{k+1} \mathbf{v}^{k+1} + \mathbf{g}^0 \Delta t) - \mathbf{v}^k - \hat{\boldsymbol{\beta}}^k_{k+1} \\ 2[(\hat{\mathbf{q}}^k_{k+1})^{-1} (\mathbf{q}^0_k)^{-1} \mathbf{q}^0_{k+1}]_{xyz} \end{bmatrix}. \end{aligned} \tag{6}$$

The covariance $\mathbf{P}^k_{k+1}$ can be calculated by iteratively linearizing the continuous-time dynamics of the error term and then updating it with discrete-time approximation:

$$\begin{aligned}\mathbf{P}^k_{t+\delta t} =& (\mathbb{I} + \mathbf{F}_t \delta t) \cdot \mathbf{P}^k_t \cdot (\mathbb{I} + \mathbf{F}_t \delta t)^{\mathrm{T}} \\ &+ (\mathbb{I} + \mathbf{G}_t \delta t) \cdot \mathbf{Q}_t \cdot (\mathbb{I} + \mathbf{G}_t \delta t)^{\mathrm{T}}\end{aligned} \tag{7}$$

with the initial condition $\mathbf{P}^k_k = \mathbf{0}$. $\mathbf{F}_t$ and $\mathbf{G}_t$ are the state transition Jacobians with respect to the states and the IMU measurement noise respectively. Detailed derivation can be found in [3].

C. Dense Tracking

Since the surrounding environment exhibits Lambertian reflection, the image intensity is supposed to be the same regardless of the viewing angles or positions. We aim to find the rigid-body transformation between state i and state j, denoted as $\mathbf{T}^j_i = \{\mathbf{t}^j_i, \mathbf{R}^j_i\} \in \mathbf{SE}(3)$, that minimizes the intensity differences:

$$\mathbf{T}^{j\,*}_i = \arg\min_{\mathbf{T}^j_i} \iint \delta\mathbf{I}(\mathbf{T}^j_i, \mathbf{u}) d\mathbf{u} \tag{8}$$

$$\delta\mathbf{I}(\mathbf{T}^j_i, \mathbf{u}) = \mathbf{I}_i[\mathbf{u}] - \mathbf{I}_j[\pi(\mathbf{R}^j_i \pi^{-1}(\mathbf{u}, d_{\mathbf{u}}) + \mathbf{t}^j_i)], \tag{9}$$

where $\mathbf{u}$ is the coordinate of a pixel, $\mathbf{I}_k[\mathbf{u}]$ is the intensity value of pixel $\mathbf{u}$ in image k, $\pi(\cdot) : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection function that projects a 3-D point $\mathbf{f} = [x, y, z]^{\mathrm{T}}$ into the image coordinate $\mathbf{u} = (u, v)$, and $\pi^{-1}(\cdot)$ is the inverse project function. As (8) is a highly non-linear function, we adopt the Gauss-Newton approach on the Lie-manifolds, which iteratively re-linearizes (8) around the current estimate $\hat{\mathbf{T}}^j_i$ and then performs incremental updates until convergence:

$$\hat{\mathbf{T}}^j_i \leftarrow \hat{\mathbf{T}}^j_i \otimes \exp(\boldsymbol{\xi}), \tag{10}$$

where $\boldsymbol{\xi} = (\delta\mathbf{t}^j_i, \delta\boldsymbol{\theta}^j_i) \in \mathfrak{se}(3)$ is the minimum dimension error state. The bridge between Lie algebra $\mathfrak{se}(3)$ and Lie group $\mathbf{SE}(3)$ is the exponential map $\exp(\boldsymbol{\xi})$. For more details about Lie algebra and Lie group, please refer to [19].

