2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Markov Parallel Tracking and Mapping for Probabilistic SLAM

Zheng Huai and Guoquan Huang

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561238

Abstract— **Parallel tracking and mapping (PTAM) as a time-efficient framework for simultaneous localization and mapping (SLAM) has been becoming popular in recent years. However, in this paper, we vigilantly point out that the favorite parallel-pipeline design realized by recent proposed SLAM algorithms may lead to inaccurate state estimates which, as a consequence, cannot always guarantee the performance of the estimators in real application. This is mainly due to the imperfect design for processing loop-closure measurements which accidentally violates the Markov assumption for probabilistic SLAM problem. To address this issue, a novel estimator design is proposed that holds the advantage of parallel processing, while striving to be consistent with the Markov property of the batch probabilistic SLAM estimator, therefore, termed *Markov parallel tracking and mapping* (MPTAM). Especially, the experiments on challenging visual-inertial datasets are employed to further demonstrate the improvements of proposed estimator in terms of accuracy and efficiency, as compared with the state-of-the-art SLAM system.**

I. Introduction and Related Work

Simultaneous localization and mapping (SLAM) has been studied for almost three decades since its pioneer work (e.g., [1], [2], [3]) was proposed around the 1990s, and has always been the core technology of many different robotic/computer vision applications, such as the visual-inertial navigation, 3D reconstruction, augmented/virtual reality (AR/VR), and also autonomous driving. Generally speaking, the primary goal of the SLAM is to learn about the states of the robot itself and its surrounding environment (e.g., a map about the places the robot has visited and its trajectory in this map). However, as those states in general are not directly observable we have to infer them based on some observations which usually come from the sensors equipped by the robot, in which the wheel odometer, camera, and inertial measurement unit (IMU) are most commonly used in real application. Moreover, in order to extract usable information about the state from the noisy sensor measurements, the probabilistic distributions are used to model these observations (e.g., assuming Gaussian noise). Then, based on that and some prior knowledge (in terms of probability distribution) about the state, we can characterize the SLAM result as a posterior probability which is formed following Bayes rule [4]. In particular, the maximum of this probability corresponds to an (optimal) inference of the state which is named maximum a posteriori (MAP) estimate. We should note that the above approach is practical for SLAM because of a *Markov assumption* [5] which specifies that the state is a complete summary of the past such that its posterior distribution is sufficient to represent the history of the robot. As a result, the corresponding Markov property is embedded in the MAP estimator designed for SLAM application.

It is well known that under certain assumptions, finding the MAP state estimate for SLAM can be cast as a nonlinear least-squares problem, which is also referred to as bundle adjustment (BA) in the literature and whose solution can be found in a batch form [6], [7]. However, the processing cost of batch estimator is increasing as more measurements are included for the estimation, and may eventually become intolerable for real-time processing even using an incremental approximation [8], [9]. In order to achieve efficiency against ever-increasing computational complexity, the filtering and the fixed-lag smoothing methods are proposed, which focus on the estimation over only a bounded-size, sliding window of recent states by marginalizing out the past states and measurements (e.g., [10], [11], [12], [13], [14], [15], [16], [17]). While those methods achieve constant processing cost and real-time performance, they suffer an ever-increasing drift in the exploration due to the inability to perform global adjustment which is usually triggered by *loop closure* (e.g., some features in the map are observed from the current location of the robot). Regarding such limitation, some approaches are further proposed by investigating the partition methods about batch SLAM formulations [18], [19]. Based on their ideas, a batch estimator is divided into two conditionally independent parts, each of which can be solved by a single thread. While those solutions take advantage of multi-thread computation, the tasks in their different threads are not always decoupled. Especially, when loop closure occurs either the exchange of information [18] or the transition between modes [19] needs to be performed in those solutions in order to have accurate state estimates. Note that, by solving the SLAM as a single problem, those solutions follow the Markov assumption, and as a result, the optimality of their estimation results is up to the approximations adopted for efficiency.

In contrast to that, another approach termed parallel tracking and mapping (PTAM) splits a single SLAM problem into two sub-problems: i) motion tracking and ii) map building, and solves them using two parallel pipelines [20], which has become popular recently. Its advantage is straightforward as the above pipelines are decoupled and can run independently, thus leading to time efficient solution. Inspired by that, most recent SLAM algorithms are designed with a high-frequency *front end* for estimating the most recent poses and observed features of the robot, and a relative low-frequency *back end* for jointly optimizing the past states in the map (e.g., [21], [22], [23], [24], [25], [26]). Typically, the estimation in the front end is formulated as a sliding-window visual(-inertial) odometry (VO/VIO), while the batch optimization is formed in the back end like a BA problem. However, we should note

This work was partially supported by the University of Delaware (UD) College of Engineering and the NSF (IIS-1924897).

The authors are with the Department of Mechanical Engineering, University of Delaware, Newark, DE, USA. {zhuai|ghuang}@udel.edu

that the following practices are adopted in those algorithms: a) loop-closure measurements are used by the front end for relocalization, where the observed map features are assumed to be perfectly known due to incomplete prior information, and b) the back end performs batch optimization over only a (heuristically) selected subset of the past states (e.g., the keyframes [20]), where the current states being estimated in the front end are excluded. Both of them violate the Markov assumption. Especially, the former causes the inconsistency issue as pointed out in [19], whereas in the latter the current states cannot be optimized with the past states together such that the loop-closing result is not jointly optimal. Thus, the accuracy of those algorithms cannot always be guaranteed.

