Optimization Based IMU Camera Calibration

Michael Fleps[†], Elmar Mair[†], Oliver Ruepp[†], Michael Suppa and Darius Burschka

[†]*The authors assert equal contribution and joint first authorship.*

***Abstract*— Inertia-visual sensor fusion has become popular due to the complementary characteristics of cameras and IMUs. Once the spatial and temporal alignment between the sensors is known, the fusion of measurements of these devices is straightforward. Determining the alignment, however, is a challenging problem. Especially the spatial translation estimation has turned out to be difficult, mainly due to limitations of camera dynamics and noisy accelerometer measurements. Up to now, filtering-based approaches for this calibration problem are largely prevalent. However, we are not convinced that calibration, as an offline step, is necessarily a filtering issue, and we explore the benefits of interpreting it as a batch-optimization problem. To this end, we show how to model the IMU-camera calibration problem in a nonlinear optimization framework by modeling the sensors' trajectory, and we present experiments comparing this approach to filtering and system identification techniques. The results are based both on simulated and real data, showing that our approach compares favorably to conventional methods.**

I. Introduction

Proper localization is a crucial issue in robotic applications. However, applications based on localization become more and more demanding regarding dynamics. Mobile robots move quicker and many applications are ported on hand-held devices. As a consequence, the need for sensors which are able to deal with such high dynamics increases permanently. The best choice to measure quick motions are inertial measurement units (IMUs), consisting of a gyroscope and an accelerometer for each of the three spatial axes.

While high quality IMUs are common, *e.g.*, in nautics and aeronautics applications, they are usually considered too expensive for robotic applications. The development of cheap gyros and accelerometers based on microelectromechanical systems (MEMS) reduced the cost of IMUs drastically and helped to introduce them to many new application areas. The drawback of these MEMS sensors is that they are quite prone to noise, and large rotations and accelerations are needed to produce measurements exhibiting a useful signal to noise ratio.

Cameras, on the other hand, have proven to provide an accurate static local and global localization by natural landmarks. However, they fail as soon as they are exposed to fast motions because of motion blur and limited frame-rates. The limitation of the frame-rate is mainly due to the large processing cost for images. Hence, cameras and IMUs are complementary sensors, which provide robust and reliable localization if correctly fused in a filter framework, like, *e.g.*, a Kalman filter. For slow motions, the camera provides a drift-free pose estimation, while for fast motions, the camera-based localization may fail, and the IMU is able to provide valuable measurements.

The complementary nature of these sensors allows for an effective combination of their measurements ([1], [2], [3], [4], [5]), which in turn requires accurate temporal and spatial registration between them. While the angular alignment between camera and gyroscopes can be computed in closed form, the translational alignment is rather difficult to compute. The only observable effect of the distance between the devices is a difference in the measured translation due to an acentric rotation axis. Fig. 1 sketches this effect of lever action. The translational measurements need to be extremely accurate in order to permit computation of the translational alignment with acceptable precision. This prerequisite is not met by the accelerometers, whose measurements are strongly corrupted by noise, bias, and the influence of gravitational acceleration. Furthermore, translation and rotation estimation based on the camera images can generally not be separated from each other, hence, errors in the rotation estimation also affect the computation of the translation.

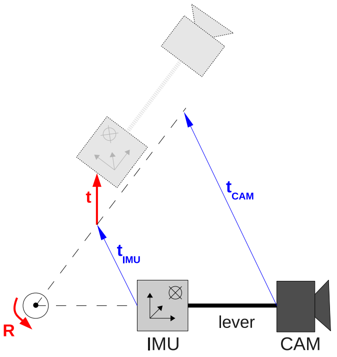


Fig. 1. The drawing sketches the motion of a camera-IMU setup. Both sensors measure the applied rotation (R) and translation (t). The only difference is the different lever action (t_{IMU}, t_{CAM}) due to the baseline introduced by the rotation.

The natural way to perform the calibration is by modeling the sensor's trajectory along with the spatial registration parameters. Most known approaches fuse the measurements in a Kalman filter and basically extend the state vector with

This work was partly supported by the German Aerospace Center (DLR) and by Technische Universität München.

E. Mair, O. Ruepp and D. Burschka are with the Department of Informatics, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany
{elmar.mair,ruepp,burschka}@cs.tum.edu

M. Fleps and M. Suppa are with the Institute for Robotics and Mechatronics, German Aerospace Center (DLR), Münchner Str. 20, 82234 Wessling, Germany
{michael.fleps,michael.suppa}@dlr.de

angular and translational alignment parameters. Thus, the alignment is approximated by the relaxed formulations of, *e.g.*, the extended or unscented Kalman filter (EKF, UKF), which only approximate the nonlinear relations. Another drawback of filter-based approaches is that the calibration parameters, which should ideally be constant over the whole sequence, are adapted continuously to best describe current measurements. Thereby, the influence of each measurement varies according to the system uncertainty. In case the sensors can be calibrated offline the data should not be processed sequentially in a filter, but the outcome should rather be optimized in a batch-like manner to consider all the information available equally. Therefore, we focus on an optimization framework in our approach, which permits a proper modeling of non-linearities and optimizes the spatial alignment over the whole sequence simultaneously.

In the next section we will sketch and discuss related approaches for IMU-camera calibration. In Section III we will describe our approach and in Section IV we explain how to get a proper parameter initialization. Finally, we will prove the functionality of our approach based on simulated and real data and compare it to state of the art solutions in Section V.

II. Related Work

Several visual-inertial calibration techniques already exist in the literature. The calibration problem is very complex and the underlying physical processes can only be modeled with limited accuracy. This makes it necessary to introduce certain assumptions which ultimately help to reduce the problem complexity. To this end, there are mainly two approaches that have been explored so far: Artificial, specialized measurement setups, which facilitate closed-form solutions, and filter-based approaches, which introduce several approximations.

Lobo and Dias make use of the gravitation vector, measured by the accelerometers, and a vertical calibration pattern to estimate the rotational alignment of the inertial-visual system [6], [7]. The translation between the devices is then estimated by using a turntable. The accelerometers have to lie in the center of the turntable so that they become the center of rotation. In this way simplified equations known from hand-eye calibration can be used to solve for the translation. While this approach has the advantage to be static and does not need dynamic motions, the necessary system setup is rather cumbersome and error-prone.

The most popular solution is to use an extended or unscented Kalman Filter (EKF, UKF) to estimate the sensor alignment. Approaches of this category implement the following states in their filter: position, orientation, linear and angular velocity, bias of the gyroscopes, bias of the accelerometer and finally translation and rotation between IMU and camera.

Kelly and Sukhatme present an UKF based calibration algorithm which can be based on artificial or natural landmarks [8], [4]. The idea is to allow a robot to simultaneously explore the environment while calibrating its sensors. If the calibration step is done offline the computational overhead of the UKF becomes irrelevant, while it provides a higher order approximation than the EKF and should, thus, be preferred.

