Real-time Motion Tracking on a Cellphone using Inertial Sensing and a Rolling-Shutter Camera

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

***Abstract*— All existing methods for vision-aided inertial navigation assume a camera with a global shutter, in which all the pixels in an image are captured simultaneously. However, the vast majority of consumer-grade cameras use rolling-shutter sensors, which capture each row of pixels at a slightly different time instant. The effects of the rolling shutter distortion when a camera is in motion can be very significant, and are not modelled by existing visual-inertial motion-tracking methods. In this paper we describe the first, to the best of our knowledge, method for vision-aided inertial navigation using rolling-shutter cameras. Specifically, we present an extended Kalman filter (EKF)-based method for visual-inertial odometry, which fuses the IMU measurements with observations of visual feature tracks provided by the camera. The key contribution of this work is a computationally tractable approach for taking into account the rolling-shutter effect, incurring only minimal approximations. The experimental results from the application of the method show that it is able to track, in real time, the position of a mobile phone moving in an unknown environment with an error accumulation of approximately 0.8% of the distance travelled, over hundreds of meters.**

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

I. Introduction

Modern smartphones have significant processing capabilities, and are equipped with a rich sensor suite that typically includes multiple cameras, gyroscopes, accelerometers, a compass, and a GPS receiver. This creates opportunities for using these devices for *location-based* applications, which use the knowledge of a device's position to provide useful information. To estimate the position, most approaches today rely on GPS and signals from known transmitters (e.g., cellular network antennas, wireless network routers). However, the estimates obtained by these methods are typically quite coarse, and cannot track fine motions of the device. By contrast, in this work we are interested in tracking the motion of a smartphone (or similar device) with high precision, without relying on any external infrastructure or prior knowledge of the area where the motion takes place.

Endowing small handheld devices with the capability to track their pose with high spatial and temporal fidelity in non-instrumented environments will create a host of new opportunities. For instance, such devices can be used for high-precision location-based services even indoors and in other areas where radio signals are unavailable; for aiding visually impaired persons navigate in unknown environments; or for tracking a person's gait with high fidelity, to help diagnose medical conditions. Additionally, we point out that a smartphone capable of real-time, high-precision pose estimation can be placed onboard a mobile robot, micro aerial vehicle, or other robotic vehicle, to provide an inexpensive localization solution. Given the widespread availability of smartphones, this would lower the barrier to entry in robotics research and development.

Towards the realization of the above goals, in this paper we propose a method for motion estimation using a device's camera and its inertial sensors (gyroscope and accelerometer). The use of visual and inertial sensing is advantageous, as both sensors are self-contained, can operate in most environments of interest, and provide measurements at high sample rates. In recent years, a significant body of literature has focused on the problem of motion estimation using cameras and inertial measurement units (IMUs) [1]–[8]. However, all prior work assumes that the camera has a global shutter, i.e., that all the pixels in an image are captured at the same time. By contrast, the cameras found on smartphones (as well as most consumer devices) typically use CMOS sensors with a *rolling shutter*, capturing each row of pixels at a slightly different time instant.

When a moving rolling-shutter camera captures an image, the changing camera pose during the capture can cause significant visual distortions (Fig. 1). In the context of motion estimation, these distortions mean that the projections of scene points on the image are at drastically different locations, compared to the locations they would have if a global shutter was used. Therefore, if one of the existing methods that assume a global shutter is used for motion estimation, its performance is severely degraded (see Section IV). Unless the rolling-shutter effects are explicitly modelled and accounted for, they can lead to outright algorithm failure.

In this paper we describe the first, to the best of our

knowledge, method for vision-aided inertial navigation using rolling-shutter cameras. Specifically, we present an extended Kalman filter (EKF)-based method for visual-inertial odometry, which fuses the IMU measurements with observations of visual feature tracks provided by the camera. No additional external signals are used, and no prior knowledge of the features' position is required. The key contribution of this work is a computationally tractable approach for taking into account the camera's motion during each image's capture. Specifically, we include the camera's linear and rotational velocities at the time of image capture in the EKF state vector, and use them to model the rolling-shutter effect in the camera's measurement model. To avoid the computational burden normally incurred by including the rotational velocity in the state vector (in which case EKF updates normally need to take place at the IMU sample rate), a novel formulation for IMU propagation is derived.

The proposed method has been tested on real-world data on a Samsung Galaxy S2 mobile phone. The attained accuracy is comparable to that obtained by high-end global-shutter cameras (e.g. [6], [9]), despite the lower quality of the sensors and the limited capabilities of the device's CPU. In datasets involving 610-m and 900-m long trajectories, the final position errors were in the order of 0.8% of the distance traveled. This high level of accuracy is achieved even though the method operates, at real-time speed, on the resource-constrained processor of the mobile phone. Moreover, simulation tests confirm that this level of accuracy is typical, and that the estimator is consistent. In what follows we describe the details of our work, after a discussion of the related literature.

II. Related Work

Most work on rolling-shutter cameras has focused on image rectification for compensating the visual distortions caused by the rolling shutter. The majority of methods in this category use the images in a video to obtain a parametric representation of the distortion, and use this to generate an undistorted video stream [10], [11]. Gyroscope measurements have also been employed for this task, since the most visually significant distortions are caused by rotational motion [12], [13]. 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. By comparison, this a less-explored topic. To the best of our knowledge, all existing approaches rely exclusively on visual data, and do not use any inertial sensors. For example, [14], [15] present algorithms for bundle adjustment with rolling-shutter cameras, [16] describes an implementation of the well-known PTAM algorithm [17] using a rolling-shutter sensor, while [18]–[20] propose methods for estimating the motion of objects by exploiting the rolling shutter distortion.

The fact that these methods use images alone, without inertial sensing, introduces limitations. For example, to estimate the linear and rotational velocity of the camera, in [14] the camera motion is assumed to vary linearly *between* the time instants images are recorded. In [15], similar assumptions are employed, but with a higher-order model for the motion. For these methods to be applicable, the camera images must be recorded at high frame rates, and the camera motion must be smooth, without any sudden changes in the motion profile. By contrast, by using inertial measurements to propagate the state estimates between images, our proposed method can operate with much lower frame rates (e.g., approximately 5 Hz in our experiments) thus reducing computational load. Moreover, the use of IMU readings enables operation even with highly-varying motions, such as those observed when a person holds a device while walking.