Solving (8) with linearization results in solving the linear system

$$\mathbf{J}^{\mathrm{T}} \mathbf{J} \boldsymbol{\xi} = \mathbf{J}^{\mathrm{T}} \mathbf{r}, \tag{11}$$

where $\mathbf{J}$ is a Jacobian matrix that forms by stacking Jacobians of the image intensity differences (8) with respect to

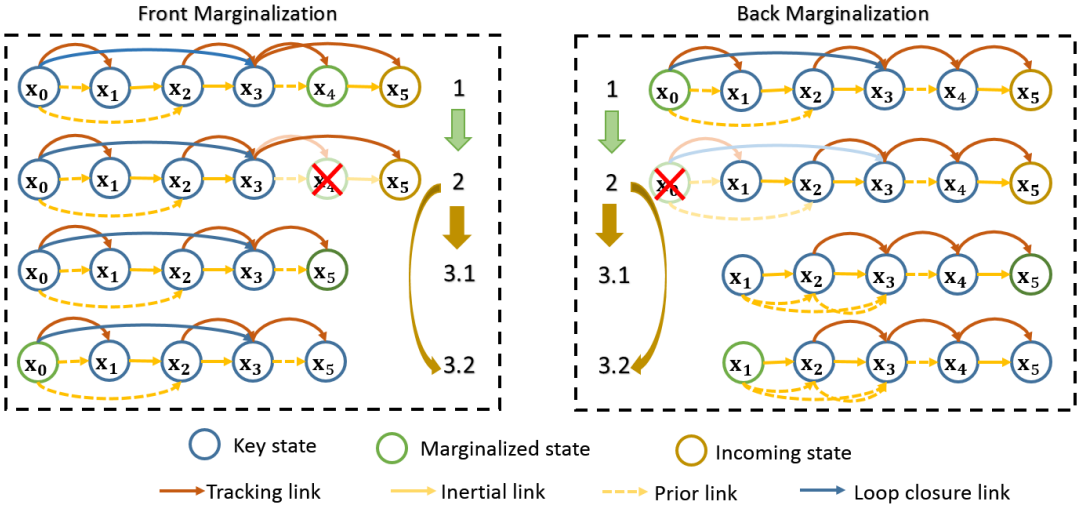


Fig. 2. The process of our two-way marginalization, which marginalizes all the available information (motion estimates from dense tracking, inertial measurement, loop closure relation and prior) into a new prior and maintains bounded computation complexity. Front marginalization marginalizes the second newest state and back marginalization marginalizes the oldest state within the sliding window.

$\boldsymbol{\xi}$, and $\mathbf{r}$ is the corresponding intensity differences vector. M is the number of valid pixels within the image coordinate range,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial}{\partial \boldsymbol{\xi}} \delta\mathbf{I}(\hat{\mathbf{T}}_i^j \otimes \exp(\boldsymbol{\xi}), \mathbf{u}_1^i) \\ \cdots \\ \frac{\partial}{\partial \boldsymbol{\xi}} \delta\mathbf{I}(\hat{\mathbf{T}}_i^j \otimes \exp(\boldsymbol{\xi}), \mathbf{u}_M^i) \end{bmatrix}_{|\xi=\mathbf{0}}, \mathbf{r} = \begin{bmatrix} \delta\mathbf{I}(\hat{\mathbf{T}}_i^j, \mathbf{u}_1^i) \\ \cdots \\ \delta\mathbf{I}(\hat{\mathbf{T}}_i^j, \mathbf{u}_M^i) \end{bmatrix}. \tag{12}$$

To better handle outliers, a weighted version of (11) is applied:

$$\mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{J}\boldsymbol{\xi} = \mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{r}, \tag{13}$$

where $\mathbf{W}$ is a diagonal matrix with weights computed according to the Huber kernel thresholding on intensity differences. Image pyramids are also adopted to increase the convergence region and handle large movements. For efficiency reasons, only pixels with noticeable gradients are used in dense tracking.

The visual measurement $\hat{\mathbf{z}}_i^j$ in (3) is $\hat{\mathbf{z}}_i^j = \mathbf{T}_i^{j*}$, and the residual function is defined as

$$r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X}) = \begin{bmatrix} \delta\mathbf{t}_i^j \\ \delta\boldsymbol{\theta}_i^j \end{bmatrix} = \begin{bmatrix} \mathbf{R}_0^j(\mathbf{p}_j^0 - \mathbf{p}_i^0) - \hat{\mathbf{t}}_i^j \\ 2[(\hat{\mathbf{q}}_i^j)^{-1}(\mathbf{q}_j^0)^{-1}\mathbf{q}_i^0]_{xyz} \end{bmatrix}, \tag{14}$$

where $\hat{\mathbf{q}}_i^j$ is the quaternion representation of $\hat{\mathbf{R}}_i^j$. The covariance $\mathbf{P}_i^j$ is set as the inverse of the Hessian matrix $\mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{J}$ at the final iteration.

