# Uncertainty-Aware Hand–Eye Calibration

Markus Ulrich and Markus Hillemann

***Abstract*—We provide a generic framework for the hand–eye calibration of vision-guided industrial robots. In contrast to traditional methods, we explicitly model the uncertainty of the robot in a stochastically founded way. Albeit the repeatability of modern industrial robots is high, their absolute accuracy typically is much lower. This uncertainty — especially if not considered — deteriorates the result of the hand–eye calibration. Our proposed framework does not only result in a high accuracy of the computed hand–eye pose but also provides reliable information about the uncertainty of the robot. It further provides corrected robot poses for a convenient and inexpensive robot calibration. Our framework is computationally efficient and generic in several regards: It supports the use of a calibration target as well as self-calibration without the need for known 3D points. It optionally enables the simultaneous calibration of the interior camera parameters. The framework is also generic with regard to the robot type, and hence supports antropomorphic as well as SCARA robots, for example. Simulated and real experiments show the validity of the proposed methods. An extensive evaluation of our framework on a public dataset shows a considerably higher accuracy than 15 state-of-the-art methods.**

## I. Introduction

HAND–EYE calibration is essential for applications that use vision-guided robots. It determines the 3D rigid transformation (pose) between the robot and the camera (hand–eye pose). This enables measurements that are performed in the camera coordinate system to be transformed into the robot coordinate system. In bin-picking applications, for example, the pose of an object is determined in the coordinate system of the camera by using 3D object recognition [1]–[4]. To grasp the object, the object pose must be transformed into the coordinate system of the robot.

In general, there are two different scenarios of vision-guided robots [5, Chapter 3.12.2]. In the first scenario, the camera is mounted at the robot tool and is moved to different positions by the robot. In the second scenario, the camera is mounted stationary outside the robot without moving with respect to the robot base. The pose that must be determined by hand–eye calibration is the pose of the camera with respect to the robot tool in the first scenario, or the pose of the camera with respect to the robot base in the second scenario. For simplicity, we focus on the first scenario with a moving camera in the following. However, our framework can be applied analogously to the second scenario with a stationary camera.

Traditional approaches for hand–eye calibration assume that the pose of the robot tool with respect to the robot base is known accurately. When discussing the accuracy of robots, it is necessary to distinguish between the pose repeatability (i.e., precision) and the pose accuracy of the robot [6]. Repeatability is a measure of the robot's ability to move the tool repeatedly to the same pose. Accuracy is a measure of the robot's ability to move its tool to a specified pose in 3D space. Modern industrial robots typically offer a very high repeatability, which is in the range of 0.02–0.15 mm [7]–[10]. For applications where the robot always moves to the exact same pose, which is trained in advance, a high repeatability is sufficient. However, for robots that are programmed offline and especially for vision-guided robots, obviously a high pose accuracy is important as well. Unfortunately, the pose accuracy of robots often is much lower than the pose repeatability. The position accuracy typically is in the range of 0.1–5.0 mm while the orientation accuracy ranges from about 0.2 deg up to several degrees [8]–[10].

It is possible to increase the accuracy with a robot calibration by up to an order of magnitude, in rare cases even up to the level of repeatability [7]. Unfortunately, robot calibration often requires high-precision measurement instruments like lasertrackers [8], and hence is expensive and time consuming. In this paper, we combine all errors of the robot under the term uncertainty, which include errors that have an impact on repeatability and on accuracy.

Almost all traditional methods for hand–eye calibration do not explicitly model erroneous robot poses. Our proposed framework explicitly models the uncertainty of the robot in a stochastically founded way resulting in a higher accuracy of the computed hand–eye pose. Furthermore, it provides reliable information about the uncertainty of the robot, which otherwise would require high-precision measurement instruments. Since the framework also provides error-corrected robot poses, it can support a convenient and inexpensive robot calibration by assuming the error-corrected robot poses as ground-truth poses for the estimation of the robot parameters (i.e., Denavit-Hartenberg parameters [11], for example).

Most existing approaches for hand–eye calibration require the acquisition of multiple images of a calibration target. A few more flexible and user-friendly solutions avoid the use of calibration targets. Our framework supports both variants, target-based calibration as well as self-calibration without the need for known 3D points. Furthermore, for both variants, the framework offers the possibility to simultaneously calibrate the interior camera parameters if desired.

This publication is an evolved paper. It extends our previously published work [12] in several aspects. We included and discussed additional relevant related work. We give details about the polynomial distortion model that supports a higher accuracy compared to the division model. We also included a comprehensive description of the structure of the parameter estimation including a visualization of the Jacobian, a flowchart, and detailed pseudocode of the approach. All necessary partial derivatives have been analytically calculated and are now given

markus.ulrich@kit.edu, markus.hillemann@kit.edu; Institute of Photogrammetry and Remote Sensing, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany

in the Appendix supporting a faster, more accurate, and hence more robust computation. A new section contains details about the estimation of the variance components. We have also improved the implementation in a way that our framework can now be applied to calibration scenarios with a large number of points and/or calibration images. These optimizations, which are explained in depth in a new section, drastically reduce both memory consumption and computation time, making our framework more widely applicable. We also executed several new experiments in order to better assess the performance of our approach. Furthermore, we applied our implementation to an additional public dataset, which enables a comparison to a larger number of state-of-the-art methods and further illustrates the advantages of our uncertainty-aware hand–eye calibration. A new chapter contains in-depth discussions and possible future research questions. Finally, our method has become part of the HALCON[1] software from version 23.05 [13] on, which is freely available for academic use[2].

We will give an overview of the related work in Sec. II. Sec. III introduces our generic framework for hand–eye calibration of uncertain robots. In Sec. IV, different aspects of our framework are extensively evaluated including a comparison to 15 state-of-the-art methods. Sec. V concludes our work.

## II. Related Work

Fig. 1 shows the coordinate systems that are relevant for the hand–eye calibration of a robot with a moving camera (for a stationary camera, see [5, Chapter 3.13], for example). Let ${}^{\mathrm{s2}}\mathbf{H}_{\mathrm{s1}}$ denote the $4 \times 4$ homogeneous transformation matrix that represents a rigid 3D transformation of points from coordinate system s1 to s2. One of the most common problem formulations of hand–eye calibration is based on closing the following chain of transformations [14]:

$$ {}^{\mathrm{b}}\mathbf{H}_{\mathrm{w}} = {}^{\mathrm{b}}\mathbf{H}_{\mathrm{t}}\,{}^{\mathrm{t}}\mathbf{H}_{\mathrm{c}}\,{}^{\mathrm{c}}\mathbf{H}_{\mathrm{w}} \quad , \tag{1} $$

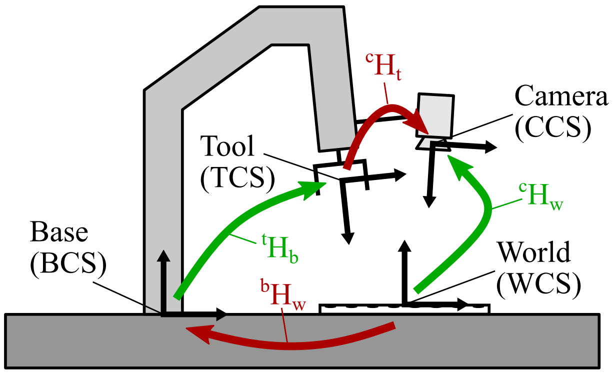
with the coordinate systems world (w, WCS), camera (c, CCS), robot tool (t, TCS), and robot base (b, BCS), as well as the unknown poses ${}^{\mathrm{t}}\mathbf{H}_{\mathrm{c}}$ (hand–eye pose) and ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{w}}$ (world–base pose). The unknown poses are typically determined by moving the robot to different robot poses and taking an image of a calibration target at each pose. The calibration target defines the WCS. At each robot pose, ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{t}}$ is queried from the robot controller and ${}^{\mathrm{c}}\mathbf{H}_{\mathrm{w}}$ is determined by PnP algorithms or camera calibration like [15]. Eq. (1) is often written as

$$ \mathbf{Y} = \mathbf{A}_i \mathbf{X} \mathbf{B}_i \tag{2} $$

with the unknown poses $\mathbf{X}$ and $\mathbf{Y}$ and the observed poses $\mathbf{A}_i$ and $\mathbf{B}_i$ for each of the $n$ robot poses ($i = 1, \ldots, n$). Since the essential unknown is $\mathbf{X}$, we can eliminate $\mathbf{Y}$ by taking one pair of different robot poses $i$ and $j$, yielding $\mathbf{A}_j^{-1}\mathbf{A}_i\mathbf{X} = \mathbf{X}\mathbf{B}_j\mathbf{B}_i^{-1}$. With $\mathbf{A} = \mathbf{A}_j^{-1}\mathbf{A}_i$ and $\mathbf{B} = \mathbf{B}_j\mathbf{B}_i^{-1}$ we obtain

$$ \mathbf{A}\mathbf{X} = \mathbf{X}\mathbf{B}, \tag{3} $$

[1]www.mvtec.com/products/halcon
[2]www.mvtec.com/company/mvtec-on-campus



${}^{\mathrm{c}}\mathrm{H}_{\mathrm{t}}$
Camera
(CCS)
Tool
(TCS)
${}^{\mathrm{c}}\mathrm{H}_{\mathrm{w}}$
Base
(BCS)
${}^{\mathrm{t}}\mathrm{H}_{\mathrm{b}}$
World
(WCS)
${}^{\mathrm{b}}\mathrm{H}_{\mathrm{w}}$

Fig. 1. Relevant 3D coordinate systems, known or observed transformations (green), and unknown (red) transformations for hand–eye calibration where a camera (eye) is rigidly mounted at the robot's end-effector (hand).

where the movement of the tool is represented by $\mathbf{A}$ and the movement of the camera is represented by $\mathbf{B}$, when moving the robot from pose $i$ to $j$.

There are several linear least-squares approaches that basically solve Eq. (2) or Eq. (3), e.g., [16]–[26]. They either determine the rotation and translation parts of the hand–eye pose successively or simultaneously. The latter has the advantage that rotation errors do not propagate and increase translation errors. Approaches that solve Eq. (3) require the selection of suitable pairs of robot poses to compute $\mathbf{A}$ and $\mathbf{B}$. Criteria for the suitability are proposed in [16], [23], and [27], for example. However, it is still difficult to ensure that the observed information ($\mathbf{A}_i$ and $\mathbf{B}_i$) is optimally exploited. Furthermore, these approaches do not explicitly model errors in the robot poses [16].

Because linear approaches typically minimize an algebraic error, their accuracy is limited. Therefore, they are often used to initialize a subsequent non-linear optimization to obtain a higher accuracy. Most non-linear approaches ( [19], [20], [26], [28]–[32], [5, Chapter 3.13.5], [33]), which either minimize an algebraic or geometric error, also do not explicitly model errors in the robot poses. Some of them face the additional problem of weighting the error components of rotation and translation appropriately with respect to each other. Handling uncertainty in hand–eye calibration is a largely neglected topic [34]. In [14], for example, a weighted sum of the rotation and translation error parts is minimized where the weights for the error components are statistically derived. While this takes errors in the robot poses ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{t}}$ into account, error-free camera poses ${}^{\mathrm{c}}\mathbf{H}_{\mathrm{w}}$ are assumed as input. In [35], rotation and translation parts of Eq. (3) are solved successively while errors in $\mathbf{A}$ and $\mathbf{B}$ are propagated to $\mathbf{X}$. In [36], uncertainties of the camera and the robot poses are also considered with covariance matrices and a Cramer-Rao Lower Bound of the uncertainty of $\mathbf{X}$ is determined. In [37] and [38], the rotation and translation parts of $\mathbf{X}$ are solved simultaneously, the uncertainty of $\mathbf{X}$ is derived analytically, and the sensitivity to robot and camera pose noise is investigated. However, in [36], [37], and [38], an algebraic error is minimized. Furthermore, no information about the uncertainty of the robot is provided. Some non-linear approaches do not require initial estimates and provide globally optimal solutions with respect to a certain cost function. In [39], this is achieved by solving a multivariate polynomial optimization problem over semi-algebraic sets using the method of convex linear matrix inequality relaxations. More recently, in [40], this is achieved by using a convex semidefinite programming relaxation. In the presence of noise, however, globally optimal solutions do not imply zero error with respect to the ground truth, but still differ from it [40].

Another class of approaches minimizes the reprojection error of 3D world points on a calibration target (e.g., [41], [42]) similar to camera calibration approaches like [15]. The appealing advantages are that it is unnecessary to explicitly estimate the camera pose in each image in pre-processing, no pre-selection of pose pairs is required, and a meaningful geometric error is minimized in the space of the observed erroneous measurements, i.e., the image points. Minimizing the reprojection error also enables the simultaneous estimation of the interior camera parameters. However, also these methods do not model errors in the robot poses. Recently, in [43], a pose graph optimization framework for hand–eye calibration was introduced, in which the errors of the robot poses are minimized in addition to the reprojection error of the points on a calibration target. Experiments showed the benefit of taking the uncertainty of robot poses into account. Unfortunately, no details are given about the stochastic model, the optimization, and how the different error components are weighted with respect to each other.

Self-calibration techniques are used to perform the hand–eye calibration without a calibration target. For the purpose of this paper, self-calibration does not mean that the interior orientation of the camera is determined but that no calibration target is used for the hand–eye calibration. In [30], unknown 3D points are tracked in the image sequence obtained from a predefined robot motion. The interior camera parameters and the 3D points are simultaneously estimated together with the hand–eye pose. A structure-from-motion (SfM) approach is used in [22], where the unknown scale factor of the SfM result is integrated into the equations. The idea is picked up in [31], where the post-processing step of enforcing the orthogonality of the rotation matrix is avoided by introducing the unknown scale factor into the equations of [19] and [21]. Because these self-calibration techniques minimize algebraic errors, the obtained accuracy is limited. Note that of course also self-calibration techniques can be applied to determine the movement of the camera to solve Eq. (3), provided that a sufficient number of image features in a non-degenerate configuration is available.

Our proposed framework combines the mentioned benefits of minimizing a reprojection error, the advantage of stochastically modelling the uncertainty of all measured observations (image points and robot poses), the flexibility of either performing target-based calibration or self-calibration, and the possibility of either using known interior camera parameters or estimating them simultaneously.

