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

Bidirectional Trajectory Computation for Odometer-Aided Visual-Inertial SLAM

Jinxu Liu, Wei Gao* and Zhanyi Hu

Abstract— **Odometer-aided visual-inertial SLAM systems typically have a good performance for navigation of wheeled platforms, while they usually suffer from degenerate cases before the first turning. In this paper, firstly we perform an observability analysis w.r.t. the extrinsic parameters before the first turning, which is a complement of the existing results of observability analyses. Secondly, inspired by the above observability analyses, we propose a bidirectional trajectory computation method, by which the poses before the first turning are refined in the backward computation thread, and the real-time trajectory is adjusted accordingly. Experimental results prove that our proposed method not only solves the problem of the unobservability of accelerometer bias and extrinsic parameters before the first turning, but also results in more accurate trajectories in comparison with the state-of-the-art approaches.**

I. INTRODUCTION

Visual-inertial SLAM (VI-SLAM) and visual-inertial odometry (VIO) approaches have received great attention in recent years. Filtering-based methods such as MSCKF [12] and optimization-based methods such as OKVIS [7] make up the two categories of state estimation methods. In general, filter-based approaches have better efficiency while optimization-based methods enjoy higher accuracy [4]. The optimization-based approaches [7], [13] typically optimize a limited number of current states to limit the amount of computation, and use marginalization to make use of previous information to better estimate the current states.

For wheeled platforms such as robots and passenger cars, the accuracy of visual-inertial navigation can be dramatically improved with the aid of wheel encoders [16], [10], [19], [20], [6], [2], [11], [15]. In some methods such as [16], [8], [19], [20], [6], [2], [11], [21] and [22], the IMU measurements and wheel encoder readings are pre-integrated individually. Approaches of this type either need at least two wheel encoders or require the front wheel angle measurement for pre-integration, and have to deal with the problem that the angular velocity provided by the sensors on wheels is always within the ground plane. Other methods such as [15], [18] and [10] jointly pre-integrates the angular velocity from IMU and the linear velocity from wheel encoder. Approaches of this type only need one wheel encoder, and the uneven terrain does not have an impact on the performance of these approaches theoretically. In this paper we use *wheel encoder* and *odometer* to denote the same thing.

However, for applications on ground vehicles, VI-SLAM and VIO approaches often suffer from degenerate cases, even with the aid of wheel encoders. [16] points out two degenerate cases under special motions. Firstly, the scale is unobservable when the platform moves with constant local linear acceleration. Secondly, the roll and pitch angles are unobservable when the platform has no rotational motion. Both the cases are related to the accelerometer bias which can not be correctly estimated under the above special motions. [16] also proves that the first degenerated case can be eliminated with the use of wheel encoder. However, it can be drawn that the second degenerate case still exists in such a case through derivation, which is straightforward and will be presented briefly in our technical report[1]. Besides, the experimental results in [10] also indicate that the accelerometer bias can not be correctly estimated until the first turning with the use of wheel encoder. In addition to the accelerometer bias, some of the extrinsic parameters can not be correctly estimated as well, when the platform has no rotational motion. [17] have proved that the translational component of camera-IMU extrinsic parameters is unobservable in a VIO system when the platform undergoes pure translation, and [22] have proved that the translational component of IMU-odometer extrinsic parameters is unobservable in an odometer-aided VIO system when the platform undergoes pure translation. However, our proposed approach adopts a sensor fusion scheme that is different from [22], i.e. in our proposed approach IMU and wheel encoder measurements are fused in the pre-integration stage. Furthermore, neither of [17] and [22] have analyzed the observability under pure translation along a straight line, which is a common case and renders another direction in extrinsic parameters unobservable. In Section III, we will give an observability analysis for the seven unobservable directions caused by the special motion pattern in extrinsic parameters for an odometer-aided VI-SLAM system, under the circumstance that the platform undergoes pure translation along a straight line. Thanks to the employment of marginalization, the optimization-based VI-SLAM approaches can make use of previous information collected since beginning. Therefore, once the platform performs rotational motion such as making a turn, the accelerometer bias and extrinsic parameters will be correctly estimated from then on. Nevertheless, before the first turning, the inaccuracy which results from the incorrectly estimated accelerometer bias and extrinsic

This work was supported in part by the National Key R&D Program of China (2016YFB0502002), and in part by the Natural Science Foundation of China (61991423, 61872361). (*Corresponding Author: Wei Gao) .

All authors are with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, and with School of Artificial Intelligence, University of Chinese Academy of Sciences, China. {jinxu.liu, wgao, huzy}@nlpr.ia.ac.cn

[1]https://arxiv.org/abs/2002.00195v3

parameters remains a problem for odometer-aided VI-SLAM.

To relieve the inaccuracy before the first turning, [10] proposes to keep the extrinsic parameters constant during nonlinear optimization until the platform has made a turn and the estimation of accelerometer bias has reached convergence. Furthermore, we may further keep the accelerometer bias constant as zero before the first turning to relieve the inaccuracy caused by the incorrectly estimated accelerometer bias. Some initialization approaches for VI-SLAM also deal with unobservability when initial motion does not render all parameters observable. [24] forces accelerometer bias to be close to zero, and [25] adds a prior with the mean value of zero w.r.t. the accelerometer bias. Extrinsic parameters are not calibrated during VI-SLAM initialization. However, since the accelerometer bias is actually not zero and the extrinsic parameters may not be very accurate, the accuracy of trajectory before the first turning may still be deteriorated, especially for outdoor scenes where the vehicle is likely to travel a long distance before the first turning.

By contrast, in this paper we propose a bidirectional trajectory computation approach to make the estimated poses before the first turning as accurate as those after the first turning. In short, after the first turning, we additionally create a backward computation thread to recalculate the poses from the first turning back to the starting point. In this way, the accuracy of estimated poses before the first turning do not suffer from the lack of rotation anymore, because the backward computation makes use of the information obtained in the first turning. Also note that by means of bidirectional trajectory computation, we can obtain more accurate overall trajectory in real time, because every time one of the poses before the first turning is updated by the backward computation thread, the real-time trajectory is also adjusted accordingly. In the following, we provide the observability analysis for the extrinsic parameters in Section III and describe the proposed bidirectional trajectory computation method in Section IV.

II. PRELIMINARIES ON ODOMETER-AIDED VISUAL-INERTIAL SLAM

The proposed bidirectional trajectory computation method is based on the odometer-aided VI-SLAM approach [10], which is a tightly-coupled approach based on sliding window optimization, where IMU and wheel encoder measurements are fused at the pre-integration stage.