To overcome the above issues, in this paper, we propose an exact parallel estimator that performs MAP estimation in both pipelines, however, following the Markov assumption. Especially, by noting that those issues are tightly related to the processing of loop-closure measurements, we introduce the following design for each pipeline: i) The front end is not responsible for relocalization, but focuses on the estimation about the current (sliding-window) states, and ii) The loop-closure measurements are fused by the back end for global optimization which includes both the past (i.e., marginalized by the front end) and the current (i.e., estimated in the front front end) states. According to that, the front end performs pure VO/VIO which does not include any past state, thus the relevant inconsistency issue is avoided. On the other hand, the back end needs to be capable of tracking the correlation between the past and current states, which, not like the batch optimization, should be performed at the same frequency as the front end. For this purpose, an asynchronous solution is developed based on the information-based MAP estimation. As a result, global optimization can be performed as a batch MAP estimator, and the global optimality of loop-closing result can be guaranteed (up to some essential approximation). As a summary, we highlight our main contributions:

- Inspired by PTAM, we propose a new parallel solution for the probabilistic SLAM problem. In particular, both pipelines are realized by the MAP estimators, while the Markov assumption is followed in our solution so as to realize an optimal parallel SLAM formulation. As both the time efficiency of PTAM and the Markov property of MAP estimation are kept in our solution, we term it Markov parallel tracking and mapping (MPTAM).
- Based on that, we implement our visual-inertial SLAM system and evaluate its performance. Especially, compared with the state-of-the-art visual-inertial approach, our system achieves better performance on both benchmark dataset and another challenging long-term dataset.

II. Probabilistic SLAM Formulation

In this section, we first introduce the probabilistic SLAM problem and then present its batch solution. Especially, we have the state vector $\mathbf{x}$ comprising the pose of the robot and the positions of the observed features at every timestep.[1] In addition, we assume that the measurements come from both visual (e.g., a camera) and inertial (e.g., an IMU) sensors.[2]

On the probabilistic SLAM problem, at every timestep we want to find the MAP estimate of $\mathbf{x}$, given current available measurements, $\mathcal{Z}$, and a prior $p(\mathbf{x})$. Especially, the posterior probability of $\mathbf{x}$ can be expressed as

$$p(\mathbf{x}|\mathcal{Z}) \propto p(\mathbf{x})p(\mathcal{Z}|\mathbf{x}) = p(\mathbf{x}) \prod_{\mathbf{z}_i \in \mathcal{Z}} p(\mathbf{z}_i|\mathbf{x}) \tag{1}$$

where the prior follows $\mathcal{N}(\hat{\mathbf{x}}, \mathbf{\Lambda})$, and for each measurement we have $\mathbf{z}_i = \mathbf{h}_i(\mathbf{x}) + \mathbf{n}_i$, with $\mathbf{h}_i(\cdot)$ the measurement model and $\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_i)$ the zero-mean white Gaussian noise. The MAP estimate corresponds to the maximum of $p(\mathbf{x}|\mathcal{Z})$, and we can find it by minimizing the negative logarithm of (1):

$$\begin{aligned} \mathbf{x}^* &= \arg\min_{\mathbf{x}} -\log p(\mathbf{x}|\mathcal{Z}) \\ &= \arg\min_{\mathbf{x}} \|\mathbf{x} - \hat{\mathbf{x}}\|^2_{\mathbf{\Lambda}} + \sum_{\mathbf{z}_i \in \mathcal{Z}} \|\mathbf{z}_i - \mathbf{h}_i(\mathbf{x})\|^2_{\mathbf{\Sigma}_i} \end{aligned} \tag{2}$$

where we have used the notation $\|\mathbf{e}\|^2_{\mathbf{\Omega}} = \mathbf{e}^\top \mathbf{\Omega}^{-1} \mathbf{e}$, that is the squared Mahalanobis norm of $\mathbf{e}$ with the covariance $\mathbf{\Omega}$.

Note that, in general, (2) is a nonlinear problem, thus we can linearize its cost function at $\hat{\mathbf{x}}$ to facilitate the solution:

$$\begin{aligned} \mathcal{C}(\hat{\mathbf{x}} + \tilde{\mathbf{x}}) &\simeq \|\tilde{\mathbf{x}}\|^2_{\mathbf{\Lambda}} + \sum_{\mathbf{z}_i \in \mathcal{Z}} \|\mathbf{z}_i - \mathbf{h}_i(\hat{\mathbf{x}}) - \mathbf{H}_i \tilde{\mathbf{x}}\|^2_{\mathbf{\Sigma}_i} \\ &= \|\tilde{\mathbf{x}}\|^2_{\mathbf{\Lambda}} + \|\mathbf{H}\tilde{\mathbf{x}} - \mathbf{e}\|^2_{\mathbf{\Sigma}} \end{aligned} \tag{3}$$

where we have stacked all of the Jacobians $\mathbf{H}_i$ and residuals $\mathbf{e}_i = \mathbf{z}_i - \mathbf{h}_i(\hat{\mathbf{x}})$ to have $\mathbf{H}$ and $\mathbf{e}$, with $\mathbf{\Sigma}$ a diagonal matrix having the (block) entries $\mathbf{\Sigma}_i$. Therefore, instead of finding the optimal estimate of $\mathbf{x}$ directly, we solve for the optimal update to the current estimate $\hat{\mathbf{x}}$:

$$\tilde{\mathbf{x}}^\oplus = \arg\min_{\tilde{\mathbf{x}}} \|\tilde{\mathbf{x}}\|^2_{\mathbf{\Lambda}} + \|\mathbf{H}\tilde{\mathbf{x}} - \mathbf{e}\|^2_{\mathbf{\Sigma}} \tag{4}$$

where the superscript $\oplus$ means this solution is optimal up to the linearization errors. Note also that, with the (upper triangular) square root matrices of $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$, we can convert (4) into *linear* least-squares form for better numerical stability:

$$\tilde{\mathbf{x}}^\oplus = \arg\min_{\tilde{\mathbf{x}}} \|\boldsymbol{I}\tilde{\mathbf{x}}\|^2 + \|\mathbf{J}\tilde{\mathbf{x}} - \mathbf{r}\|^2 \tag{5}$$

where $\boldsymbol{I} = \mathbf{\Lambda}^{-1/2}$ (i.e., the *square root information* matrix), and for $\mathbf{J}$ and $\mathbf{r}$ we have $\mathbf{J}_i = \mathbf{\Sigma}_i^{-1/2}\mathbf{H}_i$ and $\mathbf{r}_i = \mathbf{\Sigma}_i^{-1/2}\mathbf{e}_i$ with respect to each measurement $\mathbf{z}_i$. As a result, the cost function in (5) can be reshaped using QR factorization [29]:

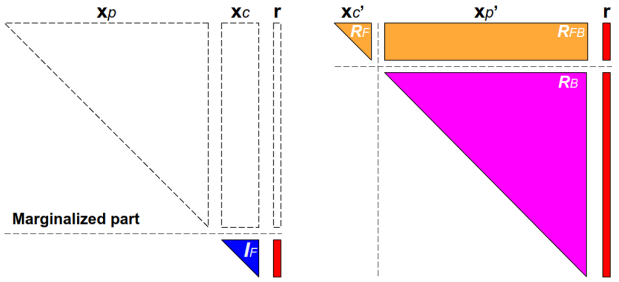
$$\begin{aligned} \mathcal{C} &= \left\| \begin{bmatrix} \boldsymbol{I} \\ \mathbf{J} \end{bmatrix} \tilde{\mathbf{x}} - \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix} \right\|^2 = \left\| \mathbf{Q} \begin{bmatrix} \boldsymbol{I}^\oplus \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} - \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix} \right\|^2 \\ &= \left\| \begin{bmatrix} \boldsymbol{I}^\oplus \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} - \mathbf{Q}^\top \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix} \right\|^2 = \left\| \begin{bmatrix} \boldsymbol{I}^\oplus \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} - \begin{bmatrix} \mathbf{r}^\oplus \\ \boldsymbol{\epsilon} \end{bmatrix} \right\|^2 \end{aligned} \tag{6}$$

By leaving $\|\boldsymbol{\epsilon}\|^2$ as the least-squares residual, we have

$$\tilde{\mathbf{x}}^\oplus = \arg\min_{\tilde{\mathbf{x}}} \|\boldsymbol{I}^\oplus \tilde{\mathbf{x}} - \mathbf{r}^\oplus\|^2 = \boldsymbol{I}^{\oplus^{-1}} \mathbf{r}^\oplus \tag{7}$$

[1] In what follows, $\hat{\mathbf{x}}$ is used to denote the estimate of $\mathbf{x}$, and $\tilde{\mathbf{x}} \triangleq \mathbf{x} - \hat{\mathbf{x}}$ is the corresponding (optimal) correction to this estimate.

[2] Here we assume that the visual measurements correspond to the pinhole camera model [27], and the inertial measurements to the IMU preintegration model [15], [28], in order for a uniform expression in the following context.



(a) Front-end optimization ($\mathcal{C}_F$) (b) Back-end optimization ($\mathcal{C}_B$)

Fig. 1: An overview of our proposed MPTAM estimator.

which can be efficiently found via back substitution, and the state estimate is then updated as: $\hat{\mathbf{x}}^{\oplus} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}^{\oplus}$. We should note that when $\mathcal{Z}$ includes loop-closure measurements, $\mathbf{J}$ has (non-zero) sub-blocks that correspond to the earlier observed features and the current pose, for which some particular case can have the complexity of the QR factorization quadratic to the number of states (i.e., all the columns in $\boldsymbol{I}$ are involved). Noting that the sparsity of $\boldsymbol{I}$ plays a dominant role for the efficiency of the estimator in that case, some approach like the variable reordering [8], [9] or information sparsification [30], [31] can be used. However, as the dimension of $\boldsymbol{I}$ grows, the computational efficiency of those approaches may adversely impact the performance of the estimator [9], [31]. Therefore, for large-scale SLAM problem an optimal batch solution may still be too computationally expensive to obtain in real time.

III. Description of Parallel Estimator

Inspired by the functional decoupling for PTAM, we split the SLAM process into two pipelines running in parallel: a) The front end estimates the most recent states by performing exact sliding-window VIO, while b) The back end performs batch optimization over the entire state when loop closure is detected. Both pipelines are realized by the MAP estimators which still use the same measurements as the batch estimator. Specifically, following the chronological order, we partition $\mathbf{x}$ into the previous states, $\mathbf{x}_p$, and current states, $\mathbf{x}_c$ (i.e., the states within the current sliding window): $\mathbf{x} = [\mathbf{x}_p^\top \ \mathbf{x}_c^\top]^\top$.

A. Front end

For sliding-window estimator, after the estimation at current timestep is finished the earliest states in the window are marginalized out of $\mathcal{C}$ before next timestep. Especially, with the square root information form we can consistently realize that by only keeping the information factor corresponding to the current states (see Figure 1a). As a result, every timestep after marginalization the information matrix of the front end is $\boldsymbol{I}_{cc}^{\oplus} =: \boldsymbol{I}_F$, and for next timestep its cost function becomes

$$\mathcal{C}_F = \|\boldsymbol{I}_F \tilde{\mathbf{x}}_c\|^2 + \|\mathbf{J}\tilde{\mathbf{x}}_c - \mathbf{r}\|^2 \tag{8}$$

Similar to (6), by minimizing $\mathcal{C}_F$, we obtain $\tilde{\mathbf{x}}_c^{\oplus} = \boldsymbol{I}_F^{\oplus\,-1}\mathbf{r}^{\oplus}$, and the current state estimate is updated as: $\hat{\mathbf{x}}_c^{\oplus} = \hat{\mathbf{x}}_c + \tilde{\mathbf{x}}_c^{\oplus}$. Especially, there are different consistent methods that can be used to realize this pipeline (e.g., [11], [12], [16], [32], [33]). Also, to reduce the workload of developing a SLAM system, the open-sourced algorithms can be directly integrated to or modified for this pipeline, as long as they can provide real-time state estimates (e.g., [13], [16], [17], [34]), which will be shown with our SLAM system used for the experiments.

B. Back end

In order to include all the states for global adjustment, the correlation between $\mathbf{x}_p$ and $\mathbf{x}_c$ is taken into account which is updated in real time in the back-end pipeline. To coordinate this high-frequency operation with the low-frequency batch optimization, an *asynchronous* update scheme is developed where reverse chronological variable ordering is adopted to reform the information factor, such that the QR factorization for tracking the state correlation and the back substitution for computing batch correction can be performed concurrently.

