Vision-aided Inertial Navigation with Rolling-Shutter Cameras

Mingyang Li and Anastasios I. Mourikis
Dept. of Electrical Engineering, University of California, Riverside
E-mail: mli@ee.ucr.edu, mourikis@ee.ucr.edu

Abstract— **In this paper, we focus on the problem of pose estimation using measurements from an inertial measurement unit and a rolling-shutter (RS) camera. The challenges posed by RS image capture are typically addressed by using approximate, low-dimensional representations of the camera motion. However, when the motion contains significant accelerations (common in small-scale systems) these representations can lead to loss of accuracy. By contrast, we here describe a different approach, which exploits the inertial measurements to avoid any assumptions on the nature of the trajectory. Instead of parameterizing the *trajectory*, our approach parameterizes the *errors* in the trajectory estimates by a low-dimensional model. A key advantage of this approach is that, by using prior knowledge about the estimation errors, it is possible to obtain *upper bounds* on the modeling inaccuracies incurred by different choices of the parameterization's dimension. These bounds can provide guarantees for the performance of the method, and facilitate addressing the accuracy-efficiency tradeoff. This RS formulation is used in an extended-Kalman-filter estimator for localization in unknown environments. Our results demonstrate that the resulting algorithm outperforms prior work, in terms of accuracy and computational cost. Moreover, we demonstrate that the algorithm makes it possible to use low-cost consumer devices (i.e., smartphones) for high-precision navigation on multiple platforms.**

I. Introduction

In this paper we focus on the problem of motion estimation by fusing the measurements from an inertial measurement unit (IMU) and a *rolling-shutter* (RS) camera. In recent years, a significant body of literature has focused on motion estimation using cameras and inertial sensors, a task often termed *vision-aided inertial navigation* (see, e.g., (Jones and Soatto, 2011; Kelly and Sukhatme, 2011; Weiss et al., 2012; Kottas et al., 2012; Li and Mourikis, 2013b) and references therein). However, the overwhelming majority of the algorithms described in prior work assume the use of a *global-shutter* (GS) camera, i.e., a camera in which all the pixels in an image are captured simultaneously. By contrast, in a RS camera the image rows are captured sequentially, each at a slightly different time instant. This can create significant image distortions (see Fig. 1), which must be modeled in the estimation algorithm.

The use of RS sensors is desirable for a number of reasons. First, the vast majority of low-cost cameras today employ CMOS sensors with RS image capture. Methods for high-precision pose estimation using RS cameras will therefore facilitate the design of localization systems for low-cost robots and MAVs. In addition to the applications in the area of robotics, methods for visual-inertial localization using RS cameras will allow tracking the position of consumer devices (e.g., smartphones) with unprecedented accuracy, even in GPS-denied environments. This can enable the development of localization aids for the visually impaired, and lead to a new generation of augmented-reality and high-precision location-based applications. We also point out that a smartphone capable of real-time, high-precision pose estimation can be mounted on a mobile robot to provide an inexpensive and versatile localization solution (see Section V-B). Given the widespread availability of smartphones, this can lower the barrier to entry in robotics research and development.

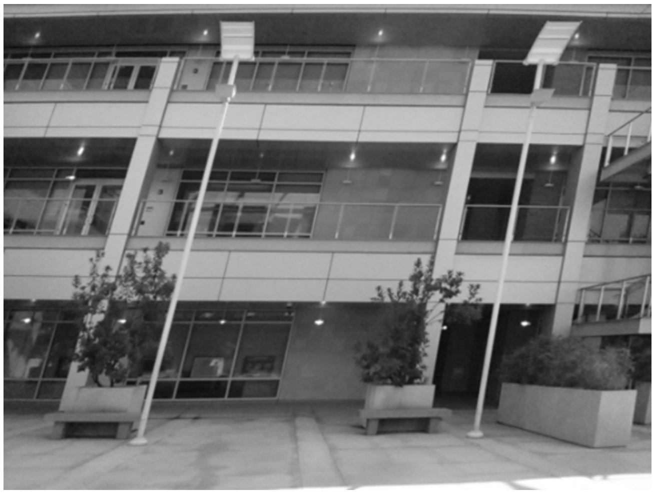

Fig. 1: An example image with rolling-shutter distortion.

In a RS camera, image rows are captured sequentially over a time interval of nonzero duration called the image *readout time*[1]. Therefore, when the camera is moving, each row of pixels is captured from a *different camera pose*. An "exact" solution to the pose estimation problem would require an estimator that includes in its state vector a separate pose for each image row. Since this is computationally intractable, existing solutions to RS-camera localization employ parametric representations of the camera motion during the readout time (e.g., constant-velocity models or B-splines). This, however, creates an unfavorable tradeoff: low-dimensional representations result in efficient algorithms, but also introduce modeling errors, which reduce the accuracy of any estimator. On the other hand, high-dimensional representations can model complex motions, but at high computational cost, which may prevent real-time estimation.

We here propose a novel method for using a RS camera for

[1] Note that alternative terms have been used to describe this time interval (e.g., shutter roll-on time (Lumenera Corporation, 2005) and line delay (Oth et al., 2013)). We here adopt the terminology defined in (Ringaby and Forssén, 2012).

motion estimation that avoids this tradeoff. To describe the main idea of the method, we start by pointing out that any estimator that employs linearization (e.g., the extended Kalman filter (EKF), iterative-minimization methods) relies on the computation of (i) measurement residuals, and (ii) linearized expressions showing the dependence of the residuals on the estimation errors. The first step involves the state *estimates*, while the second the state *errors*, which may have *different representations*[2]. Based on this observation, we here propose a method for processing RS measurements that imposes *no* assumptions on the form of the trajectory when computing the residuals, and, instead, uses a parametric representation of the *errors* of the estimates in linearization.

Specifically, for computing residuals we take advantage of the IMU measurements, which allow us to obtain estimates of the pose in the image-readout interval, given the state estimate at one instant in this interval. This makes it possible to model arbitrarily complex motions when computing residuals, as long as they are within the IMU's sensing bandwidth. On the other hand, when performing linearization, we represent the estimation errors during the readout interval using a weighted sum of temporal basis functions. By varying the dimension of the basis used, we can control the computational complexity of the estimator. The key advantage of the method is that, since the statistical properties of the errors are known in advance, we can compute *upper bounds* on the worst-case modeling inaccuracy, and use them to guide the selection of the basis dimension. We demonstrate that, in practice, a very low-dimensional representation suffices, and thus the computational cost can be kept low – almost identical to that needed for processing measurements from a GS camera.

This novel method for utilizing the RS measurements is employed for real-time visual-inertial localization in conjunction with the EKF-based estimator of (Li and Mourikis, 2012a). This is a hybrid estimator, which combines a sliding-window formulation of the filter equations with a feature-based one, to exploit the computational advantages of both. By combining the computational efficiency of the hybrid EKF with that of the proposed method for using RS measurements, we obtain an estimator capable of real-time operation even in resource-constrained systems. In our experiments, we have employed commercially available smartphone devices, mounted on different mobile platforms. The proposed estimator is capable of running comfortably in real time on the low-power processors of these devices, while resulting in very small estimation errors. These results demonstrate that high-precision visual-inertial localization with RS cameras is possible, and can serve as the basis for the design of low-cost localization systems in robotics.

II. Related Work

Most work on RS cameras has focused on the problem of image rectification, for compensating the visual distortions caused by the rolling shutter. These methods estimate a parametric representation of the distortion from the images, which is then used to generate a rectified video stream (see, e.g., (Liang et al., 2008; Forssen and Ringaby, 2010) and references therein). Gyroscope measurements have also been employed for distortion compensation, since the most visually significant distortions are caused by rotational motion (Hanning et al., 2011; Karpenko et al., 2011; Jia and Evans, 2012). In contrast to these methods, our goal is not to undistort the images (which is primarily done to create visually appealing videos), but rather to use the recorded images for motion estimation.

One possible approach to this problem is to employ estimates of the camera motion in order to "correct" the projections of the features in the images, and to subsequently treat the measurements as if they were recorded by a GS sensor. This approach is followed in (Klein and Murray, 2009), which describes an implementation of the well-known PTAM algorithm (Klein and Murray, 2007) using a RS camera. The feature projections are corrected by assuming that the camera moves at a constant velocity, estimated from image measurements in a separate least-squares process. Similarly, image corrections are applied in the structure-from-motion method of (Hedborg et al., 2011), in which only the rotational motion of the camera is compensated for. In both cases, the "correction" of the image features' coordinates entails approximations, which are impossible to avoid in practice: for completely removing the RS effects, the points' 3D coordinates as well as the camera motion would have to be perfectly known.

The approximations inherent in the approaches described above are circumvented in methods that include a representation of the camera's motion in the states to be estimated. This makes it possible to explicitly model the motion in the measurement equations, and thus no separate "correction" step is required. All such methods to date employ low-dimensional parameterizations of the motion, for mathematical simplicity and to allow for efficient implementation. For example, in (Ait-Aider et al., 2006; Ait-Aider and Berry, 2009; Magerand and Bartoli, 2010) a constant-velocity model is used to enable the estimation of an object's motion. In (Hedborg et al., 2012), a RS bundle-adjustment method is presented, which uses a constant-velocity model for the position and SLERP interpolation for the orientation. As mentioned in Section I, however, these simple representations of the camera trajectory introduce modeling inaccuracies, which can be significant if the motion is not smooth.

To demonstrate this, in Fig. 2(left) we plot the rotational velocity, $\boldsymbol{\omega}_m$, and acceleration, $\mathbf{a}_m$, measured by the IMU on a Nexus 4 device during one of our experiments. The plots show a one-second-long window of data, recorded while the device was held by a person walking at normal pace. In this time interval, 12 images were captured, each with a readout time of 43.3 ms. From the plots of $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$, it becomes clear that the device's motion is changing rapidly, and thus low-dimensional motion representations will lead to significant inaccuracies. To quantify the modeling inaccuracies, in Fig. 2(right) we plot the largest absolute

[2] For instance, this fact is commonly exploited in 3D-orientation estimation, where the estimates may be expressed in terms of a 3×3 rotation matrix or a 4×1 unit quaternion, while for the errors one typically uses a minimal, 3-element representation.

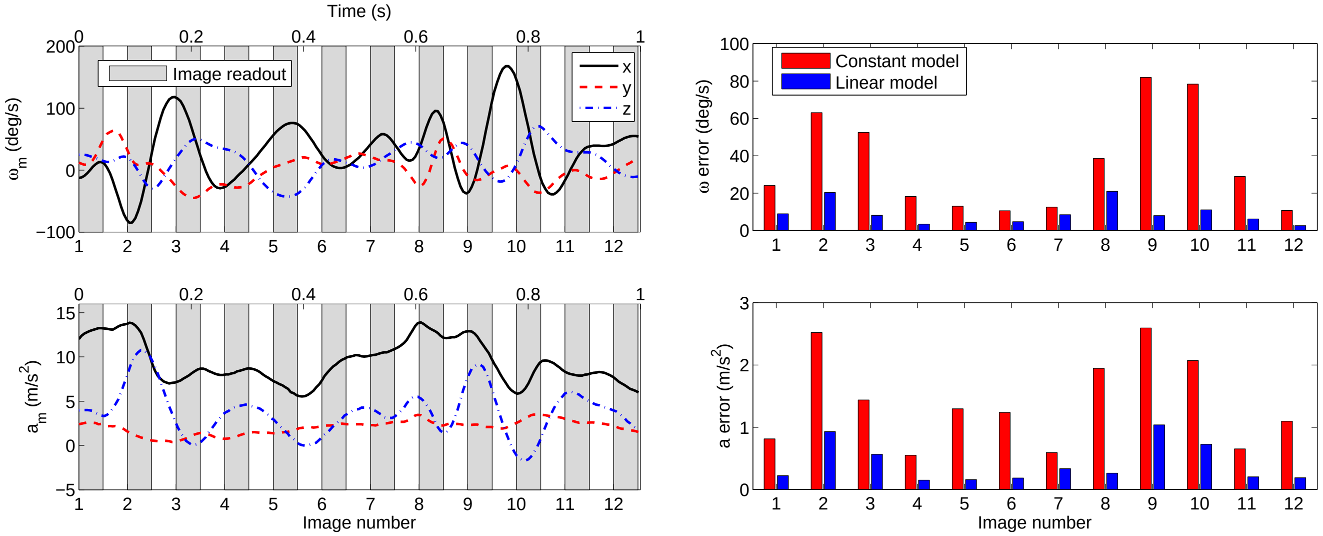

Fig. 2: Left: The rotational velocity (top) and acceleration (bottom) measurements recorded during a one-second period in one of our experiments. Right: The largest modeling errors incurred in each readout interval when using the best-fit constant (red) and linear (blue) representations of the signals.

difference between the signals and their best-fit constant and linear approximations in each readout interval. Clearly, the approximations are quite poor, especially for the rotational velocity. The modeling errors of the constant-velocity model for $\boldsymbol{\omega}_m$ (employed, e.g., in (Li et al., 2013)) reach 81.8 ^{o}/s. Even if a linear approximation of $\boldsymbol{\omega}_m$ were to be used, the modeling inaccuracies would reach 20.9 ^{o}/s. We point out that, due to their small weight, miniaturized systems such as hand-held devices or MAVs typically exhibit highly-dynamic motion profiles, such as the one seen in Fig. 2. These systems are key application areas for vision-aided inertial navigation, and thus methods that use low-dimensional motion parameterizations can be of limited utility.