Lastly, we want to highlight the differences of our framework to the approach described in [14], which at first glance could be considered similar to ours. One fundamental difference is that the approach in [14] assumes that the camera poses have been determined in advance with a calibration procedure and that these camera poses are error-free. In contrast, in our approach no camera poses need to be provided for input. The camera poses are determined only implicitly during the optimization. Therefore, by minimizing the reprojection error, uncertainties in the camera poses are considered implicitly in our approach as well. Furthermore, in [14] not the originally observed observations (image points) are modelled as erroneous quantities but rather functionally derived quantities from them, i.e., in form of one rotation and one translation error. The approach of [14] also does not allow to compute calibrated robot poses because errors in the camera poses and robot poses are mixed. Also, the weights that are automatically computed in [14] to balance translation and rotation errors might be biased by the assumption of error-free camera poses and hence cannot represent the robot accuracy. In contrast to [14], our approach models all originally observed and thus error-prone measurements directly as random variables (see Section III-F).

## III. HAND–EYE CALIBRATION OF UNCERTAIN ROBOTS

Fig. 2 shows a flowchart of our uncertainty-aware hand–eye calibration framework. It gives a rough overview of the calibration flow and the variety of use cases. In particular, the flowchart highlights four important aspects of our framework:

- the simultaneous calibration of the interior camera parameters is supported if no camera calibration is done beforehand,
- both target-based calibration as well as self-calibration are supported,
- the automatic variance components estimation and its iterative nature,
- and the variety of outputs.

In this section, we will first describe the camera model and the calibration model, i.e., the relation between 3D world points and their projection into the camera that is mounted on the end-effector of the robot. After a short introduction to parameter estimation models, we will introduce three alternative least-squares estimation models for hand–eye calibration and propose how to obtain initial values for the estimation. The section concludes with implementation details.

### A. Camera Model

We represent the camera by the perspective camera model described in [5, Chapter 3.9.1]. A 3D point $\boldsymbol{p}_\mathrm{w}$ in the WCS is transformed to a point $\boldsymbol{p}_\mathrm{c}$ in the CCS by

$$\boldsymbol{p}_\mathrm{c} = \mathtt{R}\boldsymbol{p}_\mathrm{w} + \boldsymbol{t} \tag{4}$$

where $\boldsymbol{t} = (t_\mathrm{x}, t_\mathrm{y}, t_\mathrm{z})^\top$ is a translation vector and $\mathtt{R}$ is a rotation matrix parameterized by Euler angles: $\mathtt{R} = \mathtt{R}_x(\alpha)\mathtt{R}_y(\beta)\mathtt{R}_z(\gamma)$ (see Eqs. (25)–(27)). The parameters $(t_\mathrm{x},t_\mathrm{y},t_\mathrm{z},\alpha,\beta,\gamma)$ describe the exterior orientation of the camera. With homogeneous coordinates (i.e., after extending $\boldsymbol{p}_\mathrm{w}$ and $\boldsymbol{p}_\mathrm{c}$ with a fourth coordinate of 1), the transformation can be written as a $4 \times 4$ homogeneous matrix:

$$\boldsymbol{p}_\mathrm{c} = {}^\mathrm{c}\mathtt{H}_\mathrm{w}\boldsymbol{p}_\mathrm{w} = \begin{pmatrix} \mathtt{R} & \boldsymbol{t} \\ \boldsymbol{0}^\top & 1 \end{pmatrix} \boldsymbol{p}_\mathrm{w} \; . \tag{5}$$

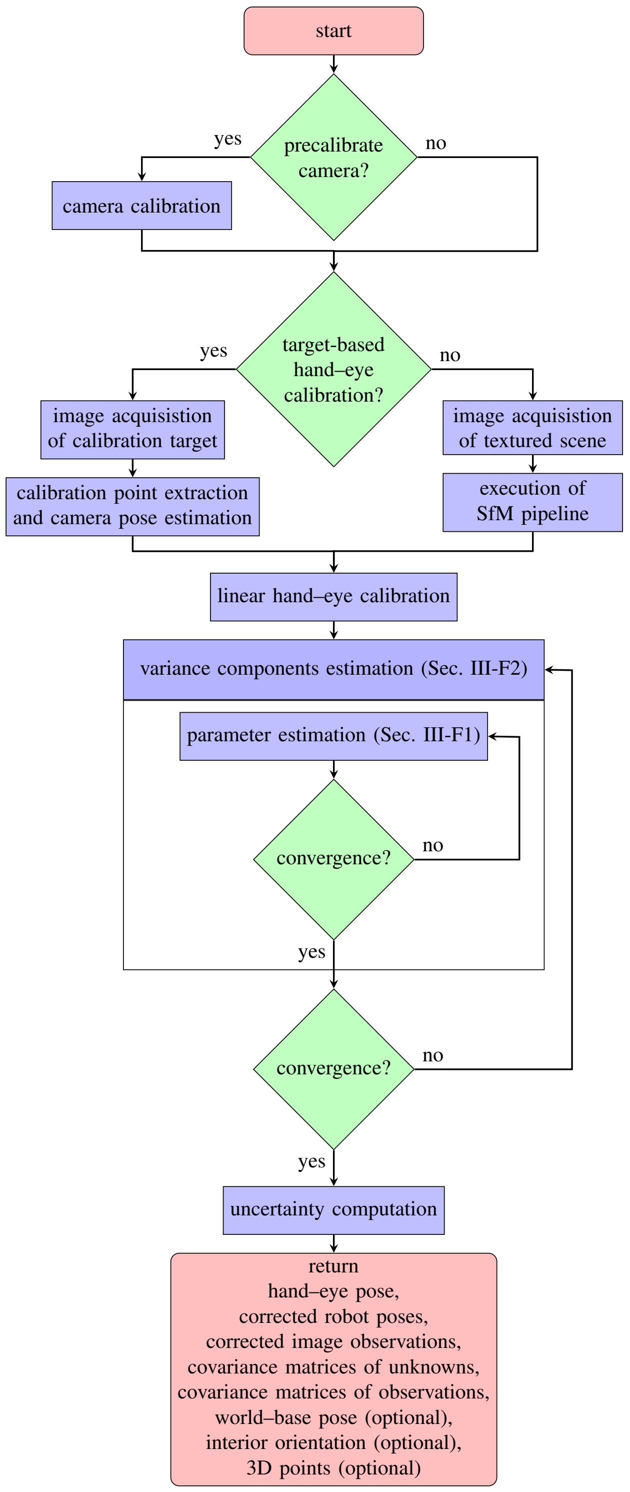

start
precalibrate camera?
yes
no
camera calibration
target-based hand–eye calibration?
yes
no
image acquisition of calibration target
image acquisition of textured scene
calibration point extraction and camera pose estimation
execution of SfM pipeline
linear hand–eye calibration
variance components estimation (Sec. III-F2)
parameter estimation (Sec. III-F1)
convergence?
no
yes
convergence?
no
yes
uncertainty computation
return
hand–eye pose,
corrected robot poses,
corrected image observations,
covariance matrices of unknowns,
covariance matrices of observations,
world–base pose (optional),
interior orientation (optional),
3D points (optional)

Fig. 2. Flowchart of our uncertainty-aware hand–eye calibration framework.

The point $\boldsymbol{p}_\mathrm{c} = (x_\mathrm{c}, y_\mathrm{c}, z_\mathrm{c})^\top$ is then projected into the image plane by

$$\begin{pmatrix} x_\mathrm{u} \\ y_\mathrm{u} \end{pmatrix} = \frac{c}{z_\mathrm{c}} \begin{pmatrix} x_\mathrm{c} \\ y_\mathrm{c} \end{pmatrix} \quad , \tag{6}$$

with the principal distance $c$ of the lens. Even if we will concentrate on a perspective lens, other camera models (e.g., for telecentric lenses) can be included in a straightforward way [5, Chapter 3.9]. Next, the undistorted point $(x_\mathrm{u}, y_\mathrm{u})^\top$ is distorted to $(x_\mathrm{d}, y_\mathrm{d})^\top$, e.g., by either the division model [44]

$$\begin{pmatrix} x_\mathrm{u} \\ y_\mathrm{u} \end{pmatrix} = \frac{1}{1+\kappa r_\mathrm{d}^2} \begin{pmatrix} x_\mathrm{d} \\ y_\mathrm{d} \end{pmatrix} \quad , \tag{7}$$

which models radial distortions by $\kappa$, or the polynomial model, which is based on [45],

$$\begin{pmatrix} x_\mathrm{u} \\ y_\mathrm{u} \end{pmatrix} = \begin{pmatrix} x_\mathrm{d}(1+K_1 r_\mathrm{d}^2 + K_2 r_\mathrm{d}^4 + K_3 r_\mathrm{d}^6) \\ \quad + (P_1(r_\mathrm{d}^2 + 2x_\mathrm{d}^2) + 2P_2 x_\mathrm{d} y_\mathrm{d}) \\ \\ y_\mathrm{d}(1+K_1 r_\mathrm{d}^2 + K_2 r_\mathrm{d}^4 + K_3 r_\mathrm{d}^6) \\ \quad + (2P_1 x_\mathrm{d} y_\mathrm{d} + P_2(r_\mathrm{d}^2 + 2y_\mathrm{d}^2)) \end{pmatrix} \quad , \tag{8}$$

which models radial distortions by $K_1$, $K_2$, and $K_3$ as well as decentering distortions by $P_1$ and $P_2$. Here, $r_\mathrm{d}^2 = x_\mathrm{d}^2 + y_\mathrm{d}^2$. While the division model can be inverted analytically, the polynomial model must be inverted by a numerical root finding algorithm [5, Chapter 3.9.1.3]. This is significantly slower than the analytical inversion in the division model. This is why we implemented the polynomial model the other way round compared to [45]. In machine vision, it is important that in online application the rectification of distorted measurements is fast, for example, when transforming image coordinates to measurements in the world. Thus, it is advantageous to compute the rectification instead of the distortion analytically [5].

Finally, the distorted point $(x_\mathrm{d}, y_\mathrm{d})^\top$ is transformed into the image coordinate system:

$$\boldsymbol{p}_\mathrm{i} = \begin{pmatrix} x_\mathrm{i} \\ y_\mathrm{i} \end{pmatrix} = \begin{pmatrix} x_\mathrm{d}/s_x + c_x \\ y_\mathrm{d}/s_y + c_y \end{pmatrix} \quad , \tag{9}$$

where $(c_x, c_y)^\top$ is the principal point and $s_x$ and $s_y$ denote the pixel pitches on the sensor ($x$ refers to the horizontal and $y$ to the vertical axis of the image).

The six parameters $\boldsymbol{i} = (c, \kappa, s_x, s_y, c_x, c_y)^\top$ for the division model or the ten parameters $\boldsymbol{i} = (c, K_1, K_2, K_3, P_1, P_2, s_x, s_y, c_x, c_y)^\top$ for the polynomial model describe the interior orientation $\boldsymbol{i}$ of the camera.

## *B. Calibration Model*

For hand–eye calibration, the tool of the robot is moved to $n_\mathrm{r}$ different poses and an image is acquired at each pose. For target-based calibration, the calibration target is placed at a fixed position within the workspace of the robot (see Fig. 1). In the self-calibration case, images are acquired from any sufficiently textured object or background scene instead. Let ${}^\mathrm{t}\mathtt{H}_{\mathrm{b},j}$ be the robot pose that is queried from the robot controller at pose $j$ ($j = 1, \ldots, n_\mathrm{r}$). Further let the 3D world points (given in the WCS) of the calibration target or in the scene be $\boldsymbol{p}_k$ ($k = 1, \ldots, n_\mathrm{w}$) and their 2D projections into the image at robot pose $j$ be $\boldsymbol{p}_{j,k}$. Then, we can describe the projection of a 3D point into an image by:

$$\boldsymbol{p}_{j,k} = \pi({}^\mathrm{c}\mathtt{H}_\mathrm{t}\,{}^\mathrm{t}\mathtt{H}_{\mathrm{b},j}\,{}^\mathrm{b}\mathtt{H}_\mathrm{w} \boldsymbol{p}_k, \boldsymbol{i}) \quad , \tag{10}$$

where $\pi(\boldsymbol{p}_\mathrm{c}, \boldsymbol{i})$ is the projection of the point $\boldsymbol{p}_\mathrm{c}$, which is given in the CCS, into the image by successive execution of Eqs. (6), the inverse of either (7) or (8), and (9), by using the parameters of the interior orientation $\boldsymbol{i}$. We denote the vector that contains the six transformation parameters of the robot poses ${}^{\mathrm{t}}\mathtt{H}_{\mathrm{b},j}$ by $\boldsymbol{e}_{\mathrm{t},j}$, that of the unknown hand–eye pose ${}^{\mathrm{c}}\mathtt{H}_{\mathrm{t}}$ by $\boldsymbol{e}_{\mathrm{c}}$, and that of the unknown pose ${}^{\mathrm{b}}\mathtt{H}_{\mathrm{w}}$ by $\boldsymbol{e}_{\mathrm{b}}$ (in analogy with ${}^{\mathrm{c}}\mathtt{H}_{\mathrm{w}}$ above).

## C. Introduction to Estimation Models

Estimation theory is used to estimate unknown parameters (unknowns) from given erroneous and redundant observed parameters (observations). Parameter estimation has a long tradition in photogrammetric computer vision and is often based on weighted least-squares estimation as a special case of Bayesian estimation [46].

Two of the most common models that are applied for this purpose are the Gauss-Markov model and the Gauss-Helmert model, both of which exist in a multitude of variations. One key advantage of these models is that the stochastic properties of the observed and unknown parameters are taken into account by the second moments of their distributions, i.e., their variances and covariances. Furthermore, a variety of established tools exist for evaluating the estimation results, making both models easy to apply in practice. The Gauss-Markov model assumes that the observations can be expressed as an explicit function of the unknowns. Since this assumption is not fulfilled in all practical cases, the more general but also less efficient Gauss-Helmert model is able to represent constraints between observations and unknowns via implicit functions [46].

