Online Camera-Gyroscope Auto-Calibration for Cellphones

Chao Jia, *Student Member, IEEE,* and Brian L. Evans, *Fellow, IEEE,*

***Abstract*—The gyroscope is playing a key role in helping estimate 3D camera rotation for various vision applications on cellphones including video stabilization and feature tracking. Successful fusion of gyroscope and camera data requires that the camera, gyroscope and their relative pose to be calibrated. Moreover, the timestamps of gyroscope readings and video frames are usually not well synchronized. Previous work performed camera-gyroscope calibration and synchronization offline after the entire video sequence has been captured with restrictions on the camera motion, which is unnecessarily restrictive for everyday users to run apps that directly use the gyroscope. In this paper, we propose an online method that estimates all of the necessary parameters while a user is capturing video. Our contributions are (1) simultaneous online camera self-calibration and camera-gyroscope calibration based on an implicit extended Kalman filter, and (2) generalization of the multiple-view coplanarity constraint on camera rotation in a rolling shutter camera model for cellphones. The proposed method is able to estimate the needed calibration and synchronization parameters online with all kinds of camera motion, and can be embedded in gyro-aided applications such as video stabilization and feature tracking. Both Monte Carlo simulation and cellphone experiments show that the proposed online calibration and synchronization method converges fast to the ground truth values.**

***Index Terms*—Camera calibration, visual-inertial sensor fusion, multiple view geometry, gyroscope, rolling shutter camera.**

I. Introduction

CELLPHONE cameras have been increasingly popular for video capture due to the portability and processing power of cellphones. An increasing number of users are getting used to record their memorable events by cellphone cameras. Beyond the video recording itself, video acquisition also provides opportunities for applications such as augmented reality and visual odometry. No matter what application mobile video capture is used for, camera motion estimation is an essential step to improve the video quality and better analyze the video content.

Hand-held mobile devices such as cellphones usually suffer from egomotion that is changing very fast, which makes it difficult to track the camera accurately using only the captured videos. For this reason, inertial sensors on cellphones such as gyroscopes and accelerometers have been used to help estimate camera motion because of their increasing accuracy, high sampling rate and robustness to lighting conditions. It has been shown that through the fusion of visual and inertial information, camera motion can be estimated more accurately and reliably [2], [3]. The fusion of camera and inertial sensors, however, requires precise calibration: the coordinate system of inertial sensors does not coincide that of camera, and the timestamps of inertial sensor readings and video frames are not well synchronized. Apart from the relative pose and timestamp delay, camera and inertial sensors themselves also have to be calibrated so that essential parameters such as focal length and sensor biases are known.

Many existing approaches in visual-inertial sensor fusion assume that calibration and synchronization have been done offline beforehand. Moreover, camera self-calibration (estimation of camera intrinsic parameters) are usually executed separately from relative pose and delay calibration between camera and inertial sensors [4], [5]. Some calibration methods can be only performed in laboratory environments with special devices (e.g. spin table and checkerboard) [6], [7], which further prevents everyday users from using cellphone cameras conveniently with the help of inertial sensors. In this paper, we focus on online calibration and synchronization of cellphone cameras and inertial sensors while users capture videos, without any prior knowledge about the devices or any special calibration hardware.

Unlike traditional cameras, most cellphone cameras do not capture the rows in a single frame simultaneously, but sequentially from top to bottom. When there is fast relative motion between the scene and the camera, a frame can be distorted because each row was captured under different 3D-to-2D projections. This is known as rolling shutter effect [8], [9], [10] and has to be considered in calibration and fusion of visual and inertial sensors.

Although some applications such as visual odometry require estimation of both camera rotation and translation, estimating rotation using only the gyroscope has been used successfully in video stabilization [11] and feature tracking [12]. When the displacement of pixels between consecutive video frames is primarily caused by camera rotation, a gyroscope-only approach successfully stabilized video and removed rolling shutter effects [11], [13]. Similarly, gyroscope measurements were used to pre-warp the frames so that the search space of the Kanade-Lucas-Tomasi (KLT) [14] feature tracker can be narrowed down to its convergence region [12]. In these proposed methods there is no need to use the accelerometer. Therefore, only the camera and the gyroscope need to be calibrated. In this paper we focus on such camera-gyroscope calibration, and our proposed approach does not assume that the camera undergoes pure rotation.

The proposed online calibration and synchronization is

C. Jia and B. L. Evans are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, 78712, USA. E-mail: cjia@utexas.edu, bevans@ece.utexas.edu

This research was supported by gift funding from Texas Instruments, Dallas, TX, USA.

Part of this paper has been presented in 2013 IEEE Global Conference on Signal and Information Processing [1]

based on an extended Kalman filter (EKF). Each video frame provides a view of the 3D scene and triggers the update of the EKF through multiple view geometry. Although we care about camera rotation only, we do not assume any degeneration in the motion of the camera. By extending the recent proposed multiple-view coplanarity constraint of camera rotation [15] to rolling shutter cameras, we propose a novel implicit measurement that involves only camera rotation. This measurement is valid when there is non-zero or zero camera translation. The implicit measurements can be effectively used in the EKF to update the estimate of state vectors.

This paper is organized as follows. Section II reviews previous algorithms on camera self-calibration, camera-inertial calibration, and camera-gyroscope calibration. Section III introduces the rolling shutter camera model and summarizes the parameters that we need to estimate in this paper. Section IV presents the coplanarity constraint on camera rotation in the rolling shutter camera model. This constraint is then used in implicit measurements by the proposed EKF-based online calibration and synchronization approach in Section V. Section VI shows and analyzes the results of Monte Carlo simulation and cellphone experiments using the proposed approach. Section VII concludes the paper.

II. Related work

Camera self-calibration has been extensively studied [16] for both global shutter camera [6] and rolling shutter camera [17], but previous work on online self-calibration is somewhat rare. In [18] full-parameter online camera self-calibration is first proposed in the framework of sequential Bayesian structure from motion using a sum of Gaussian (SOG) filter. Their work assumes a global shutter camera model and the motion of the camera has to contain large enough translation to make the structure from motion problem well-conditioned.

The inertial sensors (gyroscope and accelerometer) are widely used in camera motion estimation and simultaneous localization and mapping (SLAM) together with visual measurements [19], [4], [5]. Especially for hand-held devices such as cellphone cameras, inertial-aided approaches appear more robust in camera tracking and SLAM when compared to purely vision-based approaches [20], [21].

Relative pose between inertial sensors and camera has been successfully estimated offline with special hardware [7] or simply with a known calibration pattern [22]. Online camera-inertial calibration has also been implemented recently in the framework of SLAM or navigation [23] together with the estimation of inertial sensor biases. However, to the best of our knowledge all of the previous work assumes that the camera itself has been calibrated; i.e., the camera projection parameters are known. Moreover, rolling shutter effect was not taken into account in the fusion of inertial and visual sensors until very recently [24], [25], [26]. The timestamp delay between camera and inertial sensors was always assumed as known except for the recent work in [27] which estimates the timestamp delay online.

The SLAM framework for online calibration of camera and inertial sensors involves estimation of camera translation and 3D scene structure. In addition, camera translation estimation and accelerometer calibration require large enough camera translation to initialize absolute scale and speed estimate [28], [29]. Therefore, such methods are too complicated if we only care about camera rotation and just want to use gyroscope to estimate and track camera motion.

To calibrate the camera and gyroscope system, the method in [11] proposed to quickly shake the camera while pointing at a far-away object (e.g., a building). Feature points between consecutive frames are matched and all parameters are estimated simultaneously by minimizing the homographic reprojection errors under a pure rotation model. The calibration in [12] is also based on homography transformation of matched feature points assuming pure rotation, except that different parameters are estimated separately first and then refined through non-linear optimization. However, as shown in [12], when the camera translation is not negligible relative to the distance of the feature points to the camera, such pure rotation model becomes less accurate and the calibration results will deviate from the ground truth. Our calibration method differs with [11], [12] not only in that it is online estimation, but also in that it does not assume zero translation at all. Therefore, the proposed calibration can be performed implicitly anytime and anywhere while the camera is recording video. This is especially convenient for amateur photographers who want to take stabilized videos with smartphone cameras.

