Timestamp Offset Calibration for an IMU-Camera System Under Interval Uncertainty

Raphael Voges and Bernardo Wagner

Abstract— **To properly fuse IMU and camera information for robotics applications, the relative timestamp offset between both sensors' data streams has to be considered. However, finding the exact timestamp offset is often impossible. Thus, it is necessary to additionally consider the offset's uncertainty if we want to produce reliable results. In order to find the offset and its uncertainty, we determine orientation estimates from IMU and camera under interval uncertainty. Subsequently, these intervals are used as a common representation for our bounded-error approach that finds an interval enclosing the true offset while also modeling the uncertainty. Calibration data can be acquired in a few seconds using a simple setup of IMU, camera and camera target. Results using both simulated and real data demonstrate that we are able to determine the offset to an accuracy of 20 ms with a computation time that is suitable for future online applications. Here, our approach could be used to monitor the timestamp offset in a guaranteed way. Additionally, our method can be adapted to determine an interval for the rotation between both sensors. While this increases the computation time drastically, it also enhances the accuracy of the timestamp offset to less than 10 ms.**

I. INTRODUCTION

To localize themselves in known or unknown environments, robots are often equipped with an inertial measurement unit (IMU) and a camera. Traditionally, probabilistic approaches are employed for data fusion of these two sensors, but recent work also deals with sensor fusion in a bounded-error context [1], [2]. However, fusion of data requires knowledge about the relative timing of the sensors' measurements. Otherwise, the localization algorithm will produce less accurate or even significantly wrong results.

Since commercial off-the-shelf sensors are generally black-box systems, it is not possible to use algorithms like the Network Time Protocol (NTP) [3] to synchronize the sensors' clocks. Nevertheless, it is possible to timestamp messages as they arrive at a common host computer, use the sensor's clock to timestamp messages or to infer timestamps from the sensor's rate. However, this may lead to a temporal offset between both sensors' data streams due to the offset between the sensors' clocks, different bus topologies (e.g. USB vs. serial), signal pre-processing, etc. To determine this offset, many probabilistic approaches have been investigated.

In the case of [4], the authors describe an iterative approach to determine the timestamp offset between camera and IMU data. They formulate the calibration as a registration problem and try to find correspondences between both sensors' measurements which allow them to infer the offset. Similar approaches are investigated by Mair et al. [5]. They compare cross correlation and phase congruency - two methods that can be used to align sensor measurements in time. Rehder et al. [6] present a general framework for spatiotemporal calibration, which is the calibration of spatial transformations and temporal offsets between sensors. They translate the problem into a maximum likelihood estimation by employing analytical basis functions. Another similar method that we investigated previously, is to find the timestamp offset in post-processing [7]. This approach fuses information from different sensors and adjusts the offset such that the sensor fusion results are optimal according to some pre-defined criteria (e.g. clarity of the map).

All works presented so far have in common that they assume a constant temporal offset and neglect the offset's uncertainty. In contrast to that, Li and Mourikis [8] present an online approach to estimate the time offset for camera-IMU systems which takes its uncertainty into account. They include the timestamp offset into the state vector of their extended Kalman filter. Thereby, it is possible to track time-varying offsets and to model the uncertainty in the covariance matrix in a stochastic way. However, we are interested in finding guaranteed bounds enclosing the offset.

Marzullo and Owicki [9] introduce the interval paradigm for clock synchronization in infrastructure networks. Expressing the timestamp offset with an unknown but bounded error has the following advantages. First, jitter (i.e. random error), which is naturally bounded, can be included directly into the interval. Secondly, unknown systematic errors that are problematic in probabilistic error modelling can be naturally included in intervals. Thirdly, since the intervals are guaranteed, different computations of the offset can be combined by means of intersection without additional knowledge about the underlying data quality. Furthermore, interval methods that perform sensor fusion [1] can only guarantee results if a guaranteed timestamp offset is considered.

In this context, Olson [10] introduces a method to compute a lower bound for the timestamp offset between sensor and host computer. His approach fits the bounded-error context, although it computes no interval for the offset but a lower bound only. The only work that tries to find an interval for the relative timestamp offset between two sensors is proposed by Zaman and Illingworth [11]. The authors apply an event-based approach to constrain the interval for the systematic delay between both sensors' data streams. However, their method relies on a large number of events which have to be detected accurately during extensive experiments.

The authors are with the Real Time Systems Group (RTS), Institute for Systems Engineering, Leibniz Universität Hannover, D-30167 Hannover, Germany. {voges,wagner}@rts.uni-hannover.de

This work was supported by the German Research Foundation (DFG) as part of the Research Training Group i.c.sens [RTG 2159].

To overcome this drawback, we propose the first approach to determine a guaranteed interval for the timestamp offset using continuous sensor measurements that can be acquired in a few seconds. In order to do that, we assume a constant timestamp offset and find correspondences in the sensors' data streams using orientation measurements. These orientation measurements are generated by rotating our sensors in front of a camera target. The assumption about a constant offset is valid since we acquire data in a short time frame such that any drift between the sensors' clocks can be neglected. Naturally, the interval for the timestamp offset is guaranteed only if our assumptions about the sensors' error bounds are correct. These error bounds and our approach to determine them are depicted in Section IV.

Our method can be used in two different ways: In the one-dimensional case, an interval vector enclosing the rotation between the sensors' coordinate systems has to be estimated beforehand and only the timestamp offset is unknown. Subsequently, the one-dimensional problem can be solved with little computation time. This allows to monitor the timestamp offset or to increase its accuracy in online applications.

The four-dimensional problem, on the other hand, requires plenty of computation time, because the three-dimensional rotation between the sensors' coordinate systems has to be computed as well. Nevertheless, this leads to a more accurate solution for the timestamp offset. The results can be further utilized in online applications or to validate results from probabilistic calibration methods. In summary, the contributions of this paper are as follows:

- Similar to [12], we propose an algorithm to solve the Perspective-n-Point problem in a bounded-error context. This algorithm does not need initial estimates to converge to a guaranteed solution set (see Section V-A).
- We present an idea to determine a guaranteed interval for an IMU's relative orientation using angular velocity measurements that are affected by an unknown but bounded error (see Section V-B).
- We propose a method to determine the timestamp offset and rotation between camera and IMU in a bounded-error context using previously computed orientation estimates (see Section VI).
- We demonstrate the feasibility of our method using both simulated and real data (see Section VII).

