Ctrl-VIO: Continuous-Time Visual-Inertial Odometry for Rolling Shutter Cameras

Xiaolei Lang, Jiajun Lv, Jianxin Huang, Yukai Ma, Yong Liu, and Xingxing Zuo

Abstract—**In this letter, we propose a probabilistic continuous-time visual-inertial odometry (VIO) for rolling shutter cameras. The continuous-time trajectory formulation naturally facilitates the fusion of asynchronized high-frequency IMU data and motion-distorted rolling shutter images. To prevent intractable computation load, the proposed VIO is sliding-window and keyframe-based. We propose to probabilistically marginalize the control points to keep the constant number of keyframes in the sliding window. Furthermore, the line exposure time difference (line delay) of the rolling shutter camera can be online calibrated in our continuous-time VIO. To extensively examine the performance of our continuous-time VIO, experiments are conducted on publicly-available WHU-RSVI, TUM-RSVI, and SenseTime-RSVI rolling shutter datasets. The results demonstrate the proposed continuous-time VIO significantly outperforms the existing state-of-the-art VIO methods.**

Index Terms—**Localization, sensor fusion, visual-inertial SLAM.**

I. Introduction

A WIDE range of sensors can be applied for accurate 6-DoF motion estimation, among which camera has become a good choice due to its low cost, light weight and intuitive perception of the appearance information. While visual odometry (VO) is able to estimate the up-to-scale camera poses, it is prone to failure when facing challenges from deficient texture, light variations and violent motion, etc. By additionally fusing Inertial Measurement Unit (IMU) data, visual-inertial odometry (VIO) can estimate camera poses with absolute scale and becomes more robust against the aforementioned challenges compared to VO. The superiority of VIO has been shown in many existing works [1], [2], [3], [4], [5].

Most cameras can be divided into global shutter (GS) and rolling shutter (RS). Compared to GS cameras, the RS cameras are usually at a lower cost, thus they have been widely applied in consumer-grade electronics such as smartphones. However, unlike the GS cameras exposing all pixels simultaneously, the pixels of the RS cameras are exposed row by row and the timestamps of two consecutive image rows differ by a constant value termed as line delay [6]. Hence, significant motion distortion can be introduced in the imaging process of the RS cameras, which is named the RS effect. Some existing VIO methods just ignore the RS effect, which can severely hurt the accuracy and robustness of the estimator, especially in fast motion scenarios.

In fact, a GS image corresponds to only one camera pose, while every row of a RS image corresponds to one camera pose, inevitably leading to a sharp increase in the dimension of the states to be estimated. Therefore, it is computationally intractable to merely estimate the poses of different rows of RS images. A common way to cope with this problem is to introduce a constant velocity model, assuming the camera moves at a constant speed between two keyframes [7], [8], [9]. Another way is to parameterize the continuous-time trajectory by B-splines [10], [11], [12], [13], which is a more elegant way compared to the previous method. The constant velocity assumption does not hold in the case of large accelerations, while the continuous-time splines can instead parameterize trajectory with motions at high complexity.

To deal with the RS effect, line delay needs to be known for calculating the exact timestamp of each image row. However, in some electronic devices, line delay can not be accessed by consumers and may change with exposure configurations of the camera. To calibrate the line delay of a RS camera, there are hardware-based methods [14] and calibration-pattern-based methods [6], [15], while online self-calibration methods [16], [17] without using specific calibration patterns might be more convenient.

Inspired by the work above, we also utilize B-splines to parameterize the camera trajectory for elegantly handling the RS effect. In previous works, such continuous-time representation has been employed for RS visual odometry [10], [11], [12], [13] and offline calibration between RS cameras and IMU [6], [10], [11], [15], none of which are complete implementation of visual-inertial odometry for RS cameras, while this letter proposes continuous-time visual-inertial odometry termed as Ctrl-VIO for RS cameras with line delay online calibration, which demonstrates high accuracy even under severe RS effects. To the best of our knowledge, this is the first continuous-time visual-inertial odometry for RS cameras with line delay online calibration. Briefly, our contributions are as follows:

1) We propose a keyframe-based sliding-window visual-inertial odometry (Ctrl-VIO) for RS cameras, using continuous-time trajectory parameterized by B-splines to handle the RS effect. A dedicated marginalization strategy is proposed for the continuous-time sliding-window estimator.

Manuscript received 5 April 2022; accepted 14 August 2022. Date of publication 29 August 2022; date of current version 7 September 2022. This letter was recommended for publication by Associate Editor M. Kaess and Editor S. Behnke upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grant 6208810. *(Xiaolei Lang and Jiajun Lv contributed equally to this work.) (Corresponding authors: Yong Liu; Xingxing Zuo.)*

Xiaolei Lang, Jiajun Lv, Jianxin Huang, Yukai Ma, and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310000, China (e-mail: jerry_locker@zju.edu.cn; lvjiajun314@zju.edu.cn; 22032097@zju.edu.cn; 12132060@zju.edu.cn; yongliu@iipc.zju.edu.cn).

Xingxing Zuo is with the Department of Informatics, Technical University of Munich, 85748 Garching bei München, Germany (e-mail: xingxing.zuo@tum.de).

The codebase of this letter will also be open-sourced at https://github.com/APRIL-ZJU/Ctrl-VIO

Digital Object Identifier 10.1109/LRA.2022.3202349

2) Online calibration for line delay is naturally supported in our system, making the best of the fact that B-splines provide closed-form analytical derivatives w.r.t temporal variables [18], [19], [20].
3) We test our method on both synthetic and real-world data, demonstrating that our system performs much better with high accuracy and robustness on RS data than other state-of-the-art RS methods and GS methods which ignore the RS effect. Also, online-calibrated line delay can converge quickly and accurately.

This introduction section will be followed by the related works. Afterward, the methodology of the proposed Ctrl-VIO is elaborated in Section III. Experimental results are demonstrated and discussed in Section IV. Finally, we conclude the letter and look into the future in the last section.

II. Related Work

A. Rolling Shutter VSLAM/VISLAM

Klein et al. [7] are the first to deal with the RS effect in VSLAM. Aiming at the defects of the iPhone imaging device such as the RS effect, they make a series of modifications to PTAM [21] to run the algorithm on the iPhone. They first correct the observations on RS images based on the constant velocity assumption and then feed these corrected observations into bundle adjustment. David et al. extend DSO [22] to DSORS [8], additionally estimating the velocity at each keyframe and imposing a constant velocity prior to the photometric optimization, which prevents the ambiguity of RS images from crashing the system. The system outperforms state-of-the-art GS-method VO on challenging RS sequences. Combined with [3], DSORS is further extended to [9] by fusing inertial measurements, increasing the accuracy and stability of VO and also rendering the scale observable.