An elegant approach to the problem of motion parameterization in vision-based localization is offered by the continuous-time formulation originally proposed in (Furgale et al., 2012). This formulation has recently been employed for pose estimation and RS camera calibration in (Oth et al., 2013), and for visual-inertial SLAM with RS cameras in (Lovegrove et al., 2013). A similar approach has also been used in (Bosse and Zlot, 2009; Bosse et al., 2012), to model the trajectory for 2D laser-scanner based navigation. The key idea of the continuous-time formulation is to use a weighted sum of temporal basis functions (TBF) to model the motion. This approach offers the advantage that, by increasing the number of basis functions, one can model arbitrarily complex trajectories. Thus, highly-dynamic motion profiles can be accommodated, but this comes at the cost of an increase in the number of states that need to be estimated (see Section III-A for a quantitative analysis). The increased state dimension is not a significant obstacle if offline estimation is performed (which is the case in the aforementioned approaches), but is undesirable in real-time applications. Similar limitations exist in the Gaussian-Process-based representation of the state, described in (Tong et al., 2013).

TABLE I: List of notations

$\hat{\mathbf{x}}$	Estimate of the variable $\mathbf{x}$
$\tilde{\mathbf{x}}$	The error of the estimate $\hat{\mathbf{x}}$, defined as $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$
$\dot{\mathbf{x}}$	The first-order time derivative of $\mathbf{x}$
$\mathbf{x}^{(i)}$	The ith-order time derivative of $\mathbf{x}$
$\lfloor \mathbf{c} \times \rfloor$	The skew-symmetric matrix corresponding to the 3×1 vector $\mathbf{c}$
${}^{X}\mathbf{c}$	The vector $\mathbf{c}$ expressed in the coordinate frame $\{X\}$
${}^{X}\mathbf{p}_Y$	Position of the origin of frame $\{Y\}$ expressed in $\{X\}$
${}^{X}_{Y}\mathbf{R}$	The rotation matrix rotating vectors from frame $\{Y\}$ to $\{X\}$
${}^{X}_{Y}\bar{\mathbf{q}}$	The unit quaternion corresponding to the rotation ${}^{X}_{Y}\mathbf{R}$
$\mathbf{0}$	The zero matrix
$\mathbf{I}_n$	The $n \times n$ identity matrix
$\otimes$	Quaternion multiplication operator

We stress that, with the exception of the continuous-time formulation of (Lovegrove et al., 2013), all the RS motion-estimation methods discussed above are *vision-only* methods. To the best of our knowledge, the only work to date that presents large-scale, real-time localization with a RS camera and an IMU is that of (Li et al., 2013). The limitations of that work, however, stem from its use of a constant-velocity model for the motion during the image readout. This reduces accuracy when the system undergoes significant accelerations, but also increases computational requirements, as both the linear and rotational velocity at the time of image capture must be included in the state vector. In the experimental results presented in Section V, we show that the novel formulation for using the RS measurements presented here outperforms (Li et al., 2013), both in terms of accuracy and computational efficiency.

III. Rolling-shutter modeling

Our goal is to track the position and orientation of a moving platform equipped with an IMU and a RS camera. While several estimation approaches can be employed for this task, a common characteristic of almost all of them is that they rely on linearization. This is, for example, the case with EKF-based and iterative-minimization-based methods, which form the overwhelming majority of existing algorithms. In this section, we describe a method for processing the measurements from the RS camera, which can be employed in conjunction with any linearization-based algorithm. Table I describes the notation used in the remainder of the paper.

The defining characteristic of a RS camera is that it captures the rows of an image over an interval of duration t_r (the readout time). If the image has N rows, then the time instants these rows are captured are given by:

$$t_n = t_o + \frac{nt_r}{N}, \qquad n \in \left[-\frac{N}{2}, \frac{N}{2}\right] \tag{1}$$

where t_o is the midpoint of the image readout interval.

Let us consider a feature that is observed on the n-th row of the image. Its measurement is described by:

$$\mathbf{z} = \begin{bmatrix} z_c \\ z_r \end{bmatrix} = \mathbf{h}\left({}^I_G\bar{\mathbf{q}}(t_n), {}^G\mathbf{p}_I(t_n), \mathbf{x}_a\right) + \mathbf{n} \tag{2}$$

where z_c and z_r are the camera measurements along image columns and rows, respectively; $\mathbf{h}$ is the measurement function (e.g., perspective projection); $\mathbf{n}$ is the measurement noise, modeled as zero-mean Gaussian with covariance matrix $\sigma^2_{\text{im}}\mathbf{I}_2$; ${}^G\mathbf{p}_I(t_n)$ is the position of the IMU frame, $\{I\}$, with respect to the global reference frame, $\{G\}$, at time t_n; ${}^I_G\bar{\mathbf{q}}(t_n)$ is the unit quaternion representing the IMU orientation with respect to $\{G\}$ at t_n; and $\mathbf{x}_a$ includes all additional constant quantities that affect the measurement and are included in the estimator's state vector. These may include, for instance, the camera-to-IMU transformation, the feature position, or the camera intrinsics if these quantities are being estimated online.

In practice, image features are detected in several different rows (different values of n) in each image. To process these measurements, the direct solution would be to include in the estimator one camera pose for each value of n, which is computationally intractable. Thus, the challenge in designing a practical formulation for RS cameras is to include in the estimator only a small number of states per image, while keeping the model inaccuracies small. As explained next, the method we present here requires the IMU position, orientation, and potentially derivatives of these, at only *one* time instant, namely t_o, to be in the estimator's state vector.

We begin by noting that, in any linearization-based estimator, the processing of the measurement in (2) is based on computing the associated residual, defined by:

$$\mathbf{r} = \mathbf{z} - \mathbf{h}\left({}^I_G\hat{\bar{\mathbf{q}}}(t_n), {}^G\hat{\mathbf{p}}_I(t_n), \hat{\mathbf{x}}_a\right) \tag{3}$$

In this expression the measurement $\mathbf{z}$ is provided from the feature tracker, the estimates $\hat{\mathbf{x}}_a$ are available in the estimator's state vector, and t_n can be calculated from $\mathbf{z}$, using (1) with $n = z_r - N/2$. Thus the only "missing" part in computing the residual $\mathbf{r}$ in (3) is the estimate of the IMU pose at time t_n. In our approach, we compute this by utilizing the IMU measurements. Specifically, we include in the state vector the estimates of the IMU state at t_o, and compute ${}^G\hat{\mathbf{p}}_I(t_n)$ and ${}^I_G\hat{\bar{\mathbf{q}}}(t_n)$, $n \in [-N/2, N/2]$, by integrating the IMU measurements in the readout time interval (the method used for IMU integration is described in Section IV-B).

In addition to computing the residual, linearization-based estimators require a linear (linearized) expression relating the residual in (3) to the errors of the state estimates. To obtain such an expression, we begin by directly linearizing the camera observation model in (2), which yields:

$$\mathbf{r} \approx \mathbf{H}_{\boldsymbol{\theta}}\tilde{\boldsymbol{\theta}}_I(t_n) + \mathbf{H}_{\mathbf{p}}{}^G\tilde{\mathbf{p}}_I(t_n) + \mathbf{H}_a\tilde{\mathbf{x}}_a + \mathbf{n} \tag{4}$$

where $\mathbf{H}_{\boldsymbol{\theta}}$ and $\mathbf{H}_{\mathbf{p}}$ are the Jacobians of the measurement function with respect to the IMU orientation and position at time t_n, and $\mathbf{H}_a$ is the Jacobian with respect to $\mathbf{x}_a$. The IMU orientation error, $\tilde{\boldsymbol{\theta}}_I$, is defined in (22). Since the state at t_n is not in the estimator's state vector, we cannot directly employ the expression in (4) to perform an update – what is needed is an expression relating the residual to quantities at t_o, which do appear in the state. To obtain such an expression, we start with the Taylor-series expansions:

$${}^G\tilde{\mathbf{p}}_I(t_n) = \sum_{i=0}^{\infty} \frac{(nt_r)^i}{N^i i!}\,{}^G\tilde{\mathbf{p}}_I^{(i)}(t_o) \tag{5}$$

$$\tilde{\boldsymbol{\theta}}_I(t_n) = \sum_{i=0}^{\infty} \frac{(nt_r)^i}{N^i i!}\,\tilde{\boldsymbol{\theta}}_I^{(i)}(t_o) \tag{6}$$

The above expressions are exact, but are not practically useful, as they contain an infinite number of terms, which cannot be included in an estimator. We therefore truncate the two series to a finite number of terms:

$${}^G\tilde{\mathbf{p}}_I(t_n) \approx \sum_{i=0}^{l_p} \frac{(nt_r)^i}{N^i i!}\,{}^G\tilde{\mathbf{p}}_I^{(i)}(t_o) \tag{7}$$

$$\tilde{\boldsymbol{\theta}}_I(t_n) \approx \sum_{i=0}^{l_\theta} \frac{(nt_r)^i}{N^i i!}\,\tilde{\boldsymbol{\theta}}_I^{(i)}(t_o) \tag{8}$$

where l_p and l_θ are the chosen truncation orders for the position and orientation, respectively. Substitution in (4) yields:

$$\mathbf{r} \approx \sum_{i=0}^{l_\theta} \frac{(nt_r)^i\mathbf{H}_{\boldsymbol{\theta}}}{N^i i!}\,\tilde{\boldsymbol{\theta}}_I^{(i)}(t_o) + \sum_{i=0}^{l_p} \frac{(nt_r)^i\mathbf{H}_{\mathbf{p}}}{N^i i!}\,{}^G\tilde{\mathbf{p}}_I^{(i)}(t_o) + \mathbf{H}_a\tilde{\mathbf{x}}_a + \mathbf{n} \tag{9}$$

This equation expresses the residual as a function of the errors in the first l_θ derivatives of the orientation and the first l_p derivatives of the position. Therefore, if we include these quantities in the state vector of the estimator, we can perform an update based on the linearized expression in (9). However, this will only be useful if l_θ and l_p are small.

Clearly, any choice of truncation order in (7)-(8) will lead to an unmodeled error, and the lower the truncation order, the more significant the error will be in general. The key observation here is that, since we have prior knowledge about the magnitude of the estimation errors, we can *predict* the worst-case unmodeled error incurred by our choice of l_θ and l_p. To evaluate the importance of these unmodeled errors, we analyze the impact that they have on the residual. If the residual term due to the unmodeled truncation errors is small, compared to the measurement noise, this would indicate that the loss of modeling accuracy would be acceptable.

We start this analysis by re-writing (7)-(8) to illustrate the physical interpretation of the first few terms on the right-hand side of the series:

$$ {}^{G}\tilde{\mathbf{p}}_I(t_n) = {}^{G}\tilde{\mathbf{p}}_I(t_o) + \frac{nt_r}{N}\,{}^{G}\tilde{\mathbf{v}}_I(t_o) + \frac{(nt_r)^2}{2N^2}\,{}^{G}\tilde{\mathbf{a}}_I(t_o) + \ldots \tag{10} $$

$$ \tilde{\boldsymbol{\theta}}_I(t_n) = \tilde{\boldsymbol{\theta}}_I(t_o) + \frac{nt_r}{N}\,{}^{G}\tilde{\boldsymbol{\omega}}(t_o) + \ldots \tag{11} $$

Here ${}^{G}\tilde{\mathbf{v}}_I$ represents the error in the estimate of the IMU velocity, ${}^{G}\tilde{\mathbf{a}}_I$ represents the error in the IMU acceleration, and ${}^{G}\tilde{\boldsymbol{\omega}}$ is the error in the rotational velocity expressed in the global frame (see Appendix II for the derivation of this term).