II. INTERVAL ANALYSIS

This section introduces the basics of interval analysis - a nonprobabilistic approach to define bounds for measurement errors. We put an emphasis on contractors and their application in the SIVIA (Set Inversion via Interval Analysis) algorithm. Definitions and algorithms are derived from [13].

A. Basic Notions

A closed and connected subset of $\mathbb{R}$ is an interval

$$[x] = [\underline{x}, \overline{x}] = \{\, x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x} \,\}, \tag{1}$$

where $\underline{x}$ and $\overline{x}$ denote the lower and upper bounds of $[x]$. The width of an interval $[x]$ is defined as $w([x]) = \overline{x} - \underline{x}$. $\mathbb{IR}$ denotes the set of all intervals.

An interval box $[\mathbf{x}] \in \mathbb{IR}^n$ is defined as the Cartesian Product of n closed intervals. Accordingly, its width is defined as $w([\mathbf{x}]) = \max_{1 \leq i \leq n} \; (w([x_i]))$.

The four classical real arithmetic operators addition $(+)$, subtraction $(-)$, multiplication $(\cdot)$ and division $(/)$ are extended to interval arithmetic, e.g. $[x] + [y] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$. Similarly, elementary functions such as Sine or Cosine can be extended to interval arithmetic. This is done by introducing the notion of an inclusion function.

Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. The interval function $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^m$ is an inclusion function for $\mathbf{f}$ if

$$\mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]), \forall [\mathbf{x}] \in \mathbb{IR}^n. \tag{2}$$

We obtain the natural inclusion function for a function $\mathbf{f}$ by replacing each operator in its expression by the corresponding interval operator. To compute the inverse solution set

$$\mathbb{X} = \{\, \mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y} \,\} = \mathbf{f}^{-1}(\mathbb{Y}), \tag{3}$$

where $\mathbb{Y}$ is a subset of $\mathbb{R}^m$, we use Set Inversion Via Interval Analysis (SIVIA).

B. Contractor Programming

A Constraint Satisfaction Problem (CSP) is defined as a system of n_x variables $x_i \in \mathbb{R},\ i \in \{1, \ldots, n_x\}$ that are linked by n_f relations (or constraints). It has the form

$$f_j(x_1, x_2, \ldots, x_{n_x}) = 0,\ j \in \{1, \ldots, n_f\}, \tag{4}$$

where each variable x_i belongs to an interval $[x_i]$. This equation can be rewritten in vector form as $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{x} = (x_1, \ldots, x_{n_x})^\intercal$ and $[\mathbf{x}] = [x_1] \times \cdots \times [x_{n_x}]$ is the prior domain for $\mathbf{x}$.

The CSP $\mathcal{H}$ can be formulated as

$$\mathcal{H} : (\mathbf{f}(\mathbf{x}) = \mathbf{0},\ \mathbf{x} \in [\mathbf{x}]). \tag{5}$$

A possible solution to $\mathcal{H}$ is a mapping that assigns a value from its domain to each variable s.t. all constraints are satisfied. Accordingly, the solution set $\mathbb{S}$ of $\mathcal{H}$ is defined as $\mathbb{S} = \{\, \mathbf{x} \in [\mathbf{x}] \mid \mathbf{f}(\mathbf{x}) = \mathbf{0} \,\}$.

To gain a smaller box that still contains the solution set, a contractor $\mathcal{C}$ is applied to the CSP. This means that $[\mathbf{x}]$ is replaced by a smaller domain $[\mathbf{x}'] \subset [\mathbf{x}]$ s.t. $\mathbb{S} \subset [\mathbf{x}']$. The contractor that replaces $[\mathbf{x}]$ by the smallest box that contains $\mathbb{S}$ is denoted as optimal.

While there exist many different contractors, one of the most efficient is the forward-backward contractor that is based on constraint propagation [14]. This contractor works by decomposing all constraints in primitive constraints and taking them into account in isolation. Depending on the constraints' properties and order, this contractor may need to be applied several times to find the smallest possible box.

C. Tubes

A set of trajectories - or orientations over the course of time - can be described by means of tubes. In our case, a tube encloses an uncertain orientation $\rho(\cdot)$ over the course of time $[t_0, t_f]$. We use the formal definition given in [15]. A tube $[\rho](\cdot) : \mathbb{R} \to \mathbb{IR}^n$ is an interval of two orientation functions

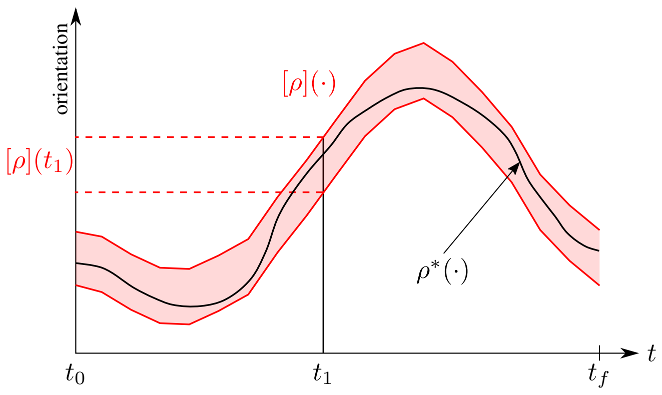


Fig. 1. An orientation tube $[\rho](\cdot)$ enclosing an orientation function $\rho^*(\cdot)$.

$[\underline{\rho}(\cdot), \overline{\rho}(\cdot)]$ such that $\forall t : \underline{\rho}(t) \leq \overline{\rho}(t)$. An orientation function $\rho(\cdot)$ belongs to the orientation tube $[\rho](\cdot)$ if $\forall t : \rho(t) \in [\rho](t)$. Fig. 1 depicts an orientation tube $[\rho](\cdot)$ enclosing an orientation function $\rho^*(\cdot)$. The interval at t_1 is $[\rho](t_1)$. If no orientation measurement is available for any point in time t_1, we use linear interpolation between the adjacent orientation measurement intervals.

III. MODIFIED RODRIGUES PARAMETERS

To represent orientations we use the Modified Rodrigues Parameters (MRP) [16]. They have the advantages that the singular points are as far away from the origin as possible and that the composition of two orientations can be calculated straightforward. These are important properties for an orientation representation that is used in conjunction with interval analysis since the evaluation of sequential rotations should not lead to a large overestimation.