The back-end information matrix is denoted as $\boldsymbol{R}$ which, regardless of the order of variables, is equivalent to $\boldsymbol{I}$ in (5). Accordingly, the information factor $\mathcal{C}_B$ can be expressed as

$$\mathcal{C}_B = \|\boldsymbol{R}\tilde{\mathbf{x}}'\|^2 = \left\| \begin{bmatrix} \boldsymbol{R}_{cc} & \boldsymbol{R}_{cp} \\ & \boldsymbol{R}_{pp} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}'_c \\ \tilde{\mathbf{x}}'_p \end{bmatrix} \right\|^2 \tag{9}$$

where the superscript $'$ means reverse chronological order. In particular, if we further denote $\boldsymbol{R}_{cc} =: \boldsymbol{R}_F$, $\boldsymbol{R}_{cp} =: \boldsymbol{R}_{FB}$ and $\boldsymbol{R}_{pp} =: \boldsymbol{R}_B$, and consider the existence of residual during the estimation, then $\mathcal{C}_B$ can be divided into two parts:

$$\overline{\mathcal{C}}_B = \|\boldsymbol{R}_F\tilde{\mathbf{x}}'_c + \boldsymbol{R}_{FB}\tilde{\mathbf{x}}'_p - \mathbf{r}_{FB}\|^2 \tag{10}$$

$$\underline{\mathcal{C}}_B = \|\boldsymbol{R}_B\tilde{\mathbf{x}}'_p - \mathbf{r}_B\|^2 \tag{11}$$

where, as illustrated in Figure 1b, $\overline{\mathcal{C}}_B$ is used for updating the correlation with the current sliding-window states, which we term information augmentation; while $\underline{\mathcal{C}}_B$ is prepared for the batch optimization until loop-closure measurements are included for update. In what follows, we use a *LEGO Brisk*-like representation to visualize the QR factorization over the information factors. Also, in our examples a sliding window which contains the pose of the robot and its observed features at current timestep only is used, for the sake of introduction.

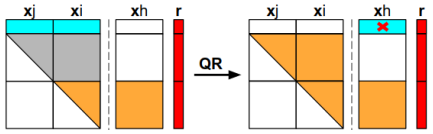
1) Information augmentation: Once $\hat{\mathbf{x}}_c^{\oplus}$ is obtained from the front end, we update $\mathcal{C}_B$ by using the same measurements whose factor $\|\mathbf{J}_{\mathcal{I}}\tilde{\mathbf{x}}' - \mathbf{r}_{\mathcal{I}}\|^2$ is linearized at $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_p^\top \ \hat{\mathbf{x}}_c^{\oplus\top}]^\top$ (see the example in Figure 2). Especially, for $\overline{\mathcal{C}}_B$ we have

$$\begin{aligned}
\overline{\mathcal{C}}_B &= \|\boldsymbol{R}_F\tilde{\mathbf{x}}'_c + \boldsymbol{R}_{FB}\tilde{\mathbf{x}}'_p - \mathbf{r}_{FB}\|^2 + \|\mathbf{J}_{\mathcal{I}}\tilde{\mathbf{x}}' - \mathbf{r}_{\mathcal{I}}\|^2 \\
&= \left\| \begin{bmatrix} \mathbf{J}_{\mathcal{I}(c)} & \mathbf{0} \\ \boldsymbol{R}_F & \boldsymbol{R}_{FB} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}'_c \\ \tilde{\mathbf{x}}'_p \end{bmatrix} - \begin{bmatrix} \mathbf{r}_{\mathcal{I}} \\ \mathbf{r}_{FB} \end{bmatrix} \right\|^2 \\
&= \|\boldsymbol{R}_F^{\oplus}\tilde{\mathbf{x}}'_c + \boldsymbol{R}_{FB}^{\oplus}\tilde{\mathbf{x}}'_p - \mathbf{r}_{FB}^{\oplus}\|^2 + \|\mathbf{J}_{\mathcal{I}(p)}^{\oplus}\tilde{\mathbf{x}}'_p - \mathbf{r}_{\mathcal{I}}^{\oplus}\|^2 \\
&= \overline{\mathcal{C}}_B^{\oplus} + \mathcal{C}_{\mathcal{I}}^{\oplus}
\end{aligned} \tag{12}$$

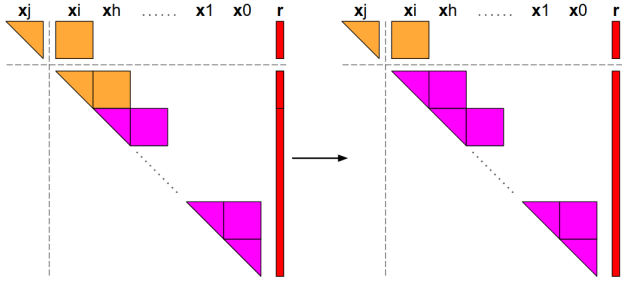
for which the following QR factorization has been applied:

$$\begin{bmatrix} \mathbf{J}_{\mathcal{I}(c)} \\ \boldsymbol{R}_F \end{bmatrix} = \mathrm{Q} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{R}_F^{\oplus} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{J}_{\mathcal{I}(p)}^{\oplus} \\ \boldsymbol{R}_{FB}^{\oplus} \end{bmatrix} = \mathrm{Q}^\top \begin{bmatrix} \mathbf{0} \\ \boldsymbol{R}_{FB} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{r}_{\mathcal{I}}^{\oplus} \\ \mathbf{r}_{FB}^{\oplus} \end{bmatrix} = \mathbf{Q}^\top \begin{bmatrix} \mathbf{r}_{\mathcal{I}} \\ \mathbf{r}_{FB} \end{bmatrix}$$

To improve the consistency of estimation [35] and reduce the computation, $\mathcal{C}_{\mathcal{I}}^{\oplus}$ will not be used to update $\underline{\mathcal{C}}_B$ but dropped (see Figure 2a). Meanwhile, $\underline{\mathcal{C}}_B$ is augmented by the factor that corresponds to the states marginalized by the front end, which does not need extra computation (see Figure 2b).



(a) Left: at timestep j, $\overline{\mathcal{C}}_B$ is updated by the factor $\|\mathbf{J}_{\mathcal{I}}\tilde{\mathbf{x}}' - \mathbf{r}_{\mathcal{I}}\|^2$: i) motion prediction and feature initialization (*gray*) and ii) feature observations w/o loop closure (*cyan*). Right: after QR factorization, the intermediate term $\mathcal{C}_{\mathcal{I}}^{\oplus}$ (cyan) is dropped (marked with '$\times$').



(b) The factor in $\overline{\mathcal{C}}_B$ which corresponds to the out-of-window state $\mathbf{x}_i$ (yellow) is moved to $\underline{\mathcal{C}}_B$, which does not incur any computation.