Let us first focus on the position errors. If we only keep the position and velocity terms in the series (i.e., $l_p = 1$), then the truncation error in (10) is given by

$$ \Delta\mathbf{p} = \frac{(nt_r)^2}{2N^2}\begin{bmatrix} {}^{G}\tilde{\mathbf{a}}_{I_x}(\tau_1) \\ {}^{G}\tilde{\mathbf{a}}_{I_y}(\tau_2) \\ {}^{G}\tilde{\mathbf{a}}_{I_z}(\tau_3) \end{bmatrix} \tag{12} $$

where $\tau_i \in [t_o, t_n]$, $i = 1, 2, 3$. The unmodeled term in the residual in (9), due to this truncation error, is given by $\mathbf{H_p}\Delta\mathbf{p}$. If the worst-case acceleration error in each direction is ϵ_a, the 2-norm of the truncation error is bounded above by $||\Delta\mathbf{p}||_2 \leq \sqrt{3}\frac{(nt_r)^2}{2N^2}\epsilon_a$, and thus the unmodeled term in the residual satisfies:

$$ \begin{aligned} \delta_p(n) &= ||\mathbf{H_p}\Delta\mathbf{p}||_2 \\ &\leq ||\mathbf{H_p}||_2 ||\Delta\mathbf{p}||_2 \\ &\leq \sqrt{3}\,H_{\mathbf{p}_u}\frac{(nt_r)^2}{2N^2} \end{aligned} $$

where $H_{\mathbf{p}_u}$ is an upper bound on $||\mathbf{H_p}||_2$. By choosing $n = \pm N/2$ we can compute the upper bound on the magnitude of the unmodeled residuals in the entire image (all n), as:

$$ \bar{\delta}_p = \frac{\sqrt{3}}{8}H_{\mathbf{p}_u}t_r^2\epsilon_a \tag{13} $$

Turning to the representation of the orientation errors, if we only maintain a *single* term in the series (i.e., $l_\theta = 0$), we similarly derive the following upper bound for the unmodeled residual term:

$$ \delta_\theta(n) \leq \sqrt{3}H_{\boldsymbol{\theta}_u}\frac{|n|t_r}{N}\epsilon_\omega \tag{14} $$

where $H_{\boldsymbol{\theta}_u}$ is an upper bound on $||\mathbf{H}_{\boldsymbol{\theta}}||_2$, and ϵ_ω is the upper bound on the rotational velocity errors. In turn, the upper bound over all rows is given by:

$$ \bar{\delta}_\theta = \frac{\sqrt{3}}{2}H_{\boldsymbol{\theta}_u}t_r\epsilon_\omega \tag{15} $$

We have therefore shown that if (i) we include in the state vector of the estimator the IMU position, orientation, and velocity at t_o, (ii) compute the measurement residual as shown in (3), and (iii) base the estimator's update equations on the linearized expression:

$$ \mathbf{r} \approx \mathbf{H}_{\boldsymbol{\theta}}\tilde{\boldsymbol{\theta}}_I(t_o) + \mathbf{H_p}\,{}^{G}\tilde{\mathbf{p}}_I(t_o) + \frac{nt_r}{N}\mathbf{H_p}\,{}^{G}\tilde{\mathbf{v}}_I(t_o) + \mathbf{H}_a\tilde{\mathbf{x}}_a + \mathbf{n} \tag{16} $$

we are *guaranteed* that the residual terms due to unmodeled errors will be upper bounded by $\bar{\delta}_\theta + \bar{\delta}_p$. The value of this bound will depend on the characteristics of the sensors used, but in any case it can be evaluated to determine whether this choice of truncation orders would be acceptable.

For example, for the sensors on the LG Nexus 4 smartphone used in our experiments (see Table III), the standard deviation of the noise in the acceleration measurements is approximately 0.04 m/s^2. Using a conservative value of $\epsilon_a = 1$ m/s^2 (to also account for errors in the estimates of the accelerometer bias and in roll and pitch), a readout time of $t_r = 43.3$ ms, and assuming a camera with a focal length of 500 pixels and 60-degree field of view observing features at a depth of 2 m, we obtain $\bar{\delta}_p = 0.12$ pixels. Similarly, using $\epsilon_\omega = 1$ ^{o}/s, we obtain $\bar{\delta}_\theta = 0.44$ pixels (see Appendix I for the details of the derivations). We therefore see that, for our system, the residual terms due to unmodeled errors when using (16) are guaranteed to be below 0.56 pixels. This (conservative) value is smaller than the standard deviation of the measurement noise, and likely in the same order as other sources of unmodeled residual terms (e.g., camera-model inaccuracies and the nonlinearity of the measurement function).

The above discussion shows that, for a system with sensor characteristics similar to the ones described above, the choice of $l_p = 1$, $l_\theta = 0$ leads to approximation errors that are guaranteed to be small. If for a given system this choice is not sufficient, more terms can be kept in the two series to achieve a more precise modeling of the error. On the other hand, we can be even more aggressive, by choosing $l_p = 0$, i.e., keeping only the camera pose in the estimator state vector, and not computing Jacobians of the error with respect to the velocity. In that case, the upper bound of the residual due to unmodeled position errors becomes:

$$ \delta'_p(n) \leq \sqrt{3}H_{\mathbf{p}_u}\frac{|n|t_r}{N}\epsilon_v \tag{17} $$

where ϵ_v is the worst-case velocity error. Using a conservative value of $\epsilon_v = 20$ cm/s (larger than what we typically observe), we obtain $\bar{\delta}'_p = 2.2$ pixels.

If these unmodeled effects were materialized, the estimator's accuracy would likely be reduced. However, we have experimentally found that the performance loss by choosing $l_p = 0$ is minimal (see Section V-A), indicating that the computed bound is a conservative one. Moreover, as shown in the results of Section V-A, including additional terms for the orientation error does not lead to a substantially improved performance. Therefore, in our implementations, we have favored two sets of choices: $l_p = 1, l_\theta = 0$, due to the theoretical guarantee of small unmodeled errors, and $l_p = 0, l_\theta = 0$, due to its lower computational cost, as discussed next.

A. Discussion

It is interesting to examine the computational cost of the proposed method for processing the RS measurements, as compared to the case where a GS camera is used. The key observation here is that, if the states with respect to which Jacobians are evaluated are already part of the state, then *no additional states* need to be included in the estimator's state vector, to allow processing the RS measurements. In this case, the computational overhead from the use of a RS camera, compared to a GS one, will be negligible[3].

To examine the effect of specific choices of l_p and l_θ, we first note that all high-precision vision-aided inertial navigation methods maintain a state vector containing at a minimum the current IMU position, orientation, velocity, and biases (Mourikis et al., 2009; Jones and Soatto, 2011; Kelly and Sukhatme, 2011; Weiss et al., 2012; Kottas et al., 2012; Li and Mourikis, 2013b). Therefore, if the measurement Jacobians are computed with respect to the *current* IMU state (e.g., as in EKF-SLAM), choosing $l_p \leq 1$ and $l_\theta = 0$ will require no new states to be added, and no significant overhead.

On the other hand, in several types of methods, Jacobians are also computed with respect to "old" states (e.g., in sliding-window methods or batch offline minimization). When a GS camera is used, these old states often only need to contain the position and orientation, while other quantities can be marginalized out. Therefore, if the proposed RS model is used, and we select $l_p \geq 1, l_\theta \geq 1$, additional states will have to be maintained in the estimator, leading to increased computational requirements. However, if $l_p = l_\theta = 0$ is chosen, once again no additional states would have to be introduced, and the cost of processing the RS measurements would be practically identical to that of a GS camera (see also Section V-A).

We next discuss the relationship of our approach to the TBF formulation of (Oth et al., 2013; Lovegrove et al., 2013; Furgale et al., 2012; Bosse and Zlot, 2009). First, we point out that the expressions in (7)-(8) effectively describe a representation of the errors in terms of the temporal basis functions $f_i(\tau) = \tau^i$, $i = 1, \ldots l_{p/\theta}$ in the time interval $[-t_r/2, t_r/2]$. This is similar to the TBF formulation, with the difference that in our case the *errors*, rather than the *states* are approximated by a low-dimensional parameterization. This difference has two key consequences. First, as we saw it is possible to use knowledge of the error properties to compute bounds on the effects of the unmodeled errors. Second, the errors are, to a large extent, *independent* of the actual trajectory, which makes the approach applicable in cases where the motion contains significant accelerations. By contrast, in the TBF formulation the necessary number of basis functions is crucially dependent on the nature of the trajectory. In "smooth" trajectories, one can use a relatively small number of functions, leading to low computational cost (see, e.g. (Oth et al., 2013)). However, with fast motion dynamics, the proposed error-parameterization approach requires a lower dimension of the state vector.

To demonstrate this with a concrete example, let us focus on the acceleration signal shown in Fig. 2(left). In (Oth et al., 2013; Lovegrove et al., 2013), fourth-order B-splines are used as the basis functions, due to their finite support and analytical derivatives. This, in turn, means that the acceleration is modeled by a linear function between the time instants consecutive knots are placed. If we place one knot every 43.3 ms (e.g., at the start and end of each image readout), we would be modeling the acceleration as a linear function during each image readout. The bottom plot in Fig. 2(right) shows the largest errors between $\mathbf{a}_m$ and the best-fit linear model during each readout time. The worst-case error is 1.03 m/s^2, which is one order of magnitude larger than the standard deviation of the accelerometer measurement noise (see Table III). Therefore, placing knots every 43.3 ms would lead to unacceptably large unmodeled terms in the accelerometer residual. To reduce the error terms to the same order of magnitude as the noise, knots would have to be placed every approximately 15 ms, or approximately three poses per image. Therefore, in this example (which involves motion dynamics common in small-scale systems) the proposed error-parameterization approach would lead to a significantly faster algorithm, due to the smaller dimension of the state vector.

IV. Motion Estimation with an IMU and a Rolling-shutter Camera

Our interest is in motion estimation in unknown, uninstrumented environments, and therefore we assume that the camera observes naturally-occurring visual features, whose positions are not known *a priori*. These measurements are processed by an EKF-based method, whose formulation is based on (Li and Mourikis, 2012a). This is a hybrid estimator that combines a sliding-window filter formulation with a feature-based one, to minimize the computational cost of EKF updates. In what follows, we briefly describe the estimator, but since the EKF algorithm is not the main contribution of this work, we refer the reader to (Li and Mourikis, 2012a) for more details.

A. EKF state vector

The hybrid estimator proposed in (Li and Mourikis, 2012a) maintains a state vector comprising the current IMU state, a sliding window of states, as well as a number of feature points. In addition to these quantities, we here include in the state vector the *spatial* and *temporal* calibration parameters between the camera and IMU. First, we include in the estimated state vector the spatial transformation between the IMU frame and the camera frame, described by the unit quaternion ${}^C_I\bar{\mathbf{q}}$ and the translation vector ${}^C\mathbf{p}_I$. This transformation is known to be observable under general motion, and including it in the estimator removes the need for an offline calibration procedure. Second, we include in the state vector the time offset, t_d, between the timestamps of the

[3] A small increase will occur, due to the IMU propagation needed to compute the estimates ${}^G\hat{\mathbf{p}}_I(t_n)$ and ${}^I_G\hat{\bar{\mathbf{q}}}(t_n)$, $n = -N/2, \ldots, N/2$. However, this cost is negligible, compared to the cost of matrix operations in the estimator.

camera and the IMU. Time-offsets between different sensors' reported timestamps exist in most systems (e.g., due to delays in the sensors' data paths), but they can be especially significant in the low-cost systems we are interested in. Performing online temporal calibration makes it possible to account for the uncertainty in the sensor timestamps, and compensate for the time offset, in a simple way (Li and Mourikis, 2013a).

Therefore, the EKF's state vector is defined as

$$\mathbf{x}(t) = \begin{bmatrix}\mathbf{x}_E^T(t) & \boldsymbol{\pi}_{IC}^T & t_d & \mathbf{x}_{I_1}^T & \cdots & \mathbf{x}_{I_m}^T & \mathbf{f}_1^T & \cdots & \mathbf{f}_{s_t}^T\end{bmatrix}^T \tag{18}$$

where $\mathbf{x}_E(t)$ is the current ("evolving") IMU state at time t, the camera-to-IMU transformation is given by $\boldsymbol{\pi}_{IC} = [{}^C_I\bar{\mathbf{q}}^T \quad {}^C\mathbf{p}_I^T]^T$, $\mathbf{x}_{I_i}$, $i = 1, \ldots, m$ are the IMU states corresponding to the time instants the last m images were recorded, and $\mathbf{f}_j$, $j = 1, \ldots, s_t$ are feature points, represented by an inverse-depth parameterization (Montiel et al., 2006). As explained in Section III, each of the IMU states $\mathbf{x}_{I_i}$ comprises the camera pose, and potentially its derivatives, at the middle of the image readout period.

In the hybrid EKF, when an IMU measurement is received, it is used to propagate the evolving state and covariance. On the other hand, when a new image is received, the sliding window of states is augmented. The images are processed to extract and match point features, and these are processed in one of two ways: if a feature's track is lost after m or fewer images, it is used to provide constraints involving the poses of the sliding window. On the other hand, if a feature is still being tracked after m frames, it is initialized in the state vector and any subsequent observations of it are used for updates as in the EKF-SLAM paradigm. At the end of the update, features that are no longer visible and old sliding-window states with no active feature tracks are removed.

In what follows, we describe each of these steps in more detail.