A. Frames and Notations

The coordinate frames of the sensors include the camera frame, the IMU frame and the odometer frame. The wheel encoder is installed on one rear wheel that always points forward. For the details of these frames the reader may refer to [10]. We use $(\cdot)^w$ to denote the world frame that is fixed since initialization, and $(\cdot)^{c_k}$, $(\cdot)^{b_k}$ and $(\cdot)^{o_k}$ to denote the camera frame, IMU frame, and odometer frame corresponding to the k^{th} image. Let $\mathbf{R}_A^B$ denote the rotation matrix that takes a vector in frame $\{A\}$ to frame $\{B\}$, and $\mathbf{q}_A^B$ is its quaternion form. $\mathbf{p}_A^B$ is the coordinate of the origin point of frame $\{A\}$ in frame $\{B\}$, and $\mathbf{v}_A^B$ is the velocity of the origin point of frame $\{A\}$ measured in frame $\{B\}$. And let $\mathbf{b}_{a_k}$ and $\mathbf{b}_{\omega_k}$ denote the accelerometer bias and gyroscope bias corresponding to image k respectively. Moreover, we use $[\cdot]_\times$ to denote the skew symmetric matrix corresponding to a vector.

B. State Estimation

The parameters to be estimated can be written as

$$\begin{aligned}\mathbf{x} &= \left[\mathbf{x}_0, \mathbf{x}_1, \dots \mathbf{x}_{K-1}, \lambda_0, \lambda_1, \dots \lambda_{m-1}, \mathbf{R}_c^b, \mathbf{p}_c^b, \mathbf{R}_o^b, \mathbf{p}_o^b\right], \\ \mathbf{x}_k &= \left[\mathbf{p}_{b_k}^w, \mathbf{v}_{b_k}^w, \mathbf{q}_{b_k}^w, \mathbf{b}_{a_k}, \mathbf{b}_{\omega_k}\right], k = 0 \dots K-1,\end{aligned} \tag{1}$$

where λ is the inverse depth of one landmark in camera frame, $\mathbf{R}_c^b$ and $\mathbf{p}_c^b$ are the camera-IMU extrinsic parameters, while $\mathbf{R}_o^b$ and $\mathbf{p}_o^b$ are the IMU-odometer extrinsic parameters. m is the number of landmarks and K is the size of sliding window.

The cost function $c(\mathbf{x})$ mainly comprises reprojection error terms, IMU-odometer error terms and the marginalization error term, which writes as

$$c(\mathbf{x}) = \sum_L \sum_{j \in \mathscr{B}_L} \mathbf{e}_{L,j}^v{}^\mathsf{T} \mathbf{W}^v \mathbf{e}_{L,j}^v + \sum_{k=0}^{K-2} \mathbf{e}_k^s{}^\mathsf{T} \mathbf{\Sigma}_{k,k+1}^{-1} \mathbf{e}_k^s + \mathbf{e}^{m\mathsf{T}} \mathbf{e}^m, \tag{2}$$

where $\mathbf{e}_{L,j}^v$ means the reprojection residual of landmark L on image j, and $\mathbf{W}^v$ is the uniform information matrix for all reprojection error terms. $\mathscr{B}_L$ is the set of images on which landmark L appears. $\mathbf{e}_k^s$ and $\mathbf{\Sigma}_{k,k+1}$ are the residual vector and covariance matrix of the IMU-odometer terms respectively, which are derived utilizing the IMU-odometer pre-integration results. $\mathbf{e}^{m\mathsf{T}}\mathbf{e}^m$ is the marginalization error term. In practice we additionally add a very small term confining the roll angle in $\mathbf{R}_o^b$, which is always an unobservable angle because the velocity of the wheel always points forward in its own coordinate frame. This term is so small that it is neglected in the following demonstrations. The nonlinear optimization is performed using Dogleg method by Ceres Solver [1]. For the details of state estimation, the reader may refer to [10].

III. OBSERVABILITY ANALYSIS

In this section, we analyze the observability of extrinsic parameters when the platform moves along a straight line with no rotation, which is often the case for a car on a straight road before its first turning. For the observability analysis we need to consider the reprojection constraints

$$\begin{aligned}&\pi_c(\mathbf{R}_b^c(\mathbf{R}_w^{b_j}(\mathbf{R}_{b_i}^w(\mathbf{R}_c^b \frac{1}{\lambda_L}\pi_c^{-1}(\begin{bmatrix}\hat{u}_{L,i}\\ \hat{v}_{L,i}\end{bmatrix}) + \mathbf{p}_c^b) + \mathbf{p}_{b_i}^w - \mathbf{p}_{b_j}^w) - \mathbf{p}_c^b)) \\ &= \begin{bmatrix}\hat{u}_{L,j}\\ \hat{v}_{L,j}\end{bmatrix}, i = 0 \dots K-1, L \in \mathscr{F}_i, j \in \mathscr{V}_l,\end{aligned} \tag{3}$$

as well as the IMU-odometer constraints

$$\mathbf{R}_w^{b_k}(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w \Delta t_k^2 - \mathbf{v}_{b_k}^w \Delta t_k) - \boldsymbol{\alpha}_{b_{k+1}}^{b_k} = \mathbf{0}, \tag{4}$$

$$\mathbf{R}_w^{b_k}(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w \Delta t_k - \mathbf{v}_{b_k}^w) - \boldsymbol{\beta}_{b_{k+1}}^{b_k} = \mathbf{0}, \tag{5}$$

$$2\left[(\boldsymbol{\gamma}_{b_{k+1}}^{b_k})^{-1} \otimes \mathbf{q}_{b_k}^{w\,-1} \otimes \mathbf{q}_{b_{k+1}}^w\right]_{vec} = \mathbf{0}, \tag{6}$$

$$\mathbf{R}_w^{b_k}(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w) - \mathbf{p}_o^b + \mathbf{R}_w^{b_k}\mathbf{R}_{b_{k+1}}^w \mathbf{p}_o^b - \boldsymbol{\eta}_{b_{k+1}}^{b_k} = \mathbf{0}, \tag{7}$$

$$\mathbf{b}_{a_{k+1}} - \mathbf{b}_{a_k} = \mathbf{0}, \tag{8}$$

$$\mathbf{b}_{\omega_{k+1}} - \mathbf{b}_{\omega_k} = \mathbf{0}, k = 0 \dots K-2, \tag{9}$$