Let $\mathbf{q}$ be a quaternion, that is defined as

$$\mathbf{q} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ \mathbf{q_{13}} \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) \\ \boldsymbol{\eta} \sin(\theta/2) \end{pmatrix}, \qquad (6)$$

where $\boldsymbol{\eta}$ is a unit vector representing the rotation axis and θ is the angle of rotation. The 3×1 MRP vector is defined as

$$\boldsymbol{\rho} = \frac{\mathbf{q_{13}}}{1 + q_0} = \boldsymbol{\eta} \tan(\theta/4). \qquad (7)$$

The 3×3 rotation matrix for $\boldsymbol{\rho}$ is

$$\mathbf{R}(\boldsymbol{\rho}) = \mathbf{I}_3 + \frac{8\,\hat{\boldsymbol{\rho}}^2 + 4\left(1 - \|\boldsymbol{\rho}\|^2\right)\hat{\boldsymbol{\rho}}}{\left(1 + \|\boldsymbol{\rho}\|^2\right)^2}, \qquad (8)$$

where $\hat{\boldsymbol{\rho}}$ is the skew matrix of $\boldsymbol{\rho}$ and $\|\boldsymbol{\rho}\|$ is the norm of $\boldsymbol{\rho}$.

Sequential rotation by $\boldsymbol{\phi}$ and then by $\boldsymbol{\rho}$ is denoted by the bullet operator ($\bullet$) and defined as

$$\boldsymbol{\rho} \bullet \boldsymbol{\phi} = \frac{\left(1 - \|\boldsymbol{\rho}\|^2\right)\boldsymbol{\phi} + \left(1 - \|\boldsymbol{\phi}\|^2\right)\boldsymbol{\rho} - 2\,\hat{\boldsymbol{\phi}}\,\boldsymbol{\rho}}{1 + \|\boldsymbol{\rho}\|^2\,\|\boldsymbol{\phi}\|^2 - 2\,\boldsymbol{\rho}^\intercal\boldsymbol{\phi}}. \qquad (9)$$

Another useful property of the MRP is that

$$\mathbf{R}^\intercal(\boldsymbol{\rho}) = \mathbf{R}(-\boldsymbol{\rho}), \qquad (10)$$

which allows direct computation of the inverse rotation.

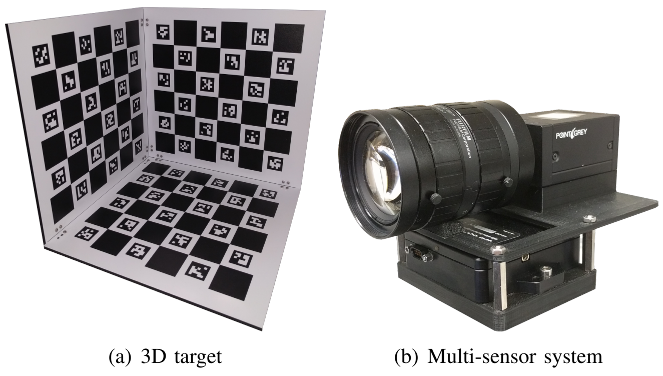


Fig. 2. Setup for the timestamp offset calibration.

IV. SETUP

Our setup for the timestamp offset calibration can be seen in Fig. 2. The 3D target that is placed in the camera's field of view is shown on the left side. It consists of three perpendicular checkerboards that are augmented by Aruco markers [17] to distinguish between them. We assume a printing accuracy of $[\Delta_W] = [-1, 1]$ mm for each checkerboard.

On the right side, the experimental setup of our multi-sensor system is depicted. It is composed of a Point Grey Grasshopper3 USB3 vision camera and a Microstrain 3DM-GQ4-45 inertial sensor. A single consumer-grade laptop (Intel Core i7 @ 2.7 GHz $\times$ 8, 16 GB RAM) is used to acquire measurement data and run the computations. Both sensors are attached via USB. To implement interval methods we use the IBEX library [18].

For our experiments we operate the camera at a resolution of 1920$\times$1200 px and a frame rate of 25 FPS. Before acquiring data, we calibrated our camera intrinsically using the algorithm proposed by Zhang [19] and its implementation in the Open Source Computer Vision Library (OpenCV). Using those intrinsic calibration parameters we removed distortion from each image before processing it. The camera's coordinate system is denoted as C with its z-axis pointing in the viewing direction, x-axis pointing to the right and y-axis pointing down. C_0 denotes the camera's initial coordinate system at the beginning of an experiment. Since not much information is available from the data sheet, we use the maximum reprojection error that occurred during camera calibration to bound the camera's error. This amounts to a maximum error of $[\Delta_C] = [-0.3, 0.3]$ px for each detection of a checkerboard feature.

The IMU is operated at a frequency of 100 Hz. We denote the IMU's coordinate system as I and its initial coordinate system at the beginning of an experiment as I_0. Similar to the camera, we acquired the IMU's maximum error from data sheets and own experiments. This error is composed of

- a 3×1 bias/noise vector $[\mathbf{b}]$ that encloses the gyroscope's systematic and random offset from $\mathbf{0}$. For our IMU: $[\mathbf{b}] = ([-0.01, 0.01]_{\times 3})^\circ$,
- a 3×1 scale factor stability vector $[\mathbf{s}]$ that encloses a proportional scaling from true velocity to measured velocity. For our IMU: $[\mathbf{s}] = ([-0.5, 0.5]_{\times 3})\,\%$.

V. Multi-Sensor System Model

We transfer the idea depicted in [4] to the field of interval analysis to determine not only the timestamp offset between camera and IMU but also its uncertainty. In order to find a relation between the data from IMU and camera, we require a common representation. For this work, we use orientation measurements that are generated by rotating the setup in front of the static camera target as a common representation. For every camera image, we calculate the pose relative to the 3D target. Afterwards, the pose estimates enable us to calculate the sequential rotation from the camera's initial pose to every other pose. For the IMU, we integrate the angular velocity measurements to find the orientation relative to the IMU's initial pose. At every point in time t_i it must hold:

$$-\boldsymbol{\rho}_C^I \bullet \boldsymbol{\rho}_I^{I_0}(t_i+\tau) \bullet \boldsymbol{\rho}_C^I = \boldsymbol{\rho}_C^{C_0}(t_i), \tag{11}$$

- τ is the timestamp offset between camera and IMU,
- $\boldsymbol{\rho}_C^I$ is the MRP vector defining the constant orientation of the camera in the IMU frame,
- $\boldsymbol{\rho}_C^{C_0}(t_i)$ is the MRP vector describing the orientation of the camera at time t_i relative to its initial orientation,
- $\boldsymbol{\rho}_I^{I_0}(t_i)$ is the MRP vector describing the orientation of the IMU at time t_i relative to its initial orientation.