Mirzaei and Roumelioutis register the camera and IMU by fusing the corner locations of a checkerboard in the camera images with the measurements of the gyroscopes and the accelerometers [9]. For that they use an error-state (indirect) Kalman Filter. An initialization stage to find good start values precedes the filtering. Further, they also explore the observability of the Kalman Filter, with the result that only rotations in two degrees of freedom are necessary to estimate the IMU camera transformation [10]. As benchmark for their experiments they use an optimization framework where they minimize the error between measurements and state estimates as described in [11]. The states and the covariance matrices are linearly propagated as within an EKF, which is similar to the approach described next.

Hol et al. [12] estimate the spatial alignment using a common system identification technique. The innovation in an EKF is minimized by standard gradient descent methods. Their experiments are promising, but one drawback is that the EKF linearizes the motion model which yields significant errors, especially with large rotations.

Our approach is also optimization-based, but contrary to the two previous methods we are not propagating the states and covariances throughout a measurement sequence, but we optimize based on the real nonlinear motion by modeling the trajectory of the sensors. Any filter-based estimation is performed sequentially instead of using the whole batch of data as it is the case in our solution. Hence, our registration results consider all available information at the same time and do not estimate the constant IMU-camera alignment on a sample-by-sample basis.

III. Description of the Algorithm

We propose to model the problem of determining the relative pose and orientation of camera and IMU as a nonlinear batch-optimization problem. Within this optimization framework, we seek to determine the trajectory of the unit together with the calibration parameters for some calibration data sequence. Since the problem is of very high complexity and many parameters and characteristics of the employed sensors are unknown, it is necessary to introduce reasonable assumptions.

In our case, the assumption is that the trajectory associated with the calibration sequence can be modeled by means of twice differentiable smooth parametric curves $\mathbf{p}(t)$ and $\mathbf{r}(t)$ describing position and orientation at time t, respectively. Quaternions will be used to represent orientations, thus $\mathbf{r}(t)$ is a four-dimensional curve, while $\mathbf{p}(t)$ is obviously three-dimensional.

The basic idea of our method is as follows: IMU and camera can be seen as sensors that deliver measurements that are a result of the same movement, observed from different coordinate systems. Given accurate measurements as well as accurate calibration data, we would be able to align the measurements based on the different base coordinate frames of the sensors. Conversely, we can evaluate the quality of calibration values by measuring the alignment between measurements. This leads directly to the idea of optimization of calibration parameters by maximizing the alignment.

To distinguish between the measurements, we are going to use successive superscripts C and I to denote samples obtained by the camera or IMU. The measurements delivered by the IMU are the rotational velocity vector $\boldsymbol{\omega}^I \in \mathbb{R}^3$ and the linear acceleration $\boldsymbol{a}^I \in \boldsymbol{R}^3$. Depending on whether the camera observes unknown or known landmarks, it can provide relative or even absolute measurements. For sake of generality we assume relative translational and rotational measurements, which we interpret as linear and rotational velocities, $\boldsymbol{v}^C \in \mathbb{R}^3$ and $\boldsymbol{\omega}^C \in \mathbb{R}^3$, by knowing the sampling period. To distinguish between the frames of reference, we are going to use leading superscripts C, I or G to denote quantities relative to the camera, IMU or the global reference frame, respectively. With ${}^I\boldsymbol{R}_C$ and ${}^I\boldsymbol{q}_C$, we denote the rotation matrix or rotation quaternion, respectively, which in this special case transforms 3D coordinates measured with respect to the camera frame to 3D coordinates within the IMU frame. The fourth element of the quaternions represents $\cos(\phi/2)$, where ϕ is the absolute rotation angle, and, thus, the only real parameter. Furthermore, we denote with ${}^I\boldsymbol{t}_{IC}$ the translation between camera and IMU frame expressed in the IMU frame.

A. *Objective Function*

Relationships between the measurements of the different sensors can be established according to [13], [14] as follows:

$$ {}^I\hat{\boldsymbol{\omega}}_t = {}^I\boldsymbol{\omega}_t^I - {}^I\boldsymbol{e}_{\boldsymbol{\omega}^I,t} = {}^I\boldsymbol{R}_C\left({}^C\boldsymbol{\omega}_t^C - {}^C\boldsymbol{e}_{\boldsymbol{\omega}^C,t}\right) \text{ and } \tag{1} $$

$$ \begin{aligned} {}^I\boldsymbol{a}_t^I - {}^I\boldsymbol{e}_{\boldsymbol{a}^I,t} + {}^I\boldsymbol{R}_G{}^G\boldsymbol{g} = {}^I\boldsymbol{R}_C\left({}^C\dot{\boldsymbol{v}}_t^C - {}^C\dot{\boldsymbol{e}}_{\boldsymbol{v}^C,t}\right) - \\ {}^I\hat{\boldsymbol{\omega}}_t \times \left({}^I\hat{\boldsymbol{\omega}}_t \times {}^I\boldsymbol{t}_{IC}\right) - {}^I\dot{\hat{\boldsymbol{\omega}}}_t \times {}^I\boldsymbol{t}_{IC} \end{aligned} \tag{2} $$

In above formulæ, ${}^I\hat{\boldsymbol{\omega}}_t$ denotes the real, error-free angular velocity, $\boldsymbol{e}_x$ represents the error of a specific measurement x, and ${}^G\boldsymbol{g}$ is the gravitation vector $(0, 0, -9.81)^T$. Measurements of cheap MEMS gyroscopes and accelerometers are not only affected by white noise, but also by a bias, which has to be modeled explicitly. We will denote the bias values for gyroscope and accelerometer by $\boldsymbol{b}_{\boldsymbol{\omega}^I}$ and $\boldsymbol{b}_{\boldsymbol{a}^I}$, respectively. Since the bias values change extremely slowly over time, we adopt the common assumption they are constant.