We note that, to the best of our knowledge, the only method that fuses visual measurements from a rolling-shutter camera and inertial sensors can be found in [21]. In that work, the visual feature observations are combined with gyroscope readings to estimate the camera's rotation *only*. By contrast, in our work we estimate the full 6-D pose of the moving platform (i.e., position and orientation), as well as the platform's velocity.

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

Our goal is to estimate the pose (position and orientation) of a device equipped with an IMU and a rolling-shutter camera. To this end, we affix a coordinate frame $\{I\}$ to the IMU, and track its motion with respect to a fixed global coordinate frame, $\{G\}$. The spatial transformation between the IMU frame and the coordinate frame of the camera, $\{C\}$, is constant and known, for example from prior calibration. This transformation is described by the rotation quaternion[1] ${}^C_I\bar{\mathbf{q}}$ and the position vector ${}^I\mathbf{p}_C$. Moreover, we assume that the intrinsic parameters of the camera are also known from an offline calibration procedure.

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*. The visual measurements are fused with measurements from a 3-axis gyroscope and a 3-axis accelerometer, which provide measurements of the rotational velocity, $\boldsymbol{\omega}_m$, and acceleration, $\mathbf{a}_m$, respectively:

$$\boldsymbol{\omega}_m = {}^I\boldsymbol{\omega} + \mathbf{b_g} + \mathbf{n_r} \tag{1}$$

$$\mathbf{a}_m = {}^I_G\mathbf{R}\left({}^G\mathbf{a} - {}^G\mathbf{g}\right) + \mathbf{b_a} + \mathbf{n_a} \tag{2}$$

where ${}^I\boldsymbol{\omega}$ is the IMU's rotational velocity vector, ${}^G\mathbf{g}$ is the gravitational acceleration vector, $\mathbf{b_g}$ and $\mathbf{b_a}$ are the gyroscope and accelerometer biases, respectively, which are modelled as random-walk processes, and finally $\mathbf{n_r}$ and $\mathbf{n_a}$ are zero-mean white Gaussian noise vectors affecting the measurements.

[1]Notation: The preceding superscript for vectors (e.g., G in ${}^G\mathbf{a}$) denotes the frame of reference with respect to which quantities are expressed. ${}^A_B\mathbf{R}$ is the rotation matrix rotating vectors from frame $\{B\}$ to $\{A\}$, ${}^A_B\bar{\mathbf{q}}$ is the corresponding unit quaternion [22], and ${}^A\mathbf{p}_B$ is the position of the origin of frame $\{B\}$ with respect to $\{A\}$. $\otimes$ denotes quaternion multiplication, $\lfloor\mathbf{c}\times\rfloor$ is the skew symmetric matrix corresponding to vector $\mathbf{c}$, while $\mathbf{0}$ and $\mathbf{I}$ are the zero and identity matrices, respectively. $\hat{a}$ is the estimate of a variable a, and $\tilde{a} = a - \hat{a}$ its error. Finally, $\hat{a}_{i|j}$ is the estimate of the state a at time step i, when all EKF updates up to time step j have been performed.

Our approach to motion estimation is based on the modified multi-state-constraint Kalman filter algorithm (MSCKF 2.0), presented in [6], [23]. This is an EKF-based visual-inertial odometry algorithm for global-shutter cameras, which has been shown to produce high-precision motion estimates in real-world settings. Before presenting our approach to motion estimation with rolling-shutter cameras, we first briefly discuss the global-shutter MSCKF 2.0 algorithm, to highlight its key components and introduce the notation used in the rest of the paper.

A. The MSCKF 2.0 algorithm for global-shutter cameras

The key idea of the MSCKF method, originally proposed in [5], is to maintain a state vector comprising a sliding window of camera poses, and to use the feature measurements to impose constraints on these poses, without explicitly including the features in the state vector. Specifically, the state vector of the MSCKF algorithm at time-step k is given by:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{I_k}^T & \boldsymbol{\pi}_{C_{k-m}}^T & \cdots & \boldsymbol{\pi}_{C_{k-1}}^T \end{bmatrix}^T \tag{3}$$

where $\mathbf{x}_{I_k}^T$ is the current IMU state, and $\boldsymbol{\pi}_{C_i} = [{}^{C_i}_{G}\bar{\mathbf{q}}^T \quad {}^{G}\mathbf{p}_{C_i}^T]^T$, for $i = k-m, \ldots, k-1$, are the camera poses at the time instants the last m images were recorded. The IMU state is defined as:

$$\mathbf{x}_I = \begin{bmatrix} {}^{I}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_I^T & {}^{G}\mathbf{v}_I^T & \mathbf{b}_\mathbf{g}^T & \mathbf{b}_\mathbf{a}^T \end{bmatrix}^T \tag{4}$$

where ${}^{I}_{G}\bar{\mathbf{q}}$ is the unit quaternion describing the rotation from the global frame to the IMU frame, while ${}^{G}\mathbf{p}$ and ${}^{G}\mathbf{v}_I$ denote the IMU position and velocity with respect to the global frame. Note that the gyroscope and accelerometer biases are also included in the state vector, since their values slowly change in time and need to be estimated online.

Propagation: Every time an IMU measurement is received, it is used to propagate the IMU state [23]. Additionally, the covariance matrix of the state estimate is propagated as:

$$\mathbf{P}_{k+1|k} = \begin{bmatrix} \boldsymbol{\Phi}_{I_k}\mathbf{P}_{II_{k|k}}\boldsymbol{\Phi}_{I_k}^T + \mathbf{Q}_d & \boldsymbol{\Phi}_{I_k}\mathbf{P}_{IC_{k|k}} \\ \mathbf{P}_{IC_{k|k}}^T\boldsymbol{\Phi}_{I_k}^T & \mathbf{P}_{CC_{k|k}} \end{bmatrix} \tag{5}$$

where $\boldsymbol{\Phi}_{I_k}$ is the error-state transition matrix for the IMU, computed in closed form as shown in [6], [23], $\mathbf{Q}_d$ is the process-noise covariance matrix, and $\mathbf{P}_{II_{k|k}}$, $\mathbf{P}_{CC_{k|k}}$, and $\mathbf{P}_{IC_{k|k}}$ are the partitions of the filter covariance matrix for the IMU state, the camera states, and the cross-terms between them, respectively.