D. Dense Tracking-based Local Loop Closure and Tracking Failure Detection

Our local loop closure module seeks possible connections between states within the sliding window in order to eliminate drift, even after very aggressive motions (Fig. 6 and Fig. 7). Loop closure detection is also done by dense tracking, but between two keyframes.

To avoid wrong loop closure links between states, cross checking is adopted so as to improve the confidence. Suppose frame k is fixed as a new keyframe to the sliding window. We enumerate all the other keyframes within the sliding window to check whether there is another keyframe, serving as reference frame, that can track frame k successfully. If keyframe i tracks frame k successfully, frame k will serve as the reference frame and track frame i. The cross checking is successful if the two corresponding rigid-body transformations are consistent. For efficiency reasons, only the coarsest pyramid level is used for the cross check. Keyframe i, after passing the cross check, will track frame k again with all pyramid levels. A link between state i and state k is added if the full-pyramid tracking is successful.

Since the tracking covariance matrix $(\mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{J})^{-1}$ tells us about the dense tracking performance, we reject dense tracking results if the covariance is greater than a certain threshold. Also, if the rigid-body transformation estimated by the dense tracking is not consistent with the initial guess from the IMU integration, the dense tracking is considered to have failed and no links from visual measurement will be added.

E. Two-way Marginalization

Due to the limited memory and computational resources of the system, we can merely maintain a certain number of states and measurements within the sliding window. We convert states that carry less information into a prior matrix $\{\boldsymbol{\Lambda}_p, \mathbf{b}_p\}$ by marginalization. Note that the effectiveness of loop closure and drift elimination depends on whether an older state is kept within the sliding window. For this reason, unlike traditional approaches which only marginalize old states, we use a two-way marginalization scheme that was first introduced in our earlier work [18] to selectively remove old or more recent states in order to enlarge the covered regions of the sliding window.

Fig. 2 illustrates the process of our two-way marginalization. Front marginalization removes the second newest

state, while back marginalization removes the oldest state. Blue circles represent key states, green circles represent the states to be marginalized and brown circles represent the incoming states. The relation between states and frames is that states include poses and velocities and take uncertainty into account, while frames include poses and images. Each frame has its corresponding state and vise versa. States are linked by IMU preintegration (inertial link), dense tracking (tracking link), loop closure (loop closure link) and the prior (prior link). To perform front marginalization, the second newest state is first linked with the incoming state (step 1) and then marginalized out (step 2). For back marginalization, the oldest state is simply marginalized out (steps 1-2). After marginalization, the third step decides which state is to be marginalized in the next round (front marginalization or back marginalization).

Mathematically, to marginalize a specific state, we remove all links related to it and then add the removed links into a prior:

$$\begin{aligned}\mathbf{\Lambda}_p =& \mathbf{\Lambda}_p + \sum_{k\in S_i^-} (\mathbf{H}_{k+1}^k)^{\mathrm{T}} (\mathbf{P}_{k+1}^k)^{-1} \mathbf{H}_{k+1}^k \\ &+ \sum_{(i,k)\in S_c^-} (\mathbf{H}_i^k)^{\mathrm{T}} (\mathbf{P}_i^k)^{-1} \mathbf{H}_i^k \qquad (15)\\ \mathbf{b}_p =& \mathbf{b}_p + \sum_{k\in {S_i}^-} (\mathbf{H}_{k+1}^k)^{\mathrm{T}} (\mathbf{P}_{k+1}^k)^{-1} r_{S_i}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X}) \\ &+ \sum_{(i,k)\in {S_c}^-} (\mathbf{H}_i^k)^{\mathrm{T}} (\mathbf{P}_i^k)^{-1} r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X}), \qquad (16)\end{aligned}$$

where S_i^- and ${S_c}^-$ are the set of removed IMU preintegration measurements and visual measurements respectively. The prior is then marginalized via the Schur complement [20].

The criterion to select whether to use front or back marginalization are based on the dense tracking performance. If the dense tracking is good and the second newest state is near to the current keyframe, the second newest state will be marginalized in the next round. Otherwise, the oldest state will be marginalized. We threshold the distance by a weighted combination of translation and rotation between the current keyframe and the second newest frame.