B. EKF propagation

Following standard practice, we define the evolving IMU state as the 16×1 vector:

$$\mathbf{x}_E = \begin{bmatrix}{}^I_G\bar{\mathbf{q}}^T & {}^G\mathbf{p}_I^T & {}^G\mathbf{v}_I^T & \mathbf{b}_\mathbf{g}^T & \mathbf{b}_\mathbf{a}^T\end{bmatrix}^T \tag{19}$$

where $\mathbf{b}_\mathbf{g}$ and $\mathbf{b}_\mathbf{a}$ are the IMU's gyroscope and accelerometer biases, modeled as random walk processes:

$$\dot{\mathbf{b}}_\mathbf{g} = \mathbf{n}_{wg}, \quad \dot{\mathbf{b}}_\mathbf{a} = \mathbf{n}_{wa} \tag{20}$$

In the above equations, $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$ represent Gaussian noise vectors with autocorrelation functions $\sigma_{wg}^2\mathbf{I}_3\delta(t_1 - t_2)$ and $\sigma_{wa}^2\mathbf{I}_3\delta(t_1 - t_2)$, respectively. Moreover, the error-state vector for the IMU state is defined as:

$$\tilde{\mathbf{x}}_E = \begin{bmatrix}\tilde{\boldsymbol{\theta}}_I^T & {}^G\tilde{\mathbf{p}}_I^T & {}^G\tilde{\mathbf{v}}_I^T & \tilde{\mathbf{b}}_\mathbf{g}^T & \tilde{\mathbf{b}}_\mathbf{a}^T\end{bmatrix}^T \tag{21}$$

where for the position, velocity, and bias states the standard additive error definition has been used (e.g., ${}^G\mathbf{v}_I = {}^G\hat{\mathbf{v}}_I + {}^G\tilde{\mathbf{v}}_I$). On the other hand, for the orientation errors we use a minimal 3-dimensional representation, defined by the equations (Li and Mourikis, 2013b):

$${}^I_G\bar{\mathbf{q}} \approx {}^I_G\hat{\bar{\mathbf{q}}} \otimes \begin{bmatrix}\frac{1}{2}\tilde{\boldsymbol{\theta}}_I \\ 1\end{bmatrix} \tag{22}$$

In the EKF, the IMU measurements are used to propagate the evolving state estimates as described in (Li and Mourikis, 2013b). Specifically, the gyroscope and accelerometer measurements are modeled respectively by the equations:

$$\boldsymbol{\omega}_m(t) = {}^I\boldsymbol{\omega}(t) + \mathbf{b}_\mathbf{g}(t) + \mathbf{n}_\mathbf{r}(t) \tag{23}$$

$$\mathbf{a}_m(t) = {}^I_G\mathbf{R}(t)\left({}^G\mathbf{a}(t) - {}^G\mathbf{g}\right) + \mathbf{b}_\mathbf{a}(t) + \mathbf{n}_\mathbf{a}(t) \tag{24}$$

where ${}^I\boldsymbol{\omega}$ is the IMU's rotational velocity, ${}^G\mathbf{g}$ is the gravitational acceleration, and $\mathbf{n}_\mathbf{r}$ and $\mathbf{n}_\mathbf{a}$ are zero-mean white Gaussian noise processes with standard deviations σ_r and σ_a on each sensing axis, respectively. The IMU orientation is propagated from time instant t_k to t_{k+1}, by numerically integrating the differential equation:

$${}^I_G\dot{\hat{\bar{\mathbf{q}}}}(t) = \frac{1}{2}\boldsymbol{\Omega}\left(\boldsymbol{\omega}_m(t) - \hat{\mathbf{b}}_\mathbf{g}(t_k)\right){}^I_G\hat{\bar{\mathbf{q}}}(t),$$

in the interval $t \in [t_k, t_{k+1}]$, assuming that $\boldsymbol{\omega}_m(t)$ is changing linearly between the samples received from the IMU at t_k and t_{k+1}. In the above, the matrix $\boldsymbol{\Omega}(\cdot)$ is defined as:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix}\lfloor\boldsymbol{\omega}\times\rfloor & \boldsymbol{\omega} \\ \boldsymbol{\omega}^T & 0\end{bmatrix}$$

The velocity and position estimates are propagated by:

$${}^G\hat{\mathbf{v}}_{k+1} = {}^G\hat{\mathbf{v}}_k + {}^G_I\hat{\mathbf{R}}(t_k)\,\hat{\mathbf{s}}_k + {}^G\mathbf{g}\Delta t \tag{25}$$

$${}^G\hat{\mathbf{p}}_{k+1} = {}^G\hat{\mathbf{p}}_k + {}^G\hat{\mathbf{v}}_k\Delta t + {}^G_I\hat{\mathbf{R}}(t_k)\,\hat{\mathbf{y}}_k + \frac{1}{2}{}^G\mathbf{g}\Delta t^2 \tag{26}$$

where $\Delta t = t_{k+1} - t_k$, and

$$\hat{\mathbf{s}}_k = \int_{t_k}^{t_{k+1}} {}^{I_k}_{I}\hat{\mathbf{R}}(\tau)\Big(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_\mathbf{a}(t_k)\Big)d\tau \tag{27}$$

$$\hat{\mathbf{y}}_k = \int_{t_k}^{t_{k+1}}\int_{t_k}^{s} {}^{I_k}_{I}\hat{\mathbf{R}}(\tau)\Big(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_\mathbf{a}(t_k)\Big)d\tau ds \tag{28}$$

The above integrals are computed using Simpson integration, assuming a linearly changing $\mathbf{a}_m$ in the interval $[t_k, t_{k+1}]$. Besides the IMU position, velocity, and orientation, all other state estimates remain unchanged during propagation. We point out that, in addition to EKF propagation, the above equations are employed for propagating the state within each readout-time interval for using the RS measurements, as described in Section III.

When EKF propagation is performed, in addition to the state estimate, the state covariance matrix is also propagated, as follows:

$$\mathbf{P}(t_{k+1}) = \boldsymbol{\Phi}(t_{k+1}, t_k)\mathbf{P}(t_k)\boldsymbol{\Phi}(t_{k+1}, t_k)^T + \mathbf{Q}_d$$

where $\mathbf{P}$ is the state covariance matrix, $\mathbf{Q}_d$ is the covariance matrix of the process noise, and $\boldsymbol{\Phi}(t_{k+1}, t_k)$ is the error-state transition matrix, given by:

$$\boldsymbol{\Phi}(t_{k+1}, t_k) = \begin{bmatrix}\boldsymbol{\Phi}_I(t_{k+1}, t_k) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}\end{bmatrix} \tag{29}$$

with $\mathbf{\Phi}_I(t_{k+1}, t_k)$ being the 15×15 error-state transition matrix for the IMU state, derived in (Li and Mourikis, 2013b).

C. State augmentation

When a new image is received, a new state must be added to the filter state vector. Let us consider the case where an image with timestamp t is received. We here assume that, by, convention, the image timestamps correspond to the midpoint of the image readout. However, recall that a time offset exists between the camera and IMU timestamps. Due to this offset, if an image with timestamp t is received, the midpoint of the image readout interval was actually at time $t+t_d$. Therefore, when a new image is received, the state is augmented with an estimate of the IMU state at $t + t_d$ (Li and Mourikis, 2013a). If a truncation order $l_p = 1$ is used, this state will comprise the IMU position, orientation, and velocity, while if $l_p = 0$, only the position and orientation are included in the state. In what follows, we present the state augmentation (and update equations) for the more general case of $l_p = 1$.

When the image is received, we propagate the EKF state up to $t + \hat{t}_d$, at which point we augment the state with the estimate $\hat{\mathbf{x}}_{I_m} = [{}^I_G\hat{\bar{\mathbf{q}}}^T(t+\hat{t}_d) \quad {}^G\hat{\mathbf{p}}_I^T(t+\hat{t}_d) \quad {}^G\hat{\mathbf{v}}_I^T(t+\hat{t}_d)]^T$. Moreover, the EKF covariance matrix is augmented to include the covariance matrix of the new state, and its correlation to all other states in the system. For this computation, an expression relating the errors in the new state to the errors of the EKF state vector is needed. This is given by:

$$\tilde{\mathbf{x}}_{I_m} = \begin{bmatrix}\mathbf{I}_9 & \mathbf{0}_{9\times 12} & \mathbf{J}_t & \mathbf{0}\end{bmatrix} \tilde{\mathbf{x}}(t+\hat{t}_d)$$

where $\mathbf{J}_t$ is the Jacobian with respect to the time offset t_d. This Jacobian, which expresses the uncertainty in the precise time instant the image was recorded, can be computed by direct differentiation of the orientation, position, and velocity, as:

$$\mathbf{J}_t = \begin{bmatrix} {}^G_I\hat{\mathbf{R}}(t+\hat{t}_d)\left({}^I\boldsymbol{\omega}_m(t+\hat{t}_d) - \hat{\mathbf{b}}_{\mathbf{g}}(t+\hat{t}_d)\right) \\ {}^G\hat{\mathbf{v}}_I(t+\hat{t}_d) \\ {}^G_I\hat{\mathbf{R}}(t+\hat{t}_d)\left(\mathbf{a}_m(t+\hat{t}_d) - \hat{\mathbf{b}}_{\mathbf{a}}(t+\hat{t}_d)\right) + {}^G\mathbf{g} \end{bmatrix} \tag{30}$$

D. EKF Update

Once state augmentation is performed, the image is processed to extract and match features. These feature measurements are processed in one of two different ways, depending on their track lengths. Specifically, the majority of features that we detect in the images can only be tracked for a small number of frames. Those features whose tracks are complete in m or fewer frames are processed without being included in the EKF state vector, by use of the multi-state-constraint Kalman filter (MSCKF) approach (Li and Mourikis, 2013b; Mourikis and Roumeliotis, 2007). On the other hand, features that are still actively being tracked after m images, are included in the EKF state vector, and their measurements are processed as in EKF-SLAM methods.

We briefly describe the two approaches, starting with the MSCKF. Let us consider the case where feature $\mathbf{f}_i$ has been observed in ℓ images, and has just been lost from tracking (e.g., it went out of the field of view). At this time, the MSCKF uses all the measurements of the feature. First, the measurements are used to compute an estimate of the feature state, via least-squares triangulation. This estimate is used, along with the estimates from the EKF's sliding window, to compute the residuals:

$$\mathbf{r}_{ij} = \mathbf{z}_{ij} - \mathbf{h}(\hat{\mathbf{x}}_{I_j}, \hat{\boldsymbol{\pi}}_{IC}, \hat{\mathbf{f}}_i) \tag{31}$$

where the index j ranges over all states from which the feature was observed. The above residual computation utilizes the state estimate $\hat{\mathbf{x}}_{I_j}$, as well as the IMU measurements in the readout time interval of the corresponding image, as described in Section III. Linearizing the above residual, we obtain:

$$\begin{aligned}\mathbf{r}_{ij} \approx \mathbf{H}_{\boldsymbol{\theta}_{ij}}\tilde{\boldsymbol{\theta}}_{I_j} + \mathbf{H}_{\mathbf{p}_{ij}}{}^G\tilde{\mathbf{p}}_{I_j} + \frac{n_{ij}t_r\mathbf{H}_{\mathbf{p}_{ij}}}{N}{}^G\tilde{\mathbf{v}}_{I_j} + \mathbf{H}_{IC_{ij}}\tilde{\boldsymbol{\pi}}_{IC} \\ + \mathbf{H}_{\mathbf{f}_{ij}}\tilde{\mathbf{f}}_i + \mathbf{n}_{ij} \quad (32) \\ = \mathbf{H}_{ij}\tilde{\mathbf{x}} + \mathbf{H}_{\mathbf{f}_{ij}}\tilde{\mathbf{f}}_i + \mathbf{n}_{ij} \quad (33)\end{aligned}$$

In the above equations, $\mathbf{H}_{IC_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$ are the Jacobians of the measurement function with respect to the camera pose and feature position, respectively, $\mathbf{n}_{ij}$ is the measurement noise, and n_{ij} is the image row on which $\mathbf{f}_i$ is observed in image j. Since the feature is not included in the MSCKF state vector, we proceed to marginalize it out. For this purpose, we first form the vector containing the ℓ residuals from all the feature's measurements:

$$\mathbf{r}_i \approx \mathbf{H}_i\tilde{\mathbf{x}} + \mathbf{H}_{\mathbf{f}_i}\tilde{\mathbf{f}}_i + \mathbf{n}_i \tag{34}$$

where $\mathbf{r}_i$ and $\mathbf{n}_i$ are $2\ell \times 1$ vectors formed by stacking the vectors $\mathbf{r}_{ij}$ and $\mathbf{n}_{ij}$, respectively, and $\mathbf{H}_i$ and $\mathbf{H}_{f_i}$ are the corresponding Jacobian matrices formed by stacking $\mathbf{H}_{ij}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$, respectively. Subsequently, a matrix $\mathbf{V}$, whose columns form a basis of the left nullspace of $\mathbf{H}_{\mathbf{f}_i}$, is used to multiply both sides of (34), leading to:

$$\mathbf{r}_i^o = \mathbf{V}^T\mathbf{r}_i = \mathbf{V}^T\mathbf{H}_i\tilde{\mathbf{x}} + \mathbf{V}^T\mathbf{n}_i = \mathbf{H}_i^o\tilde{\mathbf{x}} + \mathbf{n}_i^o \tag{35}$$

The above residual expresses the information that the observations of feature $\mathbf{f}_i$ provide for the EKF state vector. Prior to using the residual for an update, a Mahalanobis-distance test is performed, by comparing the quantity

$$\gamma_i = \mathbf{r}_i^o\left(\mathbf{H}_i^o\mathbf{P}\mathbf{H}_i^{o\,T} + \sigma_{\mathrm{im}}^2\mathbf{I}\right)^{-1}\mathbf{r}_i^o$$

to the 95-th percentile of the χ^2 distribution with $2\ell - 3$ degrees of freedom. Features whose γ_i values exceed this threshold are considered outliers and discarded from further processing.