## D. Estimation in the Gauss-Markov Model

We distinguish between the functional model and the stochastic model [46]. The functional model describes the relations between the observations and the unknowns. In the stochastic model, the observations and the unknowns are treated as random variables with uncertainties, where the uncertainties are described by (co-)variances.

For error-free robot poses, the hand–eye calibration task can be formulated in the Gauss-Markov model [46, Chapter 4.4] because all uncertain observations $\boldsymbol{l}$ can be expressed as a function $\boldsymbol{l} = \boldsymbol{f}(\boldsymbol{x})$ of the unknown parameters $\boldsymbol{x}$, where $\boldsymbol{f}$ describes the functional model and corresponds to Eq. (10).

The vector $\boldsymbol{l}$ contains the measured image points $\boldsymbol{l} = (\boldsymbol{p}_{1,1}^\top, \ldots, \boldsymbol{p}_{1,n_{\mathrm{w}}}^\top, \ldots, \boldsymbol{p}_{n_{\mathrm{r}},n_{\mathrm{w}}}^\top)^\top$. If a 3D point is not visible in a certain image, the respective entry in $\boldsymbol{l}$ is simply omitted. Let $n_{\mathrm{i}}$ denote the total number of measured image points over all images. Then, the number of observations $n_{\mathrm{l}}$ is $2n_{\mathrm{i}}$. For each observed image point, we set up two equations of the form of Eq. (10) yielding $2n_{\mathrm{i}}$ equations. The vector $\boldsymbol{x}$ depends on the application scenario and is formed according to the following recipe:

- For all scenarios, $\boldsymbol{x}$ contains the hand–eye pose: $\boldsymbol{x} := \boldsymbol{e}_{\mathrm{c}}$. Note, however, that for SCARA robots, the z component $t_z$ of the translation part of the hand–eye pose cannot be determined as all robot poses are parallel [47]. To be more specific, the optimization problem would become singular and yield a 1D manifold of equivalent solutions. Therefore, for SCARA robots, $t_z$ is (arbitrarily) set to 0, excluded from the optimization, and determined in post-processing like proposed in [26], i.e., in this case, $\boldsymbol{x}$ contains only the five remaining parameters of $\boldsymbol{e}_{\mathrm{c}}$.
- For the scenario of a target-based calibration, $\boldsymbol{x}$ must be extended by the unknown $\boldsymbol{e}_{\mathrm{b}}$: $\boldsymbol{x} := (\boldsymbol{x}^\top, \boldsymbol{e}_{\mathrm{b}}^\top)^\top$. Note that for the self-calibration scenario, we can reconstruct the 3D points directly in the BCS. Hence, we exclude $\boldsymbol{e}_{\mathrm{b}}$ from the unknowns and internally set $\boldsymbol{e}_{\mathrm{b}} = (0,0,0,0,0,0)^\top$.
- In the scenario where the interior orientation of the camera is unknown and shall be simultaneously estimated, we further extend $\boldsymbol{x} := (\boldsymbol{x}^\top, \boldsymbol{i}^\top)^\top$. Note that $s_y$ is typically excluded from the calibration [5, Chapter 3.9.4.2].
- For the self-calibration scenario, $\boldsymbol{x}$ must be further extended by the coordinates of the 3D points: $\boldsymbol{x} := (\boldsymbol{x}^\top, \boldsymbol{p}_1^\top, \ldots, \boldsymbol{p}_{n_{\mathrm{w}}}^\top)^\top$.

Let the final number of unknown variables in $\boldsymbol{x}$ be $n_{\mathrm{x}}$. It should be noted that the singularity in the Euler angle parameterization is not critical when perturbations with respect to the initial poses of $\boldsymbol{e}_{\mathrm{c}}$ and $\boldsymbol{e}_{\mathrm{b}}$ are used. These perturbations will be close to the identity transformation, such that the singularity of the Euler angle parameterization is irrelevant. Regardless, our framework supports the use of alternative pose parameterizations if desired. The initial poses are obtained by a non-iterative approach, e.g., by the dual-quaternion-based approach of [21] (cf. Sec. III-G).

The stochastic model specifies the statistical properties of the observation process. Assuming that the image points are uncorrelated and are measured with the same accuracy, we set the $n_{\mathrm{l}} \times n_{\mathrm{l}}$ weight coefficient matrix of the observations to the identity: $\mathtt{Q}_{ll} = \mathtt{I}$. Note that it is sufficient to only specify the ratio of the variances of the observations in $\mathtt{Q}_{ll}$. Their absolute values will be estimated later by scaling $\mathtt{Q}_{ll}$ with the variance factor (see Eqs. (13) and (14)).

For linearization, we compute the $2n_{\mathrm{i}} \times n_{\mathrm{x}}$ Jacobian $\mathtt{A}$[3] of $\boldsymbol{f}$ at the initial values of the unknowns $\boldsymbol{x}^{(0)}$ (cf. Sec. III-G):

$$\mathtt{A} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(0)}} \quad . \tag{11}$$

With $\Delta\boldsymbol{l} = \boldsymbol{l} - \boldsymbol{f}(\boldsymbol{x}^{(0)})$ and the weight matrix $\mathtt{P}_{ll} = \mathtt{Q}_{ll}^{-1}$, we can calculate the corrections for the unknowns by solving

$$\mathtt{A}^\top \mathtt{P}_{ll} \mathtt{A} \Delta\hat{\boldsymbol{x}} = \mathtt{A}^\top \mathtt{P}_{ll} \Delta\boldsymbol{l} \tag{12}$$

for $\Delta\hat{\boldsymbol{x}}$ by sparse Cholesky decomposition, for example[4]. For details about the Gauss-Markov model including the derivation of Eq. (12), the interested reader is referred to [46, Chapter 4.4.1].

Then, $\hat{\boldsymbol{x}}^{(1)} = \boldsymbol{x}^{(0)} + \Delta\hat{\boldsymbol{x}}$, and Eqs. (11) and (12) are repeatedly applied until convergence. This corresponds to the minimization of the reprojection error of the 3D points. After convergence, the covariance matrix of the original observations is obtained by

$$\mathtt{C}_{ll} = \hat{\sigma}_0^2 \mathtt{Q}_{ll} \quad , \tag{13}$$

with the variance factor

$$\hat{\sigma}_0^2 = \hat{\boldsymbol{v}}^\top \mathtt{P}_{ll} \hat{\boldsymbol{v}} / r \quad , \tag{14}$$

[3]Note that from here on, $\mathtt{A}$ and $\mathtt{B}$ represent Jacobians in contrast to Section II.

[4]In statistics, the hat operator ˆ denotes an estimated or fitted value.

the residuals $\hat{\boldsymbol{v}} = \mathtt{A}\Delta\hat{\boldsymbol{x}} - \Delta\boldsymbol{l}$, and the redundancy $r = n_\mathrm{l} - n_\mathrm{x}$. The corrected observations are obtained by $\hat{\boldsymbol{l}} = \boldsymbol{l} + \hat{\boldsymbol{v}}$. The covariance matrix of the estimated unknowns is obtained by variance propagation:

$$\mathtt{C}_{\hat{x}\hat{x}} = \hat{\sigma}_0^2(\mathtt{A}^\top \mathtt{P}_{ll}\mathtt{A})^{-1} \tag{15}$$

Note that the special case of this approach with target-based calibration and $\mathtt{Q}_{ll} = \mathtt{I}$ corresponds to the $rp_1$ method (fixed interior orientation) or $rp_2$ method (simultaneously estimated interior orientation) in [42].

## *E. Estimation in the Gauss-Helmert Model*

In order to consider uncertain robot poses, these must be introduced as erroneous observations in addition to the image coordinates. Hence, the observations can no longer be expressed explicitly as a function of the unknowns. For this purpose, parameter estimation must be performed in the Gauss-Helmert model [46, Chapter 4.8]. In the Gauss-Helmert model, our functional model becomes $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{l}) = \boldsymbol{0}$.

In this model, the vector $\boldsymbol{l}$ contains the observed image points as well as the robot poses $\boldsymbol{l} = (\boldsymbol{p}_{1,1}^\top, \ldots, \boldsymbol{p}_{1,n_\mathrm{w}}^\top, \ldots, \boldsymbol{p}_{n_\mathrm{r},n_\mathrm{w}}^\top, \boldsymbol{e}_{\mathrm{t},1}^\top, \ldots, \boldsymbol{e}_{\mathrm{t},n_\mathrm{r}}^\top)^\top$. The number of observation equations is still $2n_\mathrm{i}$. However, the number of observations now becomes $n_\mathrm{l} = 2n_\mathrm{i} + 6n_\mathrm{r}$. The vector $\boldsymbol{x}$ of unknowns is identical to that in the Gauss-Markov model.

In comparison to the Gauss-Markov model, the stochastic model must also consider the uncertainties of the robot poses. Tests on real systems showed that the errors in the robot poses are zero-mean Gaussian distributed [14]. But even if assuming uncorrelated observations, we still would have to know the relative accuracy between the three observation groups (image coordinates, Euler angles of the robot poses, translation components of the robot poses) to set up $\mathtt{Q}_{ll}$. The absolute accuracy values are not required as input. However, they can be computed from the estimation result. Therefore, we initialize $\mathtt{Q}_{ll}$ with reasonable values and use the concept of variance components [46] to estimate the true variances of the observations (see below). We could also initialize $\mathtt{Q}_{ll} = \mathtt{I}$, but then the number of iterations to estimate the variance components would be unnecessarily high. For a higher efficiency, in our implementations we initially set the standard deviations of the image points to $\sigma_\mathrm{i} = 0.1$ pixels, of the Euler angles to $\sigma_\mathrm{a} = 0.1$ deg, and of the translation components to $\sigma_\mathrm{t} = 1$ mm. Then,

$$\mathtt{Q}_{ll} = \mathrm{diag}(\mathrm{rep}(\sigma_\mathrm{i}^2, 2n_\mathrm{i}), \mathrm{rep}([\sigma_\mathrm{a}^2, \sigma_\mathrm{a}^2, \sigma_\mathrm{a}^2, \sigma_\mathrm{t}^2, \sigma_\mathrm{t}^2, \sigma_\mathrm{t}^2], n_\mathrm{r})) \ , \tag{16}$$

where the function $\mathrm{rep}(\boldsymbol{y}, n)$ generates a vector containing $n$ copies of $\boldsymbol{y}$.

In the Gauss-Helmert model, the Jacobian $\mathtt{A}$ is the same as in the Gauss-Markov model. However, in the Gauss-Helmert model, we have to linearize $\boldsymbol{f}$ additionally with respect to the observations yielding the $2n_\mathrm{i} \times n_\mathrm{l}$ Jacobian

$$\mathtt{B} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{l})}{\partial \boldsymbol{l}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(0)}} \ . \tag{17}$$

Note that the upper left $2n_\mathrm{i} \times 2n_\mathrm{i}$ submatrix of $\mathtt{B}$ is a diagonal matrix with all diagonal elements set to -1. With $\boldsymbol{w} = \boldsymbol{f}(\boldsymbol{x}^{(0)}, \boldsymbol{l})$, we can calculate the corrections for the unknowns by solving

$$\mathtt{A}^\top(\mathtt{B}\mathtt{Q}_{ll}\mathtt{B}^\top)^{-1}\mathtt{A}\Delta\hat{\boldsymbol{x}} = -\mathtt{A}^\top(\mathtt{B}\mathtt{Q}_{ll}\mathtt{B}^\top)^{-1}\boldsymbol{w} \tag{18}$$

for $\Delta\hat{\boldsymbol{x}}$. For details about the Gauss-Helmert model including the derivation of Eq. (18), the interested reader is referred to [46, Chapter 4.8].

After convergence, we estimate a variance component for each of the three observation groups. This yields corrected values for $\sigma_\mathrm{i}$, $\sigma_\mathrm{a}$, and $\sigma_\mathrm{t}$, and thus a corrected $\mathtt{Q}_{ll}$ (for details about variance components estimation see Sec. III-F2). Then, the whole estimation process (which is an iterative process by itself) is repeated until the variance components converge. The covariance matrix of the observations is obtained by $\mathtt{C}_{ll} = \hat{\sigma}_0^2\mathtt{Q}_{ll}$ with the redundancy $r = 2n_\mathrm{i} - n_\mathrm{x}$. The covariance matrix of the estimated unknowns is obtained by $\mathtt{C}_{\hat{x}\hat{x}} = \hat{\sigma}_0^2(\mathtt{A}^\top(\mathtt{B}\mathtt{Q}_{ll}\mathtt{B}^\top)^{-1}\mathtt{A})^{-1}$.

In essence, this model allows us to simultaneously minimize the reprojection error and the error in the robot poses by automatically weighting the error terms with respect to each other based on the computed statistics.

## *F. Estimation in the Gauss-Markov Model with Fictitious Observations*

*1) Parameter Estimation:* Because of the expensive matrix computations in the Gauss-Helmert model, we also implemented a more efficient variant of the Gauss-Markov model that is equivalent to the Gauss-Helmert model ( [48], [49]) by using the concept of fictitious observations for the robot poses.