Unfortunately, a direct comparison as outlined above is not possible, because the sensor measurements occur at different time instances. Hence, we need a model which allows to interpolate the motion at arbitrary points in time. In our optimization approach, we want to minimize the differences between the measurements and, thus, we aim to minimize the error between all samples and a motion model which inherently interpolates these measurements. The measurement errors, $\boldsymbol{\delta}$, can be formulated by following equations:

$$ \begin{aligned} \boldsymbol{\delta}_{\boldsymbol{\omega}^I,i} &= {}^I\boldsymbol{\omega}_i^I - \boldsymbol{b}_{\boldsymbol{\omega}^I} - \boldsymbol{\omega}(t_i) & (3) \\ \boldsymbol{\delta}_{\boldsymbol{a}^I,i} &= {}^I\mathbf{a}_i^I - \boldsymbol{b}_{\boldsymbol{a}^I} + {}^I\mathbf{r}(t_i) \odot \left(\boldsymbol{I}_{vq}^T{}^G\boldsymbol{g}\right) \odot {}^I\bar{\mathbf{r}}(t_i) - \\ &\qquad {}^I\ddot{\mathbf{p}}(t_i) & (4) \\ \boldsymbol{\delta}_{\boldsymbol{\omega}^C,i} &= {}^I\boldsymbol{R}_C{}^C\boldsymbol{\omega}_i^C - \boldsymbol{\omega}(t_i) & (5) \\ \boldsymbol{\delta}_{\boldsymbol{v}^C,i} &= {}^I\boldsymbol{R}_C{}^C\boldsymbol{v}_i^C + \boldsymbol{\omega}(t_i) \times {}^I\boldsymbol{t}_{IC} - {}^I\dot{\boldsymbol{p}}(t_i) & (6) \end{aligned} $$

with

$$ \begin{aligned} \boldsymbol{\omega}(t_i) &= 2\,\boldsymbol{I}_{vq}\left({}^I\bar{\mathbf{r}}(t) \odot {}^I\dot{\mathbf{r}}(t)\right), & (7) \\ \boldsymbol{I}_{vq} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, & (8) \\ {}^I\bar{\mathbf{r}}(t) &= \operatorname{diag}\begin{pmatrix} -1 & -1 & -1 & 1 \end{pmatrix} {}^I\mathbf{r}(t) & (9) \end{aligned} $$

and t_i representing the measurement time of sample i. The operator $\odot$ denotes the quaternion multiplication. Note that the formulæ are simpler if the trajectory is modeled relative to the IMU frame. The only measurement revealing information about the real orientation of the IMU relative to the world coordinate frame is the gravity vector. However, this becomes irrelevant if we also estimate the orientation of the gravity vector relative to the initial pose of the IMU ${}^{I_0}\boldsymbol{g}$. Hence, the trajectory starts at the origin of the IMU coordinate frame, aligned to its axes and can be estimated independently of the global coordinate frame. ${}^G\boldsymbol{g}$ in Eq. 4 can now be replaced by ${}^{I_0}\boldsymbol{g}$.

The errors are weighted with the pseudo Huber cost function $\mathrm{h}(\boldsymbol{\delta})$ [15], which is differentiable and robust against outliers. It is defined as

$$ \mathrm{h}(\boldsymbol{\delta}_{x,i}) = 2b_x^2\left(\sqrt{\frac{\boldsymbol{\delta}_{x,i}^T\boldsymbol{\delta}_{x,i}}{b_x^2} + 1} - 1\right). \tag{10} $$

A common choice for b_x is $3\sigma_x^2$, where σ_x^2 denotes the noise variance and x the kind of measurements for $\boldsymbol{\omega}^I$, $\boldsymbol{a}^I$, $\boldsymbol{\omega}^C$ or $\boldsymbol{v}^C$, respectively.

Concatenating the matrices of all the error vectors $\boldsymbol{\Delta}_x = (\boldsymbol{\delta}_{x,1} \ldots \boldsymbol{\delta}_{x,N_x})$, with N_x the respective number of measurements, yields the total error matrix $\boldsymbol{\Delta} = (\boldsymbol{\Delta}_{\boldsymbol{\omega}^I}\ \boldsymbol{\Delta}_{\boldsymbol{a}^I}\ \boldsymbol{\Delta}_{\boldsymbol{\omega}^C}\ \boldsymbol{\Delta}_{\boldsymbol{v}^C})$. Thus, the objective function $\mathrm{g}(\boldsymbol{\Delta})$ can be formulated as

$$ \mathrm{g}(\boldsymbol{\Delta}) = \sum_{x \in \{\boldsymbol{\omega}^I, \boldsymbol{a}^I, \boldsymbol{\omega}^C, \boldsymbol{v}^C\}} \left(\frac{1}{\sigma_x^2}\sum_{i=1}^{N_x}\mathrm{h}(\boldsymbol{\delta}_{x,i})\right). \tag{11} $$

B. *B-Spline*

The continuous, twice differentiable trajectory curve model used in our approach is a cubic B-spline curve. Using this curve model certainly constitutes a restriction, since it imposes a number of constraints on the trajectory. Note however, that it is still more general than the commonly encountered assumption of piecewise linearity of motion, which is usually made in filter-based approaches. We are now going to introduce our notation used for B-spline curves. For more detailed information about B-splines, see [16] or [17]. We define a B-spline curve as

$$\mathrm{s}(t) = \sum_{i=1}^{M} \boldsymbol{c}_i \mathrm{b}_i^k(t), \tag{12}$$

where $\mathbf{c}_i \in \mathbb{R}^d$ are the d-dimensional control point values, b_i^k denotes the i-th basis function of order k and M is the number of knots. The B-spline order k can be chosen arbitrarily as long as it is at least four. This is required because the trajectory curve has to be at least twice continuously differentiable, since we also need to compute the acceleration from the position spline. We use a cubic B-spline that is of order $k = 4$, which is equivalent to a piecewise linear approximation of the acceleration measurements. With $\mathbf{c}$, we denote the vector $(\boldsymbol{c}_1^T \, \boldsymbol{c}_2^T \ldots \boldsymbol{c}_M^T)^T \in \mathbb{R}^{Md}$ of concatenated control point values. We assume that an equidistant knot sequence is used.

It is well-known that B-splines are linear in parameters, which means that the evaluation of above formula at several parameter locations $\boldsymbol{t} = (t_0 \, t_1 \ldots t_n)$ is equivalent to computing the matrix-vector product $\mathbf{B}\,(\boldsymbol{t})\,\boldsymbol{c}$ for a suitable basis matrix $\mathbf{B}(\boldsymbol{t})$. More formally, this is expressed as

$$(\mathrm{s}(t_1)^T \, \mathrm{s}(t_2)^T \ldots \mathrm{s}(t_n)^T)^T = \mathbf{B}(\boldsymbol{t})\boldsymbol{c}. \tag{13}$$

For d-dimensional control point values, the basis matrix has the shape

$$\mathbf{B}(\boldsymbol{t}) = \begin{pmatrix} \mathrm{b}_1^k(t_1) & \mathrm{b}_2^k(t_1) & \ldots & \mathrm{b}_m^k(t_1) \\ \mathrm{b}_1^k(t_2) & \mathrm{b}_2^k(t_2) & \ldots & \mathrm{b}_m^k(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{b}_1^k(t_n) & \mathrm{b}_2^k(t_n) & \ldots & \mathrm{b}_m^k(t_n) \end{pmatrix} \otimes \mathbf{I}_d, \tag{14}$$

where $\otimes$ denotes the Kronecker matrix product, and $\mathbf{I}_d$ is the $d \times d$ identity matrix. It is obvious that if the vector of parameter locations $\boldsymbol{t}$ remains constant, so does the matrix $\mathbf{B}$. The time-stamps of the measurements are constant and, hence, the matrix $\mathbf{B}$ has to be computed only once. Furthermore, it is well-known that B-spline derivatives are again B-splines, and as such are again linear in parameters. In our optimization process, we are going to evaluate the spline and its derivatives at the time associated with the time stamps. This means that spline derivatives can also be computed by simply evaluating $\mathbf{B}_t\boldsymbol{c}$ for some appropriate matrix $\mathbf{B}_t$ representing the basis matrix of the derived spline.