Based on the constant velocity model, which is an approximate and low-dimensional representation of the camera trajectory, the RS effect can be partially addressed. However, such representation can lead to a loss of accuracy when the camera undergoes significant accelerations. [10], [11] propose a continuous-time representation to elegantly parameterize the camera trajectory with B-splines, avoiding lowering the dimension of the motion. They use this formulation to incorporate measurements from RS cameras and IMU, developing a VO system and a visual-inertial calibration system for RS cameras. Using the continuous-time representation, Kim et al. [12] extend LSD-SLAM [23] to a RS version, turning the pose estimation problem into solving control points of B-splines in photometric bundle adjustment optimization. Similarly, [13] proposes a dense spline-based continuous-time tracking and mapping method for RS RGB-D cameras, modeling RS in both the RGB and depth image by a photometric error term and a geometric error term respectively.

B. Line Delay Calibration

Geyer et al. [14] propose a line delay calibration method aided by a LED, where the RS camera is exposed with a LED flickering at high frequency and the resulting images include light and dark lines, the spatial frequency of which is related to the line delay and the known LED frequency. However, it requires expensive hardware and tends to be imprecision. Oth et al. [6] present an offline calibration procedure to determine the line delay with higher accuracy that only requires videos of a known calibration pattern, in which the continuous-time optimization is adopted. [15] also proposes a continuous-time spline-based approach for spatiotemporal calibration of the system composed of a RS camera and an IMU using a specific calibration target, where line delay is also accurately calibrated as a byproduct. However, offline calibration with the need for calibration targets might be tedious in some cases and online calibration can be a more friendly way, especially when the calibrated parameter changes every time the system starts. Built upon the discrete-time VIO, MSCKF [4], Li and Mourikis [16] propose a high-precision pose estimation algorithm for systems equipped with low-cost inertial sensors and RS cameras. The key characteristic of the proposed method is that parameters of the RS camera and IMU can be calibrated online including the line delay. Yang et al. [17] comprehensively investigate the online self-calibration of visual-inertial state estimation in the MSCKF framework, which also supports line delay calibration of RS cameras.

Compared to the aforementioned work, our method not only utilizes the continuous-time B-spline trajectory to tightly fuse visual and inertial measurements in the odometry system, but also elegantly handles the RS effect. We are among the first to propose a complete continuous-time sliding-window VIO for RS cameras, implemented as a keyframe-based estimator with probabilistic sliding-window optimization. Carefully-designed marginalization accommodating continuous-time optimization is proposed for the sliding-window estimator.

III. Methodology

We first introduce the convention in this letter. We denote the 6-DoF rigid transformation by ${}^A_B\mathbf{T} \in SE(3) \in \mathbb{R}^{4\times 4}$, which transforms the point ${}^B\mathbf{p}$ in the frame $\{B\}$ into the frame $\{A\}$. ${}^A_B\mathbf{T} = \begin{bmatrix} {}^A_B\mathbf{R} & {}^A\mathbf{p}_B \\ \mathbf{0} & 1 \end{bmatrix}$ consists of rotation ${}^A_B\mathbf{R} \in SO(3)$ and translation ${}^A\mathbf{p}_B \in \mathbb{R}^3$. $\mathrm{Exp}(\cdot)$ maps Lie Algebra to Lie Group and $\mathrm{Log}(\cdot)$ is its inverse operation. $(\cdot)_\wedge$ maps a 3D vector to the corresponding skew-symmetric matrix, while $(\cdot)_\vee$ is its inverse operation [24].

Pixels at different rows of a RS image are exposed at different time instants. The line delay between two adjacent rows is denoted by t_r, and the timestamp of a RS image is deemed as the time instant of its first row. Hence, for a RS image with timestamp t_i which has h rows, the timestamps of each row are between $[t_i, t_i + h \cdot t_r]$.

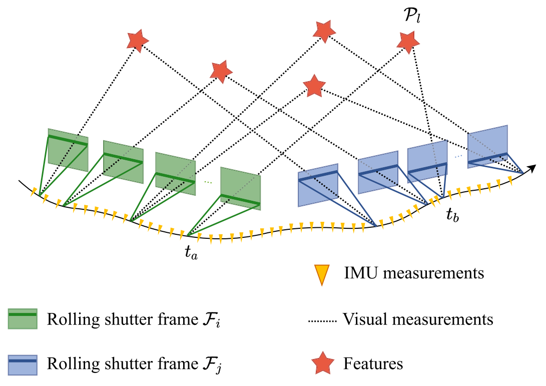
A. Continous-Time Trajectory Representation

In this letter, we adopt a split representation of continuous-time trajectory as [25], using two uniform cumulative B-splines to separately parameterize the 3D rotation and the 3D translation, that is

$$\mathbf{R}(u) = \mathbf{R}_i \cdot \prod_{j=1}^{k-1} \mathrm{Exp}\left(\zeta_j(u) \cdot \mathrm{Log}\left(\mathbf{R}_{i+j-1}^{-1}\mathbf{R}_{i+j}\right)\right) \quad (1)$$

$$\mathbf{p}(u) = \mathbf{p}_i + \sum_{j=1}^{k-1} \zeta_j(u) \cdot (\mathbf{p}_{i+j} - \mathbf{p}_{i+j-1}) \quad (2)$$

$$u(t) = \frac{t - t_0}{\Delta t} - i, t \in [t_i, t_{i+1}) \quad (3)$$



$\mathcal{P}_l$
t_a
t_b
IMU measurements
Rolling shutter frame $\mathcal{F}_i$
Visual measurements
Rolling shutter frame $\mathcal{F}_j$
Features

Fig. 1. A segment of the continuous-time trajectory of IMU in the sliding window constrained by visual measurements and IMU measurements. The extrinsic transformation between the camera and IMU is omitted for simplicity. $\mathcal{F}_i$ and $\mathcal{F}_j$ are two frames in the sliding window. A landmark $\mathcal{P}_l$, is first observed in frame $\mathcal{F}_i$ at time t_a and later observed in frame $\mathcal{F}_j$ at time t_b (see (6)).

where $\mathbf{R}_x$ and $\mathbf{p}_x$ denote **C**ontrol **P**oints (**CP**). The knots $t_0, t_1, t_2, \ldots$ of B-splines are uniformly distributed by the time interval Δt, thus cumulative matrix $\widetilde{\mathbf{M}}^{(k)} \in \mathbb{R}^{k \times k}$ is constant. $\zeta_j(u)$ is the jth row of $\widetilde{\mathbf{M}}^{(k)} \begin{bmatrix} 1 & u & \ldots & u^{k-1} \end{bmatrix}^\top$ [18]. Cubic B-Spline is adopted in this letter, which implies $k = 4$. The continuous-time trajectory of IMU in world frame $\{W\}$ can be denoted as

$$ {}^W_B\mathbf{T}(t) = \begin{bmatrix} {}^W_B\mathbf{R}(t) & {}^W\mathbf{p}_B(t) \\ \mathbf{0} & 1 \end{bmatrix}. \tag{4} $$

With the known pre-calibrated extrinsic ${}^B_C\mathbf{T}$ between IMU and camera, the camera trajectory can be calculated as

$$ {}^W_C\mathbf{T}(t) = {}^W_B\mathbf{T}(t)\,{}^B_C\mathbf{T}. \tag{5} $$

In this letter, we use $\mathbf{P}_x$ to denote the compound control point including both the rotational CP $\mathbf{R}_x$ and the positional CP $\mathbf{p}_x$. For simplicity, the four compound control points at time t_x consisting of $\{\mathbf{P}_{x1}, \mathbf{P}_{x2}, \mathbf{P}_{x3}, \mathbf{P}_{x4}\}$ are also represented by a set $\boldsymbol{\Phi}(t_x)$, and the control points in the time interval $[t_x, t_{x+1}]$ are denoted as $\boldsymbol{\Phi}(t_x, t_{x+1})$.