State augmentation: When a new image is recorded, the state vector of the filter is augmented with a copy of the current camera pose. Specifically, if a new image is recorded at time-step k, we augment the state vector with the estimate of the current camera pose:

$$\hat{\boldsymbol{\pi}}_{C_{k|k-1}} = \begin{bmatrix} {}^{C}_{G}\hat{\bar{\mathbf{q}}}_{k|k-1}^T & {}^{G}\hat{\mathbf{p}}_{C_{k|k-1}}^T \end{bmatrix}, \quad \text{with}$$

$${}^{C}_{G}\hat{\bar{\mathbf{q}}}_{k|k-1} = {}^{C}_{I}\bar{\mathbf{q}} \otimes {}^{I}_{G}\hat{\bar{\mathbf{q}}}_{k|k-1} \tag{6}$$

$${}^{G}\hat{\mathbf{p}}_{C_{k|k-1}} = {}^{G}\hat{\mathbf{p}}_{I_{k|k-1}} + {}^{I}_{G}\hat{\mathbf{R}}_{k|k-1}^T\, {}^{I}\mathbf{p}_C \tag{7}$$

and augment the state covariance matrix as:

$$\mathbf{P}_{k|k-1} \leftarrow \begin{bmatrix} \mathbf{P}_{k|k-1} & \mathbf{P}_{k|k-1}\mathbf{J}_{\boldsymbol{\pi}}^T \\ \mathbf{J}_{\boldsymbol{\pi}}\mathbf{P}_{k|k-1} & \mathbf{J}_{\boldsymbol{\pi}}\mathbf{P}_{k|k-1}\mathbf{J}_{\boldsymbol{\pi}}^T \end{bmatrix} \tag{8}$$

where $\mathbf{J}_{\boldsymbol{\pi}}$ is the Jacobian of $\boldsymbol{\pi}_{C_k}$ with respect to the state vector. Once augmentation is performed, an EKF update takes place.

MSCKF measurement model: In the MSCKF approach, all the measurements of each feature are employed simultaneously for an EKF update, once the track of this feature is complete (e.g., when the feature goes out of the field of view). Specifically, with a global-shutter camera, the observation of a feature by the camera at time step j is given by:

$$\mathbf{z}_j = \mathbf{h}({}^{C_j}\mathbf{p}_f) + \mathbf{n}_j \tag{9}$$

where $\mathbf{h}(\cdot)$ is the perspective camera model: $\mathbf{h}(\mathbf{f}) = [f_x/f_z \quad f_y/f_z]^T$, $\mathbf{n}_j$ is the measurement noise vector, modelled as zero-mean Gaussian with covariance matrix $\sigma^2\mathbf{I}_2$, and ${}^{C_j}\mathbf{p}_f$ is the position of the feature with respect to the camera. This is a function of the camera pose $\boldsymbol{\pi}_{C_j}$ and the feature position ${}^{G}\mathbf{p}_f$:

$${}^{C_j}\mathbf{p}_f(\boldsymbol{\pi}_{C_j}, {}^{G}\mathbf{p}_f) = {}^{C_j}_{G}\mathbf{R}\left({}^{G}\mathbf{p}_f - {}^{G}\mathbf{p}_{C_j}\right) \tag{10}$$

To use the feature for an update, we wait until all its measurements are collected (i.e., until the feature track is complete). Then, using all the measurements we triangulate a feature position estimate ${}^{G}\hat{\mathbf{p}}_f$, and compute the residuals

$$\mathbf{r}_j \doteq \mathbf{z}_j - \mathbf{h}(\hat{\boldsymbol{\pi}}_{C_j}, {}^{G}\hat{\mathbf{p}}_f)$$

for $j = 1 \ldots \ell$ where ℓ is the number of poses from which the feature was observed. In the above, we have explicitly expressed the fact that these residuals depend on both the camera pose estimates, and the feature position estimate. By linearizing, we can write the following expression for $\mathbf{r}_j$:

$$\mathbf{r}_j \simeq \mathbf{H}_j\tilde{\boldsymbol{\pi}}_{C_j} + \mathbf{H}_{f_j}{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n}_j,\ j = 1 \ldots \ell \tag{11}$$

where $\tilde{\boldsymbol{\pi}}_{C_j}$ and ${}^{G}\tilde{\mathbf{p}}_f$ are the estimation errors of the j-th camera pose and the feature respectively, and the matrices $\mathbf{H}_j$ and $\mathbf{H}_{f_j}$ are the corresponding Jacobians. In the MSCKF 2.0 algorithm, all the Jacobians are evaluated using the first estimate of each camera position, to ensure that the observability properties of the system model are maintained, and that the estimator remains consistent [6], [23].

Note that, since the feature errors are not included in the state vector, the residual $\mathbf{r}_j$ in (11) cannot be directly used for an EKF update. We therefore proceed to marginalize out the feature. For this purpose, we first form the vector containing the ℓ residuals:

$$\mathbf{r} \doteq \begin{bmatrix} \mathbf{r}_1^T & \mathbf{r}_2^T & \ldots & \mathbf{r}_\ell^T \end{bmatrix}^T \tag{12}$$

$$\simeq \mathbf{H}\tilde{\mathbf{x}} + \mathbf{H}_f{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n} \tag{13}$$

where $\mathbf{n}$ is a block vector with elements $\mathbf{n}_j$, and $\mathbf{H}$ and $\mathbf{H}_f$ are matrices with block rows $\mathbf{H}_j$ and $\mathbf{H}_{f_j}$, respectively. Subsequently, we define the residual vector $\mathbf{r}^o \doteq \mathbf{V}^T\mathbf{r}_i$, where $\mathbf{V}$ is a matrix whose columns form a basis of the left nullspace of $\mathbf{H}_f$. From (13), we obtain:

$$\mathbf{r}^o \doteq \mathbf{V}^T\mathbf{r} \simeq \mathbf{V}^T\mathbf{H}\tilde{\mathbf{x}} + \mathbf{V}^T\mathbf{n} = \mathbf{H}^o\tilde{\mathbf{x}} + \mathbf{n}^o \tag{14}$$

Note that (14) does not contain the feature error, and describes a residual in the standard EKF form. Thus, the residual $\mathbf{r}^o$ and the Jacobian matrix $\mathbf{H}^o = \mathbf{V}^T\mathbf{H}$ can now be used for an EKF update.