Fig. 2: Back end (Information augmentation).

2) Global optimization: Once detecting loop-closure measurements, we use them to update $\mathcal{C}_B$ with the other measurements that have been used in the front end together, and start global optimization (see the example in Figure 3). Similar to the information augmentation, those measurements form the factor $\|\mathbf{J}_{\mathcal{L}}\tilde{\mathbf{x}}' - \mathbf{r}_{\mathcal{L}}\|^2$, and we use it to first update $\overline{\mathcal{C}}_B$:

$$\begin{aligned}\overline{\mathcal{C}}_B &= \|\boldsymbol{R}_F\tilde{\mathbf{x}}'_c + \boldsymbol{R}_{FB}\tilde{\mathbf{x}}'_p - \mathbf{r}_{FB}\|^2 + \|\mathbf{J}_{\mathcal{L}}\tilde{\mathbf{x}}' - \mathbf{r}_{\mathcal{L}}\|^2 \\ &= \left\| \begin{bmatrix} \mathbf{J}_{\mathcal{L}(c)} & \mathbf{J}_{\mathcal{L}(p)} \\ \boldsymbol{R}_F & \boldsymbol{R}_{FB} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}'_c \\ \tilde{\mathbf{x}}'_p \end{bmatrix} - \begin{bmatrix} \mathbf{r}_{\mathcal{L}} \\ \mathbf{r}_{FB} \end{bmatrix} \right\|^2 \\ &= \|\boldsymbol{R}^{\oplus}_F\tilde{\mathbf{x}}'_c + \boldsymbol{R}^{\oplus}_{FB}\tilde{\mathbf{x}}'_p - \mathbf{r}^{\oplus}_{FB}\|^2 + \|\mathbf{J}^{\oplus}_{\mathcal{L}(p)}\tilde{\mathbf{x}}'_p - \mathbf{r}^{\oplus}_{\mathcal{L}}\|^2 \\ &= \overline{\mathcal{C}}_B^{\oplus} + \mathcal{C}_{\mathcal{L}}^{\oplus} \end{aligned} \tag{13}$$

for which we have applied the following QR factorization:

$$\begin{bmatrix} \mathbf{J}_{\mathcal{L}(c)} \\ \boldsymbol{R}_F \end{bmatrix} = \mathrm{Q} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{R}^{\oplus}_F \end{bmatrix}, \quad \begin{bmatrix} \mathbf{J}^{\oplus}_{\mathcal{L}(p)} \\ \boldsymbol{R}^{\oplus}_{FB} \end{bmatrix} = \mathrm{Q}^{\top} \begin{bmatrix} \mathbf{J}_{\mathcal{L}(p)} \\ \boldsymbol{R}_{FB} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{r}^{\oplus}_{\mathcal{L}} \\ \mathbf{r}^{\oplus}_{FB} \end{bmatrix} = \mathrm{Q}^{\top} \begin{bmatrix} \mathbf{r}_{\mathcal{L}} \\ \mathbf{r}_{FB} \end{bmatrix}$$

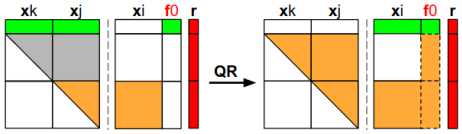
Note that $\mathcal{C}_{\mathcal{L}}^{\oplus}$ needs to be factorized in $\underline{\mathcal{C}}_B$ in order to trigger the batch optimization (see Figure 3a). Especially, this step can be finished via another QR factorization (see Figure 3b):

$$\begin{aligned}\underline{\mathcal{C}}_B &= \|\boldsymbol{R}_B\tilde{\mathbf{x}}'_p - \mathbf{r}_B\|^2 + \|\mathbf{J}^{\oplus}_{\mathcal{L}(p)}\tilde{\mathbf{x}}'_p - \mathbf{r}^{\oplus}_{\mathcal{L}}\|^2 \\ &= \|\boldsymbol{R}^{\oplus}_B\tilde{\mathbf{x}}'_p - \mathbf{r}^{\oplus}_B\|^2 + \|\boldsymbol{\epsilon}\|^2 \\ &= \underline{\mathcal{C}}_B^{\oplus} + \|\boldsymbol{\epsilon}\|^2 \end{aligned} \tag{14}$$

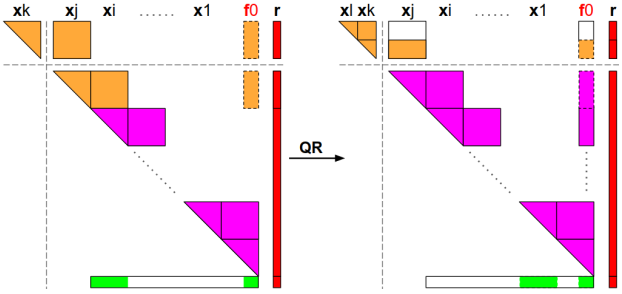
At this point, the optimal update $\tilde{\mathbf{x}}'^{\oplus}_p$ can be first obtained by minimizing $\underline{\mathcal{C}}_B^{\oplus}$:

$$\tilde{\mathbf{x}}'^{\oplus}_p = \arg\min_{\tilde{\mathbf{x}}'_p} \|\boldsymbol{R}^{\oplus}_B\tilde{\mathbf{x}}'_p - \mathbf{r}^{\oplus}_B\|^2 = \boldsymbol{R}_B^{\oplus\,-1}\mathbf{r}^{\oplus}_B \tag{15}$$

Note that during the processing of (14) and (15), $\overline{\mathcal{C}}_B^{\oplus}$ is also updated by new incoming measurements (see Figure 3b), as in (12). Therefore, once having the result of (15), we further



(a) Left: at timestep k, $\overline{\mathcal{C}}_B$ is updated by the factor $\|\mathbf{J}_{\mathcal{L}}\tilde{\mathbf{x}}' - \mathbf{r}_{\mathcal{L}}\|^2$: i) motion prediction and feature initialization (*gray*) and ii) feature observations w/ loop closure (*green*). Right: after QR factorization, the intermediate term $\mathcal{C}_{\mathcal{L}}^{\oplus}$ (green) will be further processed in $\underline{\mathcal{C}}_B$.