In our implementation we need a B-spline of dimension $d = 7$ and thus $\boldsymbol{c}_i \in \mathbb{R}^7$ to model the IMU pose $\mathrm{s}(t) = (\mathbf{p}^T(t) \;\; \mathbf{r}^T(t))^T$. Note that the quaternions are constrained to be of unit length, as described in Section III-C, which yields the expected six degrees of freedom for rigid body motion.

The control point vector of the B-spline is part of the parameter vector $\boldsymbol{\theta}$ subject to optimization. Further, this vector contains the two IMU bias terms, the initial direction of the gravity vector, the quaternion and the vector describing the translation between the IMU and camera coordinate system. For sake of generality, we also model the scale factor α of the measured camera velocity, assuming that natural landmarks are used and the scale of translation is unknown and, thus, is also estimated in our optimization.

$$\boldsymbol{\theta} = \left((\boldsymbol{c}_1^T \boldsymbol{c}_2^T \ldots \boldsymbol{c}_M^T) \;\; b_{\omega^I} \;\; b_{a^I} \;\; {}^I\boldsymbol{q}_C^T \;\; {}^{I_0}\boldsymbol{g} \;\; {}^I\boldsymbol{t}_{IC}^T \;\; \alpha \right)^T \tag{15}$$

C. Constraints and Optimization Details

There are some constraints on a few parameters which have to be satisfied for the optimization. The unit property of the control points of the B-spline and of ${}^I\boldsymbol{q}_C$ has to be ensured. Further, the gravity vector has to be of length 9.81 and the first control point of the spline is constrained to represent zero rotation and zero translation to avoid dependencies in the optimization parameters. This is because both the direction of the gravity vector and the pose of the IMU are going to be optimized – at the beginning of the trajectory one of these parameters has to be fixed to prevent redundant degrees of freedom. All these requirements can be formulated as equality constraints which have to be zero:

$$\boldsymbol{r}_{eq,1} = \sum_{i=1}^{M} (\|\boldsymbol{c}_i\|) - M \tag{16}$$

$$\boldsymbol{r}_{eq,2} = \|{}^I\boldsymbol{q}_C\| - 1 \tag{17}$$

$$\boldsymbol{r}_{eq,3} = \|{}^{I_0}\boldsymbol{g}\| - 9.81 \tag{18}$$

$$\boldsymbol{r}_{eq,4} = \boldsymbol{c}_1 - \begin{pmatrix} \boldsymbol{t}_0 \\ \boldsymbol{q}_0 \end{pmatrix} \tag{19}$$

with $\boldsymbol{t}_0 = \mathbf{0}^{3\times 1}$ and $\boldsymbol{q}_0 = (0, 0, 0, 1)^T$.

We use sequential quadratic programming (SQP) as optimization algorithm, because it is known to work well for nonlinear optimization problem and it allows to implement equality contraints [18].

For optimization purposes, it is generally necessary to compute the gradient of the objective function, as well as an appropriate Hessian approximation. The gradient of the objective function can be computed by applying the chain rule to Eq. 11 as

$$\frac{\partial \mathrm{g}(\boldsymbol{\Delta})}{\partial \boldsymbol{\theta}} = \boldsymbol{J} \sum_{x \in \{\boldsymbol{\omega}^I, \boldsymbol{a}^I, \boldsymbol{\omega}^C, \boldsymbol{v}^C\}} \left(\frac{1}{\sigma_x^2} \sum_{i=1}^{N_x} \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\Delta}} \right), \tag{20}$$

with $\boldsymbol{J}$ being the Jacobian of the error matrix $\boldsymbol{\Delta}$ relative to the parameter vector $\boldsymbol{\theta}$. The derivative of the pseudo Huber cost function is stacked by $\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\Delta}} = \left(\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\Delta}_{\boldsymbol{\omega}^I}}^T \; \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\Delta}_{\boldsymbol{a}^I}}^T \; \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\Delta}_{\boldsymbol{\omega}^C}}^T \; \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\Delta}_{\boldsymbol{v}^C}}^T \right)^T$ with $\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\Delta}_x} = \left(\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\delta}_{x,1}}^T \ldots \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\delta}_{x,N_x}}^T \right)^T$ and

$$\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x,i})}{\partial \boldsymbol{\delta}_{x,i}} = \frac{-2}{\sqrt{\frac{\boldsymbol{\delta}_{x,i}^T \boldsymbol{\delta}_{x,i}}{b_x^2} + 1}} \, \boldsymbol{\delta}_{x,i} \; . \tag{21}$$

The Jacobian $\boldsymbol{J}$ consists of following components

$$\boldsymbol{J} = \frac{\partial \boldsymbol{\Delta}}{\partial \boldsymbol{\theta}} = \left(\boldsymbol{J}_{\boldsymbol{c}} \; \boldsymbol{J}_{\boldsymbol{b}_{\omega^I}} \; \boldsymbol{J}_{\boldsymbol{b}_{a^I}} \; \boldsymbol{J}_{{}^I\boldsymbol{q}_C} \; \boldsymbol{J}_{{}^{I_0}\boldsymbol{g}} \; \boldsymbol{J}_{{}^I\boldsymbol{t}_{IC}} \; \boldsymbol{J}_{\alpha} \right) , \tag{22}$$

which can be computed in a straight-forward manner. The main part of the Jacobian consists of the control points $\boldsymbol{J}_{\boldsymbol{c}} = (\boldsymbol{J}^T_{\boldsymbol{c},\boldsymbol{\omega}^I}\ \boldsymbol{J}^T_{\boldsymbol{c},\boldsymbol{a}^I}\ \boldsymbol{J}^T_{\boldsymbol{c},\boldsymbol{\omega}^C}\ \boldsymbol{J}^T_{\boldsymbol{c},\boldsymbol{v}^C})^T$ where each $\boldsymbol{J}_{\boldsymbol{c},x}$ looks like