Note that our two-way marginalization is fundamentally different from traditional keyframe-based approaches that simply drop non-keyframes. We preserve all the information (IMU and dense tracking) from non-keyframes by only performing marginalization after the newest state comes, and the system is then updated (step 1 in front marginalization). Also by marginalization, we ensure that the time period for each IMU preintegration is bounded in order to bound the accumulated error in the IMU measurements.

F. Optimization

Based on the residual function defined in (6) and (14), we operate on the error state and optimize (3) using the Gaussian-Newton method, which iteratively minimizes

$$\begin{aligned}\min_{\delta\mathcal{X}} \ & (\mathbf{b}_p - \mathbf{\Lambda}_p \mathcal{X}) + \sum_{k\in S_i} ||r_{S_i}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X}) + \mathbf{H}_{k+1}^k \delta\mathcal{X}||^2_{\mathbf{P}_{k+1}^k} \qquad (17)\\ &+ \sum_{(i,j)\in S_c} ||r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X}) + \mathbf{H}_i^j \delta\mathcal{X}||^2_{\mathbf{P}_i^j}\end{aligned}$$

and then updates

$$\hat{\mathcal{X}} = \hat{\mathcal{X}} \oplus \delta\mathcal{X} \qquad (18)$$

until convergence. The Jacobian matrices are

$$\mathbf{H}_{k+1}^k = [\frac{\partial r_{S_i}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X})}{\partial \delta \mathbf{x}_k} \ \frac{\partial r_{S_c}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X})}{\partial \delta \mathbf{x}_{k+1}}] \qquad (19)$$

$$\frac{\partial r_{S_i}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X})}{\partial \delta \mathbf{x}_k} = \begin{bmatrix} -\mathbf{R}_0^k & -\Delta t \mathbb{I} & \lfloor \mathbf{R}_0^k \mathbf{d}_1 \times \rfloor \\ \mathbf{0} & -\mathbb{I} & \lfloor \mathbf{R}_0^k \mathbf{d}_2 \times \rfloor \\ \mathbf{0} & \mathbf{0} & -\mathbf{R}_0^{k+1} \mathbf{R}_k^0 \end{bmatrix} \qquad (20)$$

$$\frac{\partial r_{S_c}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X})}{\partial \delta \mathbf{x}_{k+1}} = \begin{bmatrix} \mathbf{R}_0^k & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_0^k \mathbf{R}_{k+1}^0 & -\mathbf{R}_0^k \mathbf{R}_{k+1}^0 \lfloor \mathbf{v}^{k+1} \times \rfloor \\ \mathbf{0} & \mathbf{0} & \mathbb{I} \end{bmatrix} \qquad (21)$$

$$\mathbf{H}_i^j = [\frac{\partial r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X})}{\partial \delta \mathbf{x}_i} \ \frac{\partial r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X})}{\partial \delta \mathbf{x}_j}] \qquad (22)$$

$$= \begin{bmatrix} -\mathbf{R}_0^j & \mathbf{0} & \mathbf{0} & \mathbf{R}_0^j & \mathbf{0} & \lfloor \mathbf{R}_0^j (\mathbf{p}_j^0 - \mathbf{p}_i^0) \times \rfloor \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbb{I} & \mathbf{0} & \mathbf{0} & -\mathbf{R}_0^i \mathbf{R}_j^0 \end{bmatrix}, \qquad (23)$$

where $\mathbf{d}_1 = (\mathbf{p}_{k+1}^0 - \mathbf{p}_k^0 + \mathbf{g}^0 \frac{\Delta t^2}{2})$ and $\mathbf{d}_2 = \mathbf{R}_{k+1}^0 \mathbf{v}^{k+1} + \mathbf{g}^0 \Delta t$.