In words, this means that at every point in time both sensors' orientation estimates have to coincide. This is because they are rigidly mounted, and consequently rotated at the same time. However, a constant timestamp offset τ between the sensors introduces a constant delay between their orientation estimates.

Since we assume an unknown but bounded error for both sensors, the variables in (11) are required to belong to their respective domain:

$$\tau \in [\tau],\ \boldsymbol{\rho}_C^I \in [\boldsymbol{\rho}_C^I],\ \boldsymbol{\rho}_I^{I_0}(\cdot) \in [\boldsymbol{\rho}_I^{I_0}](\cdot),\ \boldsymbol{\rho}_C^{C_0}(\cdot) \in [\boldsymbol{\rho}_C^{C_0}](\cdot). \tag{12}$$

For further explanation, let $\boldsymbol{\rho}_C^I = (0\ 0\ 0)^\intercal$, which means that there is no rotation between the sensors' coordinate systems. Fig. 3 depicts the idea for one dimension of (11). Let $[a](\cdot)$ be the first row of $[\boldsymbol{\rho}_C^{C_0}](\cdot)$, which is the camera's orientation tube. Similarly, let $[b](\cdot)$ be the first row of $[\boldsymbol{\rho}_I^{I_0}](\cdot)$, which is the IMU's orientation tube. Now, $[a](t_i)$ is the interval enclosing the camera's orientation at time t_i. Likewise, $[b](t_i+[\tau])$ is the interval enclosing the camera's orientation during the time interval $t_i+[\tau]$. We require that $\forall\, t_i$ the intersection $[a](t_i) \cap [b](t_i+[\tau])$ is non-empty. As can be seen, this intersection is non-empty in our example. However, a different choice of $[\tau]$ may lead to an empty intersection. We aim to find the set of all $\tau \in [\tau]$ which lead to a non-empty intersection $\forall\, t_i$.

For our experiments, we use the sensors' clocks to timestamp the data to ensure that jitter caused by non-deterministic transmission times or the operating system can be disregarded. Drift can be neglected since we acquire data in a short time frame. Before we explain the algorithm to determine an interval $[\tau]$ for τ in Section VI, we introduce our methods to find the orientation estimates for the camera (cf. Section V-A) and the IMU (cf. Section V-B).

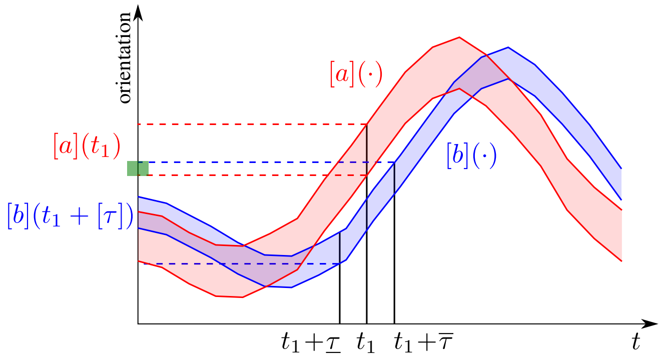


Fig. 3. Example depicting the idea of (11). $[a](\cdot)$ is the tube enclosing one dimension of the camera's orientation. Similarly, $[b](\cdot)$ is the tube enclosing one dimension of the IMU's orientation. We aim to find the set of all $\tau \in [\tau]$ such that both orientation tubes overlap for every point in time. This requirement is fulfilled for the exemplary $[\tau] = [\underline{\tau}, \overline{\tau}]$ at time t_1 since $[a](t_1)$ and $[b](t_1+[\tau])$ share a common intersection (green interval on the orientation axis).

A. Perspective-n-Point Under Interval Uncertainty

To obtain an interval for the camera's orientation we solve the Perspective-n-Point problem under interval uncertainty. Given a set of n 3D points in a predefined world coordinate system $\mathbf{X}^W = (X^W, Y^W, Z^W)^\intercal$, their corresponding 2D points $\mathbf{X}^I = (u, v)^\intercal$ in the image and intrinsic camera calibration parameters, the problem is to estimate the camera's pose in the world frame. The pose consists of a 3×3 rotation matrix $\mathbf{R}(\boldsymbol{\rho})$ that is expressed using a 3×1 MRP vector $\boldsymbol{\rho}$ (cf. (8)) and a 3×1 translation vector $\mathbf{T}$.

The rigid body transformation is expressed as follows:

$$\mathbf{X}^C = \mathbf{R}(\boldsymbol{\rho})\, \mathbf{X}^W + \mathbf{T}, \tag{13}$$

where $\mathbf{X}^C = (X^C, Y^C, Z^C)^\intercal$ are the coordinates of the projection of $\mathbf{X}^W$ in the camera frame.

By rewriting (13) and eliminating the unknown depth Z^C, two equations remain:

$$(\mathbf{R}_1 - x^C \mathbf{R}_3)\mathbf{X}^W + T_1 - x^C T_3 = 0, \tag{14}$$
$$(\mathbf{R}_2 - y^C \mathbf{R}_3)\mathbf{X}^W + T_2 - y^C T_3 = 0, \tag{15}$$

where $\mathbf{x}^C = (x^C, y^C, 1)^\intercal = (\frac{X^C}{Z^C}, \frac{Y^C}{Z^C}, 1)^\intercal$ is the normalized term of $\mathbf{X}^C$. $\mathbf{R}_i$ and T_i, $i \in \{1,2,3\}$, are the i-th row of $\mathbf{R}(\boldsymbol{\rho})$ and $\mathbf{T}$, respectively.

By applying the pinhole camera model [19], x^C and y^C can be expressed as follows:

$$x^C = \frac{u - c_x}{f_x}, \qquad y^C = \frac{v - c_y}{f_y}, \tag{16}$$

where (c_x, c_y) is the principal point and f_x, f_y are the camera's focal length expressed in pixel units.

Substituting (16) into (14)-(15) we derive two equations for every checkerboard feature. Stacking the equations, we obtain the constraints

$$\boldsymbol{\mathcal{C}}\,(\boldsymbol{\rho}, \mathbf{T}) = \mathbf{0}, \tag{17}$$

where $\boldsymbol{\mathcal{C}}$ has $2n$ rows.