$$\boldsymbol{J}_{\boldsymbol{c},x} = \begin{pmatrix} \frac{\partial \boldsymbol{\delta}_{x,1}}{\partial \boldsymbol{c}_1} & \frac{\partial \boldsymbol{\delta}_{x,1}}{\partial \boldsymbol{c}_2} & \cdots & \frac{\partial \boldsymbol{\delta}_{x,1}}{\partial \boldsymbol{c}_M} \\ \frac{\partial \boldsymbol{\delta}_{x,2}}{\partial \boldsymbol{c}_1} & \frac{\partial \boldsymbol{\delta}_{x,2}}{\partial \boldsymbol{c}_2} & \cdots & \frac{\partial \boldsymbol{\delta}_{x,2}}{\partial \boldsymbol{c}_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{\delta}_{x,N_x}}{\partial \boldsymbol{c}_1} & \frac{\partial \boldsymbol{\delta}_{x,N_x}}{\partial \boldsymbol{c}_2} & \cdots & \frac{\partial \boldsymbol{\delta}_{x,N_x}}{\partial \boldsymbol{c}_M} \end{pmatrix} \tag{23}$$

and each $\frac{\partial \boldsymbol{\delta}_{x,i}}{\partial \boldsymbol{c}_j}$ consists of two parts:

$$\frac{\partial \boldsymbol{\delta}_{x,i}}{\partial \boldsymbol{c}_j} = \begin{pmatrix} \frac{\partial \boldsymbol{\delta}_{x,i}}{\partial \boldsymbol{c}_{p,j}} & \frac{\partial \boldsymbol{\delta}_{x,i}}{\partial \boldsymbol{c}_{r,j}} \end{pmatrix} \tag{24}$$

with $\boldsymbol{c}_{p,j}$ and $\boldsymbol{c}_{r,j}$ the position and rotation component of the control points modeling $\mathbf{p}(t_j)$ and $\mathbf{r}(t_j)$, respectively. For approximating the Hessian of the system, the popular BFGS method [18] is used.

IV. Initialization

The initialization of the optimization parameters is a three-step procedure. First of all, we find the time offset between the IMU and camera measurements and an initial estimate of the orientation between IMU and camera coordinate frame ${}^I\hat{\boldsymbol{q}}_C$ similar to [19].

The second step is to determine the number of B-spline control points to be used. A high number of control points means less smoothing and higher flexibility, but also increases the complexity of the computation, and the possibility of over-fitting. For our experiments we use $0.6\,N^C$ control points, where N^C denotes the number of camera measurements. As we will show in the experiments this amount provides a good compromise between computational efficiency and accuracy. However, a more general solution would be to evaluate the measurements of the accelerometers and the gyroscopes to adapt the knot vector and the number of control points to satisfy the requirements given by the motion dynamics.

The final step is the initialization of the spline control parameter vector $\boldsymbol{c}$. A good initial estimate can be found by calculating ${}^I\hat{\mathbf{r}}(t) = {}^I\hat{\boldsymbol{q}}_G$ and ${}^I\hat{\mathbf{p}} = {}^I\hat{\boldsymbol{t}}_{IG}$ from the camera measurements according to the following equations:

$$\begin{aligned} {}^G\hat{\mathbf{q}}_C(t_i) &= \prod_{j=1}^{i}{}_{\odot}\ \mathbf{q}({}^C\boldsymbol{\omega}^C_j T_C) & (25)\\ {}^I\hat{\mathbf{q}}_G(t_i) &= {}^I\hat{\boldsymbol{q}}_C \odot {}^G\bar{\hat{\mathbf{q}}}_C(t_i) & (26)\\ {}^I\hat{\mathbf{t}}_{IG}(t_i) &= {}^I\hat{\boldsymbol{t}}_{IC} - {}^I\hat{\mathbf{R}}_G(t_i)\sum_{j=1}^{i}{}^G\hat{\mathbf{R}}_C(t_j)\,{}^C\mathbf{v}^C_j\,T_C & (27) \end{aligned}$$

whereas $i \in \{1..N^C\}$, T_C is the time period between consecutive camera measurements and $\mathbf{q}(\boldsymbol{p})$ represents the transformation of an Euler vector $\boldsymbol{p}$ to its corresponding quaternion. The bar over a quaternion $\bar{\boldsymbol{q}}$ denotes the inverse rotation as described in Eq. 9. For the sake of clarity we used ${}^I\hat{\mathbf{R}}_G(t_i)$ and ${}^G\hat{\mathbf{R}}_C(t_j)$ in Eq. 27 as the DCM representation of ${}^I\hat{\mathbf{q}}_G(t_i)$ and ${}^G\hat{\mathbf{q}}_C(t_j)$, respectively.

Subsequently, the B-spline can be approximated by following least-squares fitting equation:

$$\hat{\boldsymbol{c}} = \left(\mathbf{B}(\boldsymbol{t}^C)^T\mathbf{B}(\boldsymbol{t}^C)\right)^{-1}\mathbf{B}(\boldsymbol{t}^C)^T\begin{pmatrix}\hat{\mathbf{p}}(\boldsymbol{t}^C)\\ \hat{\mathbf{r}}(\boldsymbol{t}^C)\end{pmatrix} \tag{28}$$

with $\boldsymbol{t}^C = (t_1 \ldots t_{N^C})$.

The last step is the initialization of the gravity vector ${}^{I_0}\boldsymbol{g}$, the bias terms $\boldsymbol{b}_{\boldsymbol{\omega}^I}$ and $\boldsymbol{b}_{\boldsymbol{a}^I}$ and the relative pose vector ${}^I\boldsymbol{t}_{IC}$. Both bias values are initialized with zero vectors. A first estimate for ${}^I\boldsymbol{t}_{IC}$ can either be extracted from a CAD drawing or in case there is no drawing or other reliable information it can be set to zero too. In general there is no prior knowledge of the initial orientation of the gravity vector ${}^{I_0}\boldsymbol{g}$ and thus, we assume that the acceleration during the first few IMU measurements is negligible small so that the sensor measurements consists mainly of the gravity force. Thus, an initial estimate can be achieved by computing the negative mean of the first few, L, samples:

$${}^{I_0}\hat{\boldsymbol{g}} = -\frac{1}{L}\sum_{i=1}^{L}{}^I\boldsymbol{a}^I_i \tag{29}$$

If there is no knowledge about the used landmarks, the scale factor can be of any size. One approach to find a proper initial guess is to use the maximum measured velocity $\boldsymbol{v}^C$ of unknown scale and choose an α which yields a reasonable maximum velocity as it can be assumed during the data acquisition. If the landmarks are known also the scale is known ($\alpha = 1$) and, hence, can be removed from the parameter vector.

V. Experiments

In the following plots the X-axis components are colored blue and the Y- and Z-components are in green and red, respectively. The IMU measurements are marked with dots and the ones of the camera with crosses.