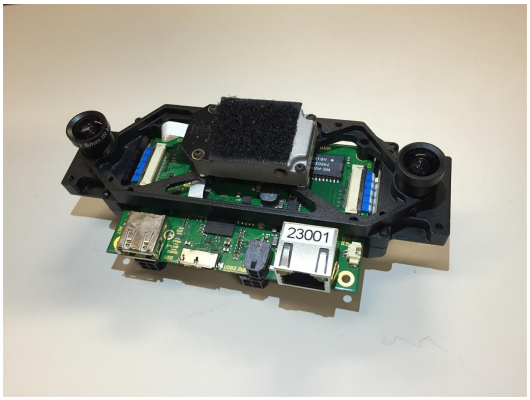


Fig. 3. The VI-Sensor from Skybotix with an 11 cm stereo baseline.

V. Experiments

All experiments are conducted with a VI-Sensor [21], which consists of an IMU and two global shutter cameras with a fronto-parallel stereo configuration, as shown in Fig. 3. The frequencies of the IMU data and stereo camera data are 200 Hz and 25 Hz respectively. The cameras have factory pre-calibrated intrinsics and extrinsics. The finest resolution for dense tracking is 320×240 and the number of levels of the pyramid is 3. The noticeable gradient threshold of the

dense tracking is 5 and the sliding window size is 30. All the experiments are run on a commodity Lenovo laptop Y510 with an i7-4720HQ CPU and 8 G of RAM.

We highlight the time required for each step in our proposed method in Table I. The block matching algorithm we use is the simplest and fastest one in OpenCV (Stereo BM). We have found that our approach is not very sensitive to depth error, and our proposed method can run up to 30 Hz if the device can support higher data frequency.

Dense Tracking Thread	Average Computation Time
Tracking	13 ms
Block Matching	5 ms
Total	18 ms

Optimization Thread	Average Computation Time
Graph Optimization	5 ms
Marginalization	3 ms
Loop Closure	18 ms
Total	26 ms

TABLE I

AVERAGE COMPUTATION TIME OF DENSE TRACKING THREAD AND OPTIMIZATION THREAD

A. Performance in large-scale environments

We test our proposed method in a large-scale environment and compare it with the performance of Google Tango. Results are shown in Fig. 4. Google Tango is tied rigidly with our device in this experiment. The total travel distance is approximately 220 meters and there are vibrations caused by hand movement during the testing. The angular rate is shown in Fig. 5. The final position drift for our proposed method is about 0.45 meters, while for Google Tango it is about 3.38 meters. The main position drift of the Google Tango estimate comes from orientation error. For translational motions, both our approach and Google Tango exhibit similar performance. However, for large and fast rotation which causes obvious image blur, our proposed method achieves significantly higher orientation precision compared to Google Tango. The maximum angular rate is about 140 degrees per second, as reported by IMU measurements.

Note that in this experiment, though dense tracking is robust, it occasionally provides wrong estimations (blue dashed boxes in Fig. 4). However, our local loop closure and failure rejection mechanism (Sect. IV-D) ensures successful recovery from failures.

B. Tracking of Aggressive Motions

This experiment is conducted in an office, with aggressive motions performed for 80 seconds. Ground truth data from the OptiTrack Flex 13 system is available in this experiment. Fig. 7 shows the detailed comparison of orientation, translation and velocity. The performance statistics are summarized in Table II. The maximum angular velocity is obtained from the IMU instantaneous measurement and the maximum linear

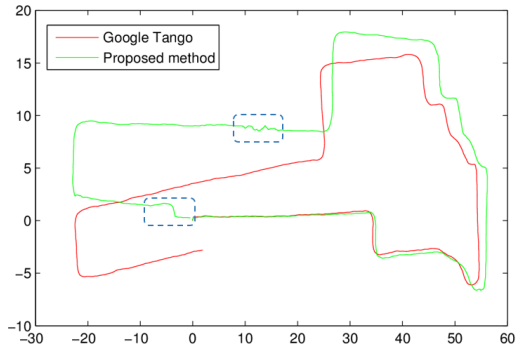


Fig. 4. We compare our proposed method with Google Tango in a large scale environment. Dashed boxes indicates failures in dense tracking.

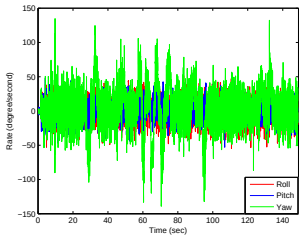


Fig. 5. The angular rate reported by the IMU during the experiment shown in Fig. 4.

velocity is reported by the OptiTrack system. Note that 1000 degrees/s is also the maximum angular velocity rate for the VI-sensor.