We assume an unknown but bounded error $[\Delta_W]$ for each 3D coordinate. Likewise, we assume an unknown but

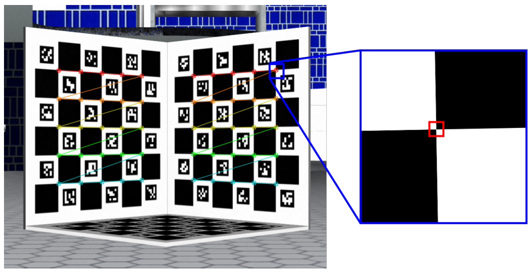


Fig. 4. Exaggerated example of a 2D bounding box for feature detection.

bounded error $[\Delta_C]$ for the checkerboard feature detection (cf. Section IV). This results in a 2D bounding box as depicted in Fig. 4. Although we do not use intervals for the camera parameters explicitly, their uncertainty is included in the features' uncertainty. $[\mathbf{X}^W]$ and $[\mathbf{X}^I]$ are computed as

$$[\mathbf{X}^W] = \mathbf{X}^W_m + [\Delta_W], \tag{18}$$

$$[\mathbf{X}^I] = \mathbf{X}^I_m + [\Delta_C], \tag{19}$$

where $\mathbf{X}^W_m$ are the 3D coordinates of a checkerboard feature and $\mathbf{X}^I_m$ is the detected checkerboard feature in pixels.

The CSP $\mathcal{H}$ can be formulated as

$$\mathcal{H} : \Big(\mathcal{C}(\boldsymbol{\rho}, \mathbf{T}) = \mathbf{0}, \\ \boldsymbol{\rho} \in [\boldsymbol{\rho}], \mathbf{T} \in [\mathbf{T}],\ \mathbf{X}^W \in [\mathbf{X}^W],\ \mathbf{X}^I \in [\mathbf{X}^I]\Big), \tag{20}$$

where $[\boldsymbol{\rho}]$ and $[\mathbf{T}]$ are the domains for the pose parameters.

Initially, we set $[\mathbf{T}] = ([-\infty, \infty], [-\infty, \infty], [0, \infty])^\intercal$ since we have no prior information about the translation parameters except that we use a front-looking camera. Accordingly, we set $[\boldsymbol{\rho}] = ([-1, 1]_{\times 3})^\intercal$ since we have no prior information about the orientation parameters.

To solve $\mathcal{H}$, a forward-backward contractor is built for the constraints $\mathcal{C}$. Subsequently, the contractor and the initial domains for the pose parameters are fed into SIVIA to compute an outer approximation for the translation parameters $[\mathbf{T}]$ and the orientation parameters $[\boldsymbol{\rho}]$. If the intervals are empty, the constraints $\mathcal{C}$ did not hold due to detection errors or wrong bounds on the measurement uncertainty. In this case, we discard the measurement and do not use it for the offset computation. Note that this procedure works as an outlier detection. However, it may happen that the error is small enough to not produce empty intervals while still exceeding the 2D/3D error bounds. In this case, we cannot detect the outlier. Nevertheless, outliers are rare in practice and exceed the error bounds drastically if they occur (e.g. in the case of a misdetection by the feature detector).

Ultimately, we are interested in the MRP vector describing the orientation of the camera relative to its initial orientation. We denote the outer approximation of the orientation parameters that were calculated for the image captured at time t with $[\boldsymbol{\rho}^C_W](t)$ since they describe the orientation of the world frame relative to the camera frame. The initial orientation is

$$[\boldsymbol{\rho}^{C_0}_W] = \bigcap_{t \leq \delta_t} [\boldsymbol{\rho}^C_W](t), \tag{21}$$

where δ_t is the time for which the system remains in its reference orientation (i.e. we do not move it).

Finally, the orientation tube describing the relative rotation from the initial orientation is

$$[\boldsymbol{\rho}^{C_0}_C](\cdot) = [\boldsymbol{\rho}^{C_0}_W] \bullet -[\boldsymbol{\rho}^C_W](\cdot). \tag{22}$$

To increase the accuracy of our orientation estimates, we perform the described method for each of the three checkerboard planes of our 3D target. Since each tube $[\boldsymbol{\rho}^{C_0}_C](\cdot)$ is guaranteed to contain the true orientation, the intersection of the three tubes that were calculated for the three checkerboards is also guaranteed to contain the true orientation.

B. IMU Orientation Under Interval Uncertainty

We assume an unknown but bounded error for the IMU's angular velocity measurements. This error is composed of a 3×1 bias/noise vector $[\mathbf{b}]$ and a 3×1 scale factor stability vector $[\mathbf{s}]$ as depicted in Section IV.

The resulting 3×1 interval vector for the IMU's angular velocity estimates is calculated as

$$[\boldsymbol{\omega}](t) = \boldsymbol{\omega}^I_m(t) + [\mathbf{b}] + [\mathbf{s}] \cdot \boldsymbol{\omega}^I_m(t), \tag{23}$$

where $\boldsymbol{\omega}^I_m(t)$ is the IMU's angular velocity measurement.

To obtain an interval for the IMU's orientation, we use the MRP kinematic differential equation:

$$\dot{\boldsymbol{\rho}}(t) = \frac{1}{4}\left(\left(1 - \|\boldsymbol{\rho}(t)\|^2\right)\mathbf{I}_3 + 2\,\hat{\boldsymbol{\rho}(t)} + 2\,\boldsymbol{\rho}(t)\right)\boldsymbol{\omega}(t), \tag{24}$$

where $\boldsymbol{\omega}(t)$ is the true 3×1 angular velocity vector.

Since our angular velocity estimates at time t are described as an interval vector $[\boldsymbol{\omega}](t)$, we use the natural inclusion function for (24). Subsequently, first-order integration of this inclusion function leads to an orientation tube that we denote by $[\boldsymbol{\rho}^{I_0}_I](\cdot)$ since it describes the IMU's relative orientation over time. Because of the integration, the interval widths increase over time, and thus reflect the orientation estimate's increasing uncertainty that is caused by drift.

VI. TIMESTAMP OFFSET CALIBRATION

The general idea to determine the temporal offset between camera and IMU is to use SIVIA to find all $\tau \in [\tau]$ that satisfy (11). At first, we determine the three-dimensional camera orientation tube $[\boldsymbol{\rho}^{C_0}_C](\cdot)$ as depicted in Section V-A. Typically, SIVIA outputs a subpaving. However, for further computations we build the camera orientation tube using the convex hull of that subpaving to reduce complexity. Additionally, we find the three-dimensional IMU orientation tube $[\boldsymbol{\rho}^{I_0}_I](\cdot)$ as depicted in Section V-B. Besides, the algorithm needs initial estimates for the timestamp offset domain $[\tau]$ and the rotation domain $[\boldsymbol{\rho}^I_C]$.