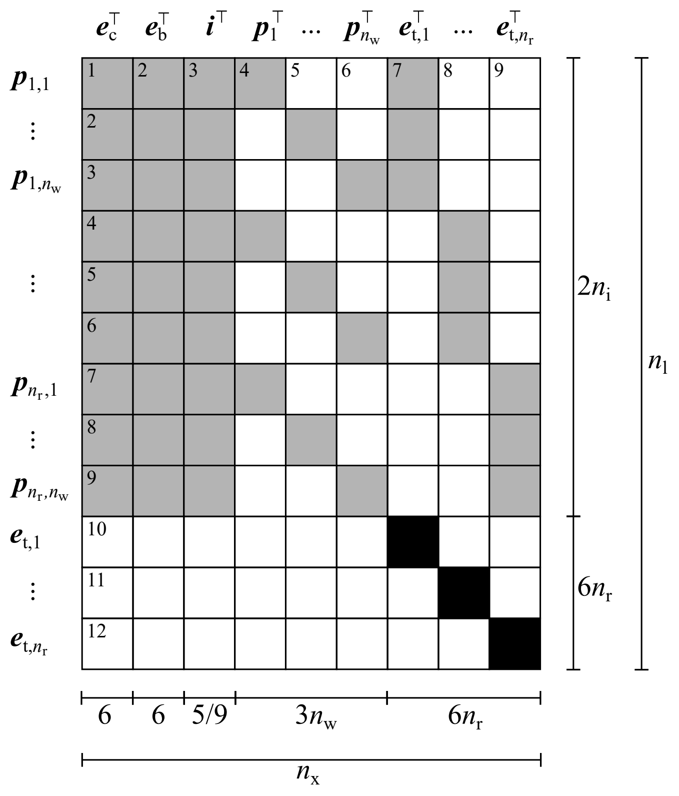
The key idea is to introduce the uncertain robot poses simultaneously as observations and as unknowns. The first part of the functional model is still $\boldsymbol{l} = \boldsymbol{f}(\boldsymbol{x})$. In contrast to Sec. III-D, however, we add 6 additional observation equations of the form $\boldsymbol{e}_{\mathrm{t},j} = \boldsymbol{f}_2(\boldsymbol{e}_{\mathrm{t},j})$ for each robot pose $j$, where $\boldsymbol{f}_2$ is the identity in our case. This results in $n_\mathrm{l} = 2n_\mathrm{i} + 6n_\mathrm{r}$ observation equations. Hence, the vector $\boldsymbol{l}$ contains the observed image points and the robot poses $\boldsymbol{l} = (\boldsymbol{p}_{1,1}^\top, \ldots, \boldsymbol{p}_{1,n_\mathrm{w}}^\top, \ldots, \boldsymbol{p}_{n_\mathrm{r},n_\mathrm{w}}^\top, \boldsymbol{e}_{\mathrm{t},1}^\top, \ldots, \boldsymbol{e}_{\mathrm{t},n_\mathrm{r}}^\top)^\top$ as in Sec. III-E. Simultaneously, we also introduce the robot poses as unknowns by extending $\boldsymbol{x} := (\boldsymbol{x}^\top, \boldsymbol{e}_{\mathrm{t},1}^\top, \ldots, \boldsymbol{e}_{\mathrm{t},n_\mathrm{r}}^\top)^\top$, yielding $n_\mathrm{x}$ unknowns.

Because $\boldsymbol{l}$ is identical to that in the Gauss-Helmert model, we apply the same stochastic model and initialize $\mathtt{Q}_{ll}$ by using Eq. (16).

The computation of the $(2n_\mathrm{i} + 6n_\mathrm{r}) \times n_\mathrm{x}$ Jacobian $\mathtt{A}$ and of the corrections $\Delta\hat{\boldsymbol{x}}$ is done by Eqs. (11) and (12). Note that in this case the lower left part of $\mathtt{A}$ contains zeros and the lower right part the identity $\mathtt{I}_{6n_\mathrm{r}\times 6n_\mathrm{r}}$. Figure 3 shows the complete structure of the Jacobian $\mathtt{A}$.

For the computation of $\mathtt{A}$, an analytic calculation of the partial derivatives is clearly preferable over a numerical calculation because in our case it is faster and the derivatives are more accurate, which leads to a faster convergence of the optimization. All partial derivatives are given in the Appendix.

After convergence, the variance components are estimated as described in Sec. III-F2. Finally, $\mathtt{C}_{ll} = \hat{\sigma}_0^2\mathtt{Q}_{ll}$, $\hat{\boldsymbol{l}}$, and $\mathtt{C}_{\hat{x}\hat{x}}$ are computed as described in Sec. III-D.



$\boldsymbol{e}_\mathrm{c}^\top$ $\boldsymbol{e}_\mathrm{b}^\top$ $\boldsymbol{i}^\top$ $\boldsymbol{p}_1^\top$ … $\boldsymbol{p}_{n_\mathrm{w}}^\top$ $\boldsymbol{e}_{\mathrm{t},1}^\top$ … $\boldsymbol{e}_{\mathrm{t},n_\mathrm{r}}^\top$
$\boldsymbol{p}_{1,1}$
$\boldsymbol{p}_{1,n_\mathrm{w}}$
$\boldsymbol{p}_{n_\mathrm{r},1}$
$\boldsymbol{p}_{n_\mathrm{r},n_\mathrm{w}}$
$\boldsymbol{e}_{\mathrm{t},1}$
$\boldsymbol{e}_{\mathrm{t},n_\mathrm{r}}$
1 2 3 4 5 6 7 8 9 10 11 12
$2n_\mathrm{i}$
$6n_\mathrm{r}$
$n_\mathrm{l}$
6 6 5/9 $3n_\mathrm{w}$ $6n_\mathrm{r}$
$n_\mathrm{x}$

Fig. 3. Structure of the Jacobian $\mathbf{A}$ in the Gauss-Markov model with fictitious observations. The rows represent observations, the columns represent unknowns. White blocks indicate that all entries are 0, gray blocks indicate partial derivatives that are typically different from 0 and black blocks indicate the identity matrix. Block rows 1 to 9 represent the measured image points, block rows 10 to 12 represent the robot poses, i.e., the fictitious observations. Block column 1 represents the hand–eye pose. Block column 2 represents the pose of the calibration target and is omitted in the self-calibration case. Block column 3 represents the interior camera parameters, which, depending on the lens distortion model, contains either 5 or 9 elements when excluding $s_y$ from the calibration. Block column 3 is omitted if the camera parameters are already known and should not be estimated. Block columns 4 to 6 represent the 3D world points, and hence are omitted in the case of target-based calibration. Block columns 7 to 9 represent the robot poses.

*2) Variance Components Estimation:* To estimate the variance components of the three observation groups we have to generalize the stochastic model of the Gauss-Markov model [46, Chapter 4.2.4]. To put it simply, it is no longer sufficient to calculate a single global variance factor $\hat{\sigma}_0^2$ as in Sec. III-D. Instead, we need to estimate an individual variance factor for each observation group. This enables the correct modeling of the stochastic properties of each observation group. Note that because we are interested in the absolute values of the variances, two variance factors that model only the ratio between the three observation groups would not be sufficient. The steps that are necessary for variance components estimation are described in this section.

1) First, we compute the weight coefficient matrix of the estimated unknowns $\hat{\boldsymbol{x}}$:

$$\mathbf{Q}_{\hat{x}\hat{x}} = (\mathbf{A}^\top \mathbf{P}_{ll} \mathbf{A})^{-1} \ . \tag{19}$$

Note that this computation step was already performed during parameter estimation (see Eq. (12) in Sec. III-D and Sec. III-F1).

2) Then, we apply error propagation to obtain the weight coefficient matrix of the corrected observations $\hat{\boldsymbol{l}}$:

$$\mathbf{Q}_{\hat{l}\hat{l}} = \mathbf{A}\mathbf{Q}_{\hat{x}\hat{x}}\mathbf{A}^\top \ . \tag{20}$$

3) From this, we compute the weight coefficient matrix of the estimated residuals $\hat{\boldsymbol{v}}$

$$\mathbf{Q}_{\hat{v}\hat{v}} = \mathbf{Q}_{ll} - \mathbf{Q}_{\hat{l}\hat{l}} \tag{21}$$

and the so-called redundancy matrix

$$\mathbf{R} = \mathbf{Q}_{\hat{v}\hat{v}}\mathbf{P}_{ll} = \mathbf{I} - \mathbf{Q}_{\hat{l}\hat{l}}\mathbf{P}_{ll} \ . \tag{22}$$

The main diagonal of $\mathbf{R}$ contains the redundancy numbers, i.e., the contribution of a single observation to the total redundancy $r$, with $\sum_{i=1}^{n_\mathrm{l}} R_i = r$ and $R_i$ being the main diagonal element of $\mathbf{R}$ at row and column $i$. Recall that the total redundancy is the difference between the number of observations and the number of unknowns $r = n_\mathrm{l} - n_\mathrm{x}$.

4) Let $I_\mathrm{i}$, $I_\mathrm{a}$, and $I_\mathrm{t}$ be the sets of observation indices that contain the observed image point coordinates, the observed Euler angles of the robot poses, and the observed translation components of the robot poses, respectively. Further, let $\hat{\boldsymbol{v}}_\mathrm{i}$, $\hat{\boldsymbol{v}}_\mathrm{a}$, and $\hat{\boldsymbol{v}}_\mathrm{t}$ be vectors that contain the subset of the elements of the residual vector $\hat{\boldsymbol{v}}$ that correspond to the respective observation group and let $\mathbf{P}_{ll,\mathrm{i}}$, $\mathbf{P}_{ll,\mathrm{a}}$, and $\mathbf{P}_{ll,\mathrm{t}}$ be matrices that contain the subset of the rows and columns of the weight matrix $\mathbf{P}_{ll}$ that correspond to the respective observation group. Then, the variance components of the three observation groups can be computed as

$$\hat{\sigma}_{0,\mathrm{i}}^2 = \frac{\hat{\boldsymbol{v}}_\mathrm{i}^\top \mathbf{P}_{ll,\mathrm{i}} \hat{\boldsymbol{v}}_\mathrm{i}}{\sum_{i \in I_\mathrm{i}} R_i}, \hat{\sigma}_{0,\mathrm{a}}^2 = \frac{\hat{\boldsymbol{v}}_\mathrm{a}^\top \mathbf{P}_{ll,\mathrm{a}} \hat{\boldsymbol{v}}_\mathrm{a}}{\sum_{i \in I_\mathrm{a}} R_i}, \hat{\sigma}_{0,\mathrm{t}}^2 = \frac{\hat{\boldsymbol{v}}_\mathrm{t}^\top \mathbf{P}_{ll,\mathrm{t}} \hat{\boldsymbol{v}}_\mathrm{t}}{\sum_{i \in I_\mathrm{t}} R_i}. \tag{23}$$

5) By multiplying the initially chosen variances of the observations with the variance components, we obtain improved estimates of the variances:

$$\sigma_\mathrm{i}^2 \leftarrow \hat{\sigma}_{0,\mathrm{i}}^2 \sigma_\mathrm{i}^2 \ , \ \sigma_\mathrm{a}^2 \leftarrow \hat{\sigma}_{0,\mathrm{a}}^2 \sigma_\mathrm{a}^2 \ , \ \sigma_\mathrm{t}^2 \leftarrow \hat{\sigma}_{0,\mathrm{t}}^2 \sigma_\mathrm{t}^2 \ . \tag{24}$$

6) They are used to update the stochastic model. The update is performed by setting up a new weight coefficient matrix $\mathbf{Q}_{ll}$ where the entries on its main diagonal are set to the new values of $\sigma_\mathrm{i}^2$, $\sigma_\mathrm{a}^2$, and $\sigma_\mathrm{t}^2$.

Then, the parameter estimation is restarted with the original observations but with the new matrix $\mathbf{Q}_{ll}$. The process is repeated until the variance components converge, i.e., $\hat{\sigma}_{0,\mathrm{i}}^2$, $\hat{\sigma}_{0,\mathrm{a}}^2$, and $\hat{\sigma}_{0,\mathrm{t}}^2$ are all sufficiently close to 1.0. This is typically the case after 3 to 5 iterations.

Note that when estimating variance components, the global variance factor $\hat{\sigma}_0^2$ does not need to be computed by using Eq. (14) because convergence of the variance components automatically ensures that $\hat{\sigma}_0^2 = 1.0$. For more details about variance components estimation, the interested reader is referred to [46].

### G. Initialization of the Unknowns

Because of the nonlinearity of the optimization problem, initial values for the unknowns need to be provided. We propose the following initialization for $\boldsymbol{x}$:

- Initial values for the hand–eye pose $\boldsymbol{e}_\mathrm{c}$ and for $\boldsymbol{e}_\mathrm{b}$ are obtained from a linear approach (e.g., [21] for antropomorphic robots or [26] for SCARA robots).
- Initial values for the interior orientation are obtained from the data sheets of the camera ($s_x$ and $s_y$) and the lens ($c$). The principal point $(c_x,c_y)^\top$ is set to the image center and the distortion coefficients are set to 0.
- For self-calibration, a SfM pipeline (e.g., COLMAP [50]) is applied to the acquired images. It returns the camera parameters $\boldsymbol{i}$, the reconstructed 3D points $\boldsymbol{p}_k$, and for each image $j$ the extracted 2D keypoints $\boldsymbol{p}_{j,k}$ and the exterior orientation ${}^\mathrm{c}\mathbf{H}_{\mathrm{w},j}$.

In the self-calibration case, we fix the inherent unknown scale factor of SfM in the initial values for $\boldsymbol{p}_k$ and ${}^\mathrm{c}\mathbf{H}_{\mathrm{w},j}$ by applying the following steps: For all pairs of robot poses $j1$ and $j2$, we compute the camera movement ${}^\mathrm{c2}\mathbf{H}_\mathrm{c1} = {}^\mathrm{c}\mathbf{H}_{\mathrm{w},j2}{}^\mathrm{c}\mathbf{H}_{\mathrm{w},j1}^{-1}$ and the tool movement ${}^\mathrm{t2}\mathbf{H}_\mathrm{t1} = {}^\mathrm{t}\mathbf{H}_{\mathrm{b},j2}{}^\mathrm{t}\mathbf{H}_{\mathrm{b},j1}^{-1}$. We then convert both rigid 3D transformations to screw parameters [21]. From the screw congruency theorem [18] we known that the screw translation parameter $d_\mathrm{t}$ of the tool movement is identical to the screw translation parameter $d_\mathrm{c}$ of the camera movement. Therefore, the ratio $d_\mathrm{t}/d_\mathrm{c}$ reflects the unknown scale factor of the SfM. For a higher robustness, the median over the ratios of all pairs is computed while ignoring ratios for which $d_\mathrm{t}$ or $d_\mathrm{c}$ is below a noise threshold. Finally, the scale factor is used to correct the initial values for $\boldsymbol{p}_k$ and ${}^\mathrm{c}\mathbf{H}_{\mathrm{w},j}$ before starting the optimization.

### H. Implementation of the Gauss-Markov Model with Fictitious Observations

Pseudocode of the proposed uncertainty-aware hand–eye calibration with the Gauss-Markov model with fictitious observations (GMF) is given in Fig. 11 in the Appendix.