where $[\hat{u}_{L,i}, \hat{v}_{L,i}]^\mathsf{T}$ and $[\hat{u}_{L,j}, \hat{v}_{L,j}]^\mathsf{T}$ are the observations of landmark L on image i and image j respectively. λ_L is the inverse depth of landmark L in the camera frame where its first observation happens. $\pi_c(\cdot)$ is the projection function, and $\pi_c^{-1}(\cdot)$ is the inverse function of $\pi_c(\cdot)$. K is the size of sliding window, $\mathscr{F}_i$ is the set of landmarks whose first observation happens on image i, and $\mathscr{V}_L$ is the set of images that can see landmark L but are not the first image seeing L. For more details of (3), the reader may refer to [13]. $\boldsymbol{\alpha}^{b_k}_{b_{k+1}}$, $\boldsymbol{\beta}^{b_k}_{b_{k+1}}$, $\boldsymbol{\gamma}^{b_k}_{b_{k+1}}$ and $\boldsymbol{\eta}^{b_k}_{b_{k+1}}$ are the nominal states from IMU-odometer pre-integration, after being compensated by the changes in estimated IMU biases and IMU-odometer extrinsic parameters. Among them $\boldsymbol{\gamma}^{b_k}_{b_{k+1}}$ is the rotation from the IMU frame of image $k+1$ to that of image k. $\boldsymbol{\eta}^{b_k}_{b_{k+1}}$ is the displacement between the origin of odometer frame of image k and that of image $k+1$, integrated using gyroscope and odometer measurements. $\boldsymbol{\alpha}^{b_k}_{b_{k+1}}$ and $\boldsymbol{\beta}^{b_k}_{b_{k+1}}$ do not have straightforward geometrical meanings, but they have the same dimensions as displacement and velocity respectively. The readers may look at (3) and (5) in [13] for details of $\boldsymbol{\alpha}^{b_k}_{b_{k+1}}$, $\boldsymbol{\beta}^{b_k}_{b_{k+1}}$ and $\boldsymbol{\gamma}^{b_k}_{b_{k+1}}$, as well as (1) and (12) in [10] for details of $\boldsymbol{\eta}^{b_k}_{b_{k+1}}$ respectively. Δt_k is the time interval between image k and image $k+1$. And $[\cdot]_{vec}$ denotes the vector part of a quaternion.

Among the extrinsic parameters $(\mathbf{R}^b_c, \mathbf{p}^b_c)$ and $(\mathbf{R}^b_o, \mathbf{p}^b_o)$, $\mathbf{p}^b_o$ is only involved in (7), which can be rewritten as

$$\mathbf{R}^{b_k}_w(\mathbf{p}^w_{b_{k+1}} - \mathbf{p}^w_{b_k}) + (\mathbf{R}^{b_k}_w \mathbf{R}^w_{b_{k+1}} - \mathbf{I})\mathbf{p}^b_o - \boldsymbol{\eta}^{b_k}_{b_{k+1}} = \mathbf{0}. \tag{10}$$

When the platform moves along a straight line with no rotation, $\mathbf{R}^{b_k}_w \mathbf{R}^w_{b_{k+1}} - \mathbf{I} = \mathbf{O}$, in which case $\mathbf{p}^b_o$ is not involved in any of the constraints, so it is unobservable.

Similarly, $\mathbf{R}^b_c$ and $\mathbf{p}^b_c$ are only involved in (3), which can be rewritten as

$$\begin{aligned}&\pi_c\Big(\frac{1}{\lambda_L}\mathbf{R}^c_b \mathbf{R}^{b_j}_w \mathbf{R}^w_{b_i} \mathbf{R}^b_c \pi_c^{-1}\Big(\begin{bmatrix}\hat{u}_{L,i}\\ \hat{v}_{L,i}\end{bmatrix}\Big) + \mathbf{R}^c_b(\mathbf{R}^{b_j}_w \mathbf{R}^w_{b_i} - \mathbf{I})\mathbf{p}^b_c \\ &+ \mathbf{R}^c_b \mathbf{R}^{b_j}_w (\mathbf{p}^w_{b_i} - \mathbf{p}^w_{b_j})\Big) = \begin{bmatrix}\hat{u}_{L,j}\\ \hat{v}_{L,j}\end{bmatrix}, i = 0 \dots K-1, L \in \mathscr{F}_i, j \in \mathscr{V}_l.\end{aligned} \tag{11}$$

When the platform moves along a straight line with no rotation, $\mathbf{p}^b_c$ is unobservable because $\mathbf{R}^{b_j}_w \mathbf{R}^w_{b_i} - \mathbf{I} = \mathbf{O}$. Moreover, in such a case, for every image pair (i, j), we have $\mathbf{R}^{b_j}_w(\mathbf{p}^w_{b_i} - \mathbf{p}^w_{b_j}) = s_{i,j}\mathbf{d}$, where $s_{i,j}$ is a scalar, and $\mathbf{d}$ is a unit vector denoting the driving direction, which is independent of (i, j). In such a case, (11) can be rewritten as

$$\pi_c\Big(\frac{1}{\lambda_L}\pi_c^{-1}\Big(\begin{bmatrix}\hat{u}_{L,i}\\ \hat{v}_{L,i}\end{bmatrix}\Big) + s_{i,j}\mathbf{R}^c_b \mathbf{d}\Big) = \begin{bmatrix}\hat{u}_{L,j}\\ \hat{v}_{L,j}\end{bmatrix}, i = 0 \dots K-1, L \in \mathscr{F}_i, j \in \mathscr{V}_l, \tag{12}$$

From (12) it is clear that the component in rotation $\mathbf{R}^b_c$ corresponding to the rotation around the driving direction $\mathbf{d}$, which is also called the roll angle in the following, is unobservable as well.

In practice, among the seven unobservable directions (three in $\mathbf{p}^b_o$, three in $\mathbf{p}^b_c$ and one in $\mathbf{R}^b_c$) when the platform moves along a straight line with no rotation, the roll angle in $\mathbf{R}^b_c$ is the direction whose observability is most relevant to whether the platform has made a turn. We firstly perform eigendecomposition of the Hessian matrix, and then compute the ratio of the eigenvalue corresponding to the eigenvector which is most approximate to direction of the perturbation of the roll angle in $\mathbf{R}^b_c$, to the largest eigenvalue. For the experiment illustrated in Figure 1, the accelerometer bias is held constant and the extrinsic parameters start to be adjusted since the beginning. It is clear from Figure 1 that after the first turning the error in roll angle of $\mathbf{R}^b_c$ dramatically decreases and that the eigenvalue ratio dramatically increases, both of which indicate that the roll angle in $\mathbf{R}^b_c$ can be estimated much better after the first turning. Compared with [23], our observability analysis computes the eigenvalue ratio inside sliding window that keeps sliding, rather than fisher information matrix inside previously divided segments, thereby illustrating the relationship between turning and the observability of extrinsic parameters more clearly. Moreover, our analysis reveals that the extrinsic parameter is still observable after the turning given that marginalization is applied, because the eigenvalue ratio does not dramatically decrease after the turning.