The process described above is repeated for all the features whose tracks were completed in the latest image. In addition to these features, the hybrid filter processes the measurements of all features observed in the most recent image that are part of the EKF state vector. For this purpose, the measurement residuals are computed similarly to (31), with the difference that the feature position estimate is obtained from the EKF state vector, rather than from a minimization process. Moreover, the Jacobians needed for using these features in the EKF are computed as in (32). All the residuals from both types of features are employed for an EKF update, as described in (Li and Mourikis, 2012a). Additionally, to improve the estimation accuracy and consistency properties, we employ the first-estimate-Jacobian approach to ensure that the observability properties of the linearized system match that of the actual system. Specifically, all the position and velocity components in the state transition matrices and measurement Jacobian matrices are evaluated using their first estimates, as explained in (Li and Mourikis, 2013b).

V. Experiments

In this section we present the results from Monte-Carlo simulations and real-world experiments, which demonstrate the performance of the proposed approach for processing RS measurements.

A. Simulations

To obtain a realistic simulation environment, we generate the ground-truth trajectory and sensor measurements in the simulator based on a real-world dataset, collected by a Nexus 4 mobile device. The device is equipped with a RS camera capturing images at 11 Hz with a readout time of 43.3 ms, and an Invensense MPU-6050 IMU, which provides inertial measurements at 200 Hz. The noise characteristics and additional details for the sensors can be found in Table III. During the data collection, the device was hand-held by a person walking at normal pace, for a duration of 4.23 min. The total distance traveled is approximately 227 m.

To generate the ground truth trajectory for the simulations, we first processed the dataset with the proposed algorithm, to obtain estimates for the IMU state, $\hat{\mathbf{x}}_E$. In the ground-truth trajectory, the IMU poses (position and orientation) at the time instants at which images are available, τ_j, $j = 1, \ldots, n_1$, are identical to the computed estimates $\hat{\mathbf{x}}_E(\tau_j)$. To obtain the complete ground truth, we must additionally determine the IMU states for all the time instants t_i, $i = 1, \ldots n_2$, at which the IMU is sampled, as well as the rotational velocity and linear acceleration at these time instants (these are needed for generating IMU measurements in the simulator). To this end we formulate an optimization problem, to determine the acceleration and rotational velocity signals which will (i) guarantee that the IMU pose at the time instants τ_j is identical to the estimates $\hat{\mathbf{x}}_E(\tau_j)$, and (ii) minimize the difference between the ground-truth acceleration and rotational velocity and the corresponding estimates obtained from the actual dataset (see (37)-(38)).

To describe this minimization problem, let us denote the vector of ground-truth quantities we seek to determine at time t_i as

$$\mathbf{x}^\star_M(t_i) = \begin{bmatrix} {}^I_G\bar{\mathbf{q}}^\star(t_i) \\ {}^G\mathbf{p}^\star_I(t_i) \\ {}^G\mathbf{v}^\star_I(t_i) \\ {}^G\mathbf{a}^\star_I(t_i) \\ {}^I\boldsymbol{\omega}^\star(t_i) \end{bmatrix} = \begin{bmatrix} \mathbf{x}^\star_s(t_i) \\ {}^G\mathbf{a}^\star_I(t_i) \\ {}^I\boldsymbol{\omega}^\star(t_i) \end{bmatrix}$$

where $\mathbf{x}^\star_s(t_i) = [{}^I_G\bar{\mathbf{q}}^\star(t_i)^T \;\; {}^G\mathbf{p}^\star_I(t_i)^T \;\; {}^G\mathbf{v}^\star_I(t_i)^T]^T$. Moreover, we denote by $\boldsymbol{\phi}$ the IMU-state propagation function, computed by numerical integration assuming linearly-changing acceleration and rotational velocity in each IMU sample interval.

The ground truth is obtained by formulating a minimization problem for each interval between two consecutive image timestamps, τ_{j-1} and τ_j. Specifically, we obtain $\mathbf{x}^\star_M(t_i), t_i \in (\tau_{j-1}, \tau_j]$ by solving the constrained-minimization problem:

$$\begin{aligned} \text{min.} & \sum_{t_i \in (\tau_{j-1}, \tau_j]} \|{}^I\boldsymbol{\omega}^\star(t_i) - {}^I\hat{\boldsymbol{\omega}}(t_i)\|_2^2 + \|{}^G\mathbf{a}^\star_I(t_i) - {}^G\hat{\mathbf{a}}_I(t_i)\|_2^2 \\ \text{s. t.} \quad & \mathbf{x}^\star_s(t_i) = \boldsymbol{\phi}(\mathbf{x}^\star_s(t_{i-1}), {}^G\mathbf{a}^\star_I(t_{i-1}, t_i), {}^I\boldsymbol{\omega}^\star(t_{i-1}, t_i)) \\ & {}^G\mathbf{p}^\star_I(\tau_j) = {}^G\hat{\mathbf{p}}_I(\tau_j), \quad {}^I_G\bar{\mathbf{q}}^\star(\tau_j) = {}^I_G\hat{\bar{\mathbf{q}}}(\tau_j) \end{aligned} \tag{36}$$

where ${}^I\hat{\boldsymbol{\omega}}$ and ${}^G\hat{\mathbf{a}}_I$ are the rotational velocity and linear acceleration computed using the EKF estimates and the IMU measurements in the real dataset:

$${}^I\hat{\boldsymbol{\omega}}(t_i) = \boldsymbol{\omega}_m(t_i) - \hat{\mathbf{b}}_{\mathbf{g}}(t_i) \tag{37}$$

$${}^G\hat{\mathbf{a}}_I(t_i) = {}^I_G\hat{\mathbf{R}}^T(t_i)\left(\mathbf{a}_m(t_i) - \hat{\mathbf{b}}_{\mathbf{a}}(t_i)\right) + {}^G\mathbf{g} \tag{38}$$

The above problem is solved sequentially for each interval $(\tau_{j-1}.\tau_j]$. The resulting ground truth is self-consistent (in the sense that integrating the ground-truth acceleration yields the ground truth velocity, integrating the velocity yields position, and so on), and closely matches the estimates obtained in the actual dataset.

The ground-truth trajectory constructed as described above is used in all Monte-Carlo trials. In each trial, different independently-sampled realizations for the IMU biases and measurement noise, the feature positions, and image measurement noise, are used. Specifically, in each trial IMU biases are generated by integrating white-noise processes as shown in (20), and IMU measurements are subsequently computed via (23)-(24). The noise vectors used in each timestep in (20), (23), and (24) are independently-sampled from Gaussian distributions with characteristics identical to those of the MPU-6050 IMU. In each simulated image, we generate new features, whose number and feature-track length distribution is identical to the corresponding actual image. The 3D position of new features is randomly drawn in each trial. Specifically, the depth of each feature is chosen equal to the estimated depth of the corresponding actual feature, while the location of its image projection is randomly sampled from a uniform distribution. Finally, each image measurement is corrupted by independently-sampled Gaussian noise.

It is worth pointing out that for generating the measurements of the RS camera in the simulator, an iterative process is necessary. This is due to the fact that the exact time instant at which a feature is observed depends on the row on which it is projected (see (2)), and for general motion, it cannot be computed analytically. Therefore, in our implementation we initially compute the projection of the feature using (2) (without noise), and assuming $t_n = t_o$. Subsequently, we update t_n using (1) with $n = z_r - N/2$, and re-compute

TABLE II: Simulation Results: RMSE and NEES for different approaches

	GS method	Li et al. 2013	Proposed method			
Position error model	N/A	N/A	$l_p = 0$	$l_p = 1$	$l_p = 0$	$l_p = 1$
Orientation error model	N/A	N/A	$l_\theta = 0$	$l_\theta = 0$	$l_\theta = 1$	$l_\theta = 1$
Position RMSE (m)	5.403	0.806	0.471	0.447	0.447	**0.427**
Orientation RMSE (deg)	15.12	1.34	0.735	**0.693**	0.723	0.695
Velocity RMSE (m/s)	0.302	0.077	0.053	**0.052**	**0.052**	**0.052**
IMU state NEES	571.2	19.78	11.05	10.64	10.56	**10.44**
Time per update (ms)	**0.85**	2.17	0.87	1.49	1.50	2.24

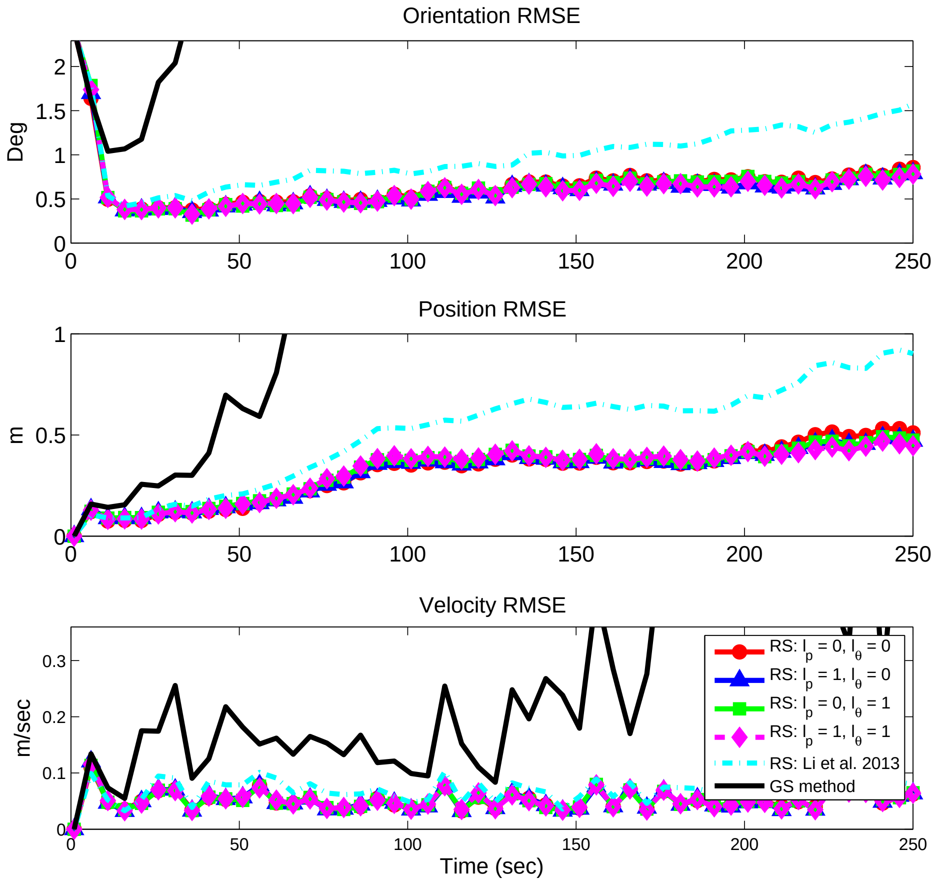

Fig. 3: Simulation results: The RMSE of the IMU orientation, position, and velocity over 50 Monte-Carlo simulation trials for different RS models. Because the performance of the four error-state models is very similar, the lines are hard to distinguish in the plot. For clarity, the numerical values of the average RMSE are provided in Table II.

the projection using the new estimate of t_n. This process is repeated until the estimate for t_n converges. Finally, measurement noise is added by sampling from a zero-mean Gaussian pdf with covariance matrix $\sigma_{\mathrm{im}}^2 \mathbf{I}_2$.

We here compare the performance of our proposed approach to the processing of RS measurements, with four different options for the modeling of the errors during the readout time, obtained by choosing $l_p = \{0, 1\}$ and $l_\theta = \{0, 1\}$. We additionally evaluate the performance of the constant-velocity approach of (Li et al., 2013), and of an approach that treats the camera as if it has a global shutter. To collect statistics for these six approaches, we carried out 50 Monte-Carlo simulation trials. Moreover, in order to isolate the effects of the RS model used, in each simulation trial all the approaches use exactly the same initial state estimate and covariance matrix (the filter initialization presented in Section V-B), the same estimator structure (the hybrid filter presented in Section IV), and process exactly the same IMU and feature measurements.

Fig. 3 shows the root mean squared errors (RMSE) (averaged over the 50 Monte-Carlo trials) in the estimates of the IMU orientation, position, and velocity computed by the six approaches. In Table II we also provide the average RMSE and the average normalized estimation error squared (NEES) for the IMU motion state (position, orientation, and velocity), averaged over all 50 trials, and over the last 25 seconds of motion[4]. Examining the NEES gives an insight into the magnitude of the unmodeled errors. Specifically, if significant unmodeled errors exist, the covariance matrix reported by the EKF will be smaller than the covariance matrix of the actual errors (i.e., the estimator will be inconsistent (Bar-Shalom et al., 2001)), and the NEES will increase. In an ideal, consistent estimator, the expected value of the NEES is equal to the dimension of the error state for which the NEES is computed, i.e., 9 in our case. Thus, by examining the deviation of the average NEES from this value, we can evaluate the significance of the unmodeled errors.

Several observations can be made based on the results of Fig. 3 and Table II. First, we clearly observe that the approach that assumes a GS camera model has very poor performance, with errors that are one order of magnitude larger than all other methods. By assuming a GS camera, the motion of the camera during the readout time is neither modeled nor compensated for, which inevitably causes unmodeled errors and degrades the estimation accuracy. The existence of large unmodeled errors is also reflected in the average NEES value, which is substantially higher than that of all other methods.