B. Visual Factor with Line Delay

We detect Shi-Tomasi corners [26] in RS images and use the KLT sparse optical flow [27] to track the detected corners between consecutive images. Since simple homographic and fundamental checking are prone to be affected by the RS effect [28], we adopt cross-checking to remove outliers in feature tracking. Specifically, features are firstly tracked forward in time and then tracked reversely. Only the tracked features located in the vicinity of their original positions within a certain threshold are considered inliers.

As shown in Fig. 1, a landmark $\mathcal{P}_l$, first observed in frame $\mathcal{F}_i$ with timestamp t_i and later observed in frame $\mathcal{F}_j$ with timestamp t_j, is parameterized in inverse depth representation. The corresponding visual factor based on reprojection error is defined as:

$$ \mathbf{r}_c = \begin{bmatrix} \mathbf{e}_1^\top \\ \mathbf{e}_2^\top \end{bmatrix} \left(\frac{\hat{\mathbf{p}}_b}{\mathbf{e}_3^\top \hat{\mathbf{p}}_b} - \pi_c\left(\begin{bmatrix} u_l^{c_b} \\ v_l^{c_b} \end{bmatrix} \right) \right) $$

$$ \hat{\mathbf{p}}_b = \left({}^W_C\mathbf{T}(t_b)\right)^\top \, {}^W_C\mathbf{T}(t_a)\, \frac{1}{\lambda_l} \pi_c\left(\begin{bmatrix} u_l^{c_a} \\ v_l^{c_a} \end{bmatrix} \right) $$

$$ t_a = t_i + v_l^{c_a} \cdot t_r,\ t_b = t_j + v_l^{c_b} \cdot t_r \tag{6} $$

where $\mathbf{e}_i$ is a 3×1 vector with its i-th element to be 1 and the others to be 0. $\pi_c(\cdot)$ denotes the back projection function which transforms a pixel to the normalized image plane. $\begin{bmatrix} u_l & v_l \end{bmatrix}^\top$ is 2D raw observation of landmark $\mathcal{P}_l$ whose inverse depth λ_l is defined in its first observed frame. The exact timestamps of the observations in $\mathcal{F}_i$ and $\mathcal{F}_j$ are t_a and t_b, respectively. Note that visual landmarks are triangulated by the specific row poses of RS keyframes in the sliding window, as depicted in Fig. 1, so as to initialize inverse depths of the landmarks.

From (6), we can find $\mathbf{r}_c$ is relevant to both t_a and t_b, thus the Jacobian of $\mathbf{r}_c$ w.r.t t_r is as follows.

$$ \frac{\partial \mathbf{r}_c}{\partial t_r} = \frac{\partial \mathbf{r}_c}{\partial t_a}\frac{\partial t_a}{\partial t_r} + \frac{\partial \mathbf{r}_c}{\partial t_b}\frac{\partial t_b}{\partial t_r}. \tag{7} $$

where $\partial \mathbf{r}_c / \partial t_a$ and $\partial \mathbf{r}_c / \partial t_b$ contain the time derivatives of splines which fortunately have analytic forms [18]. With a slight abuse of notion, we use the followings to denote the time derivative of continuous-time trajectory (refer to [10], [18] for the exact expressions):

$$ \frac{\partial {}^W_B\mathbf{R}(t)}{\partial t} = {}^W_B\dot{\mathbf{R}}(t), \quad \frac{\partial {}^W\mathbf{p}_B(t)}{\partial t} = {}^W\mathbf{v}_B(t) \tag{8} $$

Based on (6, 7, 8), we can derive the Jacobian of $\mathbf{r}_c$ w.r.t t_r straightforwardly:

$$ \begin{aligned} \frac{\partial \mathbf{r}_c}{\partial t_r} = \begin{bmatrix} \mathbf{e}_1^\top \\ \mathbf{e}_2^\top \end{bmatrix} \Big(& v_l^{c_b} \cdot {}^B_C\mathbf{R}^\top {}^W_B\dot{\mathbf{R}}^\top(t_b) {}^W_B\mathbf{R}(t_a) \hat{\mathbf{p}}_m \\ & + v_l^{c_a} \cdot {}^B_C\mathbf{R}^\top {}^W_B\mathbf{R}^\top(t_b) {}^W_B\dot{\mathbf{R}}(t_a) \hat{\mathbf{p}}_m \\ & + v_l^{c_b} \cdot {}^B_C\mathbf{R}^\top {}^W_B\dot{\mathbf{R}}^\top(t_b) \left({}^W\mathbf{p}_B(t_a) - {}^W\mathbf{p}_B(t_b)\right) \\ & + {}^B_C\mathbf{R}^\top {}^W_B\mathbf{R}^\top(t_b) \left(v_l^{c_a} \cdot {}^W\mathbf{v}_B(t_a) - v_l^{c_b} \cdot {}^W\mathbf{v}_B(t_b) \right) \Big) \end{aligned} $$

$$ \hat{\mathbf{p}}_m = {}^B_C\mathbf{R} \frac{1}{\lambda_l} \pi_c\left(\begin{bmatrix} u_l^{c_a} \\ v_l^{c_a} \end{bmatrix} \right) + {}^B\mathbf{p}_C \tag{9} $$

With the Jacobian of $\mathbf{r}_c$ w.r.t t_r, we can optimize the line delay t_r in the factor graph optimization.

C. Inertial Factors

The time derivatives of the continuous-time trajectory represented by splines lead to the angular velocity and linear acceleration, which can be easily computed [18]. Thus we directly utilize raw IMU measurements, consisting of angular velocity and linear acceleration, to formulate IMU factors at each IMU timestamp. Considering the IMU measurements within two consecutive frames $\mathcal{F}_k$ and $\mathcal{F}_{k+1}$ in the window, the IMU factor and the bias factor based on bias random walk are defined as:

$$ \mathbf{r}_i = \begin{bmatrix} {}^B\boldsymbol{\omega}(t) - {}^B\boldsymbol{\omega}_m + \mathbf{b}_{g_k} \\ {}^W_B\mathbf{R}^\top(t) \left({}^W\mathbf{a}(t) + {}^W\mathbf{g} \right) - {}^B\mathbf{a}_m + \mathbf{b}_{a_k} \end{bmatrix} \tag{10} $$

$$ \mathbf{r}_b = \begin{bmatrix} \mathbf{b}_{g_{k+1}} - \mathbf{b}_{g_k} \\ \mathbf{b}_{a_{k+1}} - \mathbf{b}_{a_k} \end{bmatrix} \tag{11} $$

where ${}^B\boldsymbol{\omega}(t) = ({}^W_B\mathbf{R}^\top(t) {}^W_B\dot{\mathbf{R}}(t))_\vee$, ${}^W\mathbf{a}(t) = {}^W\ddot{\mathbf{p}}_B(t)$ [18] are the angular velocity in $\{B\}$ frame and linear acceleration in $\{W\}$ frame, while ${}^{B}\boldsymbol{\omega}_m, {}^{B}\mathbf{a}_m$ are the raw measurements of angular velocity and linear acceleration at time t, respectively. ${}^{W}\mathbf{g} = \begin{bmatrix}0 & 0 & 9.8\end{bmatrix}^{\top}$ is the gravity vector in the world frame. Note that, the IMU measurements in $[t_k, t_{k+1})$ share the same gyroscope bias $\mathbf{b}_{g_k}$ and accelerometer bias $\mathbf{b}_{a_k}$.