During the time interval between 20 and 37 seconds, position drift caused by tracking failure is observed. However, this drift is then eliminated by loop closure.

It can be clearly seen that the estimates from our proposed method nicely fit the ground truth data. More details about the test sequence can be found in the following video: `http://1drv.ms/1Hv218d`

Duration	80 s
Maximum Angular Velocity	1000 degrees/s
Maximum Linear Velocity	4 m/s
Position-x Drift	0.0336 m
Position-y Drift	0.0814 m
Position-z Drift	0.0469 m
Velocity-x Standard Derivation	0.0264 m/s
Velocity-y Standard Derivation	0.0255 m/s
Velocity-z Standard Derivation	0.0250 m/s
Yaw Drift	1.5080 degrees
Roll Standard Deviation	0.3357 degrees
Pitch Standard Deviation	0.3595 degrees

TABLE II

PERFORMANCE STATISTICS OF OUR PROPOSED METHOD DURING AGGRESSIVE MOTIONS. DETAILED PLOTS ARE SHOWN IN FIG. 7.

C. Throw It!

We conclude by presenting the third experiment, in which we throw the VI-Sensor while walking. To the best of our knowledge, this is the toughest testing for a visual-inertial estimator that has ever been reported. The total walking

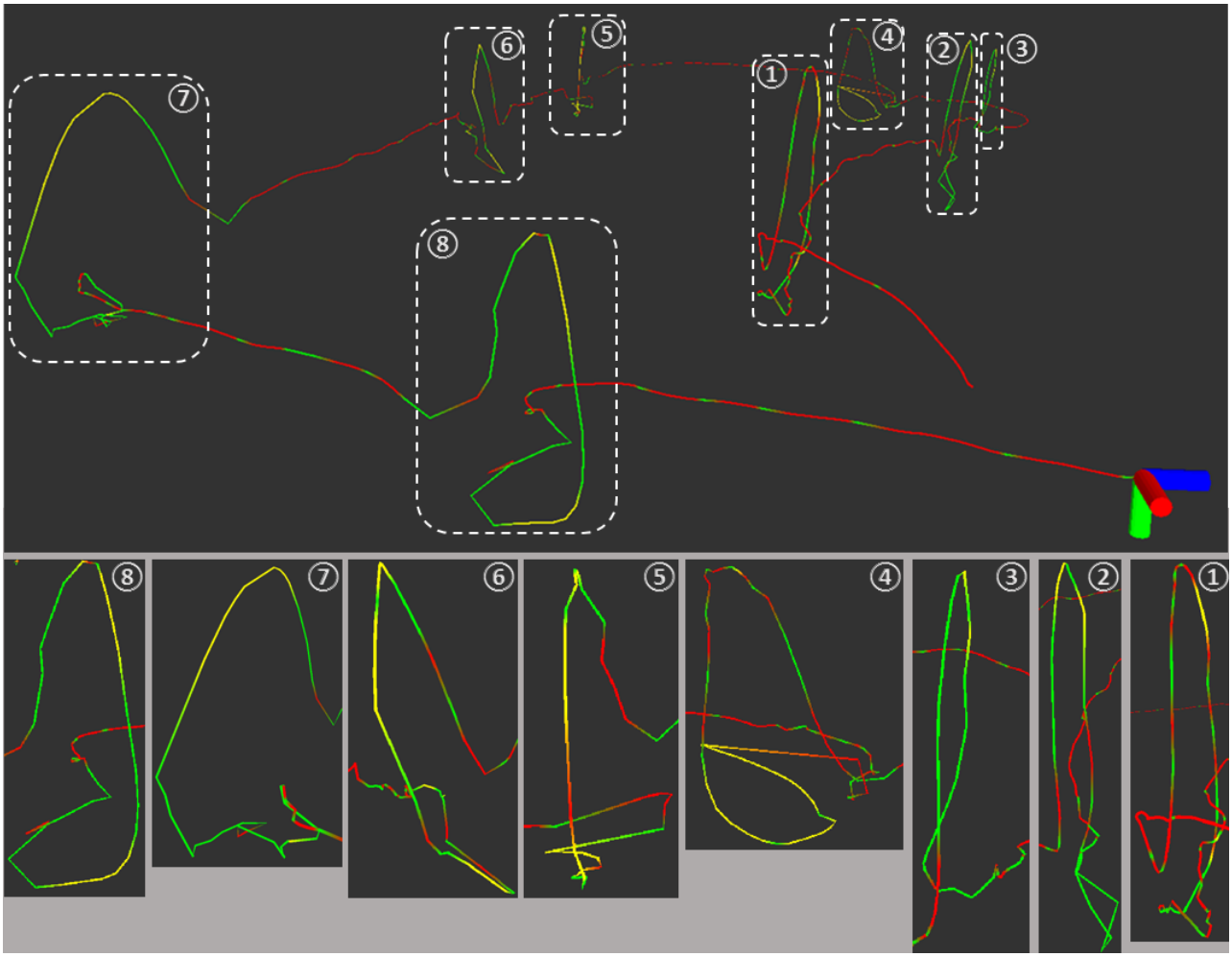


Fig. 6. We throw the VI-sensor while walking. The total walking distance is about 50 meters and the final position drift is about 1.5 meters. There are 8 occurances of throwing in total. The path consists of three kinds of color segments, which indicate 1) Red: good dense tracking without adding new keyframes; 2) Green: good dense tracking but new keyframes are added; and 3) Yellow: dense tracking failure. In all cases, our local loop closure is able to largely eliminate drifts after throwing (Sect. IV-D). A video of the experiment can be found at: `http://1drv.ms/1Hv2l8d`