In this section, details about an efficient implementation of the parameter estimation in the Gauss-Markov model with fictitious observations are given. Even though the computations in this model are less demanding than in the Gauss-Helmert model, a straightforward implementation of the presented algorithm would still result in very long computation times and a huge memory consumption, especially for a large number of observed image points. Therefore, we will describe how to optimize the implementation while preserving the exact result:

- A first set of optimizations is based on the fact that $\mathbf{Q}_{ll}$ is a diagonal matrix. Hence, $\mathbf{P}_{ll}$ is efficiently obtained by computing the reciprocals of the main diagonal elements of $\mathbf{Q}_{ll}$. Furthermore, when computing the corrections of the unknowns by solving (12), two multiplications can be sped up. Firstly, the matrix multiplication of $\mathbf{P}_{ll}$ with $\mathbf{A}$ is replaced by scaling each row of $\mathbf{A}$ by the corresponding diagonal element of $\mathbf{P}_{ll}$. Secondly, the matrix multiplication of $(\mathbf{A}^\top\mathbf{P}_{ll}\mathbf{A})^{-1}\mathbf{A}^\top$ with $\mathbf{P}_{ll}$ is replaced by scaling each column of the former matrix by the corresponding diagonal element of $\mathbf{P}_{ll}$. Each of both replacements reduces the number of calculation operations from $2n_\mathrm{l}^2n_\mathrm{x}$ to $n_\mathrm{l}n_\mathrm{x}$.
- A second set of optimizations can be applied because in Sec. III-F2 we saw that the entire redundancy matrix $\mathbf{R}$ does not need to be computed. Instead, it is sufficient to know its main diagonal to compute the variance components. Therefore, and because $\mathbf{P}_{ll}$ is a diagonal matrix as well, it follows from (22) that also for $\mathbf{Q}_{\hat{v}\hat{v}}$ only its main diagonal needs to be computed. Furthermore, we can conclude from (21) that the same holds for $\mathbf{Q}_{\hat{l}\hat{l}}$, too. Note that the right product in (20), i.e., $\mathbf{Q}_{\hat{x}\hat{x}}\mathbf{A}^\top$, is already available from the computation of the corrections of the unknowns in (12). Hence, the elements of the main diagonal of $\mathbf{Q}_{\hat{l}\hat{l}}$ can be efficiently computed by the scalar products of each row of $\mathbf{A}$ with the corresponding column of $\mathbf{Q}_{\hat{x}\hat{x}}\mathbf{A}^\top$. According to (22), each resulting diagonal element is multiplied by the corresponding diagonal element of $\mathbf{P}_{ll}$ and subtracted from 1.0 to obtain the corresponding element of $\mathbf{R}$. This drastically reduces the number of calculation operations from $2n_\mathrm{l}^4 + (2n_\mathrm{x}+1)n_\mathrm{l}^2 + 2n_\mathrm{x}^2n_\mathrm{l}$ to $2n_\mathrm{x}n_\mathrm{l}$.
- Also the computation of the numerators in (23) can be sped up by computing the element-wise product of the main diagonal of $\mathbf{P}_{ll}$ and the squared elements of $\hat{\boldsymbol{v}}$ and summing up the results. This reduces the number of calculation operations from $2n_\mathrm{l}^2 + 2n_\mathrm{l}$ to $3n_\mathrm{l}$ for each observation group[5].
- After the convergence of the variance components, the computation of the statistics can be optimized as well: The computation of $\hat{\sigma}_0^2$ is optimized in analogy with the computation of the variance components described above. The main diagonal of the covariance matrix of the original observations $\mathbf{C}_{ll}$ is obtained by scaling the main diagonal of $\mathbf{Q}_{ll}$ with $\hat{\sigma}_0^2$.

Through the described optimizations we not only speed up the calculations but also manage to avoid the computation of matrices of size $n_\mathrm{l} \times n_\mathrm{l}$. This is important because $n_\mathrm{l}$ may become huge in practice. To give one example, the optimizations reduce the overall memory consumption of the hand–eye calibration from approximately 3.9 GB to 100 MB and the computation time from 72 s to 2 s (for $n_\mathrm{r}$=25 robot poses and $n_\mathrm{w}$=200 world points, which results in $n_\mathrm{l} = 10150$).

## IV. EVALUATION

This section presents the results of multiple experiments that show the validity and the advantages of our uncertainty-aware hand–eye calibration. We will first evaluate different selected aspects of our proposed model based on simulated data that we generated. Afterwards, experiments on real data are intended to demonstrate the advantages in application scenarios. These include experiments with our own robot as well as evaluations on a public dataset of a real laboratory. Finally, a comprehensive comparison with 15 state-of-the-

[5]Note that here, $n_\mathrm{l}$ exceptionally represents the number of observations in the respective observation group

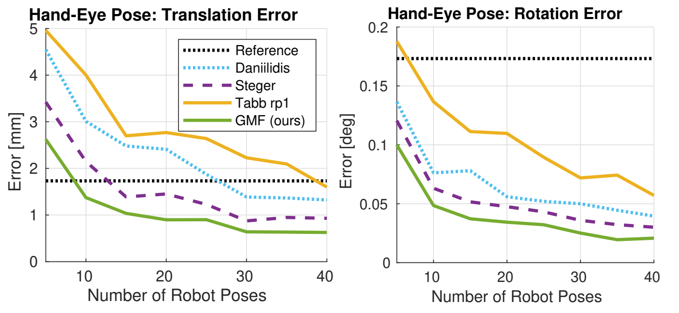


Fig. 4. Translation and rotation error of the hand–eye pose for different number of robot poses $n_\mathrm{r}$ and different approaches: Daniilidis [21], Steger [5], Tabb rp1 [42], GMF (ours). 'Reference' visualizes the uncertainty of the simulated robot, i.e., the L2-Norm of the standard deviations applied to the robot poses.

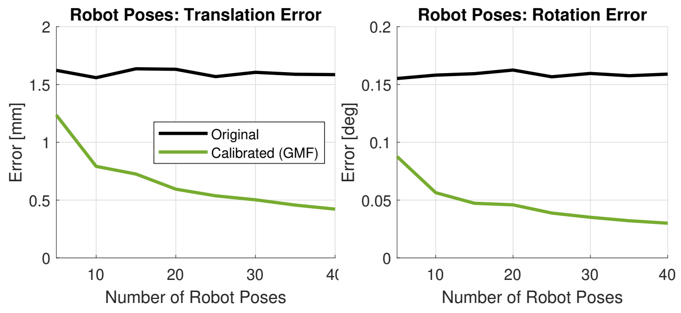


Fig. 5. Translation and rotation error of the robot poses ${}^{\mathrm{t}}\mathbf{H}_{\mathrm{b},j}$ before (black) and after (green) GMF calibration.

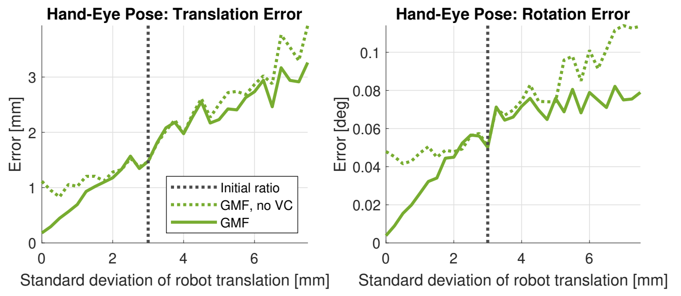


Fig. 6. Translation and rotation error of the hand–eye pose with (GMF) and without (GMF, no VC) estimating variance components. The standard deviation of the rotation components of the simulated robot was set to 0.3 deg and the standard deviation of the translation components was varied. The translation standard deviation that corresponds to the initial ratio $0.3 \cdot \sigma_\mathrm{t}/\sigma_\mathrm{a}$ mm is indicated by the vertical dotted line.

art methods based on a public dataset of synthetic images completes the section.

## A. Model Validation on Simulated Data

We simulated an uncertain robot by adding zero-mean Gaussian noise to its robot poses ${}^{\mathrm{t}}\mathbf{H}_{\mathrm{b},j}$ with a standard deviation of 1 mm in the translation components and of 0.1 deg in the orientation components. To the image points, we applied noise with a standard deviation of 0.1 pixels.

For calibration, we simulated a planar calibration target with $n_\mathrm{w} = 40$ points and a diameter of about 1 m and placed it in the workspace of the robot. We set the hand–eye pose ${}^{\mathrm{c}}\mathbf{H}_\mathrm{t}$ to different values with a translation of about 10 cm and different orientations. For the camera of resolution $1280 \times 1024$ we assumed the division model and set $c = 8$ mm, $\kappa = 2000\,\mathrm{m}^{-2}$, $s_x = 5.21\,\mu$m, $s_y = 5.2\,\mu$m, and $(c_x, c_y) = (645{,}502)$ pixels. We generated 40 random robot poses ${}^{\mathrm{t}}\mathbf{H}_{\mathrm{b},j}$ in its 1 m$^3$ workspace such that at least 90% of the calibration points were visible in the image.

First, we calibrated the interior orientation of the camera by using the acquired images and the approach of [5, Chapter 3.9.4]. Then, for hand–eye calibration we applied the linear approach of [21] (Daniilidis), the non-linear approach of [5, Chapter 3.13.5] (Steger), the approach based on the Gauss-Markov model of Sec. III-D (Tabb rp1), which is equivalent to the $rp_1$ method in [42], and the Gauss-Markov model with fictitious observations of Sec. III-F (GMF). All non-linear approaches were initialized with the result of [21]. In all experiments, the results of the Gauss-Helmert model of Sec. III-E were identical to that of GMF (up to numerical inaccuracies). This shows the validity of using the more efficient GMF instead of the Gauss-Helmert model and also makes the presentation of the results of the Gauss-Helmert model superfluous. We repeated each experiment 30 times and display the mean values in the following.

Fig. 4 shows the translation and rotation error of the computed hand–eye poses ${}^{\mathrm{c}}\mathbf{H}_\mathrm{t}$. Let $\mathbf{H}$ represent the computed pose and $\mathbf{H}_\mathrm{gt}$ the ground truth pose. We compute the translation error as the length of the translation part of $\mathbf{H} - \mathbf{H}_\mathrm{gt}$. For the rotation error, we compute the magnitude of the Rodrigues rotation of $\mathbf{H}\mathbf{H}_\mathrm{gt}^{-1}$. As expected, the errors of all approaches generally decrease when increasing the number of robot poses. GMF performs best in all cases. Furthermore, the advantage of modeling the uncertainty of the robot can be clearly seen from the difference between Tabb rp1, which is equivalent to the Gauss-Markov model of Sec. III-D, and GMF. Note that the translation error might even exceed the uncertainty of the robot especially for fewer robot poses.

Fig. 5 shows the translation and rotation error of the robot poses before and after GMF calibration (the calibrated robot poses are contained in $\hat{\boldsymbol{l}}$). GMF is able to decrease the errors in the robot poses to about 20–25%. Thus, GMF not only returns an accurate hand–eye pose but also enables a convenient vision-based robot calibration where the calibrated robot poses are assumed as ground-truth poses for the estimation of the robot parameters (e.g., Denavit-Hartenberg parameters [11]).

To analyze the benefit of estimating the variance components, we fixed the number of robot poses to 40 and set the standard deviation of the rotation components of the simulated robot poses to 0.3 deg. Then, we varied the standard deviation of the translation components of the robot from 0 to 7.5 mm and repeated each experiment 30 times. The mean error of the hand–eye pose is shown in Fig. 6. As before, we initialized $\mathbf{Q}_{ll}$ with $\sigma_\mathrm{a} = 0.1$ deg and $\sigma_\mathrm{t} = 1$ mm. The positive effect of estimating the variance components on the accuracy is obvious and increases the more the assumed ratio deviates from the true ratio of standard deviations.

Another advantage of estimating the variance components is that a meaningful accuracy of the robot is returned in $\mathbf{C}_{ll}$, which otherwise would require costly robot calibration. On average, the accuracy (i.e., the standard deviations) of the robot was computed with an error of 61.1% in translation and 53.3% in rotation without estimating variance components, and with an error of only 0.8% and 1.0% when estimating variance components. The mean standard deviation of the image points was estimated as 0.14 pixels without, and correctly as 0.10 pixels with estimating variance components.

In general, the experiments indicate that the hand–eye calibration is solvable in the proposed way. They show that the chosen problem formulation is not ill-posed and the system of normal equations (12) does not become singular. Furthermore, they show that the topography of the error surface allows the optimization to converge to the desired minimum. The experiment also indicates that the initial values that are obtained by the linear approach of [21] are sufficiently accurate.

Next, we simulated a robot with systematic errors. For this, we took the Denavit-Hartenberg (DH) parameters [11], [51] of a UR3e robot arm and added systematic errors to the parameters of each of its six axes: We applied random systematic offsets with a standard deviation of 0.05 mm to the link length parameters $a$ and to the link offset parameters $d$, random systematic offsets with a standard deviation of 0.02 deg to the link twist parameters $\alpha$ and to the joint angle parameters $\theta$, and a random systematic scaling $1+\epsilon$ of the joint angle parameters $\theta$ with a standard deviation of 0.0001 for $\epsilon$. Then, we simulated 40 robot poses by applying the forward kinematics [51] once with the erroneous DH parameters and once with the original DH parameters. We then projected the 3D points of a calibration target with 837 points for each robot pose into the camera image by using the erroneous robot pose and added noise to the image points with a standard deviation of 0.1 pixels. The robot poses that were computed with the original DH parameters were used as input to the same hand–eye calibration approaches as before. Again, all non-linear approaches were initialized with the result of [21]. For now, the camera parameters were assumed to be known. For the resulting hand–eye poses, we again computed the error in the translation part and in the rotation part. We repeated the experiment 20 times with different systematic errors in the DH parameters and computed the mean error in translation and rotation over the 20 results. Finally, we repeated the whole experiment for different noise levels by multiplying the above standard deviations of the DH parameters by a noise factor ranging from 0 to 10, while keeping the noise in the image points constant. The results are shown in Fig. 7. Note that no additional noise was added to the robot poses. The experiment shows that our approach is also able to cope with systematic robot errors very well.