IMU factors formulated with raw IMU measurements in the sliding-window optimization is displayed in Fig. 1. Apart from these IMU factors, we also have an alternative option to utilize IMU pre-integration [1], [29] factor for effective marginalization purpose (see Section III-E). With $\boldsymbol{\alpha}_{B_kB_{k+1}}, \mathbf{R}_{B_kB_{k+1}}$ and $\boldsymbol{\beta}_{B_kB_{k+1}}$ denoting the IMU pre-integration measurements based on IMU raw measurements during two keyframed images $[t_k, t_{k+1})$, the IMU pre-integration factor constraining the relative pose between two sequential keyframe images is formulated as [1], [29]:

$$\mathbf{r}_t = \begin{bmatrix} {}^{W}_{B_k}\mathbf{R}^{\top}\left({}^{W}\mathbf{p}_{B_{k+1}} - {}^{W}\mathbf{p}_{B_k} - {}^{W}\mathbf{v}_{B_k}\Delta t + \frac{1}{2}\mathbf{g}^{W}\Delta t^2\right) \\ -\boldsymbol{\alpha}_{B_kB_{k+1}} \\ \mathrm{Log}\left(\mathbf{R}^{\top}_{B_kB_{k+1}}\,{}^{W}_{B_k}\mathbf{R}^{\top}\,{}^{W}_{B_{k+1}}\mathbf{R}\right) \\ {}^{W}_{B_k}\mathbf{R}^{\top}\left({}^{W}\mathbf{v}_{B_{k+1}} - {}^{W}\mathbf{v}_{B_k} + {}^{W}\mathbf{g}\Delta t\right) - \boldsymbol{\beta}_{B_kB_{k+1}} \end{bmatrix} \tag{12}$$

where ${}^{W}_{B_k}\mathbf{R} = {}^{W}_{B}\mathbf{R}(t_k)$ and ${}^{W}\mathbf{v}_{B_k} = {}^{W}\mathbf{v}_B(t_k)$.

D. Sliding-Window Optimization

We adopt the same keyframe selection strategy as [1], and perform the discrete-time VIO initialization method [1] to boost our continuous-time VIO. Despite the RS effect, the discrete-time VIO initialization procedure can provide relatively good initial values of gyroscope bias, gravity, landmarks and initial IMU poses, which are used as initial values in the continuous-time sliding-window optimization when the system starts. After initialization, new control points with constant time interval are added and initialized by the predicted poses from IMU measurements.

As shown in Fig. 1, the RS visual observations and IMU raw measurements are used in the sliding-window optimization jointly. Assuming the N RS image frames in the current sliding window are with timestamps $\{t_s, t_{s+1}, \cdots, t_{s+N-1}\}$, and the height of RS images is H, then the control points $\boldsymbol{\Phi}(t_s, t_{s+N-1} + H \cdot t_r)$ which parameterize the trajectory in the window are going to be optimized by solving the following problem:

$$\begin{aligned} &\arg\min_{\mathcal{X}} \Big\{\sum \|\mathbf{r}_c\|^2_{\Sigma_c} + \sum \|\mathbf{r}_i\|^2_{\Sigma_i} + \sum \|\mathbf{r}_b\|^2_{\Sigma_b} \\ &\qquad + \|\mathbf{r}_p\|^2\Big\}, \\ &\mathcal{X} = \{\boldsymbol{\Phi}(t_s, t_{s+N-1} + H \cdot t_r), \mathbf{b}_{g_0} \dots \mathbf{b}_{g_{N-1}}, \\ &\mathbf{b}_{a_0} \dots \mathbf{b}_{a_{N-1}}, \lambda_{l_0} \dots \lambda_{l_{M-1}}, t_r\} \end{aligned} \tag{13}$$

where M is the number of landmarks in the sliding window. $\mathbf{r}_p$ is the prior factor obtained by marginalization (see Section III-E), and Σ_c, Σ_i, Σ_b are the corresponding covariance matrices, respectively.

E. Marginalization in Continuous-Time Sliding-Window Optimization

In order to maintain tractable computation for the sliding-window optimization, we only keep a constant number of RS image frames in the sliding window and marginalize RS frames and relevant states strategically, which is inspired by [1], [30]. When the second latest RS image frame is a non-keyframe, we directly discard visual measurements of this frame, while its corresponding control points and IMU measurements are still retained in the window. On the other hand, when the second latest image frame is a keyframe, we marginalize the oldest keyframe with its corresponding measurements, while reserving part of its control points involved in the remaining frames. That is, the control points that are older than the first control point of the second oldest keyframe $\boldsymbol{\Phi}_{\text{marg}} = \boldsymbol{\Phi}(t_s, t_{s+1}) - \boldsymbol{\Phi}(t_{s+1})$, the IMU biases $\{\mathbf{b}_{g_s}, \mathbf{b}_{a_s}\}$, and the inverse depths of the features in the oldest keyframe, are marginalized out.

In this letter, we design the following two options to marginalize the IMU relevant measurements III-C after the sliding-window optimization:

Strategy 1: pre-integrate all the intermediate IMU raw measurements between the oldest keyframe and the second oldest keyframe, that is $[t_s, t_{s+1})$, formulate one pre-integration [29] factor (12) and one bias factor (11) with $\{\mathbf{b}_{g_s}, \mathbf{b}_{a_s}, \mathbf{b}_{g_{s+1}}, \mathbf{b}_{a_{s+1}}\}$ in the sliding-window optimization, then marginalize out $\{\boldsymbol{\Phi}(t_s) - \boldsymbol{\Phi}(t_{s+1}), \mathbf{b}_{g_s}, \mathbf{b}_{a_s}\}$.