In the update that occurs after each image, all the features whose feature tracks just ended are processed to obtain residuals $\mathbf{r}_i^o$, and then all these residuals are used together to update the state vector and covariance matrix (details on how this update can be implemented in a computationally efficient way are given in [5]). Finally, the oldest camera poses, for which all the features have been processed, are removed from the state vector to keep the computational cost bounded.

B. Rolling-Shutter Camera Model

We now turn our attention to the use of a rolling-shutter camera for motion estimation. These cameras capture the rows of the image sequentially: if the first row in an image is captured at time t, the n-th row is captured at time $t+nt_d$, where t_d is the time interval between the reading of two consecutive rows of pixels. Thus, each image is captured over a time interval of non-zero duration, called the readout time of the image, t_r. For typical cameras found in consumer devices such as smartphones, t_r is usually in the order of a few tens of milliseconds, which is enough to cause significant distortions as shown in Fig. 1.

If a point feature is projected on the n-th row of the j-th image, its measurement is described by:

$$\mathbf{z}_j^{(n)} = \mathbf{h}({}^C\mathbf{p}_f(t_j + nt_d)) + \mathbf{n}_j^{(n)} \tag{15}$$

where $\mathbf{n}_j^{(n)}$ is the image measurement noise, t_j is the time instant the image capture began, and ${}^C\mathbf{p}_f(t_j + nt_d)$ is the position of the feature with respect to the camera frame at time $t_j + nt_d$:

$${}^C\mathbf{p}_f(t_j + nt_d) = {}^C_G\mathbf{R}(t_j + nt_d)\left({}^G\mathbf{p}_f - {}^G\mathbf{p}_C(t_j + nt_d)\right) \tag{16}$$

By comparing (15)-(16) to (9)-(10), the difference between a global-shutter and a rolling-shutter camera becomes clear. While previously the feature measurements in an image only depended on a single camera pose (i.e. $\boldsymbol{\pi}_{C_j}$), with a rolling shutter the image measurements in a *single* image depend on the camera pose at *multiple* time instants. To use these measurements in the MSCKF approach described in Section III-A, one would have to augment the sliding window with one camera pose per row of the image. This would require hundreds of poses to be included in the state vector, and would be computationally intractable.

In order to obtain a practical algorithm, we introduce the rotational and translational velocity of the camera into the equations. Specifically, (16) can be written without any approximation as:

$${}^C\mathbf{p}_f(t_j + nt_d) = \mathbf{R}(nt_d){}^{C_j}_G\mathbf{R} \left({}^G\mathbf{p}_f - {}^G\mathbf{p}_{C_j} - \int_{t_j}^{t_j+nt_d} {}^G\mathbf{v}_C(\tau)d\tau\right) \tag{17}$$

where ${}^G\mathbf{v}_C$ is the velocity of the camera in the global frame, and $\mathbf{R}(nt_d)$ is the rotation of the camera in the time interval $[t_j, t_j + nt_d]$, which can be computed by integration of the differential equation:

$$\dot{\mathbf{R}}(t) = -\lfloor {}^C\boldsymbol{\omega}(t)\times\rfloor\,\mathbf{R}(t), \qquad t \in [t_j, t_j + nt_d] \tag{18}$$

with initial condition $\mathbf{I}_3$. In this equation, ${}^C\boldsymbol{\omega}(t)$ denotes the rotational velocity of the camera.

Equation (17) is a function of the camera pose $\boldsymbol{\pi}_{C_j}$ and of the linear and rotational velocity of the camera in the time interval $[t_j, t_j + nt_d]$. Since none of these quantities are perfectly known, they have to be estimated. However, the camera velocities defined over a time *interval* are infinite-dimensional quantities, and thus (17) cannot be directly used to define a practically useful measurement model. To obtain a computationally tractable formulation, we employ additional assumptions on the time-evolution of the camera velocities. Specifically, we model ${}^G\mathbf{v}_C$ and ${}^C\boldsymbol{\omega}$ as being constant during the readout time of each image. This has the advantage that the rotation matrix $\mathbf{R}(nt_d)$ and the integral appearing in (17) take a simple form, leading to:

$${}^C\mathbf{p}_f(t_j + nt_d) = e^{-\lfloor nt_d\,{}^C\boldsymbol{\omega}_j\times\rfloor}\,{}^{C_j}_G\mathbf{R}\left({}^G\mathbf{p}_f - {}^G\mathbf{p}_{C_j} - nt_d\,{}^G\mathbf{v}_{C_j}\right) \tag{19}$$

The matrix exponential $e^{-\lfloor nt_d\,{}^C\boldsymbol{\omega}_j\times\rfloor}$ can be written in closed form, as shown in [24].

Note that, by assuming constant linear and rotational velocity for the camera during the capture of each image, we are introducing an approximation. However, this approximation is used for a much shorter time period (the readout time, t_r) than what is needed in vision-only methods such as [14], which assumes constant velocities for the entire time interval *between images* (e.g. in our experiments the readout time was $t_r = 32$ msec, while the average time between images was approximately 200 msec). The key benefit from assuming constant linear and rotational velocity during t_r is that we can express the measurement of each feature as a function of just four quantities (see (15) and (19)): the camera position and orientation at the time instant the first row was captured, and the two velocity vectors.