III. Rolling shutter camera model and gyroscope

Points in the camera reference space are projected according to the pinhole camera model. Assuming the 3D point coordinates in the camera reference space are $[X_c, Y_c, Z_c]^{\mathrm{T}}$, their projection onto the image plane can be represented as

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} c_x + f\frac{X_c}{Z_c} \\ c_y + f\frac{Y_c}{Z_c} \end{bmatrix}, \tag{1}$$

where f is the focal length and c_x, c_y are the principal point coordinates. Here we assume that the camera projection skew is zero and the pixel aspect ratio is 1 as in [18], which is a reasonable assumption for today's cellphone cameras. Similarly, given the pixel coordinate $[u_x, u_y]^{\mathrm{T}}$, we can invert (1) to obtain the 3D coordinates of the corresponding feature point in the camera reference space up to an unknown scale as

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \lambda \begin{bmatrix} u_x - c_x \\ u_y - c_y \\ f \end{bmatrix}. \tag{2}$$

Based on (2), we further model the radial lens distortion of camera using two distortion coefficients as

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \lambda \begin{bmatrix} (1 + \kappa_1 r^2 + \kappa_2 r^4)(u_x - c_x) \\ (1 + \kappa_1 r^2 + \kappa_2 r^4)(u_y - c_y) \\ f \end{bmatrix}, \tag{3}$$

where

$$r = \sqrt{\left(\frac{u_x - c_x}{f}\right)^2 + \left(\frac{u_y - c_y}{f}\right)^2}. \tag{4}$$

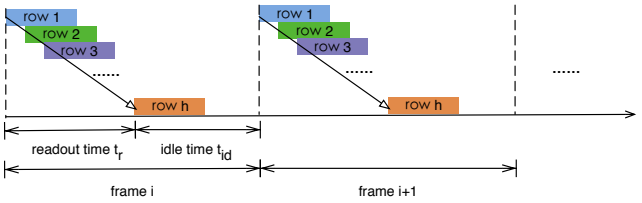


Fig. 1: Rows are captured sequentially in rolling shutter cameras. Each block represents the exposure time of a certain row.

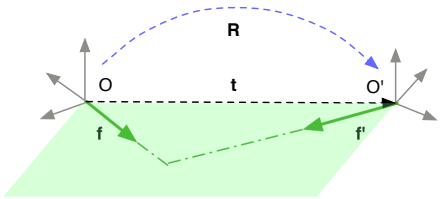


Fig. 2: The epipolar constraint on a pair of features in two viewpoints.

In rolling shutter cameras, rows in each frame are exposed sequentially from top to bottom [8], [30], as shown in Fig. 1. In Fig. 1 each block represents the exposure of a certain row. The exposure duration of each row (represented by the length of each block) depends on the lighting conditions. In this paper we ignore possible image blur and assume instantaneous exposure. Thus, the exposure moment of each row can be approximated as the left end of each block in Fig. 1. For an image pixel $\mathbf{u} = [u_x, u_y]^{\mathrm{T}}$ in frame i, the exposure time can be represented as $t(\mathbf{u}, i) = t_i + t_r \frac{u_y}{h}$, where t_i is the timestamp for frame i and h is the total number of rows in each frame. Here t_r is the readout time for each frame, which is usually about $60\% - 90\%$ of the time interval between frames.

There exists a constant delay t_d between the recorded timestamps of gyroscope and videos. Thus using the timestamps of gyroscopes as reference, the exposure time of pixel $\mathbf{u}$ in frame i should be modified as

$$t(\mathbf{u}, i) = t_i + t_d + t_r \frac{u_y}{h}. \tag{5}$$

To use the gyroscope readings we also need to know $\mathbf{q}_c$, the relative orientation of the camera in the gyroscope frame of reference (represented in unit quaternion). Finally, the bias of the gyroscope $\mathbf{b}_g$ needs to be considered. Therefore, in the online calibration we need to estimate the parameters f, c_x, c_y, κ_1, κ_2, t_r, t_d, $\mathbf{b}_g$ and $\mathbf{q}_c$.

IV. Coplanarity constraint for camera rotation

Our calibration and synchronization rely on the constraints applied to camera rotations.

A. Coplanarity constraint in global shutter cameras

First let us consider a global shutter camera in which all of the pixels in the same frame are captured at the same time. Assume the normalized 3D coordinate vectors of a certain feature in two viewpoints (frames) are $\mathbf{f}_i$ and $\mathbf{f}'_i$ (note that by (3) we cannot recover the absolute scale but only the direction of the 3D feature vector). The well-known epipolar constraint [16] is

$$(\mathbf{f}_i \times \mathbf{R}\mathbf{f}'_i) \cdot \mathbf{t} = 0, \tag{6}$$

where $\mathbf{R}$ and $\mathbf{t}$ are the relative rotation and translation between the two viewpoints. The epipolar constraint means that the vectors $\mathbf{f}_i$, $\mathbf{R}\mathbf{f}'_i$ and $\mathbf{t}$ are coplanar, as shown in Fig. 2. Now assume that three or more features are observed in these two viewpoints. By the epipolar constraint all vectors $\mathbf{f}_i \times \mathbf{R}\mathbf{f}'_i$ are perpendicular to the relative translation vector $\mathbf{t}$, and thus are coplanar ($\mathbf{t}$ is the normal vector of such plane). Such coplanarity can be expressed by the determinant of the 3×3 matrix composed by any three $\mathbf{f}_i \times \mathbf{R}\mathbf{f}'_i$ vectors being zero

$$\det[(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1)|(\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2)|(\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)] = 0. \tag{7}$$

This coplanarity was introduced in [15] and does not depend on the camera translation at all. Another desirable property of (7) is that it is still valid in the extreme case of zero translation since all vectors $\mathbf{f}_i \times \mathbf{R}\mathbf{f}'_i$ will become zero.

B. Coplanarity constraint in rolling shutter cameras

In rolling shutter cameras, however, the viewpoint is not unique for the features captured in the same frame. Here we propose a generalized coplanarity constraint for rolling shutter cameras.

First note that both the traditional epipolar constraint (6) and the coplanarity constraint (7) are expressed in terms of one of the two viewpoints. In fact, this frame of reference can be chosen arbitrarily. Once the reference is fixed, we can represent the camera orientation corresponding to any feature (determined by its exposure moment for rolling shutter cameras) in this reference. For the matched features between any two consecutive frames in rolling shutter cameras, we propose the following constraint

$$\det[(\mathbf{R}_1\mathbf{f}_1 \times \mathbf{R}'_1\mathbf{f}'_1)|(\mathbf{R}_2\mathbf{f}_2 \times \mathbf{R}'_2\mathbf{f}'_2)|(\mathbf{R}_3\mathbf{f}_3 \times \mathbf{R}'_3\mathbf{f}'_3)] = 0. \tag{8}$$

Note that in (8) $\mathbf{R}'_1$ means the camera orientation corresponding to feature 1 in the second frame, and not the transpose of $\mathbf{R}_1$. Constraint (8) does not exactly hold in general cases but only under the assumption that the relative camera translations between the exposure moments for all pair of matched features are in the same direction. The readout time of two consecutive frames is at most 66ms (for 30 fps videos) and in such short period of time the camera translation can be well approximated by a constant direction. Note that such approximation is more general than the approximation used in [25] which assumes the linear velocity (both direction and magnitude) of the camera is constant. The constraint is illustrated by Fig. 3. In Fig. 3, the first three (from left to right) frames of axes correspond to the three features detected in the current frame. The last three frames of axes correspond to the three matched features in the next frame. The different orientations of the frames of axes show the changes in camera rotation while the features are

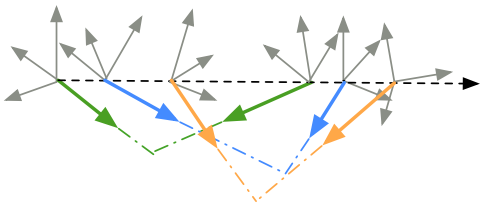


Fig. 3: Coplanarity constraint in rolling shutter cameras. The cross products of all pairs of matched features are perpendicular to the camera translation vector.