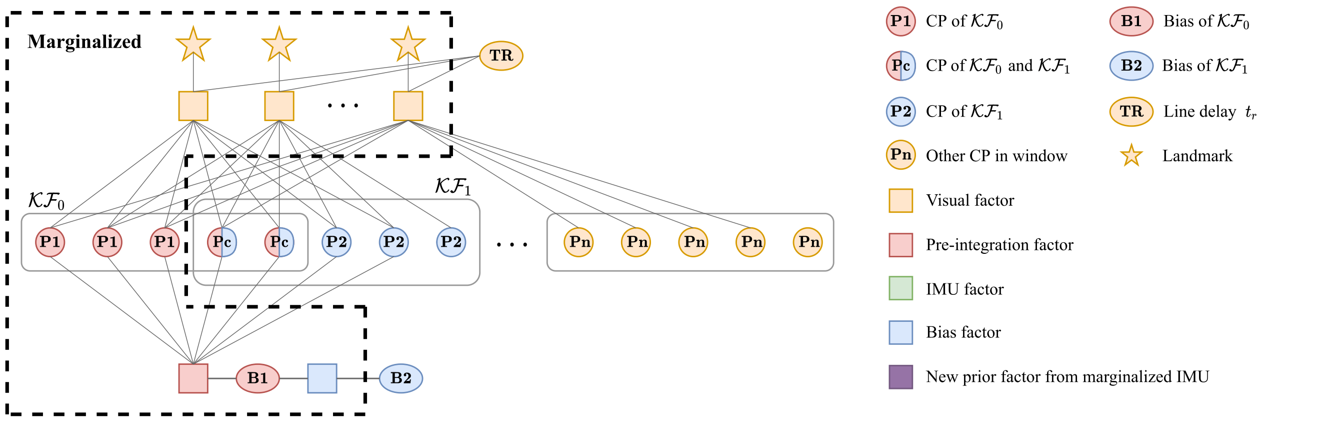
Strategy 2: directly formulate certain IMU factors (10) with the raw IMU measurements in $[t_s, t_{s+1})$ and one bias factor (11) with $\{\mathbf{b}_{g_s}, \mathbf{b}_{a_s}, \mathbf{b}_{g_{s+1}}, \mathbf{b}_{a_{s+1}}\}$, and marginalize out $\{\boldsymbol{\Phi}_{\text{marg}}, \mathbf{b}_{g_s}, \mathbf{b}_{a_s}\}$.

Strategy 2 is exactly in accordance to the optimization cost function (13), while Strategy 1 requires refactoring the cost function (13) after optimization by replacing the IMU factor cost $\sum \|\mathbf{r}_i\|^2_{\Sigma_i}$ with IMU pre-integration factor cost $\sum \|\mathbf{r}_t\|^2_{\Sigma_t}$. Although Strategy 2 is more straightforward and intuitive, Strategy 1 could be more effective. Fig. 2 illustrates the details of the marginalization and the two strategies for marginalizing IMU measurements. The main difference is that the control points of the second oldest frame $\boldsymbol{\Phi}(t_{s+1})$ are certainly involved in the pre-integration factor in Strategy 1, while rarely fully-involved in Strategy 2. After marginalization is carried out via Schur Complement [31], we get the prior factor which constrains the remaining control points and states in the window.

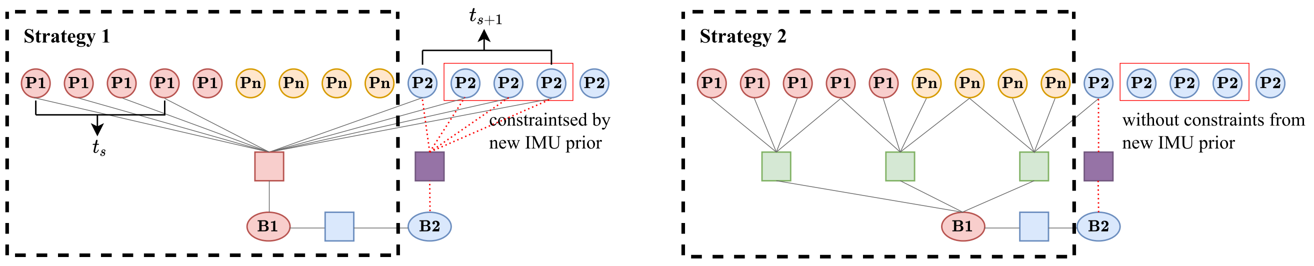
IV. Experiments

We evaluate the proposed method on a synthetic dataset WHU-RSVI [32] and two real-world datasets TUM-RSVI [9], SenseTime-RSVI [33]. The state-of-the-art monocular visual-inertial odometry that we compared against are two RS-aware methods, RS-VI-DSO [9] and RS-VINS-Mono [1], and several methods without RS-aware (GS methods) including OKVIS [34], VINS-Mono [1], ORB-SLAM3 [35], where for fairness, we disable the loop detection and pose graph optimization of VINS-Mono and ORB-SLAM3 for pure VIO evaluations. RS-VINS-Mono[1], a RS version of VINS-Mono, handles the RS effect similarly to [7] but requires known line delay and cannot perform online calibration of line delay. To study the main contribution of our proposed Ctrl-VIO, which is mainly the backend

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


Marginalized
TR
$\mathcal{KF}_0$
$\mathcal{KF}_1$
P1
Pc
P2
Pn
B1
B2
P1 CP of $\mathcal{KF}_0$
Pc CP of $\mathcal{KF}_0$ and $\mathcal{KF}_1$
P2 CP of $\mathcal{KF}_1$
Pn Other CP in window
B1 Bias of $\mathcal{KF}_0$
B2 Bias of $\mathcal{KF}_1$
TR Line delay t_r
Landmark
Visual factor
Pre-integration factor
IMU factor
Bias factor
New prior factor from marginalized IMU

(a) An overview illustration of marginalizing $\mathcal{KF}_0$ with **Strategy 1**


Strategy 1
t_{s+1}
t_s
constraintsed by new IMU prior
Strategy 2
without constraints from new IMU prior
P1
P2
Pn
B1
B2

(b) Two specific strategies for marginalizing out IMU measurements

Fig. 2. (a) An illustration of marginalizing out the oldest keyframe ($\mathcal{KF}_0$) including visual measurements of $\mathcal{KF}_0$ and IMU measurements in $[t_s, t_{s+1})$. The control points (CP) to be marginalized out are the CPs of $\mathcal{KF}_0$ (red circle) expect for the CPs both involved in the second oldest keyframe $\mathcal{KF}_1$ (bicolor circle), since they will be further optimized in the next window. (b) Two different strategies for marginalizing IMU measurements while IMU-irrelevant factors are omitted for clearness. We take the case that there are no coupled CPs between $\mathcal{KF}_0$ and $\mathcal{KF}_1$ as example, which clearly illustrates the CPs involved with the new IMU priors resulting from two marginalization strategies are different as detailed in Section III-E.

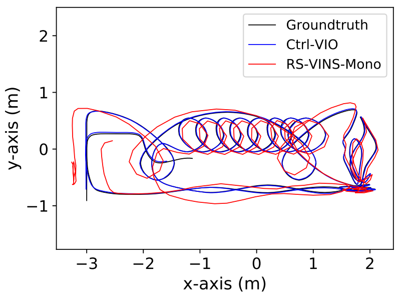
with sliding-window continuous-time optimization, we adapt the frontend of RS-VINS-Mono with cross-checking to remove outliers, and ensure the frontend of the adapted RS-VINS-Mono is the same as our Ctrl-VIO for fair comparison of the backend.