Algorithm 1 depicts our method. Instead of a contractor, we employ a check whether (11) holds for all camera orientations (cf. lines 5-15). In order to do that, it is first necessary to determine the IMU orientation at time $[t_i] = t_i + [\tau]$, where t_i is the timestamp of the current camera orientation and $[\tau]$ is the timestamp offset interval for the current iteration of SIVIA (line 7). Since $[t_i]$ is an interval, we have to find an interval for $[\boldsymbol{\rho}_I^{I_0}]\left([t_i]\right)$ that encloses all possible IMU orientations during $[t_i] = [\underline{t_i}, \overline{t_i}]$ as follows:

$$[\boldsymbol{\rho}_I^{I_0}]\left([t_i]\right) \coloneqq [\boldsymbol{\rho}_I^{I_0}]\left(\underline{t_i}\right) \cup \left(\bigcup_{\underline{t_i} \le t_i \le \overline{t_i}} [\boldsymbol{\rho}_I^{I_0}]\left(t_i\right) \right) \cup [\boldsymbol{\rho}_I^{I_0}]\left(\overline{t_i}\right), \tag{25}$$

where $[\boldsymbol{\rho}_I^{I_0}]\left(\underline{t_i}\right)$ and $[\boldsymbol{\rho}_I^{I_0}]\left(\overline{t_i}\right)$ are determined using linear interpolation on the bounds of the three-dimensional MRP interval vectors. Linear interpolation is applicable, since the IMU measurements are available with a high frequency.

Subsequently, the IMU orientation measurement needs to be transformed into the camera coordinate system. We evaluate and rearrange the single components of the expression $\mathbf{f}_{mrp} \coloneqq -\mathbf{a} \bullet \mathbf{b} \bullet \mathbf{a}$ such that each component of $\mathbf{b}$ appears only once in each component of $\mathbf{f}_{mrp}$. Subsequently, the rearranged expression allows us to determine a tighter interval for $[\boldsymbol{\rho}'][(t_i)]$ (line 8). We omit the full expression at this point due to its size.

Afterwards, the algorithm checks the set membership relation of the transformed IMU MRP vector $[\boldsymbol{\rho}'][(t_i)]$ and the camera MRP vector $[\boldsymbol{\rho}_C^{C_0}](t_i)$ (lines 9-14). If both MRP vectors do not overlap, the membership variable is set to 0 and we continue with the next iteration of SIVIA, since (11) did not hold for all camera orientations. If they overlap but $[\boldsymbol{\rho}'][(t_i)]$ is not completely inside $[\boldsymbol{\rho}_C^{C_0}](t_i)$, the membership variable is set to 1 since we need to further bisect $[\tau]$. However, if every $[\boldsymbol{\rho}'][(t_i)]$ is completely inside the corresponding $[\boldsymbol{\rho}_C^{C_0}](t_i)$, the membership variable is not changed and remains at 2. This means that (11) holds for every $\tau \in [\tau]$, and thus we can add $[\tau]$ to the solution set $\mathcal{L}$.

It is also possible to find an outer approximation for the rotation between camera and IMU frame $[\boldsymbol{\rho}_C^I]$. In order to do that, the three components of $[\boldsymbol{\rho}_C^I]$ are added as additional variables to SIVIA s.t. the stack $\mathcal{S}$ contains four-dimensional interval boxes. These four-dimensional boxes are bisected accordingly in line 21. This increases computational complexity since SIVIA has to operate in four dimensions. However, it can also lead to tighter intervals for $[\tau]$ since the intervals in line 8 get smaller, and thus some $\tau \in [\tau]$ that did not violate (11) before may violate it now. We refer to this as the four-dimensional problem, while the one-dimensional problem denotes a bisection of $[\tau]$ only.

Algorithm 1: SIVIA for timestamp offset

```
input : $[\tau_0]$, $[\boldsymbol{\rho}_C^I]$, $[\boldsymbol{\rho}_I^{I_0}](\cdot)$, $[\boldsymbol{\rho}_C^{C_0}](\cdot)$
output: $\mathcal{L}$, which is a subpaving for $[\tau]$
1  $\mathcal{S} \coloneqq \{[\tau_0]\}$ ;                                  // $\mathcal{S}$ is a stack
2  while $\mathcal{S}$ is not empty do
3      $[\tau] \coloneqq \mathrm{top}(\mathcal{S})$;
4      membership $\coloneqq 2$;
5      foreach $t_i \in T_{camera}$ do                   // for all camera orientations
6          $[t_i] \coloneqq t_i + [\tau]$;
7          $[\boldsymbol{\rho}_I^{I_0}]([t_i]) \coloneqq$ interpolate $\left([\boldsymbol{\rho}_I^{I_0}](\cdot),\, [t_i]\right)$;
8          $[\boldsymbol{\rho}'][(t_i)] \coloneqq \left(-[\boldsymbol{\rho}_C^I] \bullet [\boldsymbol{\rho}_I^{I_0}]([t_i]) \bullet [\boldsymbol{\rho}_C^I]\right)$;
9          if $[\boldsymbol{\rho}'][(t_i)] \cap [\boldsymbol{\rho}_C^{C_0}](t_i) = \emptyset$ then
10             membership $\coloneqq 0$;
11             break;
12         else if $[\boldsymbol{\rho}'][(t_i)] \setminus [\boldsymbol{\rho}_C^{C_0}](t_i) \neq \emptyset$ then
13             membership $\coloneqq 1$;
14         end
15     end
16     if membership $== 0$ then
17         continue;
18     else if $w([\tau]) < \epsilon$ or membership $== 2$ then
19         $\mathcal{L} \coloneqq \mathcal{L} \cup \{[\tau]\}$ ;
20     else if membership $== 1$ then
21         $([\tau_1], [\tau_2]) \coloneqq$ bisect $([\tau])$;
22         $\mathcal{S} \coloneqq \mathcal{S} \cup \{[\tau_1]\} \cup \{[\tau_2]\}$;
23     end
24 end
```

VII. RESULTS

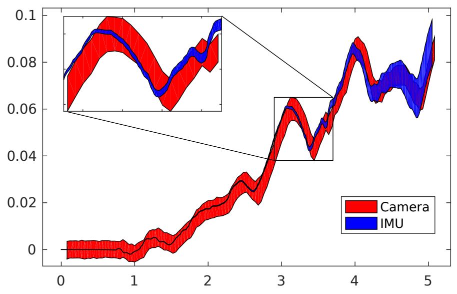
To evaluate our method we execute it on both simulated data that was generated using the robot-simulator Gazebo [20] and real data that was acquired using the setup depicted in Section IV.