For a noise factor of 1.0, we also tested the simultaneous estimation of the interior camera parameters. Table I shows the initial values for the camera parameters, the ground truth values, the results of our approach and the standard deviations, which are also returned by our approach. It can be seen that the estimated camera parameters are very close to their ground true values. Also, the standard deviations that are returned by our approach are on the same order of magnitude as the errors in the parameters, and hence are plausible.

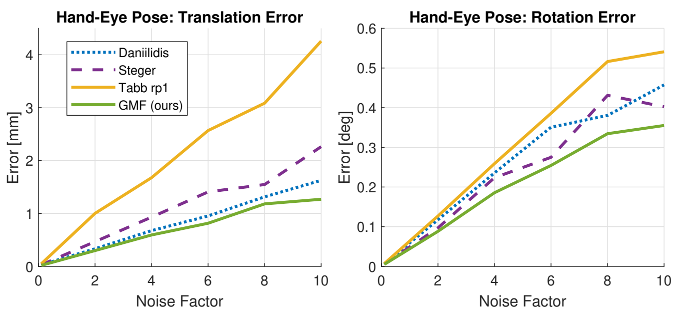


Fig. 7. Translation and rotation error of the resulting hand–eye pose for a robot with systematic errors in the Denavit-Hartenberg parameters.

TABLE I
RESULTS OF THE SIMULTANEOUS CAMERA CALIBRATION. PARAMETERS INDICATED BY * ARE EXCLUDED FROM THE CALIBRATION.

| Camera Parameter | Initial Values | Ground Truth Values | Estimated Values | Standard Deviation |
|---|---|---|---|---|
| $f$ (mm) | 8.0 | 8.4300 | 8.4303 | 0.0002 |
| $\kappa$ (m$^{-2}$) | 0.0 | 1000.00 | 999.92 | 0.54 |
| $s_x$ ($\mu$m Pixel$^{-1}$) | 5.2 | 5.21000 | 5.20997 | 0.00003 |
| $s_y$ ($\mu$m Pixel$^{-1}$)* | 5.2 | 5.20000 | 5.20000 | 0.00000 |
| $c_x$ (Pixel) | 640.0 | 660.00 | 659.99 | 0.02 |
| $c_y$ (Pixel) | 512.0 | 482.00 | 481.96 | 0.02 |

## B. Experiments on Real Data

For our experiments, we attached a $2448 \times 2048$ IDS U3-3280SE camera with an 8 mm lens at the end-effector of a 6-axis UR3e robot arm (0.5 m working radius). To test the target-based calibration, we put a planar calibration target with known control points[6] in the workspace of the robot and acquired images at 25 different robot poses (Fig. 8, top). To test the self-calibration capabilities of our model, we replaced the calibration target by a planar print and acquired 25 images by using the same robot poses (Fig. 8, bottom).

Table II displays the reprojection root mean squared error (RRMSE) and the reconstruction accuracy error (RAE), which is the average squared Euclidean error of the reconstructed calibration points, as proposed in [42]. For the evaluation, we compare different variants of GMF with the methods of [21] (Daniilidis), [5] (Steger), [42] (Tabb rp1), and [14] (Strobl) as a representative that models errors in the robot poses. For the evaluation of [14], we used the implementation of [54], which allows to chose whether the errors in the robot poses (Strobl 1) or in the camera poses (Strobl 2) are minimized. For GMF, 'fixed camera' means that camera parameters are pre-calibrated separately by using [5, Chapter 3.9.4] and are excluded from the hand–eye calibration. 'orig. poses' means that the errors are calculated with the original robot poses instead of the calibrated ones. 'self-calibration' means that no calibration target was used (in this case, RAE cannot be computed because the true 3D points are unknown). For the GMF, we again initialized $\mathbf{Q}_{ll}$ with $\sigma_i = 0.1$ pixels, $\sigma_\mathrm{a} = 0.1$ deg and $\sigma_\mathrm{t} = 1$ mm. For the calibrated robot poses,

[6]The image coordinates of the circular control points are extracted by fitting ellipses to their subpixel-accurate image edges. This causes a bias in the point positions [52]. For most applications, this distortion is so small that it can be neglected. However, the bias can be removed with the approach described in [53] if necessary.

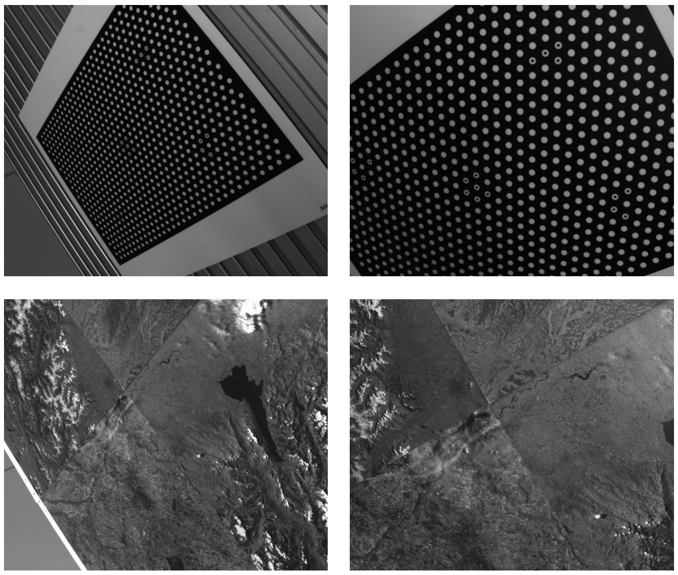


Fig. 8. Two example calibration images for the target-based hand–eye calibration (top row) and for hand–eye self-calibration (bottom row).

TABLE II
RRMSE AND RAE ERRORS ON THE UR3E DATA

| Method | RRMSE (pixel) | RAE (mm$^2$) | Time (s) |
|---|---|---|---|
| Daniilidis [21] | 0.708 | 0.00462 | 0.2 |
| Steger [5] | 0.608 | 0.00212 | 0.2 |
| Strobl 1 [14] | 1.082 | 0.04248 | 1.6 |
| Strobl 2 [14] | 1.152 | 0.07076 | 1.6 |
| Tabb rp1 [42], fixed camera | 0.567 | 0.00194 | 0.3 |
| Tabb rp1 [42] | 0.545 | 0.00212 | 0.3 |
| GMF, orig. poses, fixed camera | 0.784 | 0.02190 | 3.4 |
| GMF, orig. poses | 0.771 | 0.02103 | 3.4 |
| GMF, fixed camera | 0.196 | 0.00073 | 3.4 |
| GMF | 0.193 | 0.00073 | 3.4 |
| GMF, self-calibration | 0.178 | - | 14.2 |

GMF returns the smallest errors for target-based calibration and for self-calibration.

Table II also shows the computation times of the different methods measured on an Intel Core i7-8565U with 1.8 GHz. Compared to the methods of [21], [5], and [42], GMF is slower by a factor of approximately 10. This is mainly caused by the additional iteration that is needed in GMF for estimating the variance components. Nevertheless, the absolute value of 3.4 s is still fast enough for practical applications where the acquisition time of the calibration images takes by far the most time. In the self-calibration case, the computation time of GMF increases to 14.2 s because of the higher number of unknowns, i.e., the reconstructed 3D points.

It is also interesting to have a look at the standard deviations of the image points, the robot translations, and the robot rotations, which are returned by GMF. They are 0.23 pixels, 0.093 mm, and 0.029 deg in the target-based calibration and 0.32 pixels, 0.097 mm, and 0.025 deg in the self-calibration case. Thus, the robot uncertainty was estimated consistently. For comparison, the manufacturer specifies the repeatability of the robot as 0.03 mm. Consequently, our result suggests that the uncertainty of the robot is worse than its repeatability by a factor of 3, which is plausible. Furthermore, the image points of the calibration target are extracted with a slightly higher accuracy compared to the feature points in the prints, which is plausible as well. The resulting hand–eye poses for target-based calibration and self-calibration are shown in Table III.

TABLE III
RESULTING HAND–EYE POSES

| Method | $t_x$ (mm) | $t_y$ (mm) | $t_z$ (mm) | $\alpha$ (deg) | $\beta$ (deg) | $\gamma$ (deg) |
|---|---|---|---|---|---|---|
| GMF | 63.64 | 135.21 | 45.67 | 2.26 | 25.39 | 176.35 |
| GMF, self-calib | 63.79 | 135.18 | 45.76 | 2.24 | 25.35 | 176.35 |

TABLE IV
ESTIMATED CAMERA PARAMETERS. PARAMETERS INDICATED BY * ARE EXCLUDED FROM THE CALIBRATION.

| Camera Parameter | Initial Values | Pre-Calibration | Simultaneous Calibration | Standard Deviation |
|---|---|---|---|---|
| $f$ (mm) | 8.0 | 8.143883 | 8.143442 | 0.000296 |
| $\kappa$ (m$^{-2}$) | 0.0 | -1828.362 | -1834.429 | 2.176 |
| $s_x$ ($\mu$m pixel$^{-1}$) | 3.45 | 3.449968 | 3.449982 | 0.000026 |
| $s_y$ ($\mu$m pixel$^{-1}$)* | 3.45 | 3.450000 | 3.450000 | 0.000000 |
| $c_x$ (pixel) | 1224.0 | 1249.083 | 1249.081 | 0.066 |
| $c_y$ (pixel) | 1024.0 | 1035.332 | 1035.266 | 0.074 |

We also compared the result of camera pre-calibration using the approach of [5, Chapter 3.9.4] with simultaneous camera calibration. Table IV shows the obtained camera parameters for both variants together with the standard deviations that are returned by our approach. Additionally, the initial parameter values that were used for both variants are shown. Our results are consistent with the results obtained by pre-calibration.

To test the robustness of the estimation of the variance components to the initialization of $\sigma_\mathrm{i}$, $\sigma_\mathrm{a}$, and $\sigma_\mathrm{t}$, we modified the standard values of 0.1 pixels, 0.1 deg, and 1 mm. Since only the ratio of the three values affects initialization, we independently multiplied $\sigma_\mathrm{a}$ and $\sigma_\mathrm{t}$ by a factor of $10^P$ where $P$ is an integer ranging from -5 to +5, resulting in $11^2$ combinations. For each combination, we used $\sigma_\mathrm{i}$ and the scaled versions of $\sigma_\mathrm{a}$ and $\sigma_\mathrm{t}$ to initialize $\mathbf{Q}_{ll}$ and restarted our hand–eye calibration. For all experiments for which $-4 \leq P \leq 4$, the variance components successfully converged to the same values in at most 5 iterations. For $|P| = 5$, in some cases the least-squares estimation failed due to numeric problems, which is not surprising since the values in $\mathbf{Q}_{ll}$ differed by up to 11 orders of magnitude. The experiment shows that the estimation of the variance components is extremely robust, even for unrealistically incorrect initializations.

We also applied our approach to four datasets published in [42]. Information about these datasets is given in Table V and the results are shown in Table VI. For calibrated robot poses, GMF returns the smallest errors in almost all cases. When taking the original robot poses (GMF, orig. poses), the error is significantly higher.

## C. Comprehensive Comparison with Previous Work

For a comprehensive comparison of our framework to previous methods, we use the publicly available benchmark dataset of [33]. We selected the CS_Synthetic_3 dataset for evaluation

TABLE V
INFORMATION ABOUT THE DATASETS PUBLISHED IN [42]. CAMERA CALIBRATION ERROR REFERS TO THE REPROJECTION ROOT MEAN SQUARE ERROR OF CAMERA CALIBRATION.

| Dataset | Image Size (pixel × pixel) | Approx. Focal Length (mm) | Robot | $n_r$ | Camera Calibration Error (pixel) |
|---|---|---|---|---|---|
| 1 | 640 × 480 | 8 | Denso VS-6577GM-B | 88 | 0.185 |
| 2 | 2456 × 2058 | 8 | Denso VS-6577GM-B | 28 | 0.199 |
| 3 | 2456 × 2058 | 6 | Denso VM-60BIG | 36 | 0.540 |
| 4 | 1600 × 1200 | 6 | Denso VM-60BIG | 20 | 0.447 |

TABLE VI
RRMSE AND RAE ERRORS ON THE DATASET [42] (RESULTS OF [42] AND [43] ARE TAKEN FROM [43])

| | Dataset 1 | | Dataset 2 | | Dataset 3 | | Dataset 4 | |
|---|---|---|---|---|---|---|---|---|
| Method | RRMSE (pixel) | RAE (mm$^2$) | RRMSE (pixel) | RAE (mm$^2$) | RRMSE (pixel) | RAE (mm$^2$) | RRMSE (pixel) | RAE (mm$^2$) |
| Daniilidis [21] | 1.895 | 51.023 | 3.877 | 2.623 | 12.504 | 725.852 | 2.627 | 38.219 |
| Steger [5] | 1.411 | 5.096 | 3.578 | 0.070 | 3.624 | 9.356 | 1.524 | 2.082 |
| Strobl 1 [14] | 2.148 | 70.728 | 4.836 | 8.345 | 13.759 | 447.813 | 6.378 | 64.998 |
| Strobl 2 [14] | 2.888 | 111.251 | 4.945 | 9.971 | 13.748 | 515.157 | 6.986 | 129.097 |
| Tabb c2 [42] | 1.672 | 5.820 | 3.960 | 0.086 | 3.553 | 1.927 | 2.388 | 1.715 |
| Tabb rp1 [42] | 1.569 | 0.238 | 3.741 | 0.030 | 2.890 | 1.454 | 1.603 | 1.437 |
| Tabb rp2 [42] | 1.363 | 0.524 | 3.181 | 0.450 | 2.680 | 1.588 | 1.505 | 1.481 |
| Koide Pose Graph [43] | 1.562 | 0.214 | 3.751 | 0.038 | 2.879 | 1.337 | 1.619 | 1.380 |
| GMF, orig. poses | 1.567 | 7.715 | 3.835 | 3.519 | 11.744 | 584.722 | 1.661 | 9.090 |
| GMF | 0.076 | 0.113 | 0.176 | 0.031 | 0.479 | 1.188 | 0.389 | 1.380 |

because it is the most comprehensive one and contains the greatest variation in the orientations of the robot and the camera, which is important for accurate hand–eye calibration [16]. It is simulated with Blender [55] and consists of 30 images of a checkerboard, which is captured from different poses, and the corresponding robot poses.