We evaluate Absolute Pose Error (APE) of trajectories composed of keyframe poses by the toolbox [36]. Every method runs six rounds on each sequence and takes the average. Note that in the following experiments, Ctrl-VIO adopts Strategy 1 for marginalization by default, while Ctrl-VIO-margIMU adopts Strategy 2, and this is the only difference between the two. In our experiments, the time interval of B-splines are set as 0.03s and 0.05s for the synthetic dataset and real-world datasets respectively. The max number of features to track is 150, and the number of image frames N in the window is 11.

A. WHU-RSVI Dataset

The WHU-RSVI Dataset [32] synthesizes two trajectory profiles (see Fig. 3) at three different motion speeds (fast, medium, slow), which lead to different extent of RS effects. We name sequences with trajectory number and speed, e.g, *medium-2* is a sequence of trajectory 2 with medium speed. The simulated IMU is collected at 90 Hz and the RS camera at 30 Hz, and the groundtruth of the line delay is 69.44 us.

Firstly, we evaluate the pose estimation performance of various methods with the given ground-truth of line delay. Table I summarizes the RMSE of APE of different methods. With the same trajectory profile, GS methods tend to have larger APE as the motion speed increases. Interestingly, ORB-SLAM3 is



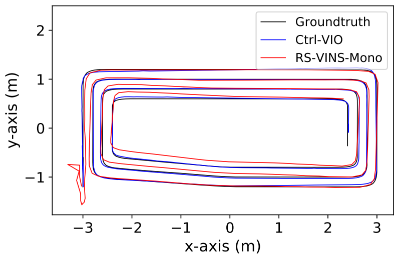


Fig. 3. Estimated trajectories compared to the groundtruth in sequences fast-1 and fast-2 in WHU-RSVI dataset.

less impacted by the RS effect, which is consistent with the experimental findings [37]. However, images in *fast-1* sequence with highly dynamic motion, especially the violent rotation, suffer from severe RS effects, resulting in the inferior performance of ORB-SLAM3. Ctrl-VIO achieves the best accuracy in almost all test sequences. It is also interesting that Ctrl-VIO

TABLE I
THE RMSE(M) OF APE RESULTS OF DIFFERENT METHODS IN WHU-RSVI DATASET. THE BEST RESULTS ARE MARKED IN BOLD

	GS Method			RS Method		
	OKVIS	VINS-Mono	ORB-SLAM3	RS-VINS-Mono	Ctrl-VIO	Ctrl-VIO-margIMU
fast-1	0.539	0.435	0.477	0.149	**0.034**	0.133
medium-1	0.472	0.242	0.074	0.133	**0.058**	0.106
slow-1	0.415	0.267	0.068	0.148	**0.061**	0.257
fast-2	0.492	0.572	0.061	0.125	**0.027**	0.077
medium-2	0.184	0.214	0.067	0.083	**0.064**	0.092
slow-2	0.168	0.074	**0.031**	0.085	0.068	0.084

TABLE II
THE MEAN AND STANDARD DEVIATION OF THE CALIBRATED LINE DELAY BY CTRL-VIO ON WHU-RSVI DATASET. THE GROUND TRUTH OF LINE DELAY IS 69.44 US

	Calibrated line delay (us)	Error (us)
fast-1	69.46 ± 0.55	0.02
medium-1	71.50 ± 0.31	2.06
slow-1	70.81 ± 0.15	1.37
fast-2	67.89 ± 0.41	-1.55
medium-2	72.06 ± 0.39	2.62
slow-2	72.45 ± 0.48	3.01

performs better on fast motion sequences. The reason we speculate might be that highly-dynamic motions make the scale of monocular VIO systems well observable [30]. On the other hand, RS-VINS-Mono performs better than VINS-Mono but is not comparable with Ctrl-VIO, mainly because it assumes constant velocity motion to remove the RS effect, which is not a reasonable hypothesis, especially under large accelerations.

Additional experiments are carried out to verify line delay online calibration accuracy. Starting from zero initial value, the line delay quickly converges within 1 second in our experiments. The calibration results are shown in Table II, where we summarize the mean and standard deviation of the line delay in the last 5 seconds of each sequence. The results demonstrate that line delay is able to be calibrated by Ctrl-VIO with high accuracy and low standard deviation.

B. TUM-RSVI Dataset

The TUM-RSVI dataset [9] is a handheld real-world dataset consisting of a GS camera at 20 Hz, a RS camera at 20 Hz and an IMU at 200Hz. All sensors are hardware synchronized and the line delay of the RS camera approximates 29.4737us, which is given as a reference by the dataset. We test GS methods on both GS and RS data, and RS methods on RS data only. It is noted that RS-VINS-Mono is run with the reference line delay given by the dataset since it does not support online calibration, while the proposed Ctrl-VIO simultaneously calibrates the line delay online from the zero initial value.

The RMSE of APE is shown in Table III, where the RMSE of VI-DSO [3] and its RS version (named RS-VI-DSO) are directly excerpted from [9]. GS methods have plausible performance on GS data, but are deteriorated substantially on RS data, especially on *seq09* and *seq10*, where angular velocities are particularly large, leading to a significant RS effect. However, RS-aware methods which explicitly address the RS effect work well on the RS data, and can achieve comparable accuracy as GS methods on GS data. Ctrl-VIO outperforms RS-VINS-Mono and RS-VI-DSO on most sequences. Moreover, averagely large angular acceleration on both *seq09* and *seq10*, makes the constant velocity assumption used by RS-VINS-Mono and RS-VI-DSO less applicable, while Ctrl-VIO elegantly handling the RS effect with continuous-time trajectory has higher accuracy.

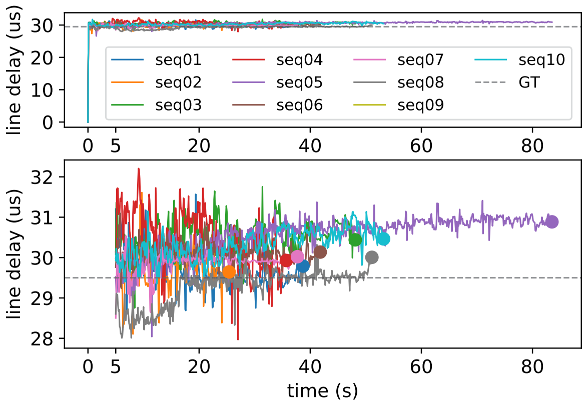


Fig. 4. The calibrated line delay along with time in each sequence of TUM-RSVI dataset, starting from zero initial value. The dashed line represents the reference. Top: line delay estimations along the entire sequence. Bottom: zoom in on [5,80] seconds.