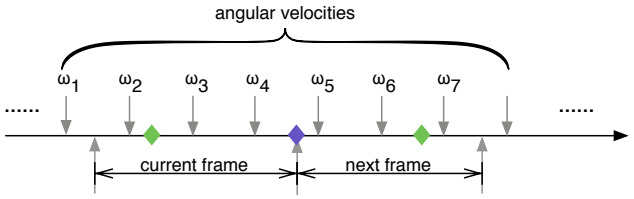


Fig. 4: Timing relationship between the gyroscope readings and the video frames.

exposed. The camera translation is approximated as the dashed ray. The three pairs of matched features are represented by green, blue, and orange arrows, respectively. By the proposed coplanarity constraint in rolling shutter cameras, the cross products of all pairs of matched features are perpendicular to the camera translation vector.

To make the constraint (8) more accurate we further apply such constraint only to groups of features that are not very far from each other in their y-axis coordinates. Based on (5) the exposure moments of features are close to each other if their y-axis coordinates are close. Assuming features $\mathbf{f}_1$, $\mathbf{f}_1'$, $\mathbf{f}_2$, $\mathbf{f}_2'$, $\mathbf{f}_3$, $\mathbf{f}_3'$ are selected in this way. Then the exposure moment difference among features in the same frame is much smaller compared to the exposure moment difference between features in adjacent frames ($\approx$ 33ms). In this way, the camera translation vectors for the three pairs of features naturally have almost the same direction. Constraint (8) is less dependent on the constant-direction assumption in camera translation between two consecutive frames.

We use the coplanarity constraint (8) as implicit measurement to estimate all the parameters in an EKF. The way to represent the camera orientation corresponding to each feature using the parameters and gyroscope readings is shown in the next section.

V. EKF-BASED ONLINE CALIBRATION AND SYNCHRONIZATION

The online calibration and synchronization is based on an extended Kalman filter. Our EKF evolves when every video frame is captured, as in [24]. The state vector is defined as

$$\mathbf{x} = [f\ c_x\ c_y\ \kappa_1\ \kappa_2\ t_r\ t_d\ \mathbf{b}_g^{\mathrm{T}}\ \mathbf{q}_c^{\mathrm{T}}]^{\mathrm{T}}. \tag{9}$$

The gyroscope in cellphones usually has a higher sampling rate than the video frame rate. Moreover, timestamps of gyroscope readings and the video frames are not aligned. We show how to compute the relative rotation corresponding to each detected feature using gyroscope readings as following

A. Computation of relative rotation

Fig. 4 illustrates the timing relationship between the gyroscope readings and the video frames. Assume a pair of matched features $\mathbf{f}_i$ and $\mathbf{f}_i'$ are detected as at moments denoted by green diamonds and the reference time is fixed as the timestamp of the next frame (shown as the purple diamond). The relative camera orientation between the reference time and the exposure moment of a certain feature can then be expressed by the angular velocities

$$\mathbf{R}_i = \prod_{n=1}^{M} \Theta(\boldsymbol{\omega}_n \Delta t_n^i), \tag{10}$$

where M is the total number of angular velocities involved in computing the relative orientation (M=7 for the example shown in Fig. 4) and Δt_n^i is the time duration that the angular velocity $\boldsymbol{\omega}_n$ is used in the integration (assuming constant angular velocity between readings). Note that not all of the M angular velocities have non-zero Δt_n^i values. For example, assume the timestamp of each angular velocity $\boldsymbol{\omega}_n$ is τ_n. Then for the feature in the next frame (right green diamond) in Fig. 4, only Δt_4^i, Δt_5^i and Δt_6^i are non-zero and they can be computed as

$$\begin{cases} \Delta t_4^i = \tau_5 - (T_{next} + t_d) \\ \Delta t_5^i = \tau_6 - \tau_5 \\ \Delta t_6^i = (T_{next} + t_d + t_r \frac{u_{y_i}}{h}) - \tau_6 \end{cases}, \tag{11}$$

where T_{next} is the framestamp of the next frame ($(T_{next}+t_d)$ corresponds to the moment for the purple diamond) and u_{y_i} is the y-axis coordinate of the feature ($(T_{next} + t_d + t_r \frac{u_{y_i}}{h})$ corresponds to the moment for the green diamond).

Each sub-relative rotation matrix can be computed by exponentiating the skew symmetric matrix formed by the product of angular velocity and its duration:

$$\Theta(\boldsymbol{\omega}_n \Delta t_n^i) = \exp(\mathrm{skew}(\boldsymbol{\omega}_n)\Delta t_n^i), \tag{12}$$

where

$$\mathrm{skew}(\boldsymbol{\omega}_n) = \begin{bmatrix} 0 & -\omega_{z_n} & \omega_{y_n} \\ \omega_{z_n} & 0 & -\omega_{x_n} \\ -\omega_{y_n} & \omega_{x_n} & 0 \end{bmatrix}. \tag{13}$$

Δt_n^i is determined by the exposure moments of $\mathbf{f}_i$ and $\mathbf{f}_i'$ computed using (5), and thus depends on the estimation variables t_r and t_d. The true angular velocities are represented as

$$\boldsymbol{\omega}_n = \hat{\boldsymbol{\omega}}_n + \mathbf{b}_g + \mathbf{n}_{g_n}, \tag{14}$$

where $\hat{\boldsymbol{\omega}}_n$ is the gyroscope reading, $\mathbf{b}_g$ is the gyroscope bias (to be estimated), and $\mathbf{n}_{g_n} \sim \mathcal{N}(0; \sigma_{\mathbf{g}})$ is the Gaussian distributed gyroscope measurement noise.

In this way, the relative camera orientation corresponding to any feature detected in the current and next frame can be expressed by the angular velocities.

B. State dynamics

All the parameters appeared in Section III except $\mathbf{b}_g$ are constant so they are just copied in state dynamics

$$\begin{cases} f(k+1) = f(k) \\ c_x(k+1) = c_x(k) \\ c_y(k+1) = c_y(k) \\ \kappa_1(k+1) = \kappa_1(k) \\ \kappa_2(k+1) = \kappa_2(k) \\ t_r(k+1) = t_r(k) \\ t_d(k+1) = t_d(k) \\ \mathbf{q}_c(k+1) = \mathbf{q}_c(k). \end{cases} \tag{15}$$

We model the dynamics of $\mathbf{b}_g$ by a random-walk process

$$\mathbf{b}_g(k+1) = \mathbf{b}_g(k) + \mathbf{m}_g(k), \tag{16}$$

where the random walk step $\mathbf{m}_g(k)$ is Gaussian distributed with zero mean and variance $\sigma_{\mathbf{b}}$.

C. State measurements

After features are matched between the current frame and the next frame, we picked N groups of features with three features in each group (without overlap). As mentioned in Section IV-B, to make the coplanarity constraint more accurate, the selection of groups of features are not completely random. The three features in the same group should have close y-axis coordinates but relatively far away x-axis coordinates.

In this way we can obtain N measurements from the coplanarity constraint shown in Section IV. For instance, the measurement formed by features 1,2 and 3 is

$$0 = \det[(\mathbf{R}_1\mathbf{f}_1 \times \mathbf{R}_1'\mathbf{f}_1')|(\mathbf{R}_2\mathbf{f}_2 \times \mathbf{R}_2'\mathbf{f}_2')|(\mathbf{R}_3\mathbf{f}_3 \times \mathbf{R}_3'\mathbf{f}_3')]. \tag{17}$$

The 3D feature locations $\mathbf{f}_i$ are computed by inverting the camera projection (3) as

$$\mathbf{f}_i = \mathbf{q}_c(k) \otimes \frac{1}{\varphi_i} \begin{bmatrix} (1+\kappa_1 r^2 + \kappa_2 r^4)(u_{x_i} + v_{x_i} - c_x(k)) \\ (1+\kappa_1 r^2 + \kappa_2 r^4)(u_{y_i} + v_{y_i} - c_y(k)) \\ f(k) \end{bmatrix}, \tag{18}$$

where $\mathbf{q}_c(k) \otimes (\cdot)$ means rotating a vector using 3D rotation defined by $\mathbf{q}_c(k)$, and φ_i is a normalization factor to make the result have unit norm. Besides normalization, there are two differences between (3) and (18): (a) we take the feature detection error $v_{x_i}, v_{y_i} \sim \mathcal{N}(0; \sigma_f)$ into account, and (b) the 3D feature is represented in the gyroscope coordinate system by multiplying the relative rotation estimate $\mathbf{q}_c(k)$ at stage k.