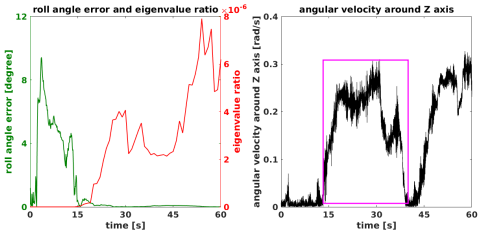


Fig. 1. Estimation error in roll angle of $\mathbf{R}^b_c$, the eigenvalue ratio and the angular velocity around the Z axis (the vertical axis) at the first turning in sequence urban34. The pinkish box indicates the first turning. The system starts at a dozen of seconds before the first turning.

Practically, when the motion of the platform is approximately but not exactly along a straight line, the errors in the unobservable directions of the extrinsic parameters do affect the accuracy of the trajectory. Table I shows the comparison of absolute trajectory error (ATE) for 4 sequences in KAIST Urban Data Set [5], either using the accurate extrinsic parameters calibrated offline or using the extrinsic parameters with added fixed error (5 degrees in the roll angle of $\mathbf{R}^b_c$). More details of observability analysis are in our arXiv technical report.

TABLE I

ATE (IN METERS) USING DIFFERENT EXTRINSIC PARAMETERS

Sequence (urban-)	22 (3.4km)	23 (3.4km)	24 (4.2km)	25 (2.5km)
EPs accurate	**8.8**	**11.1**	**15.0**	**8.0**
EPs with error	14.2	13.5	16.2	9.7

Here *EPs accurate* means using the accurate extrinsic parameters calibrated offline, and *EPs with error* means using the extrinsic parameters with added fixed error. *ATE* means absolute trajectory error, and *EPs* mean extrinsic parameters. In each of the four sequences the car moves along an approximately straight road, and the trajectories are computed using the state estimation method illustrated in Section II-B, with accelerometer bias and extrinsic parameters both held constant.

IV. METHOD

Taking into consideration that for the odometer-aided VI-SLAM system described in Section II-B, the system is not stable and the extrinsic parameters can not be correctly estimated in the beginning, and that the accelerometer bias is unobservable until the platform makes a turn, we propose a robust method to acquire accurate real-time trajectory.

A. Forward Computation Thread and Backward Computation Thread

In the very beginning, we propagate the poses and try to initialize our system. After the system is initialized as described in [10], the state estimation in sliding window is performed as in Section II-B in the main thread, which we call the forward computation thread. In the forward computation thread, before the first turning, the extrinsic parameters are held constant, and the accelerometer bias is set to zero and held constant, in order to make the system robust in the beginning. At this stage, we limit the magnitude of the marginalization term as described in Section IV-D. Once the platform has made a turn, the accelerometer bias starts to be adjusted, and as soon as the estimation of accelerometer bias reaches convergence according to the criterion adopted in [10], the extrinsic parameters starts to be adjusted. The detection criteria for a turning will be described and explained in Sect.IV-B. Thanks to the fact that the marginalization term contains historical information, especially the information gathered during the first turning, the accelerometer bias and extrinsic parameters that are engaged in state estimation will soon reach their desired values. After a time interval T_2 (30 seconds in our experiments) since the extrinsic parameters begin to be adjusted, we create a new thread named backward computation thread, meanwhile the forward computation thread keeps running. Both the two threads are independent with each other. Figure 2 is the schematic diagram illustrating forward computation and backward computation.

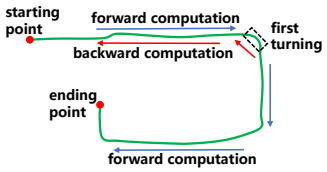
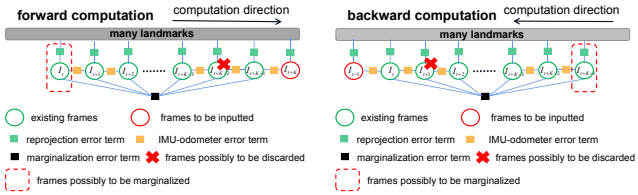


Fig. 2. Schematic diagram about forward computation and backward computation. Forward computation starts from the beginning. After the first turning backward computation starts. Forward computation continues to operate until the end. Backward computation proceeds to the starting point to work out more accurate poses.

The backward computation thread also performs state estimation in a sliding window where the parameters are as (1) and the cost function writes as (2). When creating the backward computation thread, the values of the parameters in the backward computation thread except for landmark inverse depths are copied from the forward computation thread, and the IMU-odometer terms and the marginalization term in the backward computation thread are identical to their counterparts in the forward computation thread. For the backward computation thread, the reprojection errors still take the form of (3), while the *first observation* in (3) means the observation happening on the image with the latest timestamp, instead of the one with the earliest timestamp as in the forward computation thread. Meanwhile, the inverse depth of each landmark L is shifted as

$$\lambda_L = 1/(\mathbf{e}_3^{\mathsf{T}}\mathbf{R}_b^c(\mathbf{R}_w^{b_q}(\mathbf{R}_{b_p}^w(\mathbf{R}_c^b\frac{1}{\lambda_L'}\pi_c^{-1}(\begin{bmatrix}\hat{u}_{L,p}\\ \hat{v}_{L,p}\end{bmatrix})+\mathbf{p}_c^b)+\mathbf{p}_{b_p}^w-\mathbf{p}_{b_q}^w)-\mathbf{p}_c^b)), \tag{13}$$

where λ_L and λ_L' are the inverse depths of landmark L after and before shifting respectively, $\mathbf{e}_3^{\mathsf{T}} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}$, p and q are indexes of the earliest and latest image which can see the landmark L in the sliding window respectively, and the meanings of the other symbols are the same as those in (3).



(a) sliding window in the forward computation thread (b) sliding window in the backward computation thread

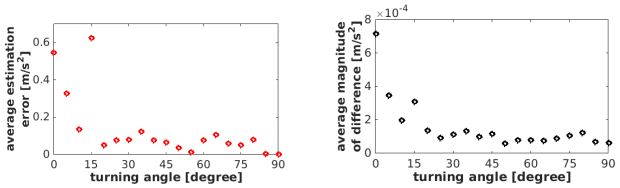
Fig. 3. Contrast between the sliding windows in forward and backward computation threads. In either of the two threads, when a new frame is inputted, the frame marked with red cross is discarded if it is not a keyframe. Otherwise, the frame in the dashed red box is marginalized.