From these results we can also see that the constant-velocity models used in (Li et al., 2013) result in significantly larger errors compared to the four models that are based on the proposed approach. Specifically, the position and orientation errors are approximately double, while the velocity errors are approximately 50% larger. As discussed in Section II, this is due to the fact that the motion in this experiment is characterized by significant variations, especially in the rotational velocity. These variations, which are to be expected in low-mass systems such as the one used here, cause the constant-velocity assumption to be severely violated, and lead to the introduction of non-negligible unmodeled errors. These errors also lead to an increase in the average NEES.

Turning to the four different error parameterizations that are based on the approach proposed here, we see that they all perform similarly in terms of accuracy. As expected, the choice $l_p = 1, l_\theta = 1$ outperforms the lower-order models, but only by a small margin. Moreover, we see that using a higher-order model leads to a lower NEES, as the unmodeled errors become smaller. The average NEES of the four different approaches is somewhat higher than the theoretically-expected value of 9. This outcome is to be anticipated, due not only to the approximations in the RS modeling, but also to the non-linear nature of the estimation problem. It is important to point out, however, that due to the use of the first-estimates-Jacobian approach of (Li and Mourikis, 2013b), all four methods' inconsistency is kept low.

TABLE III: Sensor characteristics of the mobile devices used in the experiments

Device	LG Nexus 4	Galaxy SIII	Galaxy SII
Gyroscope rate (Hz)	200	200	106
Accelerometer rate (Hz)	200	100	93
σ_r (^{o}/s)	$2.4 \cdot 10^{-1}$	$2.6 \cdot 10^{-1}$	$2.9 \cdot 10^{-1}$
σ_a (m/s^2)	$4.0 \cdot 10^{-2}$	$5.6 \cdot 10^{-2}$	$5.8 \cdot 10^{-2}$
σ_{wg} ($^o/\sqrt{\mathrm{s}^3}$)	$1.6 \cdot 10^{-3}$	$3.2 \cdot 10^{-3}$	$3.4 \cdot 10^{-3}$
σ_{wa} (m$/\sqrt{\mathrm{s}^5}$)	$7.0 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$
t_r (ms)	43.3	15.9	32.0
Frame rate (Hz)	11	20	15
Resolution (pixels)	432 × 576	480 × 640	480 × 640
σ_{im} (pixels)	0.75	0.75	0.75

Even though the accuracy and consistency of the four models is comparable, the computational cost incurred by their use is significantly different. Increasing the order of the error-model results in an increase in the dimension of the error-state vector, and therefore in a higher computational cost. The last row in Table II shows the average CPU time needed per update in each of the cases tested, measured on an Intel Core i3 2.13 GHz processor. These times show that, even though the accuracy difference between the most accurate error model ($l_p = l_\theta = 1$) and the least accurate one ($l_p = l_\theta = 0$) is less than 10%, the latter is more than 2.5 times faster. In fact, the computational cost of the method with $l_p = l_\theta = 0$ is practically identical to the cost of using a GS model, and 2.5 times faster than the model of (Li et al., 2013). Since our main interest is in resource-constrained systems, where CPU capabilities are limited and preservation of battery life is critical, our model of choice in the real-world experiments presented in the following section is $l_p = l_\theta = 0$. If additional precision were required, based on the results of Table II and the theoretical results of Section III, we would select the model with $l_p = 1, l_\theta = 0$.

B. Real world experiments

We now present results from real-world experiments, conducted on three commercially-available smartphone devices, namely a Samsung Galaxy SII, a Samsung Galaxy SIII, and an LG Nexus 4. While the proposed approach is applicable with any RS camera and IMU, the use with smartphones is of particular interest. These devices are small, inexpensive, widely available, and offer multiple sensing, processing, and communication capabilities. Therefore, they can serve as the basis for low-cost localization systems for diverse applications, which will be easy to disseminate to a large number of users with uniform hardware. The characteristics of the devices used in our experiments are shown in Table III. The noise characteristics of the IMU have been computed via offline calibration, by collecting datasets while keeping the devices stationary.

We here present results from using the smartphones to track i) a car driving on city streets, ii) a mobile robot in an outdoor setting, and iii) a person walking. In all cases, we demonstrate that the proposed approach results in high-precision estimates, and is able to do so in real time, running on the processor of the device. Videos show-

[4] Since no loop-closure information is available, the uncertainty of the position and the yaw gradually increases over time. We here plot the statistics for the last 25 seconds, to evaluate the estimators' final accuracy and consistency.

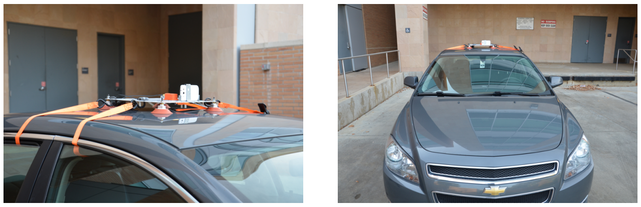

Fig. 4: Car experiment: Device setup

Fig. 5: Car experiment: Sample images

ing the images recorded by the camera in these experiments, as well as the trajectory estimates, can be found at `www.ee.ucr.edu/~mli/RollingShutterVIO`.

1) Real-world experiment I – Car localization: In this experiment, a Samsung Galaxy SIII mobile phone and a Xsens MGT-i unit (for GPS ground-truth data collection) was mounted on top of a car driving on the streets of Riverside, CA. Fig. 4 shows the setup of the devices. The total distance driven is approximately 11 km, covered in 21 min. Sample images from the experiment are shown in Fig. 5. During this experiment the sensor data were saved to disk, and later processed offline, to allow the comparison of the alternative methods. Image features are extracted by an optimized Shi-Tomasi feature extractor (Li and Mourikis, 2012b), and matched using normalized cross-correlation (NCC). A 17×17 image template is used for NCC, and a minimum matching threshold of 0.8.

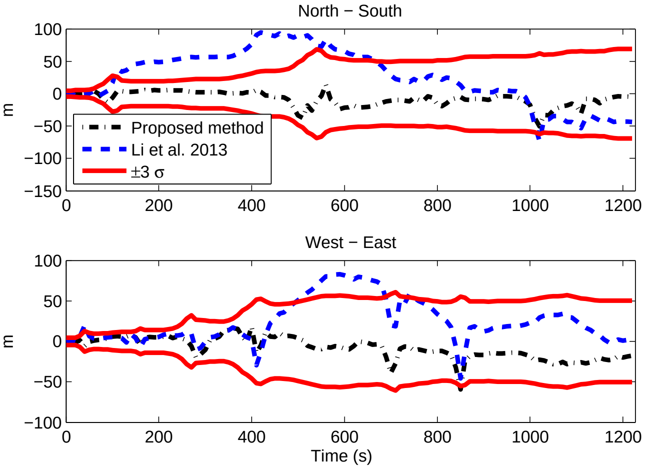

Fig. 7: Car experiment: Estimation errors for the two approaches compared.

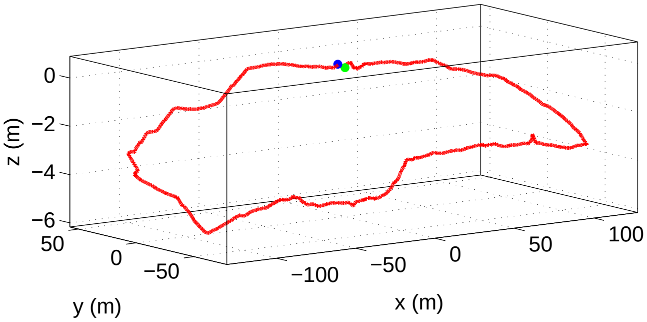

Fig. 8: Mobile robot experiment: 3D trajectory estimate. The green dot corresponds to the starting point, while the blue dot to the ending point.

For initializing the estimator, we require that during the first one second of the experiments the device is kept approximately stationary. This makes it possible to use a zero estimate for the initial velocity, and use the average accelerometer measurements during this time interval to estimate the initial roll and pitch. Since the estimates computed in this way may be somewhat inaccurate, we use conservative values for the initial standard deviations of the estimates, e.g., 0.1 m/s in each direction for the velocity (to account for any small motions that may take place), and $2\,^o$/s for roll and pitch. The estimates for the yaw and the position are initialized to zero, with zero covariance (i.e., we estimate motion with respect to the initial state). Moreover, for all remaining variables the estimates from the last successful experiment are used, with standard deviations of 0.5^o/s for the gyroscope biases, 0.1 m/s^2 for the accelerometer biases, 1^o for the camera-to-IMU rotation, 3 mm for the camera-to-IMU translation, and 4 msec for t_d.

Fig. 6 shows the trajectory estimate obtained by (i) the proposed method, (ii) the method of (Li et al., 2013), and (iii) a GPS system, which is treated as the ground truth. The estimated trajectories are manually aligned to the GPS ground truth, and plotted on a map of the area where the experiment took place. Moreover, Fig. 7 shows the position errors of the two approaches, as well as the reported uncertainty envelop for the proposed approach. This envelope corresponds to ± 3 standard deviations, computed as the square roots of the corresponding diagonal elements of the EKF's state covariance matrix. Here we only plot the position errors in the horizontal plane, as the accuracy of the ground truth in the vertical direction is not sufficiently high. From these results it becomes clear that the proposed method outperforms that of (Li et al., 2013) by a wide margin. It produces more accurate estimates (the largest position error throughout the experiment is approximately 63 m, compared to 123 m for (Li et al., 2013)), and the estimation errors agree with the estimator's reported uncertainty.

2) Real-world experiment II – Mobile robot localization: In this experiment, a ground robot moved in a closed-loop trajectory on the campus of the University of California, Riverside (UCR). For this experiment the Nexus 4 device was used, and the robot covered approximately 720 m in

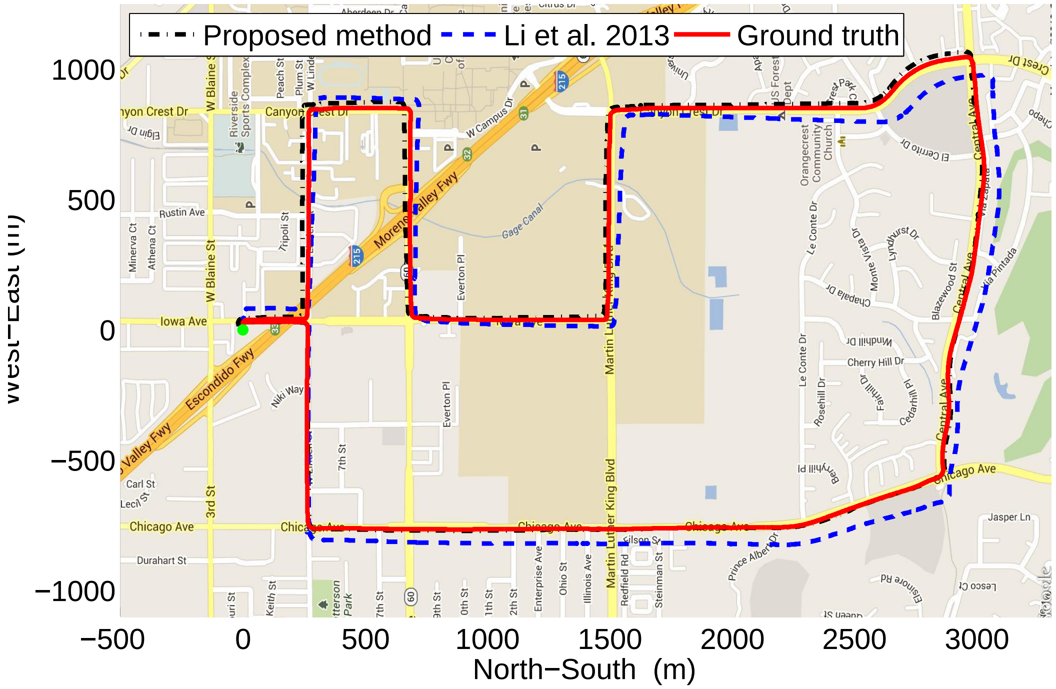

Fig. 6: Car experiment: Trajectory estimates by the proposed approach and the approach of (Li et al., 2013), compared to ground truth.

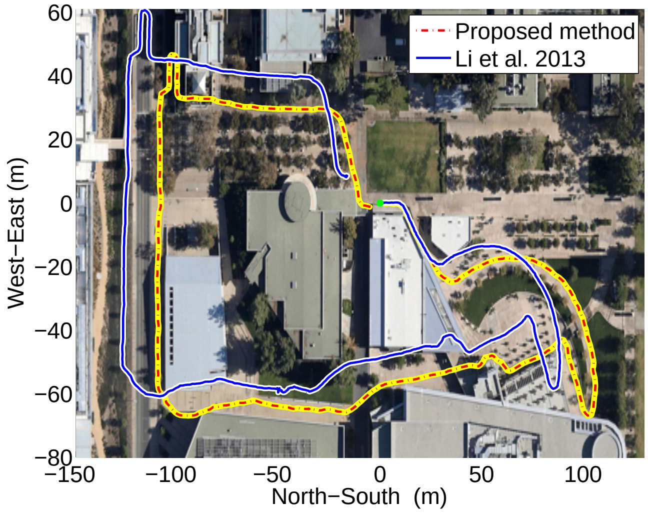

Fig. 9: Mobile robot experiment: Trajectory estimate on a satellite map. The red/yellow dashed line corresponds to the proposed approach, the blue/white solid line to the approach presented by (Li et al., 2013).

about 11.7 minutes. Similarly to the previous experiment, the data was stored for offline processing.