Using the measurement model described by (15) and (19), we can now employ a method for using the feature measurements analogous to the MSCKF. To make this possible, instead of maintaining a sliding window of camera *poses* in the state vector, we maintain a sliding window of camera *states* (cf. (3)):

$$\mathbf{x}_k = \begin{bmatrix}\mathbf{x}_{I_k}^T & \mathbf{x}_{C_{k-m}}^T & \cdots & \mathbf{x}_{C_{k-1}}^T\end{bmatrix}^T \tag{20}$$

where each camera state is defined as:

$$\mathbf{x}_{C_j} = \begin{bmatrix}{}^C_G\bar{\mathbf{q}}_j^T & {}^G\mathbf{p}_{C_j}^T & {}^G\mathbf{v}_{C_j}^T & {}^C\boldsymbol{\omega}_j^T\end{bmatrix}^T \tag{21}$$

With this change, the way of processing the features in the MSCKF, described in Section III-A, can still be employed. The only difference is that now in (11)-(13) the Jacobians with respect to the camera pose $\boldsymbol{\pi}_{C_j}$ are replaced by Jacobians with respect to the camera state $\mathbf{x}_{C_j}$. These Jacobians can be computed by differentiation of (15), using the expression in (19). After this change, the MSCKF's

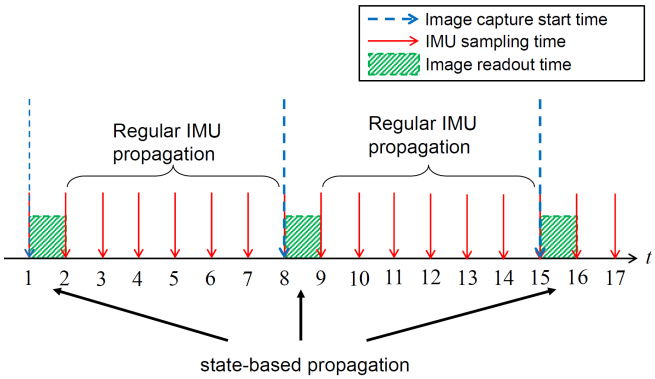

Fig. 2: Illustration of the times at which the two IMU-propagation methods are used.

technique of feature marginalization described in Section III-A can be followed.

C. State augmentation and IMU propagation

The preceding discussion shows that, by augmenting each camera state to include the linear and rotational velocity, we are able to process the measurements of the rolling-shutter camera for EKF updates. We now show how state augmentation can be performed, and how state augmentation affects the way in which the IMU is propagated. Let us consider the case where an image begins to be captured at time t_k (corresponding to time-step k in the EKF). At this time, the filter state vector must be augmented with a new camera state, containing the current camera pose and velocities. While the camera pose can be computed directly from the state estimates as shown in (6), the situation for the camera velocities is more complex. Specifically, the camera's velocity vectors are given by:

$$^G\mathbf{v}_{C_k} = {}^G\mathbf{v}_{I_k} + {}^I_G\mathbf{R}_k^T \lfloor {}^I\boldsymbol{\omega}_k \times \rfloor {}^I\mathbf{p}_C \quad (22)$$

$$^C\boldsymbol{\omega}_k = {}^C_I\mathbf{R}\ {}^I\boldsymbol{\omega}_k \quad (23)$$

The above equations contain the known camera-to-IMU transformation, the IMU velocity and orientation, *and* the IMU rotational velocity. The latter is not contained in the state vector, which introduces complications.

To see why, imagine that we use the IMU measurement at time step k to obtain an estimate for ${}^I\boldsymbol{\omega}_k$, and use this estimate in (23) for state augmentation. Now, having already used the IMU measurement, we should not re-use it for propagating the state estimate between timesteps k and $k+1$. Doing so would be using the same information twice, thus introducing unmodelled correlations which degrade accuracy. The most straightforward solution to this problem would be to explicitly include the IMU's rotational velocity in the state vector (as, for example, in [3], [25]). In this case, the gyroscope measurements are direct measurements of the state (see (1)), and must be used for EKF updates, instead of propagation. This is undesirable, as it would necessitate carrying out EKF updates (whose computational complexity is quadratic in the size of the state vector) at the IMU sample rate – typically in the order of 100 Hz. This would likely be infeasible on the resource-constrained processor of a handheld device.

To address these problems, we propose a new, "hybrid" way of using the IMU measurements: most of the time the IMU is used for propagating the state, as regular in the MSCKF, while every time a new image needs to be processed, a state-based propagation takes place. To illustrate the procedure, consider the situation shown in Fig. 2, where for simplicity we depict the camera's readout time to be equal to one IMU period (see Section III-D for the implementation details in the general case). The IMU measurements that are recorded in the time between image capture (i.e., timesteps 2-7, 9-14 and 16-17 in Fig. 2), are used for regular propagation, as described in Section III-A. On the other hand, when a new image begins to be captured (i.e., timesteps 1, 8, and 15), the following steps are taken: (i) the state vector and filter covariance matrix are augmented with a new camera state, (ii) a filter update takes place once the image becomes available, and (iii) a special state-propagation takes place, to propagate the state estimate to the next timestep. The details are described next.

1) State augmentation: If an image begins to be captured at time t_k, we compute the following estimate for the camera state, using the current filter estimates (i.e., the estimate $\hat{\mathbf{x}}_{k|k-1}$) and the current IMU measurement:

$$\hat{\mathbf{x}}_{C_{k|k-1}} = \begin{bmatrix} {}^C_G\hat{\mathbf{q}}^T_{k|k-1} & {}^G\hat{\mathbf{p}}^T_{C_{k|k-1}} & {}^G\hat{\mathbf{v}}^T_{C_{k|k-1}} & {}^C\hat{\boldsymbol{\omega}}^T_{k|k-1} \end{bmatrix}$$

where ${}^C_G\hat{\mathbf{q}}_{k|k-1}$ and ${}^G\hat{\mathbf{p}}_{C_{k|k-1}}$ are defined as in (6), and

$$^G\hat{\mathbf{v}}_{C_{k|k-1}} = {}^G\hat{\mathbf{v}}_{I_{k|k-1}} + {}^I_G\hat{\mathbf{R}}^T_{k|k-1} \lfloor (\boldsymbol{\omega}_{m_k} - \hat{\mathbf{b}}_{\mathbf{g}_{k|k-1}}) \times \rfloor {}^I\mathbf{p}_C$$

$$^C\hat{\boldsymbol{\omega}}_{k|k-1} = {}^C_I\mathbf{R}\ (\boldsymbol{\omega}_{m_k} - \hat{\mathbf{b}}_{\mathbf{g}_{k|k-1}})$$