Parameters can be estimated correctly in the backward computation thread since the backward computation thread starts, because of the information contained in the marginalization error term. Figure 3(a)-3(b) show the contrast between the sliding windows in forward and backward computation threads. In the following we illustrate how the backward computation thread operates according to Figure 3(b). Suppose that the first frame in data sequence is I_0, and that at a certain moment t, there are K frames in the sliding window, namely $I_t, I_{t+1} \ldots I_{t+K-1}$. In the backward computation thread, every time the nonlinear optimization in the sliding window finishes at the certain moment t, the next frame to be inputted is I_{t-1}, which is the one previous to the frame I_t whose timestamp is the earliest in the sliding window. The IMU-odometer pre-integration between the above two frames (I_{t-1} and I_t) is computed, and the initial value of the pose and velocity of the frame to be newly inputted (I_{t-1}) is propagated using IMU measurements between the two frames. Note that although the IMU-odometer pre-integration between the above two frames has been performed in the past in the forward computation thread, recomputation is needed because the estimated value of IMU biases and the extrinsic parameter $\mathbf{R}_o^b$ have changed, and they are engaged in pre-integration. Next, if the frame with the second earliest timestamp (I_{t+1}) is not a keyframe, it is discarded. Otherwise, the frame with the latest timestamp (I_{t+K-1}) is marginalized. The criterion to judge whether an

image frame is a keyframe is the same as that in [13]. The backward computation terminates when the first frame in data sequence I_0 has been inputted into the sliding window. The IMU and wheel encoder measurements and the feature points used in backward computation are recorded previously during the forward computation. When the pose of a certain frame is estimated in the backward computation thread, it is used to substitute the corresponding pose previously estimated in the forward computation thread, because the poses estimated in the backward computation thread are more accurate.

B. Turning Detection

Since the local optimizations are always performed inside a sliding window, for turning detection we consider the turning angle inside a sliding window. We determine how large a turning angle should reach to result in good estimation of accelerometer bias, because the estimation of accelerometer bias is very relevant to the turning of the platform, which is evident from Figure 6 and [10]. We simulate 19 data sequences, each of which contains a single turning that can be contained inside a sliding window, and the turnings vary every 5 degrees from 0 to 90 degrees. We run the odometer-aided visual-inertial SLAM with only forward computation on each sequence, and the average estimation error in accelerometer biases after turning, as well as the average magnitude of difference between every two successively estimated accelerometer biases after turning are shown in Figure 4(a)-4(b).

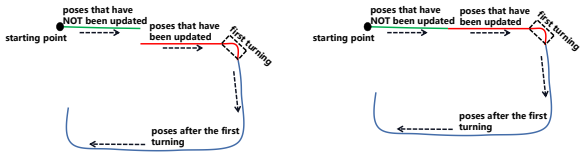


(a) (b)

Fig. 4. (a) average estimation error in accelerometer biases w.r.t. ground-truth accelerometer biases after turning and (b) average magnitude of difference between every two successively estimated accelerometer biases after turning. Each dot in the figures represents the average value in a data sequence that contains a certain turning angle. The difference between every two successively estimated accelerometer biases is also expected to be small, because the accelerometer bias is a slow time-varying quantity.

From Figure 4(a)-4(b) we can see that when the turning angle exceeds 20 degrees, both the average estimation error in accelerometer biases after turning and the average magnitude of difference between every two successively estimated accelerometer biases after turning do not further decrease as the turning angle increases, indicating that the estimation of accelerometer bias is good enough under such circumstances. Therefore, we consider the platform has made a turn if and only if it has turned larger than 20 degrees within a sliding window. We do not use eigenvalue ratio to detect turning, as eigendecomposition is time-consuming. On the data sequence urban25, eigendecomposition takes 10.5ms on average after each optimization, and updating the turning angle takes 0.0007ms on average between two consecutive optimizations.

C. Computation of Real-time Trajectory



(a) Trajectory comprising poses $(\mathbf{R}^{lw}_{c_k}, \mathbf{p}^{lw}_{c_k})$ (b) Real-time trajectory comprising poses $(\mathbf{R}'^{w}_{c_k}, \mathbf{p}'^{w}_{c_k})$

Fig. 5. Schematic diagram of real-time trajectory computation. The black dotted arrows represent the moving direction of the platform.

Although the backward computation directly updates the past poses, every time the backward computation thread updates the pose of a certain frame, the current pose is also adjusted accordingly. As can be seen from Figure 5, The real-time trajectory is computed by keeping the starting point unchanged and keeping the trajectory continuous. For a certain image frame k, its pose $(\mathbf{R}^{lw}_{c_k}, \mathbf{p}^{lw}_{c_k})$ is first computed by forward computation thread, and if it is recomputed in the backward computation thread, its pose $(\mathbf{R}'^{w}_{c_k}, \mathbf{p}'^{w}_{c_k})$ is updated by the backward computation thread, and in this case the pose previously computed by the forward computation thread is denoted as $(\hat{\mathbf{R}}^{w}_{c_k}, \hat{\mathbf{p}}^{w}_{c_k})$. Let j denote the frame that has just been updated by the backward computation thread. For the frames before frame j, i.e. $i < j$, the real-time pose $(\mathbf{R}''^{w}_{c_i}, \mathbf{p}''^{w}_{c_i})$ equals $(\mathbf{R}'^{w}_{c_i}, \mathbf{p}'^{w}_{c_i})$, that is to say they are unchanged. For the frames after frame j, i.e. $i \geq j$, the real-time pose $(\mathbf{R}''^{w}_{c_i}, \mathbf{p}''^{w}_{c_i})$ is computed as

$$\begin{aligned} \mathbf{R}''^{w}_{c_i} &= \hat{\mathbf{R}}^{w}_{c_j} \mathbf{R}'^{c_j}_{w} \mathbf{R}'^{w}_{c_i}, \\ \mathbf{p}''^{w}_{c_i} &= \hat{\mathbf{R}}^{w}_{c_j} \mathbf{R}'^{c_j}_{w} (\mathbf{p}'^{w}_{c_i} - \mathbf{p}'^{w}_{c_j}) + \hat{\mathbf{p}}^{w}_{c_j}, \text{ for } i \geq j. \end{aligned} \tag{14}$$

D. Bounded Marginalization Term

The marginalization residual takes the form of $\mathbf{e}^m = \mathbf{r}^m - \mathbf{J}^m \delta\mathbf{x}$. $\mathbf{r}^m$ and $\mathbf{J}^m$ are computed in the marginalization process, and $\delta\mathbf{x}$ is the step to update the parameters $\mathbf{x}$. We have observed the phenomenon that the marginalization error keeps growing and thus the total error keeps growing before the first turning, when the accelerometer bias and extrinsic parameters are held constant. In order to reduce the accumulation of the error caused by inaccurate extrinsic parameters and accelerometer bias in the marginalization error term and prevent the above error from dominating the state estimation, before the first turning we multiply $\mathbf{r}^m$ and $\mathbf{J}^m$ by a ratio μ once the ratio of marginalization error $\mathbf{e}^{m\top}\mathbf{e}^m$ to total error $c(\mathbf{x})$ in (2) after optimization rises beyond a threshold r. In our experiment μ is set to 0.85 and r is set to 0.4.