Fig. 8 shows the 3D-plot of the trajectory estimate computed by the proposed approach. This plot shows the significant elevation difference in the trajectory, and motivates the use of a visual-inertial localization method, as opposed to one based on 2D odometry. Since the robot was moving under trees and close to buildings, the GPS system was unable to report reliable ground truth. However, the robot started and ended at the same position, which allows us to calculate the final error of the trajectory estimate. For the proposed approach, the final position error along the three axes is $[5.33 \quad 1.15 \quad 0.2]$ m (0.73% of the traveled distance), while the corresponding 3σ reported by the filter are $[6.44 \;\; 4.67 \;\; 0.21]$ m. For the approach of (Li et al., 2013), the final position error is $[-16.49 \quad 8.76 \quad 0.41]$ m.

Fig. 9 shows the trajectory estimates of the two methods, overlaid on a satellite image of the area. Close inspection of this figure reveals that the result of (Li et al., 2013) is incorrect, while the proposed method generates an estimate that closely follows the actual path of the robot. It is worth noting that the superior performance of the proposed method (both in this and the previous experiment) is due to two main factors: the use of a more accurate RS model, but also the use of the hybrid EKF estimator, as opposed to the "pure" MSCKF method employed in (Li et al., 2013). This hybrid estimator makes it possible to better use features that are tracked for long periods, and thus reduces the rate of error accumulation.

3) Real-world experiment III – Indoor personal localization: In the previous experiments, the data was stored for offline processing, to facilitate comparison with the method of (Li et al., 2013). We now report the results from a "live" estimation experiment, conducted using the visual and inertial measurements on the Nexus 4 device. In this experiment, the device was held by a person walking indoors, on three floors of the UCR Science Library building. In Fig. 10 a sample image from this experiment is shown. The total distance covered was approximately 523 m, and

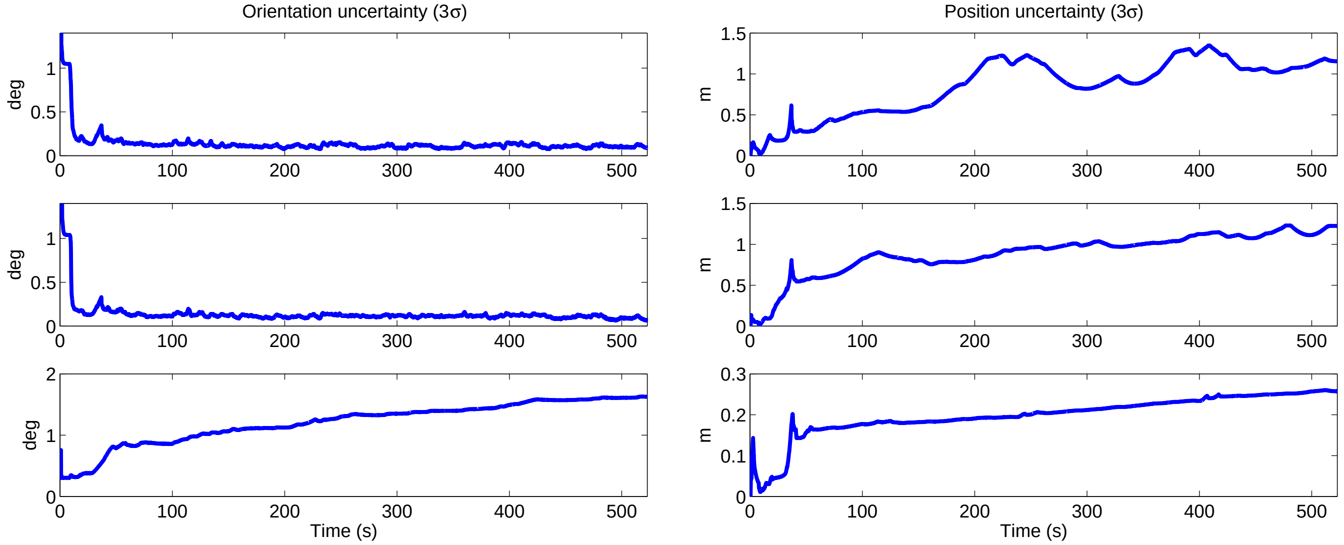

Fig. 12: Indoor personal-localization experiment: Reported uncertainty. The plot on the left shows the orientation uncertainty (3 standard deviations, computed from the corresponding diagonal elements of the EKF's state covariance matrix) about the x-y-z axes (roll-pitch-yaw), and the plot on the right shows the uncertainty in x-y-z position. The z axis corresponds to elevation, while the x and y axes to the errors in the horizontal plane.

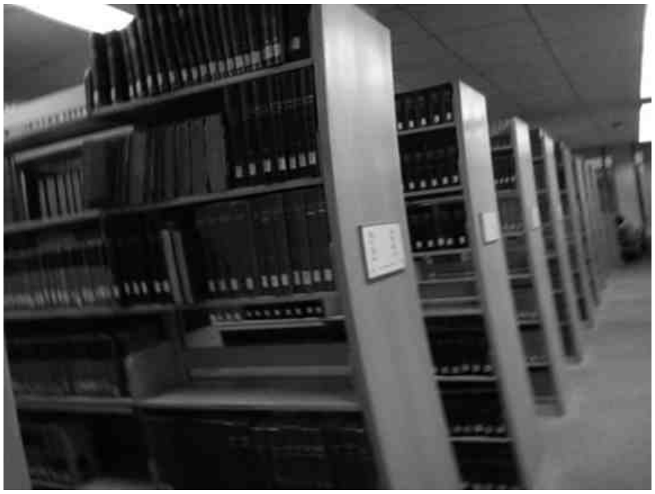

Fig. 10: A sample image from the indoor experiment, showing significant rolling-shutter distortion.

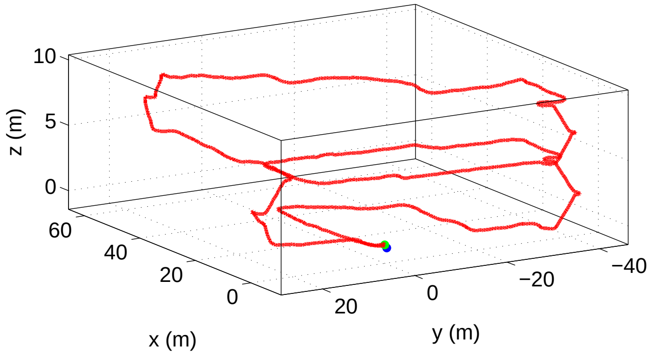

Fig. 11: Indoor personal-localization experiment: 3D trajectory estimate. The green dot corresponds to the starting point, while the blue dot to the ending point.

the duration of the experiment was 8.7 min. The data from the phone's camera and IMU were processed on-line on the device's processor, and the average time needed per EKF update was 13.6 ms, comfortably within the requirements for real-time operation.

Fig. 11 shows the trajectory estimate computed in real-time on the phone. Similarly to the two previous experiments, the trajectory starts and ends at the same point, which allows us to calculate the final position error. This error was $[-0.80 \quad -0.07 \quad -0.17]$ m, which only corresponds to approximately 0.16% of the trajectory length. Moreover, Fig. 12 shows the orientation and position uncertainty (3σ) reported by the hybrid EKF during this experiment. As expected, the orientation uncertainty about the x and y axes (roll and pitch) remains small throughout the trajectory, since the direction of gravity is observable in vision-aided inertial navigation (Martinelli, 2012; Jones and Soatto, 2011; Kelly and Sukhatme, 2011). On the other hand, the uncertainty in yaw and in the position gradually increases over time, since these quantities are unobservable and no loop-closure information is used. It is important to note that in this indoor environment the rate of uncertainty increase is substantially smaller than what was observed in the previous outdoor experiments (cf. Fig. 7). This is due to the smaller average feature depth, which results in larger baseline for the camera observations, and thus more information about the camera's motion.

4) Comparison on the datasets of (Li et al., 2013) : Finally, we report the results of our new method on the two datasets used in the experimental validation of (Li et al.,

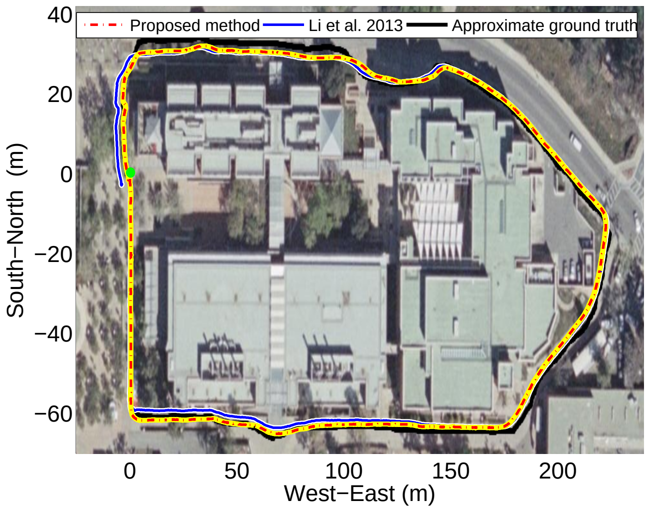

Fig. 13: Comparison on a dataset used in (Li et al., 2013): Trajectory estimates on a satellite map. The red/yellow dashed line corresponds to the proposed approach, the blue/white line to the approach presented by (Li et al., 2013), and the black solid line to the *approximate* ground truth.

2013). In these experiments a Samsung Galaxy SII device was hand-held, while a person walked in outdoor areas of the UCR campus and surrounding areas. The first dataset involves an approximately 900-m, 11-minute long trajectory, while the second one a 610-m, 8-minute long trajectory. Applying the proposed method on these two datasets yields a reduction of the final position error in both cases. Specifically, in the first dataset the final position error is reduced from 5.30 m to 3.22 m, while in the second one the error is reduced from 4.85 m to 2.31 m. To visualize the difference in the resulting estimates, in Fig. 13, we plot the results of the two methods in the second dataset, as well as the *approximate* ground-truth, overlaid manually on the image based on the locations of the walkable paths. We can observe that the method described here leads to an improvement in accuracy, especially visible in the final part of the trajectory.

VI. Conclusion

In this paper, we have presented a new method for combining measurements from a rolling-shutter camera and an IMU for motion estimation. The key idea in this approach is the use of the inertial measurements to model the camera motion during each image readout period. This removes the need for low-dimensional motion parameterizations, which can lead to loss of accuracy when the camera undergoes significant accelerations. Instead of a parameterization of the *trajectory*, our approach employs a parameterization of the *errors* in the trajectory estimates, when performing linearization. Using prior knowledge of the error characteristics, we can compute bounds on the unmodeled errors incurred by the representation, and address the tradeoff between the modeling accuracy and the computational cost in a principled manner. Our results demonstrate that the proposed approach achieves lower estimation errors than prior methods, and that its computational cost can be made almost identical to the cost of methods designed for global-shutter cameras. Moreover, we show that, coupled with a computationally-efficient EKF estimator, the proposed formulation for the RS measurements makes it possible to use low-cost consumer devices, to achieve high-performance 3D localization on a multitude of mobile platforms.

Appendix I

In this section, we show how the upper bounds on the unmodeled residual terms can be computed. We start by presenting the camera measurement model and calculating the measurement Jacobian matrices. Specifically, the camera measurement model (2) can be written as[5]:

$$\mathbf{z} = \begin{bmatrix} c_u + a_u\, u \\ c_v + a_v\, v \end{bmatrix} + \mathbf{n}, \qquad \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{{}^C z_f} \begin{bmatrix} {}^C x_f \\ {}^C y_f \end{bmatrix} \tag{39}$$

$${}^C\mathbf{p}_f(t_n) = \begin{bmatrix} {}^C x_f \\ {}^C y_f \\ {}^C z_f \end{bmatrix} = {}^C_I\mathbf{R}\,{}^I_G\mathbf{R}(t_n)\left({}^G\mathbf{p}_f - {}^G\mathbf{p}_I(t_n)\right) + {}^C\mathbf{p}_I \tag{40}$$

where ${}^C\mathbf{p}_f$ is the position of the feature expressed in the camera frame, (a_u, a_v) represent the camera focal length measured in horizontal and vertical pixels, and (c_u, c_v) is the principal point. By computing the partial derivatives of the measurement function $\mathbf{h}$ with respect to the IMU's position and orientation, we obtain the Jacobian matrices $\mathbf{H_p}$ and $\mathbf{H}_{\boldsymbol{\theta}}$, respectively:

$$\mathbf{H_p} = \mathbf{A}\,\mathbf{J}\,{}^C_I\mathbf{R}\,{}^I_G\hat{\mathbf{R}}(t_n) \tag{41}$$

$$\mathbf{H}_{\boldsymbol{\theta}} = \mathbf{A}\,\mathbf{J}\,{}^C_I\mathbf{R}\,{}^I_G\hat{\mathbf{R}}(t_n)\lfloor {}^G\mathbf{p}_f - {}^G\hat{\mathbf{p}}_I(t_n)\times\rfloor \tag{42}$$

where

$$\mathbf{A} = \begin{bmatrix} a_u & 0 \\ 0 & a_v \end{bmatrix}, \qquad \mathbf{J} = \frac{1}{{}^C\hat{z}_f}\begin{bmatrix} 1 & 0 & -\frac{{}^C\hat{x}_f}{{}^C\hat{z}_f} \\ 0 & 1 & -\frac{{}^C\hat{y}_f}{{}^C\hat{z}_f} \end{bmatrix} \tag{43}$$