(b) Left: $\underline{\mathcal{C}}_B$ is augmented by the information factor corresponding to $\mathbf{x}_j$ (yellow), and ready for the QR factorization with $\mathcal{C}_{\mathcal{L}}^{\oplus}$. Right: during the QR factorization on $\underline{\mathcal{C}}_B$, $\overline{\mathcal{C}}_B$ is available to be updated by the factors of new incoming measurements (e.g., at timestep l).

Fig. 3: Back end (Global optimization).

obtain the optimal update $\tilde{\mathbf{x}}'^{\oplus}_c$ by minimizing $\overline{\mathcal{C}}_B^{\oplus}$:

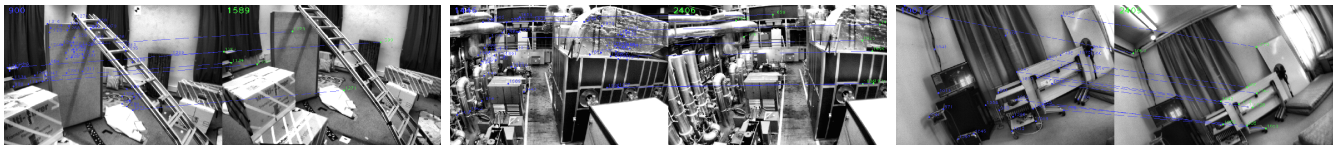
$$\begin{aligned}\tilde{\mathbf{x}}'^{\oplus}_c &= \arg\min_{\tilde{\mathbf{x}}'_c} \|\boldsymbol{R}^{\oplus}_F\tilde{\mathbf{x}}'_c + \boldsymbol{R}^{\oplus}_{FB}\tilde{\mathbf{x}}'^{\oplus}_p - \mathbf{r}^{\oplus}_{FB}\|^2 \\ &= \boldsymbol{R}_F^{\oplus\,-1}(\mathbf{r}^{\oplus}_{FB} - \boldsymbol{R}^{\oplus}_{FB}\tilde{\mathbf{x}}'^{\oplus}_p) \end{aligned} \tag{16}$$

where both $\tilde{\mathbf{x}}'^{\oplus}_p$ and $\tilde{\mathbf{x}}'^{\oplus}_c$ can be efficiently found via back substitution. After rearranging $\tilde{\mathbf{x}}'^{\oplus}_p$ and $\tilde{\mathbf{x}}'^{\oplus}_c$ in chronological order, we have loop-closing correction: $\tilde{\mathbf{x}}^{\oplus} = [\tilde{\mathbf{x}}_p^{\oplus\top}\ \tilde{\mathbf{x}}_c^{\oplus\top}]^{\top}$, and global optimization can be performed as: $\hat{\mathbf{x}}^{\oplus} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}^{\oplus}$. It should be noted that the immediate estimate $\hat{\mathbf{x}}_c^{\oplus}$ obtained from the front end is further updated by loop closure ($\hat{\mathbf{x}}_c^{\oplus} \leftarrow \hat{\mathbf{x}}_c^{\oplus} + \tilde{\mathbf{x}}_c^{\oplus}$, with slight abuse of notation), which, in another word, demonstrates that global optimization in our solution *simultaneously* improves the accuracy of both pipelines.

3) Resource-aware settings (optional): Although the back end is allowed to run at a low processing speed, we prefer to have loop-closing results as fast as possible so that the drifts in the map can be corrected timely. As mentioned earlier the sparsity of $\boldsymbol{R}$ dominates the efficiency of batch optimization, thus we consider two approaches to improve its speed.

First, as suggested in [35], $\boldsymbol{R}_B$ can be approximated as a diagonally banded matrix which favors efficient factorization. To this end, we can only keep the off-diagonal entries of $\boldsymbol{R}_B$ which are within a certain *bandwidth* (e.g., in Figure 2b the bandwidth of $\boldsymbol{R}_B$ is one (sliding) window) every time after finishing global optimization.

Second, we should notice that the square-root information form makes single-precision floating-point arithmetic available for numerical computation [14], where a 32-bit floating-point value can be treated as zero if it is smaller than 10^{-7}. Thus, after QR factorization (e.g., Givens method [29]), some parts of $\boldsymbol{R}$ may numerically approach to zero (e.g., the dotted sub-blocks in Figure 2 and 3). By zeroing those parts after QR factorization the sparsity of $\boldsymbol{R}$ is also improved.



(a) EuRoC dataset (Vicon room) (b) EuRoC dataset (Machine hall) (c) Vicon Loops dataset

Fig. 4: Loop-closure detection results of our system on each dataset (the correspondences are connected by *blue* lines).

TABLE I: Real-time pose accuracy (RMSE) and average computation time (of the front end) on EuRoC dataset.

	VINS-Mono					MPTAM				
	w/o loop closure		w/ loop closure			w/o loop closure		w/ loop closure		
	Orientation	Translation	Orientation	Translation	Time	Orientation	Translation	Orientation	Translation	Time
	[deg]	[m]	[deg]	[m]	[ms]	[deg]	[m]	[deg]	[m]	[ms]
V1_01_easy	6.30	0.08	**6.07**	0.14	69	6.25	0.10	**5.67**	**0.09**	13
V1_02_medium	3.27	0.11	**2.92**	**0.08**	50	3.67	0.09	**2.23**	**0.08**	9
V1_03_difficult	6.07	0.18	7.39	0.31	36	4.76	0.18	**2.29**	**0.15**	5
V2_01_easy	2.06	0.08	**1.72**	0.10	36	4.70	0.18	**1.14**	**0.15**	10
V2_02_medium	4.24	0.16	**3.44**	**0.14**	47	2.69	0.18	**1.71**	**0.10**	7
V2_03_difficult	3.21	0.27	**3.09**	**0.20**	28	3.78	0.40	**2.92**	**0.38**	3
MH_01_easy	1.20	0.15	**1.09**	0.16	68	4.58	0.28	**2.41**	**0.24**	11
MH_02_easy	1.20	0.18	1.32	0.22	64	5.61	0.45	**3.50**	**0.41**	6
MH_03_medium	1.55	0.19	1.78	**0.11**	65	4.93	0.30	**1.14**	**0.16**	11
MH_04_difficult	1.49	0.34	**1.32**	0.41	58	2.35	0.77	**2.29**	**0.67**	4
MH_05_difficult	0.69	0.30	**0.69**	**0.27**	61	5.10	0.69	**1.14**	**0.46**	10

The underlined value means the corresponding accuracy is *not* improved as expected after loop-closure corrections.