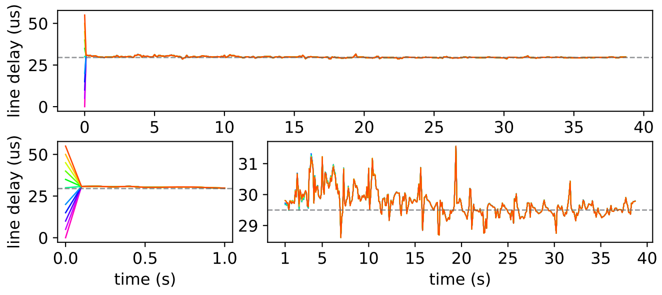


Fig. 5. Consistent calibration results of the line dealy are achieved from various initial values, in *seq01* of TUM-RSVI dataset. The dashed line represents the reference value. Top: line delay estimations along the entire sequence. Bottom: zoom in on [0,1] seconds and [1,40] seconds.

Fig. 4 also shows the convergence of line delay over time on different sequences starting from zero initial value. The line delay parameter quickly converges to a near reference value and keeps nearly static in the remaining trajectory. In addition, we further examine the line delay calibration starting from various initial values. Fig. 5 displays the line delay calibrated from initial values of 0∼55 us with the interval of 5 us. The line delay can stably converge to a near reference value with various initial values.

C. SenseTime-RSVI Dataset

The SenseTime-RSVI Dataset [33] provides RS visual-inertial data collected by a mobile phone, Xiaomi Mi 8. The RS images are captured at 30Hz and IMU data collected at 400Hz. No reference value of line delay is given in this dataset. Fortunately, the proposed Ctrl-VIO is able to run with calibrating line delay online, which is initialized from zero. Note that, we run RS-VINS-Mono with the line delay estimated by Ctrl-VIO. Compared to TUM-RSVI Dataset, SenseTime-RSVI Dataset is collected under more smooth and slow motion, without violent shake and significant accelerations like the former.

TABLE III
THE RMSE(M) OF APE RESULTS OF GS METHODS ON GS DATA, AND BOTH GS AND RS METHODS ON RS DATA IN TUM-RSVI DATASET. THE BEST RESULTS ARE MARKED IN BOLD AND THE SECOND BEST RESULTS ARE MARKED UNDERLINED

Image Type	Method	RS-Aware	*seq01*	*seq02*	*seq03*	seq04	*seq05*	*seq06*	*seq07*	*seq08*	*seq09*	*seq10*
GS	OKVIS	✗	0.199	0.227	0.168	0.151	0.138	0.212	0.203	0.156	0.157	0.134
	VINS-Mono	✗	0.171	0.129	0.273	**0.092**	0.098	0.064	0.083	1.310	0.224	0.130
	ORB-SLAM3	✗	**0.020**	0.020	0.037	0.162	**0.036**	**0.017**	**0.036**	**0.025**	**0.044**	**0.107**
	VI-DSO	✗	0.038	**0.018**	**0.027**	0.137	0.060	0.135	0.061	0.070	0.128	0.111
RS	OKVIS	✗	0.895	0.641	0.435	0.216	0.301	0.576	0.246	0.185	1.002	1.683
	VINS-Mono	✗	114.710	2.230	72.850	53.130	122.690	1.570	155.930	0.409	130.950	159.910
	ORB-SLAM3	✗	2.100	-	0.601	0.204	0.269	0.936	0.260	0.365	-	-
	VI-DSO	✗	79.591	40.725	1.803	0.970	0.683	2.352	28.336	0.501	218.152	482.021
	RS-VINS-Mono	✓	0.129	0.102	0.075	0.049	0.071	0.115	0.102	0.060	0.146	0.189
	RS-VIDSO	✓	0.040	0.044	**0.028**	0.079	0.049	**0.017**	0.075	0.168	0.168	0.246
	Ctrl-VIO	✓	**0.027**	**0.033**	0.047	**0.042**	**0.041**	0.054	**0.057**	**0.058**	**0.062**	**0.079**
	Ctrl-VIO-margIMU	✓	0.065	0.049	0.048	0.046	0.069	0.039	0.083	0.067	0.097	0.128

TABLE IV
THE RMSE(M) OF APE RESULTS OF DIFFERENT METHODS IN SENSETIME-RSVI DATASET. THE BEST RESULTS ARE IN BOLD

	GS Method			RS Method		
	OKVIS	VINS-Mono	ORB-SLAM3	RS-VINS-Mono	Ctrl-VIO	Ctrl-VIO-margIMU
A0	0.072	0.075	0.089	**0.062**	0.068	0.065
A1	0.088	0.161	0.071	0.044	0.036	**0.034**
A2	0.068	0.057	0.107	0.045	0.047	**0.039**
A3	0.023	0.024	0.024	0.016	0.016	**0.015**
A4	0.147	0.022	0.028	0.018	**0.017**	0.030
A5	0.078	0.050	0.080	0.059	**0.046**	**0.046**
A6	0.064	0.027	0.045	0.022	0.019	**0.018**
A7	0.047	0.020	0.032	0.012	**0.011**	0.016

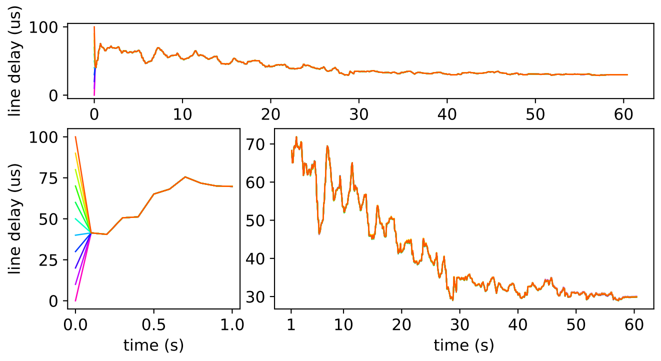


Fig. 6. Consistent results of the line dealy are achieved with various initial values, over *A6* sequence. Top: line delay estimations along the entire sequence. Bottom: zoom in on [0,1] seconds and the remaining.

Table IV shows the RMSE of APE of different methods, where the RMSE of OKVIS and VINS-Mono are directly from [33]. Ctrl-VIO gets more accurate pose estimation on almost all the sequences. It is also interesting to find that GS methods achieve satisfying accuracy mainly because of lowly-dynamic motions of the dataset. It can be easily found that RS-VINS-Mono with the line delay calibrated by Ctrl-VIO gets better performance than VINS-Mono without RS-aware. Fig. 6 also shows the estimated line delay over time on sequence A6, starting from different initial values (from 0 to 100 us with an interval of 10 us). The final converged value of line delay in all trails is about 30 us which is a reasonable value and matches the conclusion in [6] that the line delay of RS cameras of smartphones is around 25∼60 us. However, the convergence of the line delay takes longer time on this dataset, partially due to the insufficient RS effect of the lowly-dynamic motion.