A. Simulated Data

First, we will show the results of our approach on simulated data to evaluate its performance based on ground truth. In our simulation we assume that the camera provides images with a frame rate of 15 Hz, while the IMU measurements arrive with a rate of 120 Hz. The length of the simulated registration sequence is about 10 s, which is similar to our real registration sequences. The simulated rotational and translational motion corresponds to three superimposed sine waves with varying frequencies and amplitudes for each degree of freedom. The signals are corrupted by noise with the following standard deviations, which have been chosen to be similar to the ones measured in the real data: $\sigma_{\boldsymbol{\omega}^I} = 28.6\,°/\text{s}$, $\sigma_{\boldsymbol{a}^I} = 0.5\,\text{m/s}^2$, $\sigma_{\boldsymbol{\omega}^C} = 4.3\,°/\text{s}$ and $\sigma_{\boldsymbol{v}^C} = 0.05\,\text{m/s}$. The covariance matrix Σ_P of the estimated parameters is calculated according to [15] by

$$\boldsymbol{\Sigma}_P = \left(\boldsymbol{J}^T\boldsymbol{\Sigma}_X^{-1}\boldsymbol{J}\right)^+ \tag{30}$$

with $\boldsymbol{\Sigma}_X = \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\omega}^I}\ \boldsymbol{\sigma}^2_{\boldsymbol{a}^I}\ \boldsymbol{\sigma}^2_{\boldsymbol{\omega}^C}\ \boldsymbol{\sigma}^2_{\boldsymbol{v}^C})$. A comparison of the estimated values and their standard deviation with the simulated values shows that the results are coherent. Fig. 2 and the table below illustrate the result of the presented optimization approach – ${}^{I}\phi_C$ denotes the Euler angles corresponding to ${}^{I}\boldsymbol{q}_C$.

run	${}^{I}\boldsymbol{t}_{IC}$ $[mm]$			${}^{I}\phi_C$ [°]		
	x	y	z	x	y	z
simulated	10	0	-5.0	90	0	0
estimated	10.4	3.0	-5.0	89.7	0.8	-0.8
std. dev. σ	5.3	5.5	5.6	0.4	0.4	0.4

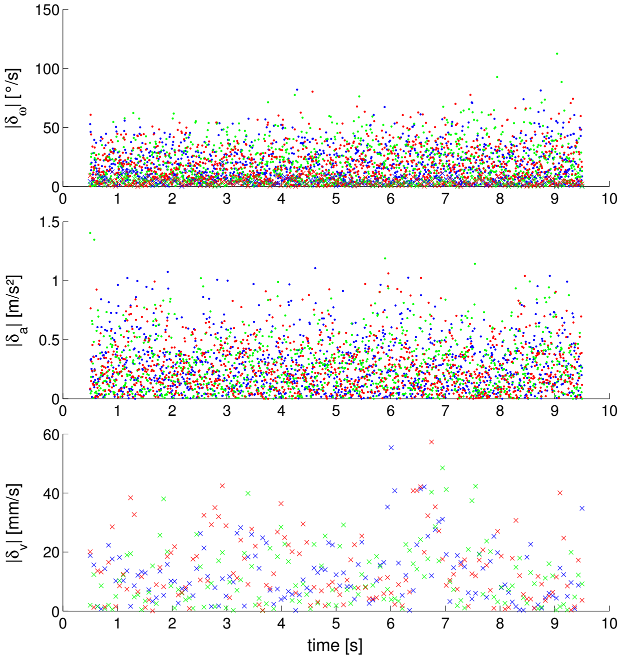


Fig. 2. Error between the simulated measurements and the estimated trajectories. The upper plot illustrates both angular velocities $\boldsymbol{\omega}^I$ and $\boldsymbol{\omega}^C$, the middle plot shows $\boldsymbol{a}^I$ and the lower one $\boldsymbol{v}^C$. The errors of the estimation correspond to the simulated noises.

B. Real Data

The IMU-camera setup used for our experiments is shown in Fig. 3. The PointGrey Flea2G camera is equipped with a wide angle lens with approximately 120° aperture angle. It is mounted on a Xsens-MTi IMU. In eight runs the camera was moved in front of a checkerboard acquiring images and inertial data for 10 s each. The camera rotation is computed by extracting the corners of the checkerboard and estimating the camera position in an optimization-framework using Calde and Callab [20]. The optimization is initialized as described in Section IV. Even though the scale factor α is known in this special case due to known dimensions of the checkerboard, we still optimize it as well. The resulting value of α has turned out to be a good indicator for the success of an optimization run: A value close to 1.0 usually means that a reasonable data alignment could be achieved, while large deviations of the optimal value indicate a failure of the algorithm. Errors in the intrinsic calibration, the accelerometer calibration or the checkerboard dimensions typically also cause slight deviations of α from the ideal value. The scale factor is initialized to 1.0.

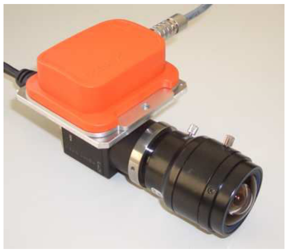


Fig. 3. A PointGrey Flea2G camera with wide angle lens and a Xsens-MTi IMU (orange box) as it is used in our experiments.

The results of all eight runs are illustrated in the table below. It shows the estimated values for ${}^{I}\boldsymbol{t}_{IC}$ and the corresponding standard deviations $\boldsymbol{\sigma}_{{}^{I}\boldsymbol{t}_{IC}}$:

run	${}^{I}\boldsymbol{t}_{IC}$ $[mm]$			$\boldsymbol{\sigma}_{{}^{I}\boldsymbol{t}_{IC}}$ $[mm]$		
	x	y	z	x	y	z
1	-20.6	69.6	-30.7	37.9	49.6	60.1
2	-3.0	62.9	-31.3	25.9	33.7	38.2
3	-18.7	64.6	-39.4	28.2	32.2	36.9
4	-18.9	68.5	-33.2	18.4	24.5	22.4
5	-21.3	52.5	-33.4	6.5	9.0	8.2
6	-17.9	58.1	-33.5	4.7	12.3	4.3
7	-25.8	60.4	-30.9	6.7	8.5	6.7
8	-20.5	46.9	-25.1	13.9	41.4	16.7

We will now compare our results with the system identification (grey box) approach described in [12] and a filter approach using an UKF as presented in [4]. The only modification for the grey box approach is that we do not use an EKF but a more accurate UKF instead to propagate and update the system state. The parameter vector $\boldsymbol{\theta}_{\text{UKF}}$ of the optimization consists of:

$$\boldsymbol{\theta}_{\text{UKF}} = \left(\boldsymbol{b}_{\boldsymbol{\omega}^I}^T \; \boldsymbol{b}_{\boldsymbol{a}^I}^T \; {}^{I_0}\boldsymbol{g}^T \; {}^{I}\boldsymbol{q}_C^T \; {}^{I}\boldsymbol{t}_{IC}^T\right)^T \tag{31}$$

The boxplots (box-and-whisker diagrams) in Fig. 4 illustrate the calibration results of the three different approaches. The estimation of the relative pose ${}^{I}\boldsymbol{t}_{IC}$ by the B-spline clearly outperforms the filter-based methods, while the angular alignment ${}^{I}\boldsymbol{q}_C$ yields comparable results. A closer look at the results shows that especially the runs with less dynamics account for this difference. The B-spline approach seems to be more sensitive and, thus, it achieves adequate results even if the dynamics of the calibration motion are low.