The relative rotation matrix $\mathbf{R}_i$ is computed according to (10). In this way, the right hand side of (17) can be expressed as a function of the state variables.

All of the N coplanarity constraints generates N implicit measurements at a stage k according to (17)

$$\begin{cases} 0 = z_1(k) = h_1(\mathbf{x}(k), \{\hat{\boldsymbol{\omega}}_n\}, \{\mathbf{u}_i\}, \{\mathbf{n}_{gn}\}, \{\mathbf{v}_i\}), \\ 0 = z_2(k) = h_2(\mathbf{x}(k), \{\hat{\boldsymbol{\omega}}_n\}, \{\mathbf{u}_i\}, \{\mathbf{n}_{gn}\}, \{\mathbf{v}_i\}), \\ \qquad \vdots \\ 0 = z_N(k) = h_N(\mathbf{x}(k), \{\hat{\boldsymbol{\omega}}_n\}, \{\mathbf{u}_i\}, \{\mathbf{n}_{gn}\}, \{\mathbf{v}_i\}), \end{cases} \tag{19}$$

where $\{\hat{\boldsymbol{\omega}}_n\}$ are the gyroscope readings during the exposure time of two consecutive frames, $\{\mathbf{u}_i\}$ are the 2D coordinates of all of the observed features. $\{\mathbf{n}_{gn}\}$ and $\{\mathbf{v}_i\}$ are the gyroscope measurement noise and feature observation noise, respectively, as shown in (14) and (18). Please note in the measurement equations the measurement noise appears implicitly as non-additive noise.

While the typical formulation of the EKF involves the assumption of additive measurement noise, this assumption is not necessary for EKF implementation. For the general form of observation

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k) \tag{20}$$

with Gaussian noise $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$, the innovation covariance is computed as

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^{\mathrm{T}}, \tag{21}$$

where $\mathbf{H}_k = \frac{\partial h}{\partial \mathbf{x}}\big|_{\hat{\mathbf{x}}_{k|k-1}}$, $\mathbf{V}_k = \frac{\partial h}{\partial \mathbf{v}}\big|_{\hat{\mathbf{x}}_{k|k-1}}$. The rest steps are the same as EKF with additive measurement noise. For details please see [31].

In our case, the measurement noise $\mathbf{v}_k$ consists of the gyroscope measurement noise and the feature observation noise. Its covariance $\mathbf{R}_k$ is a constant matrix.

The state update is performed right after state prediction. Only one round of state prediction and update is needed once a new frame is read and all features are tracked.

D. Extended Kalman filter computation

In EKF state vector estimate is predicted using dynamic equations and then updated using measurement equations. Prediction and Update rely on the Jacobian matrices of the dynamic and measurement equations with respect to the state vector and the system noise. The linear dynamic equations (15) and (16) lead to very simple Jacobian matrices (identity matrix). The Jacobian matrices of the measurement equations can also be computed analytically in closed-form. We show the derivations in Appendix A.

The camera-to-gyroscope orientation is represented by unit quaternion $\mathbf{q}_c$. Traditional extended Kalman filter cannot guarantee unit norm of the quaternion after estimate update. Therefore we use a minimal 3-element representation $\delta\boldsymbol{\theta}$ for the estimate error of $\mathbf{q}_c$ as in [32]. The true value of $\mathbf{q}_c$ can be represented as

$$\mathbf{q}_c = \delta\mathbf{q} \otimes \hat{\mathbf{q}}_c, \tag{22}$$

where $\hat{\mathbf{q}}_c$ is the estimate and

$$\delta\mathbf{q} = \begin{bmatrix} \delta\boldsymbol{\theta}/2 \\ \sqrt{1 - ||\delta\boldsymbol{\theta}/2||_2^2} \end{bmatrix}. \tag{23}$$

With such error representation we can update the estimate in a multiplicative way and guarantee the unit norm of the estimate. For more details please see [32].

In practice EKF update is executed every other frame (or less often to reduce complexity). The reason is that the measurement equation (17) involve features detected from two consecutive frames. If EKF is updated every frame then the features in each frame are used twice, which causes correlation between feature detection errors and the state estimate. One

can augment the state vector to track the feature detection errors. However, such augmentation will further increase the computational burden, while updating state estimate every other frame can easily avoid such correlation without augmenting the state vector.

E. State initialization

The state vector needs to be initialized carefully to make the EKF work properly. We initialize the principal point coordinates c_x, c_y to be the center of the frame. The focal length is initialized using the horizontal view angle provided by the smart phone operating system. If the operation system of the smartphone does not provide the value of horizontal view angle, SOG filters can be used with several initial guesses as in [18]. The readout time t_r is initialized as 0.0275 ms which is about 82.5% of the entire interval between frames. The coordinate system of the gyroscope is defined relative to the screen of the phone in its default orientation in all Android phones. Thus we can obtain the initial guess of $\mathbf{q}_c$ depending on whether we are using the front or rear camera. This initial guess is usually accurate enough, but our calibration is necessary since the camera is sometimes not perfectly aligned with the screen of the phone. The initial values of all other parameters (t_d and $\mathbf{b}_g$) are just set as 0.

To make sure that the true value lies in the 3σ intervals of the initial Gaussian distributions, we initialize the standard deviation of c_x, c_y, f, t_r as 6.67 pixels, 6.67 pixels, 20 pixels, and 0.00167 s, respectively. t_d is initialized as a sum of Gaussian distribution because of the highly non-linearity of the measurements with respect to t_d. The set of Gaussian distributions are initialized uniformly in the range of $\pm 30ms$. The standard deviation of each element in $\mathbf{b}_g$ is initialized as 0.006. The standard deviation of the estimate error of $\mathbf{q}_c$ is initialized as 0.5 degrees along each axis. The standard deviation of the radial distortion parameters κ_1 and κ_2 is initialized as 0.1. We set the standard deviation of gyroscope measurement noise and feature detection error as 0.003 rads/s and 1 pixels, respectively. The standard deviation of gyroscope measurement noise is determined from computing the reading variance while the cellphone is put still. Due to the sum-of-Gaussian initialization of t_d, we start from a SOG filter but it quickly converge to a single EKF using pruning of distributions with low weights [18].

VI. Experimental results

In this section we test the proposed algorithm with both Monte Carlo simulation and cellphone experiments.

A. Monte Carlo simulation

In the Monte Carlo simulation we randomly locate 1000 3D feature points distributed in range $X \in [-30, 30]$ meters, $Y \in [-20, 20]$ meters, $Z \in [30, 60]$ meters, respectively. The ground truth value of the parameters are set as $f = 690$ pixels, $c_x = 355$ pixels, $c_y = 220$ pixels, $\kappa_1 = 0.111$, $\kappa_2 = -0.303$, $t_r = 0.02$ s, $t_d = 0.02$ s, $\mathbf{q}_c = [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0, 0]^{\mathrm{T}}$ respectively. $\mathbf{b}_g$ is initialized as $[-0.008, 0.002, 0.017]^{\mathrm{T}}$ rads/s and then simulated by random walk. All of these values come from the parameters of a real cellphone camera. The ground truth motion of camera is fixed with a randomly generated sequence of angular velocities and linear velocities. The angular velocity and linear velocity sampling rate is set as 100 Hz. With the ground truth motion and camera/gyroscope parameters, we artificially generate a video with 250 frames at frame rate 30fps. Note that each video frame is not a real image but a sparse 2D point cloud.