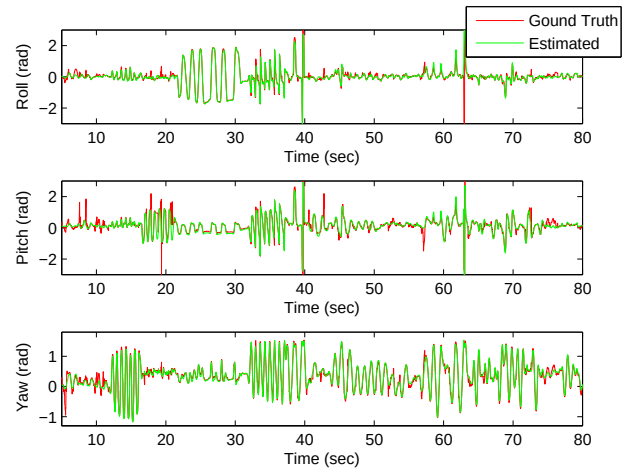
distance is about 50 meters and the final position drift is about 1.5 meters. We throw the VI-Sensor eight times as shown in Fig. 6. The trajectory contains three different color segments corresponding to the three situations of the system. The first situation (red segments) is that the current keyframe-to-frame dense tracking is good and the current frame is near to the latest keyframe (front marginalization is applied). The second situation (green segments) is that the current keyframe-to-frame dense tracking is good but the current frame is far (in terms of either translation or rotation) from the latest keyframe. In such a case, the current frame is fixed as a new keyframe and back marginalization is applied. The last situation (yellow segments) is that the current keyframe-to-frame dense tracking is bad and there are only inertial links between consecutive states. In such a case, the current frame is fixed as a new keyframe and back marginalization is applied.

From this extremely challenging experiment, we can see that while dense tracking is robust and able to handle fast motion, it is still subject to failure when motions become more and more aggressive. Inertial measurement from the IMU is the last resort in this case, and it provides crucial links between consecutive states to ensure continuous operation of the estimator. In all cases, our local loop closure is able to largely eliminate drifts after throwing (Sect. IV-D).