In addition to the state, the covariance matrix of the filter must be augmented. To this end, we linearize $\mathbf{x}_{C_k}$ with respect to the state estimate and the gyroscope measurement, to obtain the following expression for the error in $\hat{\mathbf{x}}_{C_{k|k-1}}$:

$$\begin{aligned} \tilde{\mathbf{x}}_{C_{k|k-1}} &\simeq \begin{bmatrix} \mathbf{J_q} & \mathbf{J_p} & \mathbf{J_v} & \mathbf{J}_{\mathbf{b}_g} & \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}}_{k|k-1} + \mathbf{J}_{\boldsymbol{\omega}} \mathbf{n}_{\mathbf{r}_k} \\ &= \mathbf{J_x} \tilde{\mathbf{x}}_{k|k-1} + \mathbf{J}_{\boldsymbol{\omega}} \mathbf{n}_{\mathbf{r}_k} \end{aligned}$$

where $\mathbf{n}_{\mathbf{r}_k}$ is the noise in the measurement $\boldsymbol{\omega}_{m_k}$; $\mathbf{J_q}$, $\mathbf{J_p}$, $\mathbf{J_v}$, and $\mathbf{J}_{\mathbf{b}_g}$ are the Jacobians of $\mathbf{x}_{C_k}$ with respect to the IMU orientation, position, velocity, and gyroscope bias; and $\mathbf{J}_{\boldsymbol{\omega}}$ is the Jacobian with respect to $\boldsymbol{\omega}_{m_k}$. Based on the above expression, we can augment the covariance matrix as:

$$\mathbf{P}_{k|k-1} \leftarrow \begin{bmatrix} \mathbf{P}_{k|k-1} & \mathbf{P}_{k|k-1}\mathbf{J}_{\mathbf{x}}^T \\ \mathbf{J_x}\mathbf{P}_{k|k-1} & \mathbf{J_x}\mathbf{P}_{k|k-1}\mathbf{J}_{\mathbf{x}}^T + \mathbf{Q}_{r_k} \end{bmatrix} \quad (24)$$

with $\mathbf{Q}_{r_k}$ being the covariance matrix of the noise $\mathbf{n}_{\mathbf{r}_k}$.

2) State-based propagation: After state augmentation, we wait for the image to be fully captured, and perform an EKF update as described in Section III-A, to obtain $\hat{\mathbf{x}}_{k|k}$. Once the update is complete, the next operation that must be performed is to propagate the IMU state, to obtain $\hat{\mathbf{x}}_{k+1|k}$, where t_{k+1} is the time instant the image capture is complete (e.g., 2, 9, or 16 in Fig. 2). Normally, we would use the IMU measurements $\mathbf{a}_{m_k}$ and $\boldsymbol{\omega}_{m_k}$ for this step; however, as we mentioned $\boldsymbol{\omega}_{m_k}$ has already been used for the state augmentation, and should not be re-used, to avoid unmodelled correlations. Instead, for propagation we use the accelerometer measurement $\mathbf{a}_{m_k}$ and the estimate of the

Algorithm 1

Processing IMU measurements: When an IMU measurement is received at a time when an image is not being captured, use it to propagate the IMU state [5], [6] and the covariance matrix (see (5)).

Processing images:

- When an image starts to be captured, initialize a new camera state using the current IMU state estimate and gyroscope measurement (Section III-C.1).
- Once the image becomes available, perform an MSCKF update using all features whose tracks are complete (Section III-A).
- State management: Remove from the state vector old camera states, whose feature measurements have all been used.
- Propagate the state vector and covariance matrix using the accelerometer reading and the angular velocity of the latest camera state (Section III-C.2).

IMU's rotational velocity, computed as ${}^{I}_{C}\mathbf{R}\,{}^{C}\hat{\boldsymbol{\omega}}_{k|k}$ (note that ${}^{C}\hat{\boldsymbol{\omega}}_{k|k}$ is included in the EKF's state vector). Using these, the IMU kinematic state can be propagated as:

$$
\begin{aligned}
{}^{I}_{G}\hat{\bar{\mathbf{q}}}_{k+1|k} &= e^{\frac{1}{2}\boldsymbol{\Omega}\left({}^{I}_{C}\mathbf{R}\,{}^{C}\hat{\boldsymbol{\omega}}_{k|k}t_r\right)}\,{}^{I}_{G}\hat{\bar{\mathbf{q}}}_{k|k} \\
{}^{G}\hat{\mathbf{p}}_{I_{k+1|k}} &= {}^{G}\hat{\mathbf{p}}_{I_{k|k}} + {}^{G}\hat{\mathbf{v}}_{I_{k|k}}t_r + \frac{1}{2}{}^{G}\mathbf{g}t_r^2 + \\
&\qquad \int_{t_k}^{t_{k+1}}\int_{t_k}^{\tau} {}^{I}_{G}\hat{\mathbf{R}}^{T}(s)\left(\mathbf{a}_{m_k} - \hat{\mathbf{b}}_{a_{k|k}}\right)ds\,d\tau \\
{}^{G}\hat{\mathbf{v}}_{I_{k+1|k}} &= {}^{G}\hat{\mathbf{v}}_{I_{k|k}} + {}^{G}\mathbf{g}t_r + \int_{t_k}^{t_{k+1}} {}^{I}_{G}\hat{\mathbf{R}}^{T}(s)\left(\mathbf{a}_{m_k} - \hat{\mathbf{b}}_{a_{k|k}}\right)ds
\end{aligned}
$$

while all remaining quantities in the state vector stay unchanged. In the above equations $\boldsymbol{\Omega}$ is a 4×4 matrix given in [22]. In our implementation, the integrals in the position and velocity propagation are computed numerically using the trapezoid rule. To propagate the state covariance, we compute the state transition matrix by computing Jacobians of the above expressions with respect to all state variables (which now include ${}^{C}\boldsymbol{\omega}_k$), obtaining an expression analogous to (5).

D. Algorithm and implementation details

The entire algorithm for vision-aided inertial navigation with a rolling-shutter camera is described in Algorithm 1. Note that, to maintain the clarity of the presentation, in Section III-C it was assumed that, at the time an image's capture begins, an IMU measurement is simultaneously available. In practice, however, the sensor measurements are not sampled at the same time. In fact, in consumer-grade devices, where the gyroscope and accelerometer are usually separate units, even these two sensors' data arrive at different time instants. For these reasons, in our implementation we interpolate the inertial measurements, so that (i) the gyroscope and accelerometer measurements are available simultaneously, and (ii) an inertial measurement is available at the time an image capture begins. Moreover, since the image readout time is not, in general, equal to one IMU period, the measurement used for the state augmentation and state-based propagation is computed by averaging the inertial measurements that arrive during the readout time.