V. EXPERIMENTS

We evaluate the effect of the bidirectional trajectory computation method proposed in this paper on KAIST Urban Data Set [5], which is a publicly available dataset containing

data in complex urban scenes collected on a rear wheel drive passenger car. The sensors in the dataset include stereo cameras, one IMU and two wheel encoders mounted on two rear wheels. The frequencies of the captured images, IMU measurements and wheel encoder measurements are 10Hz, 100Hz and 100Hz respectively. The proposed approach is compared with the stereo inertial version of the state-of-the-art VI-SLAM system VINS-Fusion [14], the standard VIO [9], and the odometer-aided VI-SLAM approaches [19], [20] and [10]. The proposed approach and [10] use a monocular camera, one IMU and one wheel encoder. The approaches [19] and [20], as reported in their papers, use a monocular camera, one IMU and two wheel encoders. The stereo inertial version of VINS-Fusion uses stereo cameras and one IMU. To our knowledge, there are no other approaches intentionally dealing with the unobservability after initialization and before the first turning. All the experiments presented are performed on a PC with Intel Core i7 3.6GHz $\times$ 6 core CPU and 64GB memory. The extrinsic parameters provided in the dataset are adopted as initial values, which may not be very accurate.

A. Average Positioning Error by Aligning the Starting Frame

The primary concern of our bidirectional trajectory computation method is to improve the accuracy at initial stage, which matters a lot supposing we only know the position and orientation of the vehicle at the starting point. In our first evaluation, we align the position and orientation of the starting image frames for the resulting trajectory from VI-SLAM approaches and the ground truth trajectory, and compute the average positioning error of every frame in the data sequence. The practice of aligning the starting frames is also adopted in the evaluation criteria on KITTI dataset [3]. Here our proposed approach is mainly compared with [10], which our approach is based on. The work [10] starts to optimize accelerometer bias from the beginning, and fix the extrinsic parameters until the platform has made a turn and the estimation of accelerometer bias has reached convergence, in order to reduce the instability in the very beginning. In order to make an exhaustive comparison on different strategies dealing with accelerometer bias and extrinsic parameters that are two instability factors, we derive some adapted versions from [10], that are: (i) both accelerometer bias and extrinsic parameters are fixed until the first turning (FAFE), (ii) the extrinsic parameters starts to be optimized from the beginning, and accelerometer bias is fixed until the first turning (FAOE), (iii) both accelerometer bias and extrinsic parameters starts to be optimized from the beginning (OAOE). Our proposed approach is firstly compared against [10] (OAFE) and its three adapted versions in the above, as well as VINS-Fusion [14]. To make a fair comparison, we select the image frame when the vehicle has traveled 100 meters as the starting frame, to avoid being affected by some erroneous pose estimations from some approaches in the very beginning. This comparison is made on all the 15 sequences with stereo cameras and with complexity level 3 (middle) or level 4 (high) in [5], namely urban25-urban39.

TABLE II
COMPARISON OF AVERAGE POSITIONING ERROR (IN METERS) BY ALIGNING THE STARTING FRAME

Sequence (urban-)	Proposed	FAFE	FAOE	OAOE	OAFE [10]	VINS-Fusion [14]
*25 (2.5km)	11.6	**11.3**	72.7	62.1	15.3	862.7
26 (4.0km)	**20.8**	41.0	42.8	29.8	41.7	52.9
27 (5.4km)	**19.0**	49.5	73.4	91.9	44.5	63.1
28 (11.5km)	32.6	61.5	47.7	**27.7**	104.8	103.4
29 (3.6km)	51.3	44.3	**12.1**	13.3	40.1	122.6
30 (6.0km)	**24.8**	34.6	36.8	45.5	43.0	×
31 (11.4km)	**389.9**	1108	995.9	1072	1230	1739
32 (7.1km)	**66.9**	407.5	140.6	149.1	422.6	257.9
33 (7.6km)	**31.8**	177.3	81.9	130.9	221.3	696.7
34 (7.8km)	118.9	**98.8**	160.9	122.6	168.7	×
*35 (3.2km)	61.1	**49.3**	290.9	253.4	57.5	×
36 (9.0km)	**218.0**	333.1	221.5	281.0	283.7	×
*37 (11.8km)	462.9	**371.0**	1989	2221	677.0	1126
38 (11.4km)	**27.6**	123.4	151.9	44.8	101.3	134.6
39 (11.0km)	**13.5**	953.7	22.0	36.8	42.1	×

Here *Proposed* means the proposed approach in this paper and '×' means failure. The *Sequence* column firstly contains the sequence number which is *urban*XX, followed by the trajectory length in the parentheses. The sequences marked with '*' do not contain turnings, so the difference in accuracies on those sequences between the proposed approach and *FAFE* is only resulted by restricting the marginalization error as described in Section IV. *FAFE* refers to fixing accelerometer bias and fixing extrinsic parameters. *FAOE* refers to fixing accelerometer bias and optimizing extrinsic parameters; *OAOE* refers to optimizing accelerometer bias and optimizing extrinsic parameters. *OAFE* refers to optimizing accelerometer bias and fixing extrinsic parameters.

The comparison of average positioning error by aligning the starting frame is shown in Table II.

Table II indicates that the proposed approach outperforms all the other five approaches on 9 out of the 15 sequences. Among the rest six sequences, urban25, urban35 and urban37 do not contain turnings, as a result the proposed bidirectional trajectory computation does not come in handy on these sequences in our proposed approach. The accuracy of the proposed approach on the above three sequences is generally higher than FAOE, OAOE and the stereo VI-SLAM [14], and comparable with OAFE [10], but slighterly lower than FAFE. That is because the manipulation described in Section IV-D can cause information loss, given that when the trajectory does not contain turnings, the only difference between the proposed approach and FAFE lies in the utilization of the manipulation in Section IV-D. However, in view of the good performance of the proposed approach on the other sequences with turnings where the bidirectional trajectory computation comes in handy, the benefit of the manipulation in Section IV-D dramatically outweighs the cost. Generally speaking, the accuracy of the proposed approach is higher than [10], as well as its adapted versions that deal with the accelerometer bias and extrinsic parameters differently.