To evaluate the sensitivity of the presented algorithm regarding the quality of the initialization, we used a bad initialization to run the optimization. For that, we assumed no knowledge about the relative and the absolute orientation, setting ${}^{I}\boldsymbol{q}_C$ to zero and ${}^{I_0}\boldsymbol{g}$ to $(0\;0\;-9.81)$. The errors between the trajectories and the measurements resulting from the two different initializations are illustrated in Fig. 5(a) and Fig. 5(b). The wrong initialization of the rotation between camera and IMU becomes apparent in the angular velocity plots. The bad initialization of the gravity vector becomes obvious in the much larger scale of the acceleration plot

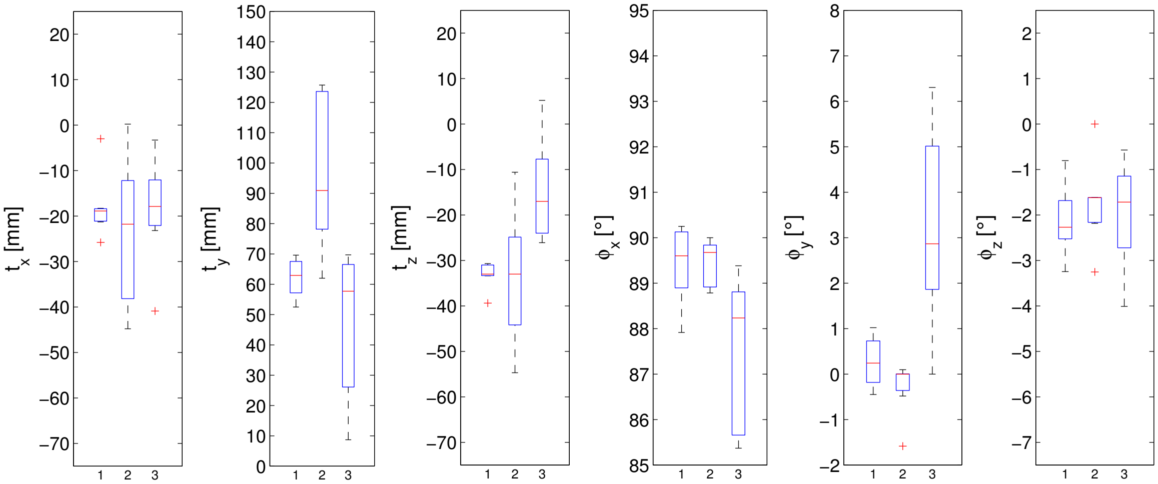


Fig. 4. Estimation of ${}^{I}\mathbf{t}_{IC}$ and the Euler angles ${}^{I}\boldsymbol{\phi}_{C}$ corresponding to ${}^{I}\boldsymbol{q}_{C}$ (boxplots). *1-left*: B-spline, *2-middle*: grey box, *3-right*: UKF.

in Fig. 5(b) - the gravitation component in the acceleration measurements can not be compensated properly. Nevertheless, in our experiments the optimization converges always to the same result, shown in Fig. 5(c), which indicates good convergence properties of this problem. The same experiment was performed using the grey box and UKF approach. Both could handle the bad initial orientation, but ran into serious problems with the bad initial gravity ${}^{I_0}\boldsymbol{g}$ and, thus, could not estimate the relative pose properly.

The last experiment shall illustrate the sensitivity of the optimization with respect to the number of control points. The relative pose ${}^{I}\boldsymbol{t}_{IC}$ and the scale factor α react most sensitive to changes of these parameters and, thus, they are used to evaluate the quality of the estimation. Fig. 6 shows the convergence of the optimization results with increasing number of knots. According to this experiment, there are no significant improvements to expect using more than 0.6 knots per camera frame for our registration sequences. If the spline has too few knots, it will not allow accurate modeling of the acceleration measurements and, thus, the velocity measured by the camera does not fit the approximated acceleration. A large number of knots increases the number of control points and, consequently, the overall optimization duration.

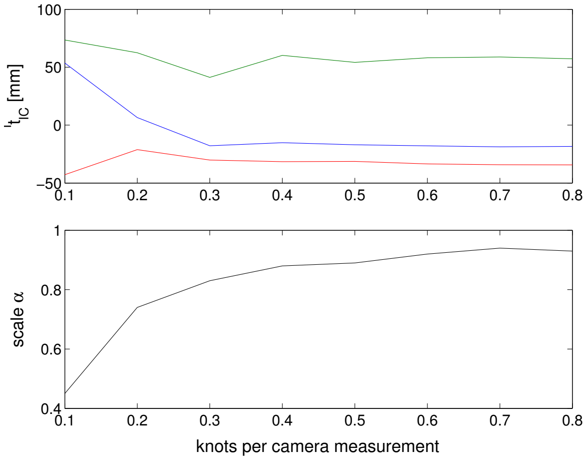


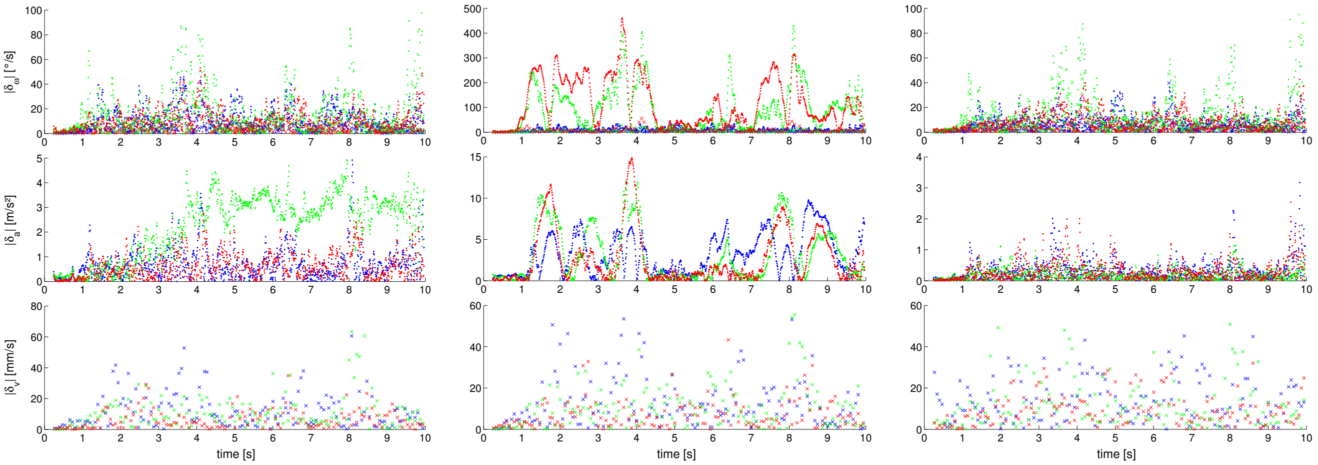
Fig. 6. Estimated relative pose ${}^{I}\boldsymbol{t}_{IC}$ (above) and scale α (below) with respect to the number of knots.