To simplify the derivations, we here assume that the pixel aspect ratio is equal to one (a valid assumption for most cameras), i.e., $a_u = a_v$. With this simplification, the 2-norm of the position Jacobian matrix becomes:

$$||\mathbf{H_p}||_2 = ||\mathbf{A}\,\mathbf{J}\,{}^C_I\mathbf{R}\;{}^I_G\hat{\mathbf{R}}(t_n)||_2 \tag{44}$$

$$= a_u||\mathbf{J}||_2 = a_u\sqrt{\lambda_{max}(\mathbf{J}^T\mathbf{J})} \tag{45}$$

where we used the fact that unitary matrices (e.g., rotation matrices) preserve the 2-norm, and $\lambda_{max}(\mathbf{J}^T\mathbf{J})$ represents the largest eigenvalue of the matrix $\mathbf{J}^T\mathbf{J}$. This can be computed analytically as:

$$\lambda_{max}(\mathbf{J}^T\mathbf{J}) = \frac{1}{{}^C\hat{z}_f^4}\left({}^C\hat{x}_f^2 + {}^C\hat{y}_f^2 + {}^C\hat{z}_f^2\right) \tag{46}$$

Substituting (46) into (45) and denoting

$$\phi = \operatorname{atan}\left(\frac{{}^C\hat{x}_f^2 + {}^C\hat{y}_f^2}{{}^C\hat{z}_f^2}\right) \tag{47}$$

[5] We here ignore the radial/tangential distortion parameters to simplify the theoretical analysis, however, these parameters are explicitly modeled in our estimator.

we obtain:

$$||\mathbf{H}_{\mathbf{p}}||_2 = \frac{a_u}{{}^C\hat{z}_f}\sqrt{1+\tan^2(\phi)} \tag{48}$$

Turning to $||\mathbf{H}_{\boldsymbol{\theta}}||_2$, we note that in practice, the distance between the IMU and the camera is typically much smaller than the distance between the features and the camera (e.g., in the devices used in our experiments, the IMU and camera are approximately 4 cm apart, while feature depths are typically in the order of meters). Therefore, to simplify the calculations, we use the approximation ${}^C\hat{\mathbf{p}}_f - {}^C\mathbf{p}_I \approx {}^C\hat{\mathbf{p}}_f$, leading to:

$$\begin{aligned}
||\mathbf{H}_{\boldsymbol{\theta}}||_2 &\approx a_u||\mathbf{J}\left\lfloor {}^C\hat{\mathbf{p}}_f\times\right\rfloor {}^I_G\hat{\mathbf{R}}^T(t_n)\,{}^C_I\mathbf{R}^T||_2 && (49)\\
&= a_u||\mathbf{J}\lfloor {}^C\hat{\mathbf{p}}_f\times\rfloor||_2 && (50)\\
&= \frac{a_u}{{}^C\hat{z}_f^2}\left({}^C\hat{x}_f^2+{}^C\hat{y}_f^2+{}^C\hat{z}_f^2\right) && (51)\\
&= a_u\left(1+\tan^2(\phi)\right) && (52)
\end{aligned}$$

Note that the angle ϕ defined in (47) is the angle between the camera optical axis and the optical ray passing through the feature. Therefore, ϕ is upper bounded by one half the angular field-of-view, F of the camera. As a result, we can obtain upper bounds for the 2-norms of the Jacobians as follows:

$$||\mathbf{H}_{\mathbf{p}}||_2 \leq H_{\mathbf{p}_u} = \frac{a_u}{{}^C\hat{z}_f}\sqrt{1+\tan^2\left(\frac{F}{2}\right)}$$

$$||\mathbf{H}_{\boldsymbol{\theta}}||_2 \leq H_{\boldsymbol{\theta}_u} = a_u\left(1+\tan^2\left(\frac{F}{2}\right)\right)$$

Using $a_u = 500$, $F = 60^o$, and ${}^C\hat{z}_f = 2$, and substituting the above values into (13), (15), and (17), we obtain the worst-case bounds for the unmodeled residuals given in Section III.

Appendix II

In this section, we derive the time derivative of the orientation error term $\dot{\tilde{\boldsymbol{\theta}}}_I$. By converting the quaternion-multiplication expression (22) into the rotation-matrix form, we obtain:

$${}^I_G\mathbf{R} = {}^I_G\hat{\mathbf{R}}\left(\mathbf{I}-\lfloor\tilde{\boldsymbol{\theta}}_I\times\rfloor\right) \tag{53}$$

Taking the time derivative on both sides of the above equation leads to:

$${}^I_G\dot{\mathbf{R}} = {}^I_G\dot{\hat{\mathbf{R}}}\left(\mathbf{I}-\lfloor\tilde{\boldsymbol{\theta}}_I\times\rfloor\right) - {}^I_G\hat{\mathbf{R}}\lfloor\dot{\tilde{\boldsymbol{\theta}}}_I\times\rfloor \tag{54}$$

The derivative of the rotation matrix, ${}^I_G\dot{\mathbf{R}}$, can be written as:

$${}^I_G\dot{\mathbf{R}} = -\lfloor {}^I\boldsymbol{\omega}\times\rfloor {}^I_G\mathbf{R} \tag{55}$$

Substituting (55) into (54), we thus obtain:

$$\lfloor {}^I\boldsymbol{\omega}\times\rfloor {}^I_G\mathbf{R} = \lfloor {}^I\hat{\boldsymbol{\omega}}\times\rfloor {}^I_G\hat{\mathbf{R}}\left(\mathbf{I}-\lfloor\tilde{\boldsymbol{\theta}}_I\times\rfloor\right) + {}^I_G\hat{\mathbf{R}}\lfloor\dot{\tilde{\boldsymbol{\theta}}}_I\times\rfloor \tag{56}$$

which can be re-formulated by substituting ${}^I_G\mathbf{R}$ using (53):

$$\lfloor\dot{\tilde{\boldsymbol{\theta}}}_I\times\rfloor = \lfloor {}^I_G\hat{\mathbf{R}}^T\,{}^I\tilde{\boldsymbol{\omega}}\times\rfloor - \lfloor {}^I_G\hat{\mathbf{R}}^T\,{}^I\tilde{\boldsymbol{\omega}}\times\rfloor\lfloor\tilde{\boldsymbol{\theta}}_I\times\rfloor \tag{57}$$

By ignoring the quadratic error term $\lfloor {}^I_G\hat{\mathbf{R}}^T\,{}^I\tilde{\boldsymbol{\omega}}\times\rfloor\lfloor\tilde{\boldsymbol{\theta}}_I\times\rfloor$, we obtain the following expression for $\dot{\tilde{\boldsymbol{\theta}}}_I$:

$$\dot{\tilde{\boldsymbol{\theta}}}_I \approx {}^I_G\hat{\mathbf{R}}^T\,{}^I\tilde{\boldsymbol{\omega}} = {}^G\tilde{\boldsymbol{\omega}} \tag{58}$$

Acknowledgments

This work was supported by the National Science Foundation (grant no. IIS-1117957 and IIS-1253314) and the UC Riverside Bourns College of Engineering.

References

[Ait-Aider et al., 2006] Ait-Aider, O., Andreff, N., and Lavest, J. M. (2006). Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *Proceedings of the European Conference on Computer Vision*, pages 56–68.

[Ait-Aider and Berry, 2009] Ait-Aider, O. and Berry, F. (2009). Structure and kinematics triangulation with a rolling shutter stereo rig. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1835 –1840.

[Bar-Shalom et al., 2001] Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons.

[Bosse and Zlot, 2009] Bosse, M. and Zlot, R. (2009). Continuous 3d scan-matching with a spinning 2d laser. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4312–4319, Anchorage. IEEE.

[Bosse et al., 2012] Bosse, M., Zlot, R., and Flick, P. (2012). Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, 28(5):1104–1119.

[Forssen and Ringaby, 2010] Forssen, P. and Ringaby, E. (2010). Rectifying rolling shutter video from hand-held devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 507 –514, San Francisco, CA.

[Furgale et al., 2012] Furgale, P., Barfoot, T., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2088–2095, Minneapolis, MN.

[Hanning et al., 2011] Hanning, G., Forslow, N., Forssen, P., Ringaby, E., Tornqvist, D., and Callmer, J. (2011). Stabilizing cell phone video using inertial measurement sensors. In *IEEE International Conference on Computer Vision Workshops*, pages 1 –8, Barcelona, Spain.

[Hedborg et al., 2012] Hedborg, J., Forssen, P., Felsberg, M., and Ringaby, E. (2012). Rolling shutter bundle adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1434 –1441, Providence, RI.

[Hedborg et al., 2011] Hedborg, J., Ringaby, E., Forssen, P., and Felsberg, M. (2011). Structure and motion estimation from rolling shutter video. In *IEEE International Conference on Computer Vision Workshops*, pages 17 –23, Barcelona, Spain.

[Jia and Evans, 2012] Jia, C. and Evans, B. (2012). Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, Banff, Canada.

[Jones and Soatto, 2011] Jones, E. and Soatto, S. (2011). Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *International Journal of Robotics Research*, 30(4):407–430.

[Karpenko et al., 2011] Karpenko, A., Jacobs, D., Back, J., and Levoy, M. (2011). Digital video stabilization and rolling shutter correction using gyroscopes. Technical report, Stanford University.

[Kelly and Sukhatme, 2011] Kelly, J. and Sukhatme, G. (2011). Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research*, 30(1):56–79.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *The International Symposium on Mixed and Augmented Reality*, pages 225–234, Nara, Japan.

[Klein and Murray, 2009] Klein, G. and Murray, D. (2009). Parallel tracking and mapping on a camera phone. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 83–86, Orlando.

[Kottas et al., 2012] Kottas, D. G., Hesch, J. A., Bowman, S. L., and Roumeliotis, S. I. (2012). On the consistency of vision-aided inertial navigation. In *Proceedings of the International Symposium on Experimental Robotics*, Quebec City, Canada.

[Li et al., 2013] Li, M., Kim, B., and Mourikis, A. I. (2013). Real-time motion estimation on a cellphone using inertial sensing and a rolling-shutter camera. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4697–4704, Karlsruhe, Germany.

[Li and Mourikis, 2012a] Li, M. and Mourikis, A. I. (2012a). Optimization-based estimator design for vision-aided inertial navigation. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia.

[Li and Mourikis, 2012b] Li, M. and Mourikis, A. I. (2012b). Vision-aided inertial navigation for resource-constrained systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1057–1063, Vilamoura, Portugal.

[Li and Mourikis, 2013a] Li, M. and Mourikis, A. I. (2013a). 3-D motion estimation and online temporal calibration for camera-IMU systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5689–5696, Karlsruhe, Germany.

[Li and Mourikis, 2013b] Li, M. and Mourikis, A. I. (2013b). High-precision, consistent EKF-based visual-inertial odometry. *International Journal of Robotics Research*, 32(6):690–711.

[Liang et al., 2008] Liang, C.-K., Chang, L.-W., and Chen, H. (2008). Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing*, 17(8):1323 –1330.

[Lovegrove et al., 2013] Lovegrove, S., Patron-Perez, A., and Sibley, G. (2013). Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling-shutter cameras. In *Proceedings of the British Machine Vision Conference*, Bristol, UK.

[Lumenera Corporation, 2005] Lumenera Corporation (2005). Rolling shutter timing. `http://www.lumenera.com/support/pdf/LA-2104-ExposureVsShutterTypeTimingAppNote.pdf`.

[Magerand and Bartoli, 2010] Magerand, L. and Bartoli, A. (2010). A generic rolling shutter camera model and its application to dynamic pose estimation. In *International Symposium on 3D Data Processing, Visualization and Transmission*, Parris, France.

[Martinelli, 2012] Martinelli, A. (2012). Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28(1):44 –60.

[Montiel et al., 2006] Montiel, J., Civera, J., and Davison, A. (2006). Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems*, pages 81–88, Philadelphia, PA.

[Mourikis and Roumeliotis, 2007] Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3565–3572, Rome, Italy.

[Mourikis et al., 2009] Mourikis, A. I., Trawny, N., Roumeliotis, S. I., Johnson, A. E., Ansar, A., and Matthies, L. H. (2009). Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*, 25(2):264–280.

[Oth et al., 2013] Oth, L., Furgale, P., Kneip, L., and Siegwart, R. (2013). Rolling shutter camera calibration. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Portland, OR.

[Ringaby and Forssén, 2012] Ringaby, E. and Forssén, P.-E. (2012). Efficient video rectification and stabilisation for cell-phones. *International journal of computer vision*, 96(3):335–352.

[Tong et al., 2013] Tong, C. H., Furgale, P., and Barfoot, T. (2013). Gaussian Process GaussNewton for non-parametric simultaneous localization and mapping. *The International Journal of Robotics Research*, 32(5):507–525.

[Weiss et al., 2012] Weiss, S., Achtelik, M., Lynen, S., Chli, M., and Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 957–964, St Paul, MN.