B. Absolute Trajectory Error (ATE) Comparison

We also make an extensive comparison with more approaches, including the odometer-aided VI-SLAM approaches [19], [20] and [10], the stereo VI-SLAM system

VINS-Fusion [14], the standard VIO [9], and the batch and incremental stereo VI-SLAM approach ICE-BA [26]. The comparison is made in terms of absolute trajectory error (ATE), which is the rooted mean square error (RMSE) of the positions after a 6-DoF trajectory alignment with the ground truth. The experiments are conducted on the sequences urban26, urban28, urban38 and urban39, because only the ATEs of these four sequences are reported in the paper [20]. The comparison results are shown in Table III.

Table III indicates that the proposed approach is more accurate than other approaches in terms of ATE on 3 out of the 4 sequences, and on the rest one sequence urban26, the accuracy of the proposed approach is second only to [10] by a small difference.

C. Evaluation of Effects on Estimating Accelerometer Bias and Extrinsic Parameter

We examine the effects on estimating accelerometer bias and extrinsic parameter using the proposed bidirectional trajectory computation approach, in order to reveal why this approach improves the accuracy. The proposed approach is compared with the approach OAOE in Section V-A, which optimizes accelerometer bias and extrinsic parameters from the beginning and only performs forward computation. Figure 6 shows the comparison on estimated values of accelerometer bias and the estimation error in roll angle of $\mathbf{R}_c^b$ between the above two approaches, at the first turning in each sequence of urban27 and urban28. As same as in Section III, here the system also starts at a dozen seconds before the first turning in each sequence instead of starting from the very beginning. Figure 6 indicates that the estimation error in roll angle of $\mathbf{R}_c^b$ is much smaller using the proposed approach, and that the estimated value of accelerometer bias is more stable over time, which is more reasonable because the accelerometer bias is a slow time-varying quantity. Besides, both approaches can estimate the z-component of the accelerometer bias well. That is because both of the two unobservable directions of accelerometer bias before the first turning, which compose the orthogonal basis spanning over the 2D column space of $\mathbf{R}_w^{b_0}[\mathbf{g}^w]_\times$, have very small values in their respective z-components, in consideration of the fact that $\mathbf{R}_w^{b_0}$ is a rotation mainly around the Z axis for a ground vehicle and the gravity direction $\mathbf{g}^w$ is exactly along the Z axis. The ground-truth value of $\mathbf{R}_c^b$ is obtained offline, and the method to compute error in the roll angle is the same as that in Section III.

D. Computation of Real-time Trajectory

To illustrate the effect of computing the real-time trajectory, we take the sequence urban33 for example. Figure 7(a)-7(d) displays the real-time trajectories after 0, 160, 320 and 480 seconds since backward computation starts respectively, compared with the ground truth trajectory. We can see that as backward computation proceeds, the real-time trajectory becomes closer and closer to the ground truth trajectory gradually.

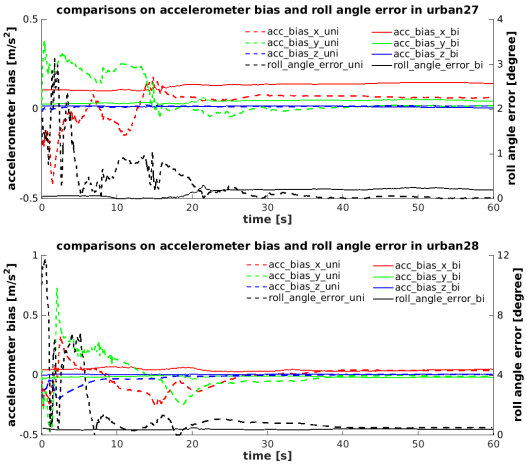


Fig. 6. Comparison on estimated accelerometer bias and the estimation error in roll angle of $\mathbf{R}_c^b$. *acc_bias_x_uni*, *acc_bias_y_uni*, *acc_bias_z_uni* and *roll_angle_error_uni* are respectively the three components of accelerometer bias and the roll angle error estimated by the unidirectional computation method *OAOE*. *acc_bias_x_bi*, *acc_bias_y_bi*, *acc_bias_z_bi* and *roll_angle_error_bi* are the corresponding quantities estimated by our proposed bidirectional trajectory computation method. Here *acc* refers to accelerometer, and *uni* and *bi* mean unidirectional and bidirectional trajectory computation respectively.

TABLE III

COMPARISON OF ATE (IN METERS) AMONG DIFFERENT APPROACHES

Sequence (urban-)	proposed	[10]	[20]	[19]	VINS-Fusion [14]	[9]	ICE-BA [26]
26 (4.0km)	12.0	**11.9**	14.8	16.1	22.5	32.8	22.1
28 (11.5km)	**15.4**	27.8	25.0	33.1	93.3	34.7	×
38 (11.4km)	**11.8**	16.0	33.5	43.0	90.0	55.5	×
39 (11.0km)	**7.5**	8.0	21.3	24.0	×	33.4	×

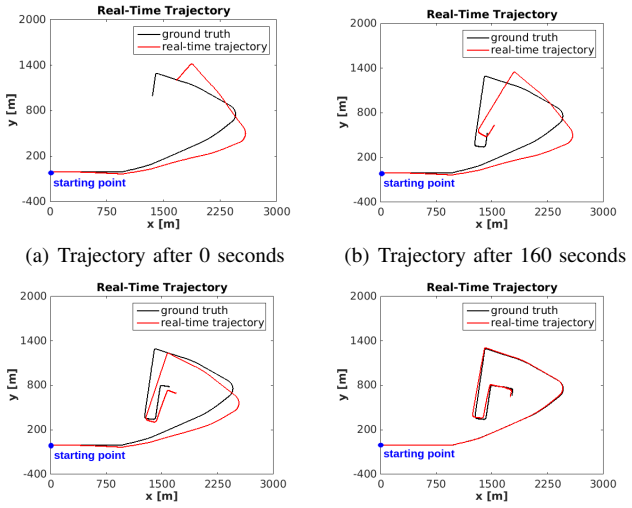
Here *Proposed* means the proposed approach in this paper and '×' means failure. *Sequence* column firstly contains the sequence number which is *urbanXX*, followed by the trajectory length in the parentheses. Results for [20], [19] and [9] are obtained from the reported results in [20]. *ATE* refers to absolute trajectory error.