In each trial of Monte Carlo simulation we generate Gaussian random gyroscope measurement noise and feature detection errors according to the variances shown in Section V-E. In this way, we can artificially add the noise and simulate the gyroscope readings and feature detections. Then we run EKF calibration in each trial, with state estimate initialized randomly within 3σ range around the ground truth values (note that this initialization method is different from that in Section V-E, which is used for cellphone experiments). In state update we use only 150 virtual features (50 measurements) picked from the feature pool.

We run 50 Monte Carlo trials to compare the proposed online estimation with the online estimation proposed in our earlier work [1]. The proposed estimation differs from [1] primarily in lens distortion modeling, Jacobian matrices computation ([1] computed them numerically) and selection of features ([1] selected the features completely randomly without considering the y-axis coordinate distance). We also compare the online calibration with a batch optimization using all of the frames. The batch optimization is solved via Levenberg-Marquardt algorithm [33].

Table I shows the root mean square (RMS) error of the parameter estimation before calibration and after calibration (with 250 frames). The estimation error of the gyroscope bias $\mathbf{b}_g$ is not shown since it is time-varying. The estimation error of $\mathbf{q}_c$ is converted to a single angle (computed as the L_2 norm of the minimal 3-element error representation). We find that batch optimization performs the best. The proposed EKF-based calibration method is also able to successfully converge to the ground truth value. With the modifications proposed in this paper, we can achieve a better calibration compared with [1]. Please note that although batch optimization gives the closest estimate, the EKF-based online calibration can be implemented in real time and enable immediate use of gyroscope in vision applications. More importantly, online calibration is able to deal with time-varying parameters, such as varying f due to zoom and varying t_d due to clock drift.

In Fig. 5 we show the estimation error along EKF-based calibration in one trial, with blue lines representing the estimation error and red lines representing the 99.7% (3σ) uncertainty bounds. For the relative orientation $\mathbf{q}_c$ we only show the estimation error after converting to a single angle as in table I. From Fig. 5 we can observe that the proposed method is able to accurately estimate the parameters.

B. Cellphone experiments

In our cellphone experiments, we use a Google Nexus S Android smartphone that is equipped with a three-axis

TABLE I: RMS error of 50 Monte Carlo simulation trials.

RMS error				
state variable	before calibration	batch optimization	online estimation	[1]
f (pixels)	17.3	0.910	2.28	3.70
c_x (pixels)	8.26	0.645	2.55	3.45
c_y (pixels)	6.55	0.576	0.96	2.40
t_r (ms)	1.8	0.031	0.078	0.15
t_d (ms)	17.1	0.027	0.041	0.089
$\mathbf{q}_c$ (degrees)	0.60	0.076	0.196	0.285
κ_1	0.099	0.0014	0.0042	N/A
κ_2	0.076	0.0026	0.0060	N/A

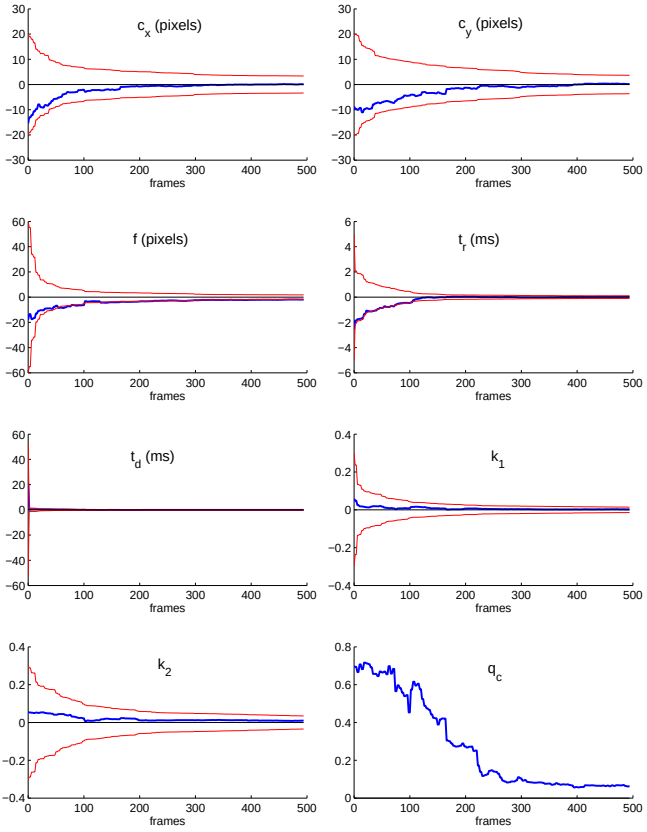
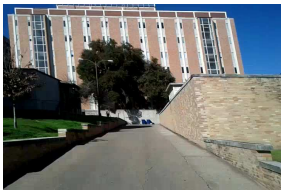


Fig. 5: Estimation error over time in one Monte Carlo simulation trial.

gyroscope. We capture the videos and the gyroscope readings from the cellphone and run the proposed online calibration and synchronization in MATLAB. The feature points are tracked using KLT tracker. We divide the frame into 4 equally sized bins and perform outlier rejection locally within each bin by computing a homography transformation using RANSAC [34], as in [35]. We estimate the ground truth of camera projection parameters (with lens distortion) using the offline camera calibration method in [6]. The ground truth of timestamp delay t_d is obtained by offline calibration in [12]. The ground truth of rolling shutter readout time t_r is obtained by batch optimization under pure rotational camera motion as in [11]. The estimated values are not guaranteed to be equal to the ground truth values so we only use them as a reference to

Running sequence



Panning sequence

Fig. 6: Examples of frames extracted from the test sequences.

TABLE II: Absolute estimation error for the running sequence.

Absolute estimation error		
state variable	before calibration	after online calibration
f (pixels)	26.5	4.16
c_x (pixels)	5.2	2.32
c_y (pixels)	13.57	1.50
t_r (ms)	3.72	0.21
t_d (ms)	13.2	0.14
κ_1	0.111	0.012
κ_2	0.303	0.045

roughly examine the accuracy of the proposed algorithm. We test the performance of the proposed method on various video sequences and show the results on two typical sequences: one shot while running forward and the other shot while panning the camera in front of a building. Fig. 6 shows two frames extracted from the two test sequences [1].

The running sequence (with 250 frames) is used to test the performance of the algorithm under arbitrary camera motion, including very high frequency shake and non-zero translation. The absolute estimation errors before and after online calibration and synchronization are shown in Table II. We can observe that the proposed method is able to estimate the parameters that are close to offline separate calibration.

In the second test video sequence (with 241 frames) we simply pan the camera in front of a building. This video is used to test the algorithm under (almost) zero camera translation (pure rotation). The estimation errors are shown in Table. III. The proposed algorithm works equally well compared to the running sequence.

To better display the difference before and after synchronization of the timestamps between video frames and gyroscope readings, we show the rates of 2D translation of pixels compared to the gyroscope data as in [11]. If we ignore the rolling shutter effect and the camera rotation around z-axis, the average rate of pixel translation can be approximated as

$$\begin{cases} \dot{u_x}(t) \approx f \cdot \omega_y(t + t_d) \\ \dot{u_y}(t) - \approx f \cdot \omega_x(t + t_d), \end{cases} \tag{24}$$

where $\omega_x(t)$ and $\omega_y(t)$ are angular velocities around x-axis and y-axis. These two angular velocity sequences can be obtained discretely from the gyroscope readings (after adding the gyroscope bias and transformed by $\mathbf{q}_c$). The pixel translation rate on the left hand side of (24) is approximated by finite

[1]The videos can be found at http://users.ece.utexas.edu/~bevans/papers/2015/autocalibration/.

TABLE III: Absolute estimation error for the panning sequence.