IV. Experimental Results

To demonstrate the performance of our proposed MPTAM in real application, we implement our visual-inertial SLAM system based on our previous work, R-VIO [16], a sliding-window filtering-based odometry algorithm written in C++. Specifically, as the front end we modify it according to [36] to jointly estimate the observed features in the sliding window. The implementation of the back end follows Section III-B, where QR factorization is carried out using the algorithms provided by Eigen [37]. In addition to that, the loop-closure detector running with the front end is implemented using i) DBoW2 [38] (place recognition), ii) BRIEF [39] (descriptor matching), and iii) epipolar geometry-based outlier rejection to provide loop-closure measurements. Also, the parameters for loop-closure detection are the same for both systems.

We compare our MPTAM with the state-of-the-art VINS-Mono [25]. Different with our system, the relinearization is enabled by default in VINS-Mono such that the nonlinearity of the SLAM formulation can be approximated better. Then, the comparison in this section shall also release the priority between a) compliance of the Markov property and b) use of relinearization, for a SLAM estimator. Especially, two challenging visual-inertial datasets are used for our evaluation (see Figure 4), and all the tests run on a laptop with Intel Core i7-4710MQ 2.5GHz×8 CPU in *real time*.

A. EuRoC Dataset

As a benchmark for evaluating the performance of SLAM estimator, this dataset provides calibrated camera (20Hz) and IMU (200Hz) measurements which are recorded with a VI-sensor on the micro aerial vehicle. Ground truth is captured by the Vicon system and aligned with the coordinate of VI-sensor. Especially, for MPTAM 200 features are tracked per image, and the sliding window contains 15 recent poses and at most 50 observed features, while for VINS-Mono we use its default settings to guarantee the best performance.

For real-time applications, we focus more on the accuracy of the front-end estimates. Especially, for a parallel estimator this is able to be improved by performing global adjustment, which, however, is realized by VINS-Mono and our MPTAM in different ways: a) For VINS-Mono, its front-end estimates are adjusted based on a map (in terms of keyframe) which is refined by its back-end optimization with loop closure, while b) For MPTAM, as mentioned in Section III-B.2, the states estimated in both pipelines are corrected simultaneously by doing global optimization. Therefore, we can also term them a) *non-probabilistic* and b) *probabilistic* feedback schemes.

To compare the performance of these two approaches, for both estimators we record pose estimates of their front ends with and without loop-closure corrections at the same time. The corresponding RMSE with respect to the ground truth are computed and presented in Table I. We can clearly find that in all the sequences the pose errors of proposed MPTAM are decreased as expected with loop-closure corrections. In contrast to that, such trend is not reflected from the results of VINS-Mono. In particular, we should notice that over half of the sequences its pose errors are unexpectedly increased after having loop-closure corrections, although the relinearization is performed by VINS-Mono for its both pipelines. It should also be noted that our approach improves the accuracy of the front end and still keeps its computational efficiency.

B. Vicon Loops Dataset

As the traveling distances in EuRoC dataset are short (e.g., the longest sequence is only 130m), we further evaluate both estimators on a long-term dataset [13], where a handheld VI-sensor undergoes 6-DOF highly dynamic motion and travels 1200m in 14 minutes. The cameras run at 20Hz and an IMU runs at 800Hz. Similar to EuRoC dataset, the ground truth is

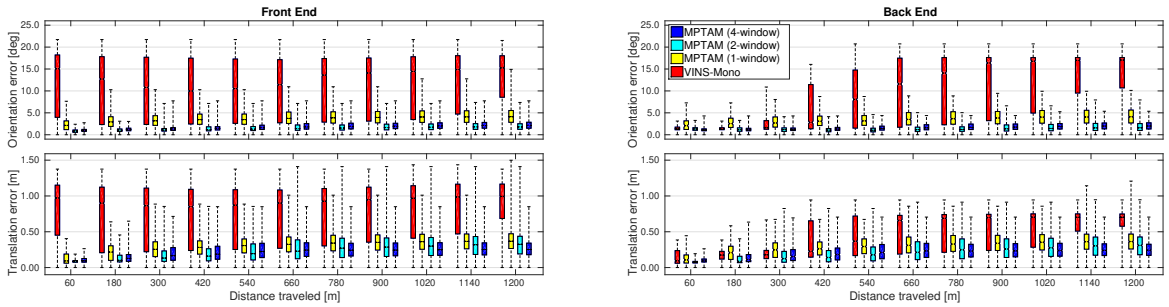


Fig. 5: Absolute pose error statistics in terms of extremum, median, 1st and 3rd quartiles, for Vicon Loops dataset.

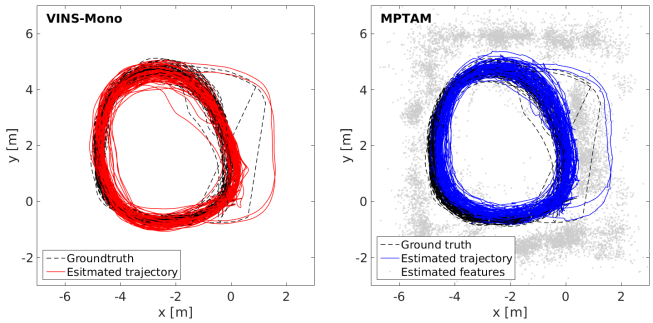


Fig. 6: The estimated maps on Vicon Loops dataset: note that in VINS-Mono the features are associated to the keyframes, and not involved in its back-end (pose-graph) optimization.

provided by the Vicon system. Only the left-camera images are used, for which 100 features are tracked (per image) by both estimators. As loop closure can be detected frequently in this dataset, the sliding window of MPTAM has been resized to include 15 recent poses and at most 20 observed features. By noting that the information matrix fill-ins resulting from the frequent loop closures impact the efficiency of MPTAM, we enable resource-aware options. In particular, three types of bandwidth (1-, 2- and 4-window) are tested for $\boldsymbol{R}_B$. For VINS-Mono, the same settings for EuRoC dataset are used, except for the number of features to be tracked.