Fig. 3: The estimation results on the first dataset. The start/end point is denoted by a green square.

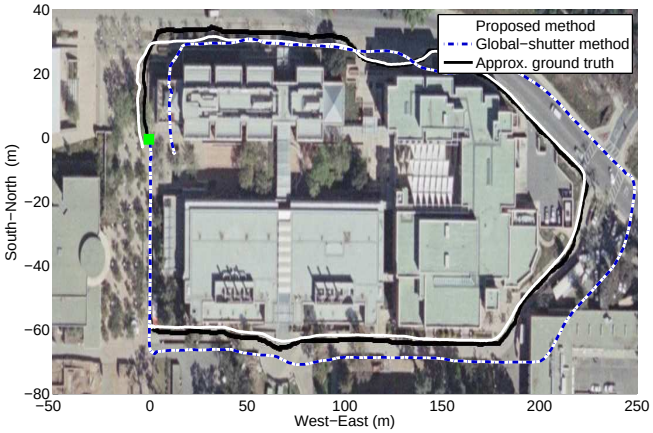

Fig. 4: The estimation results on the second dataset. The start/end point is denoted by a green square.

IV. Experimental Results

A. Real-World Experiments

To test the performance of our proposed algorithm, we collected data using a mobile phone (Samsung Galaxy S2) that is equipped with a rolling-shutter camera, a 3-axis gyroscope and a 3-axis accelerometer. We here report results in two datasets collected while a person walked, holding the phone with the camera facing forward. The device provides gyroscope measurements at 106 Hz and accelerometer measurements at 93 Hz. Both data streams were resampled to 90 Hz as described in Section III-D. On the other hand, images were captured at 15 Hz, and the readout time for the sensor was 32 msec. To reduce the computational load, we only use a new image whenever the device has moved by at least 20 cm from the location the last image was recorded (the motion distance is determined via the IMU propagation). This results in an effective average frame rate of 5.05 Hz in both datasets. In fact, we noticed that using images more frequently does not result in any meaningful performance improvement. For feature extraction we used an optimized version of the Shi-Tomasi algorithm, and performed normalized cross-correlation for feature matching, as

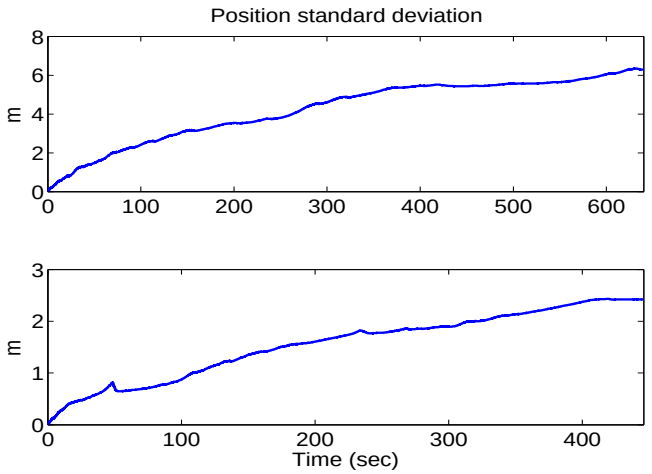

Fig. 5: The reported standard deviation for the position estimates (along the axis of largest uncertainty) in the two real-world datasets.

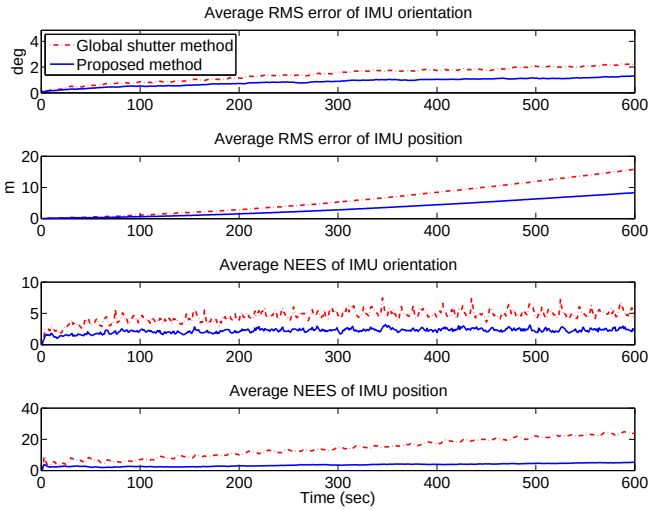

Fig. 6: The average RMS error and NEES over 50 Monte-Carlo simulations.

TABLE I: Simulation statistics

	Global Shutter		Rolling Shutter	
	Orient.	Pos.	Orient.	Pos.
RMS error	1.43 (deg)	6.25 (m)	0.83 (deg)	3.30 (m)
NEES	4.29	14.23	2.21	3.51

described in [26]. The data was stored onboard the cellphone, and processed offline on the same device.

The first dataset was collected walking on the streets of a residential area. The trajectory was approximately 900-m long, lasting approximately 11 minutes. The second dataset consists of a loop around the Engineering building at the University of California Riverside, in an approximately 610-m long trajectory over 8 minutes. The trajectory estimation results for both datasets are shown in Fig. 3 and Fig. 4. In these plots, we compare the estimates computed by the proposed method to the estimates computed by the approach of [6], which assumes a global shutter. Note that due to the nature of the datasets, a precise ground truth is difficult to obtain. For visualization purposes, we manually drew the approximate ground-truth path on the same figures.

The results of both experiments show that the proposed method computes trajectory estimates that accurately follow the (approximate) ground truth. These estimates are significantly more accurate than those of the method in [6], which demonstrates the importance of modelling the rolling-shutter effects. Even though precise ground truth for the entire trajectories is not known, it is known that in both cases the final position is identical to the starting one. Using this knowledge we can compute the final error to be 5.30 m, or 0.58% of the trajectory length in the first dataset, and 4.85 m, or 0.80% of the trajectory, in the second. These figures are comparable to the accuracy obtained in our previous work, with high-quality global-shutter cameras, and significantly more expensive IMUs [6]. In addition, we stress that the algorithm is capable of real-time operation: the average processing needed per image (including image processing and estimation) is 146 msec on the cellphone's processor, as compared to an average image period of 198 msec.