We only present results for the timestamp offset calibration introduced in Section VI. The camera orientation tube $[\boldsymbol{\rho}_C^{C_0}](\cdot)$ and IMU orientation tube $[\boldsymbol{\rho}_I^{I_0}](\cdot)$ are calculated beforehand according to Section V-A and Section V-B. We set ϵ, which influences the bisection depth of SIVIA, to 0.001, which corresponds to an interval width of 1 ms or 0.06°, respectively. This proved to be an adequate compromise between accuracy and computation time. Furthermore, we set the initial interval for the offset as $[\tau_0] = [-500, 500]$ ms. All depicted interval boxes are the outer enclosure of the corresponding subpaving.

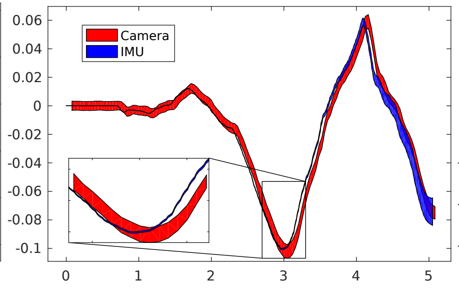
A. Simulated Data

Since there is no possibility to recover the true timestamp offset using our real sensors, we recreated our setup in Gazebo using the error bounds from the sensors' data sheets. This allows us to simulate different offsets which should be enclosed by our algorithm. The true orientation of the camera in the IMU frame is $\boldsymbol{\rho}_C^I = -\left(\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}\right)^\intercal$ for all simulations.

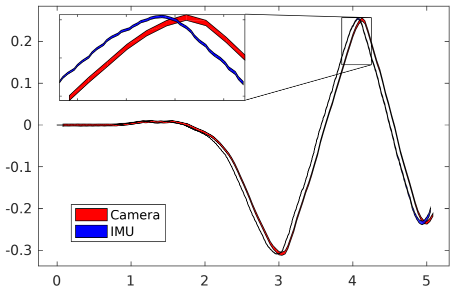
Although we used different data sets to evaluate our approach, we only present results for one data set. Here we rotated our setup around all three axis at the same time for 3 s with a velocity of 2 rad/s. We show results for both the one-dimensional problem and the four-dimensional problem. For the one-dimensional problem we bisected only the interval for the timestamp offset $[\tau]$ in Algorithm 1 and set $[\boldsymbol{\rho}_C^I] = -([0.32, 0.34]_{\times 3})^\intercal$, which corresponds to an angle uncertainty of $w([\theta]) = 6°$. For the four-dimensional problem we bisected also the intervals for the rotation between both sensors in Algorithm 1 starting from an initial domain of $[\boldsymbol{\rho}_C^I] = -([0.1, 0.5]_{\times 3})^\intercal$, which corresponds to an initial angle uncertainty of $w([\theta]) = 124°$.



(a) ρ_1 over time.



(b) ρ_2 over time.



(c) ρ_3 over time.

Fig. 5. Spatially aligned orientation tubes for the third trial that are non-overlapping due to a timestamp offset.

The results can be seen in Table I. $[\tau]_{1D}$ corresponds to the one-dimensional problem while $[\tau]_{4D}$ corresponds to the four-dimensional problem. Note, that the true offset τ is always enclosed in the corresponding intervals $[\tau]_{1D}$ and $[\tau]_{4D}$. If we solve the four-dimensional problem, the rotation between the sensors' coordinate systems is also recovered with an accuracy of $w([\theta]) = 19.85°$ (cf. (7)). Furthermore, it can be seen that the true offset's deviation from zero has no effect on the computed offset's accuracy. Small fluctuations can be explained by the nature of the bisection in our algorithm.

TABLE I
SIMULATED TRUE OFFSETS (1D+4D)

τ (ms)	$[\tau]_{1D}$ (ms)	$[\tau]_{4D}$ (ms)
10.0	$[-0.8, 27.3]$	$[0.8, 21.1]$
-10.0	$[-21.1, 7.8]$	$[-18.8, 1.6]$
20.0	$[8.6, 37.5]$	$[10.9, 31.3]$
50.0	$[39.1, 67.2]$	$[41.4, 60.9]$

B. Real Data

To gather real data we used the setup depicted in Section IV with the interval error bounds as explained. Overall, we collected four different data sets of different characteristics. The first three data sets were recorded while rotating the setup by hand and ensuring that the 3D target remained in the camera's field of view. In order to do that, we placed our setup on the floor in front of the 3D target for initialization, picked it up after roughly 1 s and rotated it for about 10 s. We aimed to increase the rotational velocity from trial to trial, s.t. the first trial contains low velocity rotations and the third trial contains high velocity rotations. For the fourth trial, we added a motor to our setup s.t. we could conduct an experiment with rotation around one axis only. We set the motor's velocity to 1 rad/s.

Similar to our simulation studies, we decided to solve the one-dimensional problem at first, which means that we used a fixed interval vector for the rotation and only bisected the interval for the timestamp offset. The results can be seen in Table II. We estimated the three-dimensional MRP vector by hand and added the following measurement uncertainties. We chose a rather small rotation uncertainty of $w([\theta]) = 6°$ for the column containing $[\tau]_1$ and a large rotation uncertainty of $w([\theta]) = 18°$ for the column containing $[\tau]_2$. Fig. 5 shows the orientation tubes for the third trial after aligning them spatially using the estimated bounded rotation. It can be seen that they are shifted due to the timestamp offset.

To provide some reference data, we computed the IMU's and camera's orientation using traditional methods and applied cross-correlation to temporally align the data series as described by Mair et al. [5]. The results, which can be seen in the fourth column of Table II, are always enclosed in the corresponding interval for the timestamp offset.

TABLE II
TIMESTAMP OFFSET ONLY (1D)

Trial	$[\tau]_1$ (ms)	$[\tau]_2$ (ms)	τ_{xcorr} (ms)
1	$[1.0, 94.7]$	$[-23.4, 126.0]$	48.8
2	$[25.4, 76.2]$	$[2.0, 102.5]$	48.2
3	$[29.3, 60.5]$	$[5.9, 77.1]$	45.8
4	$[41.0, 68.4]$	$[37.1, 73.2]$	50.9

The rotational velocity's influence on the width of the timestamp offset interval can be observed. However, it makes no difference whether we rotate our setup by hand around all three axis or employ a motor to achieve a rotation around one axis only. Furthermore, note that all computed intervals overlap, which shows that the results are consistent and the calibration is repeatable. Since the results should be guaranteed, it is inevitable that all computed intervals share a common intersection. If this were not the case, we would have discovered an inconsistency in our system. The common intersection for our experiments is $[\tau] = [41.0, 60.5]$ ms. The final uncertainty for our one-dimensional studies is $w([\tau]) = 19.5$ ms. Naturally, this accuracy can be improved by estimating more accurate rotation parameters.