E. Comparison of Efficiency Against Batch and Incremental Solver

We compare the efficiency of our proposed approach against the batch and incremental solver ICE-BA [26] by comparing the total consumption of time of computing the whole trajectory for each sequence, for which the results are shown in Table IV. From Table IV we can see that our proposed approach consumes far less time than ICE-BA [26]. From the above experiments we can draw that our proposed approach outperforms the incremental solver ICE-BA [26] in both accuracy and efficiency in KAIST Urban Data Set [5].

VI. CONCLUSION

In this paper, we propose a bidirectional trajectory computation method for VI-SLAM aided with wheel encoder. Firstly, we perform an observability analysis on the degenerate case that an odometer-aided VI-SLAM system deployed on a car possibly encounters before the first turning. Secondly, we describe our proposed backward computation thread which refines the poses before the first turning, as well



(a) Trajectory after 0 seconds (b) Trajectory after 160 seconds

(c) Trajectory after 320 seconds (d) Trajectory after 480 seconds

Fig. 7. Real-time trajectories in urban33 at different moments.

TABLE IV

COMPARISON OF TIME CONSUMPTION (IN SECONDS)

Sequence	urban25	urban26	urban27	urban28	urban29
Proposed	107.4	**566.8**	**1133.5**	**2102.1**	**434.1**
ICE-BA [26]	**103.3**	967.9	3339.8	6940.5	1612.9
Sequence	urban30	urban31	urban32	urban33	urban34
Proposed	**1274.2**	**1113.6**	**1090.7**	**1335.9**	**659.2**
ICE-BA [26]	2093.3	3980.5	4789.7	6250.3	2352.1
Sequence	urban35	urban36	urban37	urban38	urban39
Proposed	**184.4**	**375.2**	**543.5**	**2159.8**	**1858.5**
ICE-BA [26]	572.4	905.8	1398.4	13349.6	7220.3

Here *Proposed* means the proposed approach in this paper.

as the method to adjust the real-time trajectory. Experimental results show the higher accuracy of the whole trajectory, the correctly estimated parameters before the first turning, and the effects of real-time trajectory adjustment. Although in this paper wheel encoder is used, we also believe that the proposed bidirectional trajectory computation method can be applied on VI-SLAM systems that are not aided with wheel encoders as well.

REFERENCES

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. `http://ceres-solver.org`, 2017.

[2] Z. Dang, T. Wang, and F. Pang. Tightly-coupled data fusion of vins and odometer based on wheel slip estimation. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1613–1619. IEEE, 2018.

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[4] G. Huang. Visual-inertial navigation: A concise review. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9572–9582. IEEE, May 2019.

[5] J. Jeong, Y. Cho, Y. Shin, H. Roh, and A. Kim. Complex urban dataset with multi-level sensors from highly diverse urban environments. *The International Journal of Robotics Research*, 38(6):642–657, 2019.

[6] R. Kang, L. Xiong, M. Xu, J. Zhao, and P. Zhang. Vins-vehicle: A tightly-coupled vehicle dynamics extension to visual-inertial state estimator. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3593–3600. IEEE, 2019.

[7] 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*, 34(3):314–334, 2015.

[8] D. Li, K. Eckenhoff, K. Wu, Y. Wang, R. Xiong, and G. Huang. Gyro-aided camera-odometer online calibration and localization. In *2017 American Control Conference (ACC)*, pages 3579–3586. IEEE, 2017.

[9] M. Li and A. I. Mourikis. Optimization-based estimator design for vision-aided inertial navigation. In *2013 Robotics: Science and Systems*, pages 241–248, 2013.

[10] J. Liu, W. Gao, and Z. Hu. Visual-inertial odometry tightly coupled with wheel encoder adopting robust initialization and online extrinsic calibration. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5391–5397. IEEE, 2019.

[11] F. Ma, J. Shi, Y. Yang, J. Li, and K. Dai. Ack-msckf: Tightly-coupled ackermann multi-state constraint kalman filter for autonomous vehicle localization. *Sensors*, 19(21):4816, 2019.

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

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

[14] T. Qin, J. Pan, S. Cao, and S. Shen. A general optimization-based framework for local odometry estimation with multiple sensors. *arXiv preprint arXiv:1901.03638*, 2019.

[15] M. Quan, S. Piao, M. Tan, and S. Huang. Tightly-coupled monocular visual-odometric slam using wheels and a mems gyroscope. *IEEE Access*, 7:97374–97389, 2019.

[16] K. J. Wu, C. X. Guo, G. Georgiou, and S. I. Roumeliotis. Vins on wheels. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5155–5162. IEEE, 2017.

[17] Y. Yang, P. Geneva, K. Eckenhoff, and G. Huang. Degenerate motion analysis for aided ins with online spatial and temporal sensor calibration. *IEEE Robotics and Automation Letters*, 4(2):2070–2077, 2019.

[18] W. Ye, R. Zheng, F. Zhang, Z. Ouyang, and Y. Liu. Robust and efficient vehicles motion estimation with low-cost multi-camera and odometer-gyroscope. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4490–4496. IEEE, 2019.

[19] M. Zhang, Y. Chen, and M. Li. Vision-aided localization for ground robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2455–2461. IEEE, 2019.

[20] M. Zhang, X. Zuo, Y. Chen, and M. Li. Localization for ground robots: On manifold representation, integration, re-parameterization, and optimization. *arXiv preprint arXiv:1909.03423v2*, 2019.

[21] X. Zuo, M. Zhang, Y. Chen, Y. Liu, G. Huang, and M. Li. Visual-inertial localization for skid-steering robots with kinematic constraints. *arXiv preprint arXiv:1911.05787*, 2019.

[22] W. Lee, K. Eckenhoff, Y. Yang, P. Geneva, and G. Huang. Visual-inertial-wheel odometry with online calibration. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4559–4566, 2020.

[23] T. Schneider, M. Li, C. Cadena, J. Nieto, and R. Siegwart. Observability-aware self-calibration of visual and inertial sensors for ego-motion estimation. *IEEE Sensors Journal*, 19(10):3846–3860, 2019.

[24] C. Campos, J. M. M. Montiel, and J. D. Tardós. Fast and robust initialization for visual-inertial slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1288–1294, 2019.

[25] C. Campos, J. M. M. Montiel, and J. D. Tardós. Inertial-only optimization for visual-inertial initialization. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 51–57, 2020.

[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 *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1974–1982, 2018.