In addition to the trajectory estimates, in Fig. 5 we plot the reported standard deviation of the position estimates, along the axis on the plane with largest uncertainty. The top plot in this figure corresponds to the first dataset, while the bottom plot to the second. These plots show that the proposed method can obtain high-precision position estimates, without the use of any external reference signal. Note that, as expected of a visual-inertial odometry method, the position uncertainty gradually increases with time. In an application where GPS measurements, or other absolute position information (e.g., from a feature map database) can be obtained intermittently, they can be used in conjunction with the proposed method to prevent long-term drift.

B. Monte-Carlo simulations

To examine whether the results obtained in our real-world data are typical, and whether the estimates provided by the proposed method are consistent, we performed Monte-Carlo simulation tests. In these, we model a scenario similar to that of the real-world data, in which a person uses a hand-held mobile phone for pose estimation while walking fast. To simulate the periodical pattern of human walking, we designed linear acceleration and angular velocity trajectories following sinusoidal curves, similar to what we observed in the real-world data. In addition, we randomly generated features in front the camera with characteristics (e.g., depths, feature track lengths) similar to those of the real datasets. The sensor characteristics were chosen to be identical to those of the mobile phone used in the real-world experiments, and the trajectory is 10-minutes, 870-m long.

We carried out 50 Monte-Carlo simulation trials, and in each trial our proposed method and the method of [6] process the same data. To evaluate the performance of the two methods, we computed both the RMS error and the normalized estimation error squared (NEES) for the position and orientation, averaged over all trials at each time step. The RMS error gives as a measure of accuracy, while the

NEES is a measure of the consistency of the methods [27]. If an estimator is consistent, i.e., if it reports an appropriate covariance matrix for its state estimates, the average NEES should be close to the dimension of the variable being estimated (i.e., equal to 3 for the position and orientation).

Fig. 6 plots the RMS and NEES values over the duration of the trajectory, while Table I gives the average values for the entire trajectory for both methods tested. These results agree with the results observed in the real world data: when the rolling-shutter effect is properly accounted for, the estimation accuracy is significantly better. At the end of the trajectory, the RMS position error is 8.33 m, or 0.96% of the trajectory length, in the same order of magnitude as what was observed in the real-world experiments. In addition, the average NEES values for the position and orientation in the proposed approach are significantly smaller than that of the global-shutter MSCKF, and are close to the theoretically expected values for a consistent estimator.

V. Conclusion

In this paper, we have presented an algorithm for visual-inertial odometry using a rolling-shutter camera. To model the rolling-shutter effects the proposed method includes the camera's linear and rotational velocities at the time of image capture in the EKF state vector. This makes it possible to explicitly model the camera motion's effect on the projections of points in the images. The proposed approach has computational complexity linear in the number of features, which makes it suitable for real-time estimation on devices with limited resources, such as cellphones. Our experimental testing has shown that the method is capable of producing high-precision state estimates in real time: in our tests, the algorithm tracked the motion of a walking person with errors of up to 0.8% of the traveled distance, in trajectories that were hundreds of meters long, while running on a mobile phone's processor. These results indicate that the proposed approach is suitable for real-time pose estimation of miniature devices.

Acknowledgments

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

References

[1] A. Wu, E. Johnson, and A. Proctor, "Vision-aided inertial navigation for flight control," *AIAA Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 9, pp. 348–360, Sep. 2005.

[2] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. H. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 264–280, Apr. 2009.

[3] E. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, Apr. 2011.

[4] P. Pinies, T. Lupton, S. Sukkarieh, and J. Tardos, "Inertial aiding of inverse depth SLAM using a monocular camera," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 2797–2802.

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

[6] M. Li and A. I. Mourikis, "Improving the accuracy of EKF-based visual-inertial odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation*, St Paul, MN, May 2012, pp. 828–835.

[7] J. Kelly and G. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, Jan. 2011.

[8] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "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*, St Paul, MN, May 2012, pp. 957–964.

[9] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, Jul. 2012.

[10] C.-K. Liang, L.-W. Chang, and H. Chen, "Analysis and compensation of rolling shutter effect," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1323 –1330, Aug. 2008.

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

[12] A. Karpenko, D. Jacobs, J. Back, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," Stanford University, Tech. Rep., 2011.

[13] G. Hanning, N. Forslow, P. Forssen, E. Ringaby, D. Tornqvist, and J. Callmer, "Stabilizing cell phone video using inertial measurement sensors," in *IEEE International Conference on Computer Vision Workshops*, Barcelona, Spain, Nov. 2011, pp. 1 –8.

[14] J. Hedborg, P. Forssen, M. Felsberg, and E. Ringaby, "Rolling shutter bundle adjustment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, June 2012, pp. 1434 –1441.

[15] J. Hedborg, E. Ringaby, P. Forssen, and M. Felsberg, "Structure and motion estimation from rolling shutter video," in *IEEE International Conference on Computer Vision Workshops*, Barcelona, Spain, Nov. 2011, pp. 17 –23.

[16] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, Orlando, Oct. 2009, pp. 83–86.

[17] ——, "Parallel tracking and mapping for small AR workspaces," in *The International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007, pp. 225–234.

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

[19] O. Ait-Aider and F. Berry, "Structure and kinematics triangulation with a rolling shutter stereo rig," in *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2009, pp. 1835 –1840.

[20] L. Magerand and A. Bartoli, "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, May 2010.

[21] C. Jia and B. Evans, "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, Sept. 2012.

[22] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 6D pose estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep. 2005-002, Jan. 2005.

[23] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *International Journal of Robotics Research*, 2013, to appear.

[24] J. J. Craig, *Introduction to robotics: mechanics and control*, 3rd ed. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2005.

[25] D. Strelow, "Motion estimation from image and inertial measurements," Ph.D. dissertation, Carnegie Mellon University, Nov. 2004.

[26] M. Li and A. I. Mourikis, "Vision-aided inertial navigation for resource-constrained systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, Oct. 2012, pp. 1057–1063.

[27] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 2006, pp. 3562–3568.