TABLE V
THE MEAN AND STANDARD DEVIATION OF RUNTIME OF CTRL-VIO AND RS-VINS-MONO ON *SEQ1* OF TUM-RSVI DATASET

	Frontend (ms)	Optimization (ms)	Marginalization (ms)
RS-VINS-Mono	27.74(16.24)	18.96(10.29)	3.72(4.22)
Ctrl-VIO	27.31(16.89)	123.01(111.96)	11.07(12.43)

D. Comparison of marginalization strategies for IMU

On the SenseTime-RSVI dataset with lowly-dynamic motion, Ctrl-VIO and Ctrl-VIO-margIMU have very similar performances on pose estimation accuracy (see Table IV). However, as the pose estimation results are shown in Table I and III over WHU-RSVI and TUM-RSVI dataset, the Ctrl-VIO with marginalization Strategy 1 outperforms Ctrl-VIO-margIMU with marginalization Strategy 2. The difference is caused by the different factor graph formulations in Strategy 1 and Strategy 2. The pre-integration factor in Strategy 1 and IMU factors in Strategy 2 involve different control points (described in Section III-E). After IMU information between the oldest keyframe and the second oldest keyframe is marginalized out, for Strategy 2, the new prior factor obtained might not constrain all control points corresponding to the timestamp of the second oldest keyframe, while Strategy 1 indeed does. Although we admit that both marginalization strategies are rational theoretically, they can have different numerical performances in the non-linear least-square optimizations.

E. Runtime analysis

We compare the runtime of Ctrl-VIO and RS-VINS-Mono on *seq1* of the TUM-RSVI Dataset, and Table V shows the result. Both methods are executed on the desktop PC with an Intel i7-8700 CPU @ 3.2Ghz and 32GB RAM. The frequency of RS-VINS-Mono and Ctrl-VIO are 19.83Hz and 6.19Hz, respectively. And the time cost of optimization and marginalization of Ctrl-VIO have relatively high standard deviations since the sliding window duration varies. The number of control points to be optimized increases when the time span of the sliding window is large, and fewer control points will be optimized if the time span is small. For the former situation, the motion is usually gentle and we do not have to uniformly add so many control points to constrain the trajectory. Thus, we consider using a non-uniform B-spline in future work to improve the computational efficiency.

V. Conclusions and Future Work

This letter proposes continuous-time visual-inertial odometry dedicated to RS cameras, dubbed Ctrl-VIO. The VIO is implemented as a keyframe-based sliding-window estimator, which accommodates the continuous-time state optimization and elegantly handles the RS effects. RS visual and IMU measurements are tightly coupled to optimize the trajectory in the sliding window. A specific marginalization method for the continuous-time sliding-window estimator is investigated, and two different ways of marginalizing IMU measurements are compared and discussed. Besides, line delay can be online calibrated in the proposed Ctrl-VIO, which is useful when the line delay is unknown. Verified by simulation and real-world experiments, our system demonstrates high accuracy on RS data, superior to the existing state-of-the-art VIOs.

Although Ctrl-VIO has good performance regarding accuracy and robustness, it cannot be run in real-time currently. The control points in the current implementation are distributed uniformly over time, which is unnecessary when the motion is slow and smooth. It is worthwhile investigating the non-uniform B-spline to parameterize the trajectory with fewer control points to improve efficiency further.

References

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

[2] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4666–4672.

[3] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2510–2517.

[4] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 3565–3572.

[5] J. Mo and J. Sattar, "Continuous-time spline visual-inertial odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 9492–9498.

[6] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1360–1367.

[7] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proc. IEEE 8th Int. Symp. Mixed Augmented Reality*, 2009, pp. 83–86.

[8] D. Schubert, N. Demmel, V. Usenko, J. Stuckler, and D. Cremers, "Direct sparse odometry with rolling shutter," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 682–697.

[9] D. Schubert, N. Demmel, L. von Stumberg, V. Usenko, and D. Cremers, "Rolling-shutter modelling for direct visual-inertial odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 2462–2469.

[10] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras," in *Proc. Brit. Mach. Vis. Conf.*, vol. 113, no. 3, pp. 208–219.

[11] A. Patron-Perez, S. Lovegrove, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," *Int. J. Comput. Vis.*, vol. 113, no. 3, pp. 208–219, 2015.

[12] J.-H. Kim, C. Cadena, and I. Reid, "Direct semi-dense SLAM for rolling shutter cameras," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1308–1315.

[13] C. Kerl, J. Stuckler, and D. Cremers, "Dense continuous-time tracking and mapping with rolling shutter rgb-d cameras," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2264–2272.

[14] C. Geyer, M. Meingast, and S. Sastry, "Geometric models of rolling-shutter cameras," in *Proc. 6th OmniVis WS*, 2005, vol. 1, no. 4.

[15] J. Huai, Y. Zhuang, Y. Lin, G. Jozkow, Q. Yuan, and D. Chen, "Continuous-time spatiotemporal calibration of a rolling shutter camera-imu system," *IEEE Sensors J.*, vol. 22, no. 8, pp. 7920–7930, Apr. 2022.

[16] M. Li, H. Yu, X. Zheng, and A. I. Mourikis, "High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 409–416.

[17] Y. Yang, P. Geneva, X. Zuo, and G. Huang, "Online self-calibration for visual-inertial navigation systems: Models, analysis and degeneracy," 2022, *arXiv:2201.09170*.

[18] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers, "Efficient derivative computation for cumulative b-splines on lie groups," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11148–11156.

[19] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1280–1286.

[20] G. Cioffi, T. Cieslewski, and D. Scaramuzza, "Continuous-time vs. discrete-time vision-based SLAM: A comparative study," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2399–2406, Apr. 2022.

[21] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proc. IEEE 6th ACM Int. Symp. Mixed Augmented Reality*, 2007, pp. 225–234.

[22] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[23] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.

[24] T. D. Barfoot, *State Estimation for Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[25] A. Haarbach, T. Birdal, and S. Ilic, "Survey of higher order rigid body motion interpolation methods for keyframe animation and continuous-time trajectory estimation," in *Proc. IEEE Int. Conf. 3D Vis.*, 2018, pp. 381–389.

[26] J. Shi and Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1994, pp. 593–600.

[27] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[28] J. Hedborg, P.-E. Forssén, M. Felsberg, and E. Ringaby, "Rolling shutter bundle adjustment," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1434–1441.

[29] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration theory for fast and accurate visual-inertial navigation," *IEEE Trans. Robot.*, pp. 1–18, 2015.

[30] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Initialization-free monocular visual-inertial state estimation with application to autonomous MAVs," in *Experimental Robotics*. Berlin, Germany: Springer, 2016, pp. 211–227.

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

[32] L. Cao, J. Ling, and X. Xiao, "The WHU rolling shutter visual-inertial dataset," *IEEE Access*, vol. 8, pp. 50771–50779, 2020.

[33] L. Jinyu, Y. Bangbang, C. Danpeng, W. Nan, Z. Guofeng, and B. Hujun, "Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality," *Virtual Reality Intell. Hardware*, vol. 1, no. 4, pp. 386–410, 2019.

[34] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, 2015.

[35] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[36] M. Grupp, "EVO: Python package for the evaluation of odometry and SLAM," 2017. [Online]. Available: https://github.com/MichaelGrupp/evo

[37] F. Johansson and S. Svensson, "Evaluation of monocular visual SLAM methods on UAV imagery to reconstruct 3D terrain," Linköping Univ., 2021. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1575660/FULLTEXT01.pdf