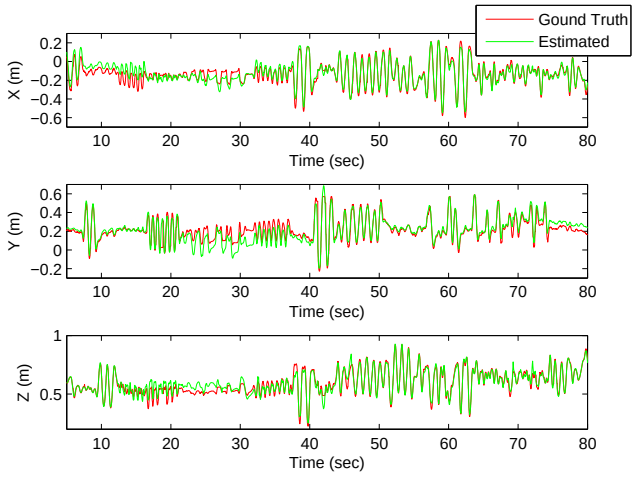
VI. Conclusion and Future Work

We have proposed real-time dense visual-inertial odometry that is able to handle aggressive motions. Our proposed method benefits from recent advances in direct dense tracking, IMU preintegration, and graph-based optimization. The multi-thread framework enables a fast and stable estimate using only the CPU of an off-the-shelf laptop. Online experiments have verified the performance of the visual-inertial odometry even with extremely challenging motions.

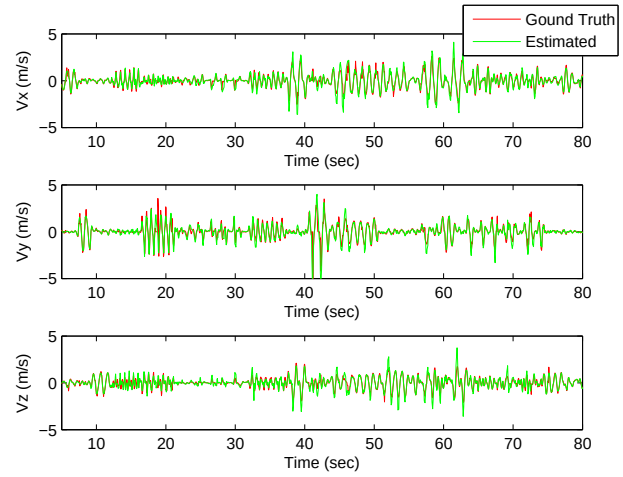
In the future, we will integrate our proposed visual-inertial odometry into real applications, such as autonomous flight of unmanned aerial vehicles (UAV) or augmented reality (AR), whose performances are still unsatisfactory and unstable during challenging motions using existing algorithms.



(a) Orientation



(b) Position



(c) Velocity

Fig. 7. Performance of our proposed method compared with ground truth obtained by OptiTrack. (a)Orientation. (b)Position. (c)Velocity.

REFERENCES

[1] M. Li and A. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Intl. J. Robot. Research*, vol. 32, no. 6, pp. 690–711, May 2013.

[2] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 158–176, Feb. 2014.

[3] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, WA, May 2015.

[4] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: dense tracking and mapping in real-time," in *IEEE International Conference on Computer Vision*, 2011, pp. 2320–2327.

[5] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[6] J. Shi and C. Tomasi, "Good features to track," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Seattle, WA, June 1994, pp. 593–600.

[7] D. Scaramuzza and F. Fraundorfer, "Visual Odometry: Part I - The First 30 Years and Fundamentals," *IEEE Robot. Autom. Mag.*, vol. 18, 2011.

[8] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, "Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robot. Autom. Mag.*, vol. 21, no. 3, 2014.

[9] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor," in *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, 2013.

[10] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial SLAM using nonlinear optimization," in *Proc. of Robot.: Sci. and Syst.*, Berlin, Germany, June 2013.

[11] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. of the Intl. Sym. of Robot. Research*, Flagstaff, AZ, Aug. 2011.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[13] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proc. of the IEEE Intl. Conf. Comput. Vis.*, Sydney, Australia, December 2013.

[14] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2013.

[15] A. I. Comport, E. Malis, and P. Rives, "Real-time quadrifocal visual odometry," in *Intl. J. Robot. Research*, 2010.

[16] S. Omari, M. Bloesch, P. Gohl, and R. Siegwart, "Dense visual-inertial navigation system for mobile robots," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2015.

[17] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 61–76, Feb. 2012.

[18] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Initialization-free monocular visual-inertial estimation with application to autonomous MAVs," in *Proc. of the Intl. Sym. on Exp. Robot.*, Marrakech, Morocco, 2014.

[19] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer Science & Business Media, 2012, vol. 26.

[20] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *J. Field Robot.*, vol. 27, no. 5, Sept. 2010.

[21] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2014, pp. 431–437.