The final outputs along with the ground truth are shown in Figure 6, where the map of our MPTAM consists of 16431 poses of the sensor and 11453 observed features. In order to evaluate the accuracy of the estimators, we adopt the metric proposed in [13]. According to that, the absolute pose errors are grouped regarding different traveling distances which are increased from the origin, such that the estimation accuracy is able to be illustrated by the resultant error statistics. The corresponding results with respect to 11 traveling distances $\{60, 180, 300, 420, 540, 660, 780, 900, 1020, 1140, 1200\}$ are presented in Figure 5, where the evolution of pose errors has been described by a series of box-plots. Based on the above results, the followings can be pointed out for VINS-Mono:

- Non-probabilistic feedback does not always improve the accuracy of the front end. Except for {60,180,300}, the pose errors (medians) for its back end are going up that is in response to a similar trend for its front end.
- Batch optimization in its back end leads to biased state estimates. As the factors in the pose graph are weighted equally, its result is prone to local minimum upon iterations, as is reflected by {660,780,900,1020,1140,1200}.

TABLE II: Pose estimation accuracy on Vicon Loops dataset (w/ 1-, 2-, and 4-window bandwidths for $\boldsymbol{R}_B$ in MPTAM).

		VINS-Mono	MPTAM
Number of loop-closure updates		445	372 / 382 / 381
Front end	Orien. MAE [deg]	12.48	4.24 / 2.02 / **2.24**
	Trans. MAE [m]	0.82	0.37 / 0.32 / **0.26**
Back end	Orien. MAE [deg]	13.55	4.14 / 1.88 / **2.16**
	Trans. MAE [m]	0.60	0.35 / 0.30 / **0.26**

In contrast, for our proposed MPTAM we can find that

- Our probabilistic feedback lets the front end achieve the same level of accuracy as the back end (the translation errors (medians) are consistently smaller than 0.5m that is 0.04% of the total traveling distance).
- The accuracy of loop closing can be guaranteed even by the conservatively approximated global optimization. It can be expected that by expanding the bandwidth of $\boldsymbol{R}_B$ our MPTAM shall realize higher accuracy.

The corresponding mean absolute errors (MAE) are given in Table II which shows that as compared with VINS-Mono our MPTAM achieves higher accuracy with less loop-closure updates. In particular, the computation costs (in average) of MPTAM are 19ms for the front end, and 0.63/0.79/1.09s for the back end, with respect to the above bandwidth options.

V. Conclusion

Inspired by the functional decoupling principle of PTAM, we propose the Markov parallel tracking and mapping (MPTAM) for probabilistic SLAM. In contrast to the recent proposed SLAM algorithms which perform parallel estimation, in our solution, we explicitly follow the Markov assumption that is the basis of the MAP solution of probabilistic SLAM problem. Both pipelines of MPTAM are realized by the MAP estimators, and the issues about i) estimation consistency and ii) joint optimality for the loop-closing result, which exist in the recent proposed SLAM algorithms have been addressed by properly processing loop-closure measurements. Through the experimental evaluation, we further validate our analysis and demonstrate the superiority of our proposed MPTAM on the accuracy and efficiency in real SLAM application.

ACKNOWLEDGMENT

We would like to thank Kejian Wu and Tong Ke for their valuable discussions about this work. We would also like to thank the anonymous reviewers for their detailed comments.

REFERENCES

[1] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The international journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[2] P. Cheeseman, R. Smith, and M. Self, "A stochastic map for uncertain spatial relationships," in *International Symposium on Robotic Research*. MIT Press Cambridge, 1987, pp. 467–474.

[3] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, vol. 3, 1991, pp. 1442–1447.

[4] P. S. Maybeck, *Stochastic models, estimation, and control*. Academic press, 1982.

[5] S. Thrun, W. Burgard, and D. Fox, "Probalistic robotics," *Kybernetes*, 2006.

[6] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[7] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International Workshop on Vision Algorithms*. Springer, 1999, pp. 298–372.

[8] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[9] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[10] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572.

[11] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[12] G. Huang, M. Kaess, and J. J. Leonard, "Towards consistent visual-inertial navigation," in *IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 4926–4933.

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

[14] K. Wu, A. Ahmed, G. A. Georgiou, and S. I. Roumeliotis, "A square root inverse filter for efficient vision-aided inertial navigation on mobile devices." in *Robotics: Science and Systems*, vol. 2, 2015.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[16] Z. Huai and G. Huang, "Robocentric visual–inertial odometry," *The International Journal of Robotics Research*, 2019. [Online]. Available: https://journals.sagepub.com/doi/10.1177/0278364919853361

[17] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 4666–4672.

[18] S. Williams, V. Indelman, M. Kaess, R. Roberts, J. J. Leonard, and F. Dellaert, "Concurrent filtering and smoothing: A parallel architecture for real-time navigation and full smoothing," *The International Journal of Robotics Research*, vol. 33, no. 12, pp. 1544–1568, 2014.

[19] T. Ke, K. J. Wu, and S. I. Roumeliotis, "Rise-slam: A resource-aware inverse schmidt estimator for slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 354–361.

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

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

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[23] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1885–1892.

[24] 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.

[25] 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.

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

[27] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

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

[29] G. H. Golub and C. F. Van Loan, *Matrix Computations*. JHU Press, 2012, vol. 3.

[30] J. Vial, H. Durrant-Whyte, and T. Bailey, "Conservative sparsification for efficient and consistent approximate estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 886–893.

[31] G. Huang, M. Kaess, and J. J. Leonard, "Consistent sparsification for graph optimization," in *European Conference on Mobile Robots*. IEEE, 2013, pp. 150–157.

[32] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 158–176, 2013.

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

[34] 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, 2018.

[35] K. J. Wu and S. I. Roumeliotis, "Inverse schmidt estimators," MARS, Tech. Rep., 2016, http://mars.cs.umn.edu/tr/inverseschmidt.pdf.

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

[37] G. Guennebaud *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[38] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[39] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European Conference on Computer Vision*. Springer, 2010, pp. 778–792.