Absolute estimation error		
state variable	before calibration	after online calibration
f (pixels)	26.5	3.57
c_x (pixels)	5.2	1.04
c_y (pixels)	13.57	2.29
t_r (ms)	3.72	0.056
t_d (ms)	21.7	0.33
κ_1	0.111	0.014
κ_2	0.303	0.055

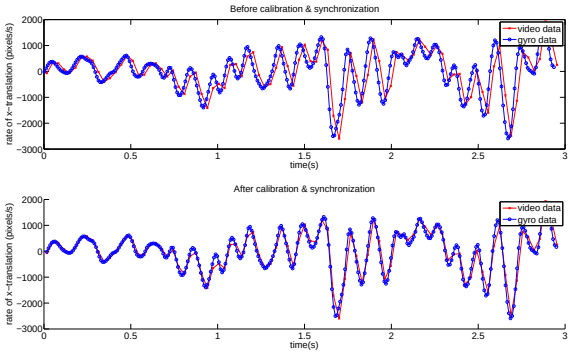


Fig. 7: Horizontal pixel translation rate $\dot{u_x}(t)$ (red) and $f \cdot \omega_y(t + t_d)$ (blue) for the running sequence.

differences between consecutive frames. In Fig. 7 and Fig. 8 we show the pixel translation rates and the angular velocities (right hand side of (24)) for the running sequence. We only plot a 3-second duration sequence in order to make the difference look more obvious. We can observe that after calibration and synchronization, the curve from video data and gyro data align much better, which indicates the effectiveness of the proposed algorithm.

In Fig. 9 and Fig. 10 we show the same comparison for the panning sequence. Again, the pixel translations computed from the video and gyroscope readings align very well after the proposed online calibration and synchronization.

C. Rolling shutter artifact rectification after calibration

We apply the proposed online calibration and synchronization algorithm in rectifying the rolling shutter artifact in video sequences. After calibration and synchronization the camera rotation can be directly obtained from gyroscope readings. The rolling shutter artifact is rectified by warping each row in the frame so that all of the rows are captured at the same moment (we fix this moment as the starting time of each frame). Fig. 11 and Fig. 12 show that the gyroscope readings can effectively correct the rolling shutter artifact after sensor calibration.

D. Run time

The current running speed of the proposed algorithm implemented in MATLAB (where feature detection and tracking are implemented using mex functions of an OpenCV implementation [36]) is 20.95 fps on a laptop with 2.3GHz Intel

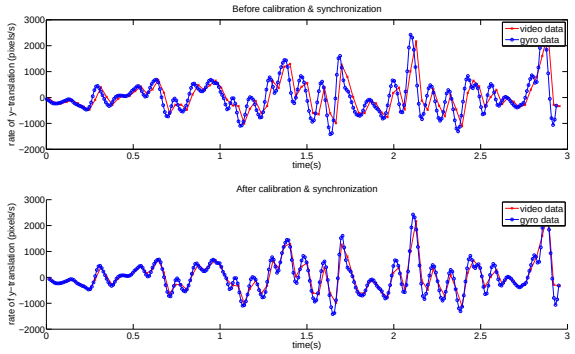


Fig. 8: Vertical pixel translation rate $\dot{u_y}(t)$ (red) and $-f \cdot \omega_x(t + t_d)$ (blue) for the running sequence.

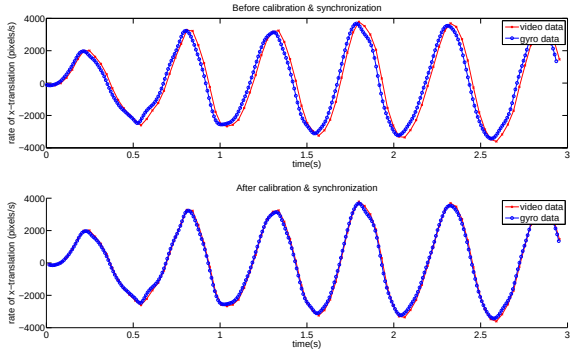


Fig. 9: Horizontal pixel translation rate $\dot{u_x}(t)$ (red) and $f \cdot \omega_y(t + t_d)$ (blue) for the panning sequence.

i5 processor. In our simulation, we had run the algorithm on every other pair of adjacent frames. However, we can run the calibration less often than using every other pair of adjacent frames, which allows a scaling back of the calculations to meet real-time constraints.

VII. Conclusions

In this paper we propose an online calibration and synchronization algorithm for cellphones that is able to estimate not only the camera projection parameters, but also the gyroscope bias, the relative orientation between the camera and gyroscope, and the delay between the timestamps of the two sensors. The proposed algorithm is based on the generalization of the coplanarity constraint of the cross products of matched features in a rolling shutter camera model. The proposed algorithm can also be naturally extended to a global shutter camera model by forcing the readout time for each frame t_r to be zero. Monte Carlo simulation and experiments run on real data collected from cellphones show that the proposed algorithm can successfully estimate all of the needed parameters with different kinds of motion of the cellphones. This online calibration and synchronization of rolling shutter camera and gyroscope make it more convenient for high quality video recording, gyro-aided feature tracking, and visual-inertial navigation.

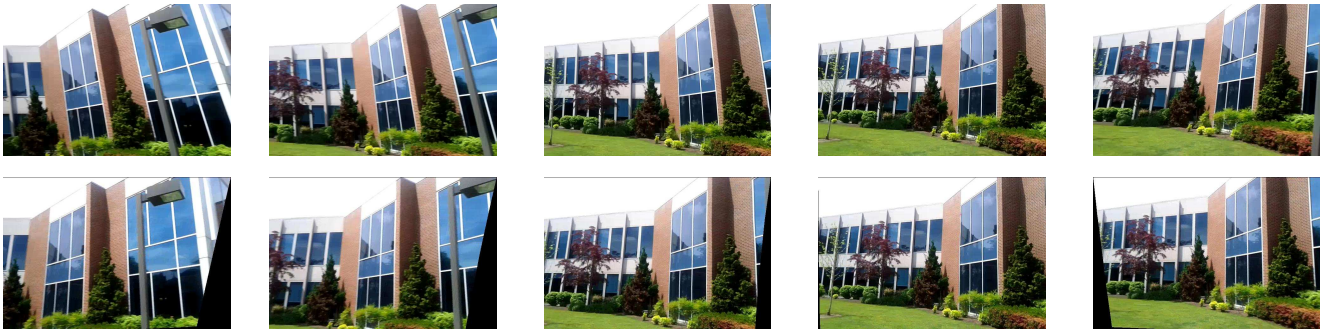


Fig. 11: Rolling shutter artifact rectification for the running sequence using the gyroscope readings after sensor calibration and synchronization. Top: five consecutive frames with rolling shutter artifact. Bottom: the rectified frames.

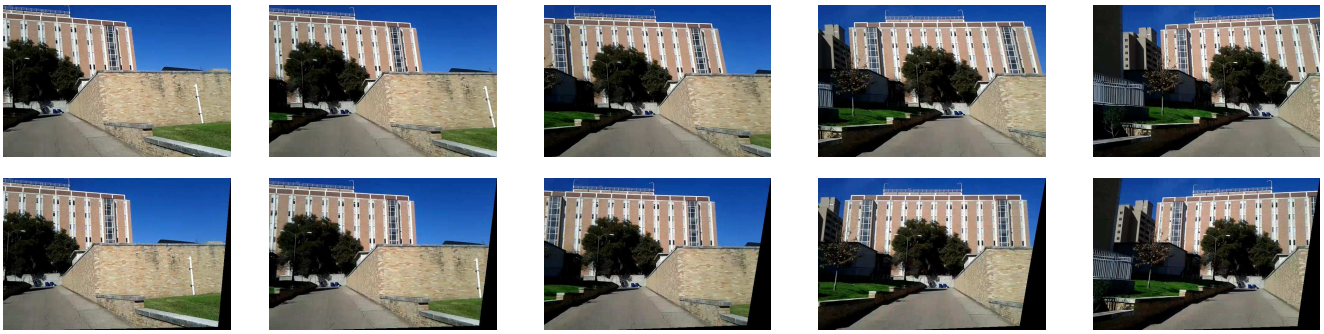


Fig. 12: Rolling shutter artifact rectification for the panning sequence using the gyroscope readings after sensor calibration and synchronization. Top: five consecutive frames with rolling shutter artifact. Bottom: the rectified frames.