The authors of [33] aim to introduce realistic noise to get as close as possible to real data. They add Gaussian visual noise to the image points and introduce pseudo-realistic noise to the robot poses, which consists of a systematic and a random (Gaussian) component and is different for each axis. The noise magnitude of the components is derived from a real robot by reverse engineering. The size of the systematic components is 0.06 mm, -0.05 mm, and -0.04 mm for the robot position in X, Y, and Z as well as 0.0032 deg, -0.0002 deg, and 0.0002 deg for the robot rotation about these axes. The standard deviation of the random components is 0.22 mm, 0.18 mm, and 0.17 mm for the position in X, Y, and Z as well as 0.0177 deg, 0.0161 deg, and 0.0110 deg for the rotation about these axes. Note that our proposed framework models the robot noise by a single random component. However, it can be easily extended to more complex noise models in a straightforward way.

Fig. 9 visualizes the translation and rotation errors[7] defined in [33] of the hand–eye pose for varying visual noise. For this experiment, the pseudo-realistic robot pose noise was set to the above fixed values. For each level of visual noise, the computations were repeated 1000 times with different random noise. The mean results over all repetitions are shown in Fig. 9. Dotted lines visualize linear approaches: Tsai [16], Horaud [19], Park [28], Li [24], Shah [25], Daniilidis [21]. Dashed lines visualize non-linear approaches: Ali Xc1 and Xc2 [33], Tabb Zc1 and Zc2 [42], Steger [5], Wise (R+C+H) [40]. Solid lines visualize non-linear approaches that minimize the reprojection error: Ali RX and RZ [33], Tabb rp1 [42], GMF (ours) with a fixed camera. The implementations of the approaches are taken from the benchmark in [33] or from the source that is given in the respective publication.

[7]Note that the definition of the rotation error of [33] slightly differs from our definition in Section IV-A.

Our framework clearly outperforms all previous approaches. The evaluation also shows that all approaches that minimize the reprojection error are more robust to visual noise, which agrees with the results of [33]. Of the linear methods, Li [24] is the most robust to high visual noise.

Fig. 10 visualizes the translation and rotation error defined in [33] of the hand–eye pose for varying robot pose noise. Now, the visual noise was set to a fixed value of 0.5 pixels. Rotation and translation components of the robot pose noise were varied simultaneously. For each level of robot pose noise, the computations were repeated 100 times with different random noise. The mean results over all repetitions are shown in Fig. 10. It shows that our framework outperforms all other approaches also in the presence of robot pose noise. In this experiment, the performance gain to the second best method is almost one order of magnitude. All other approaches that minimize the reprojection error are sensitive to robot pose noise. Here, approaches that minimize a pose error might have an advantage. This clearly shows the benefit of explicitly considering the robot uncertainty during hand–eye calibration as our approach minimizes both visual and robot pose noise.

## D. Discussion and Future Work

At this point, we would like to highlight again the advantages of estimating absolute values for the uncertainties. The result matrix $\mathbf{C}_{ll}$ contains the uncertainty of the robot, which includes both absolute accuracy and repeatability. This information is usually not available since most robot manufacturers only specify the repeatability. For vision-guided robots,

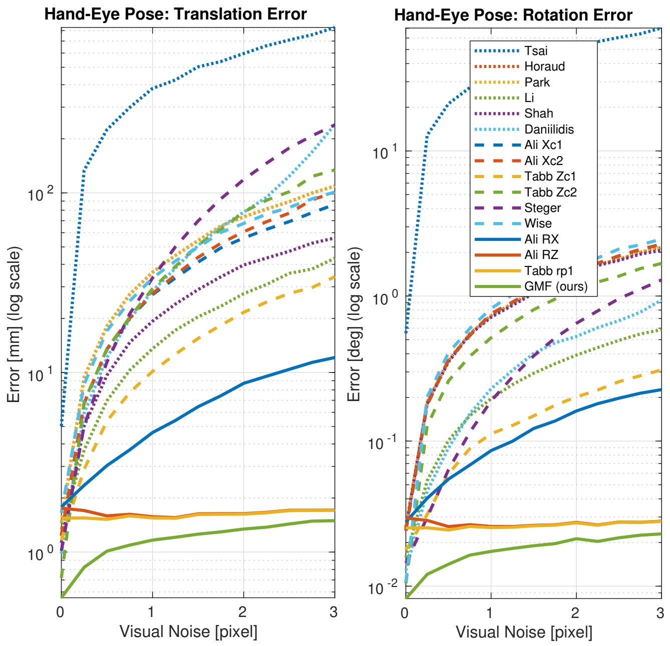


Fig. 9. Translation and rotation error of the hand–eye pose for varying visual noise and constant pseudo-realistic robot pose noise calculated on the benchmark dataset of [33]. Dotted lines visualize linear approaches, dashed lines non-linear approaches, and solid lines non-linear approaches that minimize the reprojection error.

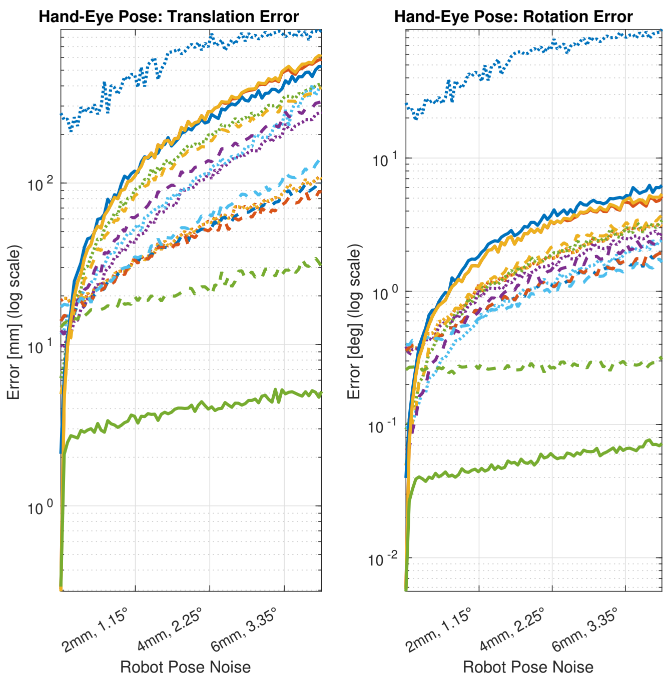


Fig. 10. Translation and rotation error of the hand–eye pose for varying robot pose noise and constant visual noise calculated on the benchmark dataset of [33] (see Fig. 9 for the legend).

however, it is not the repeatability but the total uncertainty of the robot that is crucial. Therefore, our method can be used to easily determine the robot uncertainty, which in turn can be used, for example, to decide whether the robot is suited for a specific application with a certain accuracy requirement. The absolute accuracy of the hand–eye pose, which is returned in $\mathbf{C}_{\hat{x}\hat{x}}$, is of interest in practical applications of a vision-guided robot as well for the same reason. The accuracy information can be the base for the decision whether the accuracy of the estimated hand–eye pose is sufficiently high for a certain application. The absolute accuracy of the hand–eye pose might also be helpful to analyze whether the robot poses cover enough variations for an accurate hand–eye calibration. If, for example, all robot tool poses were more or less parallel to each other, the standard deviation in $t_z$ would be very high. Finally, the obtained uncertainty information about the robot and the hand–eye pose is valuable as input for applying error propagation. For example, the robot can be used for stereo reconstruction by moving it to two specified different poses and acquiring an image at each pose. The uncertainty of the robot pose and that of the hand–eye pose can then be propagated together with the uncertainty of the measured image points to the uncertainty of the reconstructed 3D point.

We model the uncertainty of the robot by assuming that the errors in the robot poses are zero-mean Gaussian distributed. While tests on real systems showed the validity of this assumption [14], the assumption might be violated for some robots. The evaluations presented in this section have shown that the accuracy of the hand–eye calibration benefits from modeling the uncertainty in the proposed way even when the distribution of the errors in the robot poses deviates from a zero-mean Gaussian distribution. Obviously, a wrong assumption about the exact shape of the error distribution is still better than ignoring the errors completely. However, the modelling and the accuracy of the hand–eye calibration could be further improved by directly modelling the uncertainty of the robot parameters of the individual joints rather than the robot pose, which result from the robot parameters by applying the forward kinematics. This is a topic for future research.

Our current implementation assumes that there are no outliers in the observations. For target-based calibration, the robust calibration mark extraction ensures that outliers are suppressed. For self-calibration, outliers are already identified and eliminated during the SfM pipeline, e.g., by using RANSAC [56]. Consequently, our optimization framework does not need to deal with outliers in both cases. However, if the correspondences would be obtained in a different way where outliers cannot be suppressed completely, it would be necessary to extend the optimization by well-established methods for robust estimation. A comprehensive overview about robust estimation and outlier detection is given in [46, Chapter 4.7].

We currently assume that all image points are equally accurate. While this is a common assumption in computer vision, the assumption does not perfectly hold in practice. Therefore, we would like to investigate whether considering a point-specific accuracy improves the accuracy of the hand–eye calibration. This could be easily established in our framework by setting individual entries in $\mathbf{Q}_{ll}$ including covariances of row and column coordinates. For target-based calibration, we could use the size and shape of the elliptic projections of the circular calibration marks in the image to derive an individual variance scaling for each image point, for example. For self-calibration, an appropriate point extractor (e.g., [57]) could be used that returns information about the point's accuracy.

Because there is no bi-invariant distance metric in SE(3), i.e., the Euclidean group of rigid-body motions [58], there is a direction-dependence in our formulation of modeling the robot poses. Therefore, in future research we want to investigate whether this direction-dependence has a significant impact on the results, and, if so, investigate possible solutions.

## V. Conclusion

Explicitly modelling the uncertainty of a robot is beneficial for robot hand–eye calibration. It improves the accuracy, returns calibrated robot poses, and gives information about the uncertainty of the robot. The proposed parameter estimation in the Gauss-Markov model with fictitious observations combined with the estimation of variance components enables a stochastically sound representation of the problem. Different scenarios of hand–eye calibration (e.g., target-based calibration, self-calibration, calibration of antropomorphic or SCARA robots, unknown or known interior orientation) can be easily represented by adding or removing the respective parameters to or from the parameter vectors. By this, a large number of applications can benefit from our approach. Implementation details including several optimizations facilitate an easy, fast, and memory efficient realization of the framework making it applicable also in scenarios with a large number of calibration images and points. Finally, the experiments clearly demonstrate the advantages of our uncertainty-aware hand–eye calibration.

## Appendix
## Pseudocode

Fig. 11 shows pseudocode of our generic uncertainty-aware hand–eye calibration, which uses parameter estimation in the Gauss-Markov model with fictitious observations (GMF).

## Appendix
## Calculation of the Partial Derivatives

### A. Definitions

To write down the partial derivatives and to keep the notation as simple as possible, we use the following definitions:

- $\boldsymbol{p}_\mathrm{w} = (x_\mathrm{w}, y_\mathrm{w}, z_\mathrm{w})^\top$, $\boldsymbol{p}_\mathrm{b} = (x_\mathrm{b}, y_\mathrm{b}, z_\mathrm{b})^\top$, $\boldsymbol{p}_\mathrm{t} = (x_\mathrm{t}, y_\mathrm{t}, z_\mathrm{t})^\top$, and $\boldsymbol{p}_\mathrm{c} = (x_\mathrm{c}, y_\mathrm{c}, z_\mathrm{c})^\top$ is a 3D point in the WCS, BCS, TCS, and CCS, respectively.
- $\boldsymbol{p}_\mathrm{i} = (x_\mathrm{i}, y_\mathrm{i})^\top$ is the corresponding projected 2D image point.
- The 3D rigid transformation of a point in the WCS to the BCS is given by $\boldsymbol{p}_\mathrm{b} = {}^\mathrm{b}\mathbf{R}_\mathrm{w}\boldsymbol{p}_\mathrm{w} + {}^\mathrm{b}\boldsymbol{t}_\mathrm{w}$ with the translation vector ${}^\mathrm{b}\boldsymbol{t}_\mathrm{w} = ({}^\mathrm{b}tx_\mathrm{w}, {}^\mathrm{b}ty_\mathrm{w}, {}^\mathrm{b}tz_\mathrm{w})^\top$ and the rotation matrix ${}^\mathrm{b}\mathbf{R}_\mathrm{w} = \mathbf{R}_x({}^\mathrm{b}\alpha_\mathrm{w})\mathbf{R}_y({}^\mathrm{b}\beta_\mathrm{w})\mathbf{R}_z({}^\mathrm{b}\gamma_\mathrm{w})$. The rotation matrices are parameterized by Euler angles:

$$\mathbf{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} , \tag{25}$$

$$\mathbf{R}_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} , \tag{26}$$

$$\mathbf{R}_z(\gamma) = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} . \tag{27}$$

The 3D rigid transformations $\boldsymbol{p}_\mathrm{t} = {}^\mathrm{t}\mathbf{R}_\mathrm{b}\boldsymbol{p}_\mathrm{b} + {}^\mathrm{t}\boldsymbol{t}_\mathrm{b}$ and $\boldsymbol{p}_\mathrm{c} = {}^\mathrm{c}\mathbf{R}_\mathrm{t}\boldsymbol{p}_\mathrm{t} + {}^\mathrm{c}\boldsymbol{t}_\mathrm{t}$ are defined accordingly.