The accuracy of the spatial registration depends strongly on the measured motion. No calibration framework will be able to improve the result beyond the calibration errors which affect the fusion. Thus, assuming that the observability conditions mentioned in [10] are met, the accuracy of the spatial alignment strongly depends on the signal to noise ratio. In general, the noise of the sensors cannot be reduced further, thus, one should aim to provide a calibration run with high dynamic motions. If at least the same dynamics as a specific application requires are provided, the residual calibration error can be neglected because the resulting fusion error is not significant compared to the measured motion.

VI. Conclusion

In this work we presented a batch-optimization based solution for the problem of IMU-camera registration. Our experiments have shown that our approach compares favorably to conventional methods. This is due to a more general approximation of motion as in Kalman filters, which allows to model the non-linearities more accurately. The approach has shown to provide good convergence properties independently of the parameter initialization. As a byproduct of the registration one yields the smooth trajectory of the sensors as B-spline.

The presented approach is not real-time capable like conventional filtering solutions, but if an offline calibration is feasible an optimization based calibration should be preferred as the experiments illustrate. By modeling the trajectory with a B-spline one can avoid the first order approximation of the state as in [11], [12]. Especially such high dynamic motions as required for the IMU-camera calibration yield



(a) good initialization

(b) bad initialization

(c) estimation result

Fig. 5. The first two plot-columns show the errors of $\boldsymbol{\omega}^I$, $\boldsymbol{\omega}^C$ (both in the upper row), $\boldsymbol{a}^I$ (middle row) and $\boldsymbol{v}^C$ (lower row) relative to a good and a bad B-spline initialization as described in the text. Note the difference in the angular velocity plots and the difference in the scale of the acceleration plots. The rightmost plot denotes the estimation result, which is the same for both initializations.

large linearization errors. Varying the order of the B-spline and the number of the knots allows for an easy adaption to the dynamics of the calibration sequence. Of course, a higher order and more knots result in longer processing times - hence, a trade-off has to be found.

High dynamics in the calibration motion increase the signal to noise ratio and, thus, the accuracy of the calibration result. The use of a checkerboard limits the dynamics of the motion. Thus, in future we want to evaluate sequences with natural landmarks. However, without any knowledge about the environment we can only estimate the camera translation up to scale. The presented framework already provides such a scale parameter which makes it easy to adapt. We wonder if the higher signal to noise ratio of the IMU would allow for still more accurate calibration even with less accurate landmark tracking.

The number and the location of the B-spline knots has been determined empirically in this work. Estimating the optimal number and location of the knots would speed up processing and make it adaptive to arbitrary dynamics.

Another possible improvement is a direct fusion of the landmark locations in the images with the IMU motion, avoiding the currently preceding camera pose estimation. While this would increase the complexity of the optimization framework, we would expect increased accuracy.

VII. Acknowledgments

This work was partially funded by the DLR internal funding for image-based navigation systems.

References

[1] A. Chilian and H. Hirschmuller. Stereo camera based navigation of mobile robots on rough terrain. In *IEEE/RSJ IROS*, Oct 2009.

[2] K.H. Strobl, E. Mair, T. Bodenmüller, S. Kielhöfer, W. Sepp, M. Suppa, D. Burschka, and G. Hirzinger. The self-referenced dlr 3d-modeler. In *IEEE/RSJ IROS*, Oct 2009.

[3] K. Konolige and M. Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5), 2008.

[4] J. Kelly and G.S. Sukhatme. Visual-inertial simultaneous localization, mapping and sensor-to-sensor self-calibration. In *Proceeding IEEE Int. Symp. Computational Intelligence in Robotics and Automation*, pages 360–368, Dec 2009.

[5] E. Mair, K.H. Strobl, T. Bodenmüller, M. Suppa, and D. Burschka. Real-time image-based localization for hand-held 3d-modeling. *Künstliche Intelligenz*, 24, May 2010.

[6] J. Lobo and J. Dias. Relative pose calibration between visual and inertial sensors. In *IEEE ICRA Workshop on Integration of Vision and Inertial Sensors - 2nd InerVis*, Apr 2005.

[7] J. Lobo and J. Dias. Relative pose calibration between visual and inertial sensors. *International Journal of Robotic Research*, 26(6):561–575, Jun 2007.

[8] J. Kelly and G.S. Sukhatme. Fast relative pose calibration for visual and inertial sensors. In *Proc. 11th Int. Symposium Experimental Robotics*, Jul 2008.

[9] F.M. Mirzaei and S.I. Roumeliotis. A kalman filter-based algorithm for imu-camera calibration. In *IEEE/RSJ IROS*, pages 2427 –2434, Oct 2007.

[10] F.M. Mirzaei and S.I. Roumeliotis. A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics*, 24(5):1143 –1156, Oct 2008.

[11] F.M. Mirzaei and S.I. Roumeliotis. *IMU-Camera Calibration: Bundle Adjustment Implementation*, Aug 2007.

[12] J.D. Hol, T.B. Schön, and F. Gustafsson. Modeling and calibration of inertial and vision sensors. *Int. Journal of Robotic Research*, 29, Feb 2010.

[13] J.J. Craig. *Introduction to Robotics*. Pearson Education, 3 edition, 2006.

[14] J. Wendel. *Integrierte Navigationssysteme - Sensordatenfusion, GPS und Inertiale Navigation*. Oldenburg, 2007.

[15] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2nd edition, 2003.

[16] H. Prautzsch, W. Boehm, and H. Paluszny. *Bezier and B-Spline Techniques*. Springer, Berlin, Heidelberg, New York, 2002.

[17] C. De Boor. *A practical guide to splines*. Springer, New York, 1985.

[18] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, Aug 2000.

[19] R.Y. Tsai and R.K. Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. In *IEEE Transactions on Robotics and Automation*, 1989.

[20] K.H. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. DLR CalDe and DLR CalLab.