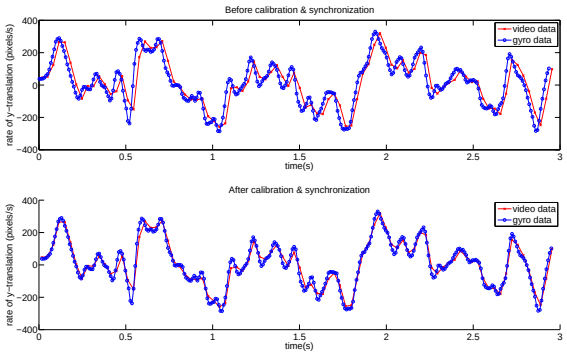


Fig. 10: Vertical pixel translation rate $\dot{u_y}(t)$ (red) and $-f \cdot \omega_x(t+t_d)$ (blue) for the panning sequence.

APPENDIX A
DERIVATION OF JACOBIAN MATRICES

In this appendix we derive how Jacobian matrices of the measurement equation can be computed analytically. As shown in (19), the measurement equation $h()$ can be decomposed into several independent components $\{h_j()\}$ for each single coplanarity constraint. Therefore, we only need to show $\frac{\partial h_j}{\partial \mathbf{x}}$ and $\frac{\partial h_j}{\partial \mathbf{v}}$, where $\mathbf{v}$ contains both gyroscope measurement noise and feature detection noise.

Each single measurement equation $h_j()$ can be represented in form of (17). Let $\mathbf{a}_i$ denote $\mathbf{R}_i\mathbf{f}_i$ and let $\mathbf{b}_i$ denote $\mathbf{R}'_i\mathbf{f}'_i$. Then we have

$$h_j(\mathbf{x},\mathbf{v}) = \det[(\mathbf{a}_1 \times \mathbf{b}_1)|(\mathbf{a}_2 \times \mathbf{b}_2)|(\mathbf{a}_3 \times \mathbf{b}_3)]. \quad (25)$$

$\frac{\partial h_j}{\partial \mathbf{x}}$ can be computed as $\sum_{i=1}^{3} \frac{\partial h_j}{\partial \mathbf{a}_i}\frac{\partial \mathbf{a}_i}{\partial \mathbf{x}} + \frac{\partial h_j}{\partial \mathbf{b}_i}\frac{\partial \mathbf{b}_i}{\partial \mathbf{x}}$. $\frac{\partial h_j}{\partial \mathbf{v}}$ can be computed in the same way. Without loss of generality, we only show how to compute $\frac{\partial h_j}{\partial \mathbf{b}_1}$, $\frac{\partial \mathbf{b}_1}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{b}_1}{\partial \mathbf{v}}$.

Based on the definition of matrix determinant we have

$$\frac{\partial h_j}{\partial \mathbf{b}_1} = [(\mathbf{a}_2 \times \mathbf{b}_2) \times (\mathbf{a}_3 \times \mathbf{b}_3)]^{\mathrm{T}}\mathrm{skew}(\mathbf{a}_1), \quad (26)$$

where skew() is defined as in (13).

To simplify the representation, we define

$$\mathbf{d}_1 = \begin{bmatrix} (1+\kappa_1 r^2+\kappa_2 r^4)(u'_{x_1}+v'_{x_1}-c_x) \\ (1+\kappa_1 r^2+\kappa_2 r^4)(u'_{y_1}+v'_{y_1}-c_y) \\ f \end{bmatrix} \quad (27)$$

and $\mathbf{e}_1 = \frac{1}{||\mathbf{d}_1||_2}\mathbf{d}_1$. Note that here

$$r = \sqrt{\left(\frac{u'_{x_1}+v'_{x_1}-c_x}{f}\right)^2 + \left(\frac{u'_{y_1}+v'_{y_1}-c_y}{f}\right)^2}. \quad (28)$$

In this way, we have

$$\mathbf{b}_1 = \mathbf{R}'_1(\mathbf{q}_c \otimes \mathbf{e}_1) \quad (29)$$

according to (17) and (18). The rotation matrix $\mathbf{R}'_1$ is not affected by the camera intrinsic parameters. So we have

$$\frac{\partial \mathbf{b}_1}{\partial c_x} = \mathbf{R}'_1[\mathbf{q}_c \otimes (\frac{\partial \mathbf{e}_1}{\partial \mathbf{d}_1}\frac{\partial \mathbf{b}_1}{\partial c_x})], \quad (30)$$

where

$$\frac{\partial \mathbf{b}_1}{\partial c_x} = \begin{bmatrix} -(2\kappa_1 + 4\kappa_2 r^2)\frac{(c_x - u'_{x_1} - v'_{x_1})^2}{f^2} - (1 + \kappa_1 r^2 + \kappa_2 r^4) \\ (2\kappa_1 + 4\kappa_2 r^2)\frac{(c_x - u'_{x_1} - v'_{x_1})(u'_{y_1} - v'_{y_1} - c_y)}{f^2} \\ 0 \end{bmatrix}. \tag{31}$$

Similarly we can obtain $\frac{\partial \mathbf{b}_1}{\partial c_y}$, $\frac{\partial \mathbf{b}_1}{\partial f}$, $\frac{\partial \mathbf{b}_1}{\partial \kappa_1}$ and $\frac{\partial \mathbf{b}_1}{\partial \kappa_2}$.

As mentioned in Section V-D, we use a minimal 3-element error representation $\delta\boldsymbol{\theta}$ for $\mathbf{q}_c$ and have

$$\frac{\partial \mathbf{b}_1}{\partial \delta\boldsymbol{\theta}} = -\mathbf{R}'_1 \text{skew}(\mathbf{q}_c \otimes \mathbf{e}_1). \tag{32}$$

For more details about the minimal 3-element error representation please see [32].

Recall that the rotation matrix $\mathbf{R}'_1$ can be computed as in (10)

$$\mathbf{R}'_1 = \prod_{n=1}^{M} \Theta(\boldsymbol{\omega}_n \Delta t_n). \tag{33}$$

Different from (10), (33) only contains angular velocities with non-zero Δt_n. Similar to (11) which shows how to compute Δt_n for the example shown in Fig. 4, we have

$$\begin{cases} \Delta t_1 = \tau_2 - (T + t_d) \\ \Delta t_n = \tau_{n+1} - \tau_n, n = 2, \cdots, M-1 \\ \Delta t_M = (T + t_d + t_r \frac{u'_{y_1}}{h}) - \tau_M \end{cases} \tag{34}$$

where T is the framestamp for the frame in which the feature $[u'_{x_1}, u'_{y_1}]^{\mathrm{T}}$ appears. Please note that t_d and t_r only affect the value of Δt_1 and Δt_M.

By defining $\Gamma_n = \prod_{m=1}^{n-1} \Theta(\boldsymbol{\omega}_m \Delta t_m)$ and $\gamma_n = [\prod_{m=n+1}^{M} \Theta(\boldsymbol{\omega}_m \Delta t_m)](\mathbf{q}_c \otimes \mathbf{e}_1)$, we have

$$\mathbf{b}_1 = \Gamma_n \Theta(\boldsymbol{\omega}_n \Delta t_n)\gamma_n, \forall n = 1, \cdots, M. \tag{35}$$

It is not difficult to show that

$$\frac{\partial \mathbf{b}_1}{\partial \Delta t_n} = -\Gamma_n \text{skew}(\gamma_n)\boldsymbol{\omega}_n. \tag{36}$$

Therefore, we can compute $\frac{\partial \mathbf{b}_1}{\partial t_d}$ and $\frac{\partial \mathbf{b}_1}{\partial t_r}$ as

$$\begin{cases} \frac{\partial \mathbf{b}_1}{\partial t_d} = -\Gamma_M \text{skew}(\gamma_M)\boldsymbol{\omega}_M + \Gamma_1 \text{skew}(\gamma_1)\boldsymbol{\omega}_1 \\ \frac{\partial \mathbf{b}_1}{\partial t_r} = -\frac{u'_{y_1}}{h}\Gamma_M \text{skew}(\gamma_M)\boldsymbol{\omega}_M. \end{cases} \tag{37}$$

Given (14) we can compute $\frac{\partial \mathbf{b}_1}{\partial \mathbf{b}_g}$ as

$$\frac{\partial \mathbf{b}_1}{\partial \mathbf{b}_g} = \sum_{n=1}^{M} \frac{\partial \mathbf{b}_1}{\partial \boldsymbol{\omega}_n}, \tag{38}$$

where

$$\frac{\partial \mathbf{b}_1}{\partial \boldsymbol{\omega}_n} = -\Delta t_n \Gamma_n \text{skew}(\gamma_n). \tag{39}$$

So far we have derived the derivative $\frac{\partial \mathbf{b}_1}{\partial \mathbf{x}}$ analytically as in (30), (31), (32), (37) and (38). Next we compute the derivative of $\mathbf{b}_1$ with respect to the measurement noise.