For the results in Table III we solved the four-dimensional problem, meaning that we bisected both the interval for the timestamp offset and the three-dimensional MRP interval vector for the rotation. We chose the same initial rotation uncertainty of $w([\theta]) = 124°$ as for our simulation studies.

As can be seen, we are not able to estimate the rotation to a reasonable accuracy if we employ the motor to rotate around one axis only. Nevertheless, we manage to contract the three-dimensional MRP interval vectors for trials one to

TABLE III

TIMESTAMP OFFSET AND ROTATION (4D)

Trial	$[\tau]$ (ms)	$w([\theta])$ (deg)
1	$[27.3, 68.4]$	12.70
2	$[35.2, 57.6]$	12.24
3	$[37.1, 51.8]$	12.93
4	$[43.0, 65.4]$	100.72

three. Again, the intervals for the timestamp offset share a common intersection $[\tau] = [43.0, 51.8]$ ms, which is also consistent with the result of our one-dimensional studies. Thus, the final accuracy we achieve is $w([\tau]) = 8.8$ ms. In contrast to the one-dimensional problem, which requires around 0.1 s computation time, the computation time for the four-dimensional problem ranges from 10 min for the third trial to 1 h for the first trial. The computation time for the fourth trial amounts to 10 h since the rotation parameters do not converge. Fig. 6 shows the camera and IMU orientation intervals which were aligned using the computed rotation between both sensors' coordinate systems. Note the IMU's increasing uncertainty due to integration.

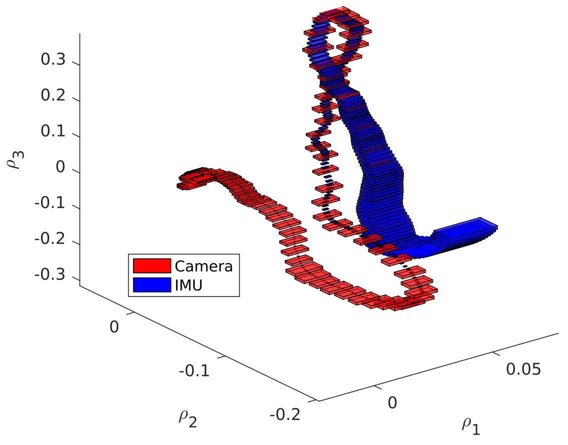


Fig. 6. Aligned camera and IMU orientation intervals after calibration.

VIII. CONCLUSIONS

In this paper, we have presented a method to determine an interval that is guaranteed to contain the timestamp offset between IMU and camera if the sensor errors are modeled correctly. Furthermore, the interval allows us to make statements about the timestamp offset's accuracy, which is not considered in most approaches. For our experiments we put an emphasis on the influence of motion characteristics during data acquisition and showed that we are able to determine the timestamp offset to an accuracy of less than 10 ms using a very basic setup. Moreover, our algorithm is able to recover an outer approximation for the rotation between the two sensors starting from a very rough estimate.

As future work, we aim to further improve the computation time of our method by designing a contractor on tubes. Furthermore, we strive to employ our algorithm in online applications by using features in the real world instead of features on the 3D target. Afterwards, we plan to apply our method to other sensor combinations, which have to be modeled in a bounded-error context beforehand. Additionally, we aim to incorporate the offset interval into sensor fusion approaches to account not only for the timestamp offset but also its uncertainty.

REFERENCES

[1] A. Bethencourt and L. Jaulin, "3D reconstruction using interval methods on the kinect device coupled with an IMU," *International Journal of Advanced Robotic Systems*, vol. 10, no. 2, p. 93, 2013.

[2] M. Langerwisch and B. Wagner, "Global indoor localization assuming unknown but bounded sensor errors," *Field and Assistive Robotics - Advances in Systems and Algorithms*, 2014.

[3] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[4] J. Kelly, N. Roy, and G. S. Sukhatme, "Determining the time delay between inertial and visual sensor measurements," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1514–1523, 2014.

[5] E. Mair, M. Fleps, M. Suppa, and D. Burschka, "Spatio-temporal initialization for IMU to camera registration," *2011 IEEE International Conference on Robotics and Biomimetics*, 2011.

[6] J. Rehder and R. Siegwart, "Camera/IMU calibration revisited," *IEEE Sensors Journal*, vol. 17, no. 11, pp. 3257–3268, 2017.

[7] R. Voges, C. S. Wieghardt, and B. Wagner, "Finding timestamp offsets for a multi-sensor system using sensor observations," *Photogrammetric Engineering & Remote Sensing*, vol. 84, no. 6, pp. 357–366, 2018.

[8] M. Li and A. I. Mourikis, "Online temporal calibration for camera–IMU systems: Theory and algorithms," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 947–964, 2014.

[9] K. Marzullo and S. Owicki, "Maintaining the time in a distributed system," *ACM SIGOPS Operating Systems Review*, vol. 19, no. 3, pp. 44–54, 1985.

[10] E. Olson, "A passive solution to the sensor synchronization problem," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[11] M. Zaman and J. Illingworth, "Interval-based time synchronisation of sensor data in a mobile robot," *2004 Intelligent Sensors, Sensor Networks and Information Processing Conference*, 2004.

[12] I.-F. Kenmogne, V. Drevelle, and E. Marchand, "Image-based UAV localization using interval methods," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[13] L. Jaulin, M. Kieffer, O. Didrit, and É. Walter, *Applied Interval Analysis*. Springer London, 2001.

[14] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget, "Revising hull and box consistency," *Proceedings of the 16th International Conference on Logic Programming*, pp. 230–244, 1999.

[15] A. Bethencourt and L. Jaulin, "Solving non-linear constraint satisfaction problems involving time-dependant functions," *Mathematics in Computer Science*, vol. 8, no. 3-4, pp. 503–523, 2014.

[16] M. D. Shuster, "Survey of attitude representations," *Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993.

[17] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[18] J. Ninin, "Global optimization based on contractor programming: An overview of the IBEX library," *Mathematical Aspects of Computer and Information Sciences. MACIS 2015. Lecture Notes in Computer Science*, pp. 555–559, 2016.

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

[20] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004, pp. 2149–2154.