The gyroscope measurement noise $\{\mathbf{n}_{gn}\}$ appears in (33) through (14). As a result we have

$$\frac{\partial \mathbf{b}_1}{\partial \mathbf{n}_{gn}} = \frac{\partial \mathbf{b}_1}{\partial \boldsymbol{\omega}_n} = -\Delta t_n \Gamma_n \text{skew}(\gamma_n). \tag{40}$$

The feature detection noise $\{\mathbf{v}_i\}$ appears in (27). Also note that $\mathbf{b}_1$ is only affected by v'_{x_1} and v'_{y_1}. As a result we have

$$\begin{cases} \frac{\partial \mathbf{b}_1}{\partial v'_{x_1}} = \mathbf{R}'_1[\mathbf{q}_c \otimes (\frac{\partial \mathbf{e}_1}{\partial \mathbf{d}_1}\begin{bmatrix}1\\0\\0\end{bmatrix})] \\ \frac{\partial \mathbf{b}_1}{\partial v'_{y_1}} = \mathbf{R}'_1[\mathbf{q}_c \otimes (\frac{\partial \mathbf{e}_1}{\partial \mathbf{d}_1}\begin{bmatrix}0\\1\\0\end{bmatrix})] \end{cases}. \tag{41}$$

In this way, the derivative $\frac{\partial \mathbf{b}_1}{\partial \mathbf{v}}$ can be computed analytically.

ACKNOWLEDGMENT

The authors would like to thank Dr. Hamid Sheikh at Texas Instruments for introducing us to open research problems in video acquisition in cellphone cameras.

REFERENCES

[1] C. Jia and B. L. Evans, "Online calibration and synchronization of cellphone camera and gyroscope," in *Proc. IEEE Global Conf. Signal and Information Processing*, Dec. 2013.

[2] D. Strelow and S. Singh, "Motion estimation from image and inertial measurements," *Intl. Journal Robotics Research*, vol. 23, no. 12, pp. 1157–1195, 2004.

[3] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Intl. Conf. Robotics and Automation*, Apr. 2007.

[4] D. Strelow and S. Singh, "Online motion estimation from image and inerital measurements," in *Proc. Workshop Integration of Vision and Inertial Sensors*, Jun. 2003.

[5] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," *Intl. Journal Robotics Research*, vol. 26, no. 6, pp. 519–535, 2007.

[6] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[7] J. Lobo and J. Dias, "Relative pose calibration between visual and inertial sensors," *Intl. Journal Robotics Research*, vol. 26, no. 6, pp. 561–575, 2007.

[8] C. Geyer, M. Meingast, and S. Sastry, "Geometric models of rolling-shutter cameras," in *Proc. Workshop Omnidirectional Vision*, 2005.

[9] S. Baker, E. Bennett, K. S. B., and R. Szeliski, "Removing rolling shutter wobble," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2010.

[10] P.-E. Forssén and E. Ringaby, "Rectifying rolling shutter video from hand-held devices," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2010.

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

[12] M. Hwangbo, J.-S. Kim, and T. Kanade, "Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and GPU implementation," *Intl. Journal Robotics Research*, vol. 30, no. 14, pp. 1755–1774, 2011.

[13] G. Hanning, N. Forslöw, P.-E. Forssén, E. Ringaby, D. Törnqvist, and J. Callmer, "Stabilizing cell phone video using inertial measurement sensors," in *Proc. IEEE Intl. Workshop Mobile Vision*, Nov. 2011.

[14] B. Lucas and T. Kanade, "An iterative image registration technique with application to stereo vision," in *Proc. Intl. Joint Conf. Artificial Intelligence*, 1981, pp. 674–679.

[15] L. Kneip, R. Siegwart, and M. Pollefeys, "Finding the exact rotation between two images independently of the translation," in *Proc. European Conf. Computer Vision*, Oct. 2012.

[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[17] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proc. IEEE Intl. Conf. Computer Vision and Pattern Recognition*, Jun. 2013.

[18] J. Civera, D. Bueno, A. Davison, and J. Montiel, "Camera self-calibration for sequetial Bayesian structure from motion," in *Proc. IEEE Intl. Conf. Robotics and Automation*, 2009.

[19] S.-H. Jung and C. Taylor, "Camera trajectory estimation using inertial sensor measurements and structure from motion results," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, Dec. 2001, pp. 732–737.
[20] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. IEEE Intl. Conf. Computer Vision*, vol. 2, Oct. 2003, pp. 1403–1410.
[21] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Intl. Conf. Computer Vision and Pattern Recognition*, Apr. 2004.
[22] F. Mirzaei and S. Roumeliotis, "A Kalman filter-based algorithm for IMU-camera calibration," *IEEE Trans. Robotics*, vol. 24, no. 5, pp. 1143–1156, 2008.
[23] J. Kelly and G. Sukhatme, "Visual-inertial sensor fusion: localization, mapping and sensor-to-sensor self-calibration," *Intl. Journal Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.
[24] C. Jia and B. L. Evans, "Probabilistic 3-D motion estimation for rolling shutter video rectification from visual and inertial measurements," in *Proc. IEEE Intl. Workshop Multimedia Signal Processing*, Sep. 2012.
[25] M. Li, B. Kim, and A. Mourikis, "Real-time motion tracking on a cellphone using inertial sensing and a rolling shutter camera," in *Proc. IEEE Intl. Robotics and Automation*, May 2013.
[26] S. Lovegrove, A. Patron-Perez, and G. Sibley, "A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras," in *Proc. British Machine Vision Conf.*, Sep. 2013.
[27] M. Li, B. Kim, and A. Mourikis, "3-D motion estimation and online temporal calibration for camera-IMU systems," in *Proc. IEEE Intl. Robotics and Automation*, May 2013.
[28] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart, "Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence," in *Proc. IEEE Intl. Conf. Robotics and Automation*, May 2011.
[29] A. Martinelli, "Vision and IMU data fusion: closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 44–60, 2012.
[30] C.-K. Liang, L.-W. Chang, and H. Chen, "Analysis and compensation of rolling shutter effect," *IEEE Trans. Image Processing*, vol. 17, no. 8, pp. 1323–1330, Aug. 2008.
[31] D. Simon, *Optimal State Estimation*. John Wiley & Sons, 2006.
[32] N. Trawny and S. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," *Univ. of Minnesota Tech. Report*, 2005.
[33] J. More, "The Levenberg-Marquardt algorithm, implementation and theory," *Numerical Analysis*, pp. 105–116, 1977.
[34] M. Fischler and R. Bolles, "Random sample consensus, a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[35] M. Grundmann, V. Kwatra, D. Castro, and I. Essa, "Calibration-free rolling shutter removal," in *Proc. IEEE Intl. Conf. Computational Photography*, Apr. 2012.
[36] K. Yamaguchi, "mexopencv," http://www.cs.stonybrook.edu/~kyamagu/mexopencv/.

PLACE PHOTO HERE

Chao Jia Biography text here.

PLACE PHOTO HERE

Brian L. Evans Biography text here.