### B. Observation Equations

To build the Jacobian $\mathbf{A}$, we essentially need the partial derivatives of the observation equations (10) with respect to the unknowns. With the previously introduced notation, equation (10) can be split up into the following functions that are executed consecutively:

1) Transformation from WCS to BCS:
$$\boldsymbol{p}_\mathrm{b} = {}^\mathrm{b}\mathbf{R}_\mathrm{w}\boldsymbol{p}_\mathrm{w} + {}^\mathrm{b}\boldsymbol{t}_\mathrm{w} \tag{28}$$
2) Transformation from BCS to TCS:
$$\boldsymbol{p}_\mathrm{t} = {}^\mathrm{t}\mathbf{R}_\mathrm{b}\boldsymbol{p}_\mathrm{b} + {}^\mathrm{t}\boldsymbol{t}_\mathrm{b} \tag{29}$$
3) Transformation from TCS to CCS:
$$\boldsymbol{p}_\mathrm{c} = {}^\mathrm{c}\mathbf{R}_\mathrm{t}\boldsymbol{p}_\mathrm{t} + {}^\mathrm{c}\boldsymbol{t}_\mathrm{t} \tag{30}$$
4) Projection into the image plane:
$$\begin{pmatrix} x_\mathrm{u} \\ y_\mathrm{u} \end{pmatrix} = \frac{c}{z_\mathrm{c}} \begin{pmatrix} x_\mathrm{c} \\ y_\mathrm{c} \end{pmatrix} \tag{31}$$
5) Applying lens distortions (inverse of (7)):
$$\begin{pmatrix} x_\mathrm{d} \\ y_\mathrm{d} \end{pmatrix} = \frac{2}{1 + \sqrt{1 - 4\kappa(x_\mathrm{u}^2 + y_\mathrm{u}^2)}} \begin{pmatrix} x_\mathrm{u} \\ y_\mathrm{u} \end{pmatrix} \tag{32}$$
6) Transformation into the image coordinate system:
$$\begin{pmatrix} x_\mathrm{i} \\ y_\mathrm{i} \end{pmatrix} = \begin{pmatrix} x_\mathrm{d}/s_x + c_x \\ y_\mathrm{d}/s_y + c_y \end{pmatrix} \tag{33}$$

Note that in step 5, we focus on the division model since the polynomial model cannot be inverted analytically. If the polynomial model should be used, its partial derivatives can be computed numerically. Alternatively, the minimization of the reprojection error can be performed in the (distorted) image plane instead of in the image coordinate system (for details, see [5, Chapter 3.9.4.2]).

In the following representations of the derivatives, zeros and ones are explicitly considered for efficiency reasons.

### C. Derivatives of Step 1: Transformation from WCS to BCS

The partial derivatives of $x_\mathrm{b}$, $y_\mathrm{b}$, and $z_\mathrm{b}$ with respect to ${}^\mathrm{b}\alpha_\mathrm{w}$, ${}^\mathrm{b}\beta_\mathrm{w}$, ${}^\mathrm{b}\gamma_\mathrm{w}$, ${}^\mathrm{b}tx_\mathrm{w}$, ${}^\mathrm{b}ty_\mathrm{w}$, ${}^\mathrm{b}tz_\mathrm{w}$, $x_\mathrm{w}$, $y_\mathrm{w}$, and $z_\mathrm{w}$ are given in Table VII. Note that rows 1–3 of Table VII are the transposed of $[-{}^\mathrm{b}\mathbf{R}_\mathrm{w}\boldsymbol{p}_\mathrm{w}]_\times\mathbf{M}$ with

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & \sin{}^\mathrm{b}\beta_\mathrm{w} \\ 0 & \cos{}^\mathrm{b}\alpha_\mathrm{w} & -\sin{}^\mathrm{b}\alpha_\mathrm{w}\cos{}^\mathrm{b}\beta_\mathrm{w} \\ 0 & \sin{}^\mathrm{b}\alpha_\mathrm{w} & \cos{}^\mathrm{b}\alpha_\mathrm{w}\cos{}^\mathrm{b}\beta_\mathrm{w} \end{pmatrix} \tag{34}$$

and $[\boldsymbol{x}]_\times$ denoting the skew-symmetric 3×3 matrix of $\boldsymbol{x}$ for that $[\boldsymbol{x}]_\times\boldsymbol{y} = \boldsymbol{x}\times\boldsymbol{y}$ holds (see [59] for details). Rows 4–6 obviously are the identity matrix and rows 7–9 are the transposed of ${}^\mathrm{b}\mathbf{R}_\mathrm{w}$.

Generic GMF Algorithm for Hand–Eye Calibration

```
1: procedure GMF(Bool DoTargetBasedCalibration, Bool DoPrecalibrateCamera)
2:     if DoPrecalibrateCamera then
3:         Perform camera calibration $\rightarrow$ interior orientation $\boldsymbol{i}$
4:     end if
5:     if DoTargetBasedCalibration then
6:         Place calibration target with $n_\mathrm{w}$ 3D points $\boldsymbol{p}_k$ $(k = 1, \ldots, n_\mathrm{w})$ in robot workspace
7:         for $j = 1$ to $n_\mathrm{r}$ do
8:             Move robot tool to predefined pose ${}^\mathrm{t}\mathtt{H}_{\mathrm{b},j}^{(0)}$
9:             Acquire camera image
10:            Extract coordinates of calibration points $\boldsymbol{p}_{j,k}$ in the acquired image
11:            Compute ${}^\mathrm{c}\mathtt{H}_{\mathrm{w},j}$ from correspondences $\boldsymbol{p}_k$, $\boldsymbol{p}_{j,k}$, and $\boldsymbol{i}$ with PnP algorithm
12:        end for
13:    else
14:        Provide textured scene elements in robot workspace
15:        for $j = 1$ to $n_\mathrm{r}$ do
16:            Move robot tool to pose ${}^\mathrm{t}\mathtt{H}_{\mathrm{b},j}^{(0)}$
17:            Acquire and save camera image
18:        end for
19:        Apply SfM pipeline (e.g., COLMAP) to camera images $\rightarrow \boldsymbol{i}^{(0)}$, $n_\mathrm{w}$, $\boldsymbol{p}_k^{(0)}$, $\boldsymbol{p}_{j,k}$, ${}^\mathrm{c}\mathtt{H}_{\mathrm{w},j}$
20:        Fix the SfM scale ambiguity in $\boldsymbol{p}_k^{(0)}$ and ${}^\mathrm{c}\mathtt{H}_{\mathrm{w},j}$ as described in Section III-G
21:    end if
22:    Perform linear hand–eye calibration of [21] based on ${}^\mathrm{t}\mathtt{H}_{\mathrm{b},j}^{(0)}$ and ${}^\mathrm{c}\mathtt{H}_{\mathrm{w},j}$ $\rightarrow$ initial values for ${}^\mathrm{c}\mathtt{H}_\mathrm{t}^{(0)}$, ${}^\mathrm{b}\mathtt{H}_\mathrm{w}^{(0)}$
23:    Build parameter vectors ${}^\mathrm{c}\mathtt{H}_\mathrm{t}^{(0)} \rightarrow \boldsymbol{e}_\mathrm{c}^{(0)}$, ${}^\mathrm{b}\mathtt{H}_\mathrm{w}^{(0)} \rightarrow \boldsymbol{e}_\mathrm{b}^{(0)}$, ${}^\mathrm{t}\mathtt{H}_{\mathrm{b},j}^{(0)} \rightarrow \boldsymbol{e}_{\mathrm{t},j}^{(0)}$    ▷ for self-calibration, ${}^\mathrm{b}\mathtt{H}_\mathrm{w}^{(0)} = \mathtt{I}$
24:    Set vector of observations $\boldsymbol{l} = (\boldsymbol{p}_{1,1}^\top, \ldots, \boldsymbol{p}_{1,n_\mathrm{w}}^\top, \ldots, \boldsymbol{p}_{n_\mathrm{r},n_\mathrm{w}}^\top, \boldsymbol{e}_{\mathrm{t},1}^{(0)\top}, \ldots, \boldsymbol{e}_{\mathrm{t},n_\mathrm{r}}^{(0)\top})^\top$
25:    Build vector of unknowns $\boldsymbol{x}^{(0)} \leftarrow (\boldsymbol{e}_\mathrm{c}^{(0)\top}, \boldsymbol{e}_{\mathrm{t},1}^{(0)\top}, \ldots, \boldsymbol{e}_{\mathrm{t},n_\mathrm{r}}^{(0)\top})^\top$
26:    if not(DoPrecalibrateCamera) then
27:        Extend vector of unknowns $\boldsymbol{x}^{(0)} \leftarrow (\boldsymbol{x}^{(0)}, \boldsymbol{i}^{(0)\top})^\top$
28:    end if
29:    if DoTargetBasedCalibration then
30:        Extend vector of unknowns $\boldsymbol{x}^{(0)} \leftarrow (\boldsymbol{x}^{(0)}, \boldsymbol{e}_\mathrm{b}^{(0)\top})^\top$
31:    else
32:        Extend vector of unknowns $\boldsymbol{x}^{(0)} \leftarrow (\boldsymbol{x}^{(0)}, \boldsymbol{p}_1^{(0)\top}, \ldots, \boldsymbol{p}_{n_\mathrm{w}}^{(0)\top})^\top$
33:    end if
34:    Initialize variances of the three observation groups $\sigma_\mathrm{i}^2 \leftarrow 0.1^2$ pixel, $\sigma_\mathrm{a}^2 \leftarrow 0.1^2$ deg, $\sigma_\mathrm{t}^2 \leftarrow 1.0^2$ mm
35:    repeat
36:        Build $\mathtt{Q}_{ll}$ by using Eq. (16) and compute $\mathtt{P}_{ll} \leftarrow \mathtt{Q}_{ll}^{-1}$
37:        Initialize vector of unknowns $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}^{(0)}$
38:        repeat
39:            $\Delta\boldsymbol{l} \leftarrow \boldsymbol{l} - \boldsymbol{f}(\hat{\boldsymbol{x}})$    ▷ $\boldsymbol{f}(\hat{\boldsymbol{x}})$ is Eq. (10) extended by the current values of the unknown robot poses
40:            Compute Jacobian $\mathtt{A}$ shown in Fig. 3 by using Eq. (11) and the partial derivatives given in the Appendix
41:            Solve Eq. (12) for $\Delta\hat{\boldsymbol{x}}$
42:            $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} + \Delta\hat{\boldsymbol{x}}$
43:        until convergence of the parameter estimation $(\Delta\hat{\boldsymbol{x}} == 0)$
44:        Compute $\mathtt{Q}_{\hat{x}\hat{x}}$, $\mathtt{Q}_{\hat{l}\hat{l}}$, $\mathtt{Q}_{\hat{v}\hat{v}}$, and $\mathtt{R}$ with Eqs. (19), (20), (21), and (22), respectively
45:        Compute the variance components $\hat{\sigma}_{0,\mathrm{i}}^2$, $\hat{\sigma}_{0,\mathrm{a}}^2$, and $\hat{\sigma}_{0,\mathrm{t}}^2$ with Eqs. (23) and update $\sigma_\mathrm{i}^2, \sigma_\mathrm{a}^2, \sigma_\mathrm{t}^2$ with Eqs. (24)
46:    until convergence of the variance component estimation $(\hat{\sigma}_{0,\mathrm{i}}^2 == \hat{\sigma}_{0,\mathrm{a}}^2 == \hat{\sigma}_{0,\mathrm{t}}^2 == 1 \Rightarrow \hat{\sigma}_0^2 = 1)$
47:    Compute the covariances $\mathtt{C}_{ll}$ of the original observations with Eq. (13)    ▷ provides information about robot uncertainty
48:    Compute the covariances $\mathtt{C}_{\hat{x}\hat{x}}$ of the estimated unknowns with Eq. (15)
49:    return $\hat{\boldsymbol{x}}$, $\mathtt{C}_{\hat{x}\hat{x}}$, $\mathtt{C}_{ll}$
50: end procedure
```

Fig. 11. Pseudocode of our generic uncertainty-aware hand–eye calibration (GMF) for the cases target-based hand–eye calibration (DoTargetBasedCalibration == `True`), self-calibration (DoTargetBasedCalibration == `False`), using a pre-calibrated camera (DoPrecalibrateCamera == `True`), and simultaneous camera calibration (DoPrecalibrateCamera == `False`).

### *D. Derivatives of Step 2: Transformation from BCS to TCS*

The partial derivatives of $x_\mathrm{t}$, $y_\mathrm{t}$, and $z_\mathrm{t}$ with respect to ${}^\mathrm{t}\alpha_\mathrm{b}$, ${}^\mathrm{t}\beta_\mathrm{b}$, ${}^\mathrm{t}\gamma_\mathrm{b}$, ${}^\mathrm{t}tx_\mathrm{b}$, ${}^\mathrm{t}ty_\mathrm{b}$, ${}^\mathrm{t}tz_\mathrm{b}$, $x_\mathrm{b}$, $y_\mathrm{b}$, and $z_\mathrm{b}$ are analogous to those of step 1. Then, we apply the chain rule to obtain the partial derivatives of $x_\mathrm{t}$, $y_\mathrm{t}$, and $z_\mathrm{t}$ with respect to the parameters of the previous steps (see Table VIII). Note that rows 1–3 of Table VIII are the transposed of $-{}^\mathrm{t}\mathbf{R}_\mathrm{b}[{}^\mathrm{b}\mathbf{R}_\mathrm{w}\boldsymbol{p}_\mathrm{w}]_\times\mathtt{M}$, rows 3–5 are the transposed of ${}^\mathrm{t}\mathbf{R}_\mathrm{b}$, and rows 7–9 are the transposed of ${}^\mathrm{t}\mathbf{R}_\mathrm{b}{}^\mathrm{b}\mathbf{R}_\mathrm{w}$.

### *E. Derivatives of Step 3: Transformation from TCS to CCS*

The partial derivatives of $x_\mathrm{c}$, $y_\mathrm{c}$, and $z_\mathrm{c}$ with respect to ${}^\mathrm{c}\alpha_\mathrm{t}$, ${}^\mathrm{c}\beta_\mathrm{t}$, ${}^\mathrm{c}\gamma_\mathrm{t}$, ${}^\mathrm{c}tx_\mathrm{t}$, ${}^\mathrm{c}ty_\mathrm{t}$, ${}^\mathrm{c}tz_\mathrm{t}$, $x_\mathrm{t}$, $y_\mathrm{t}$, and $z_\mathrm{t}$ are analogous to those of step 1. Then, we apply the chain rule to obtain the partial

TABLE VII
DERIVATIVES OF STEP 1: TRANSFORMATION FROM WCS TO BCS

| | $\partial x_{\mathrm{b}}/\ldots$ | $\partial y_{\mathrm{b}}/\ldots$ | $\partial z_{\mathrm{b}}/\ldots$ |
|---|---|---|---|
| $\ldots/\partial {}^{\mathrm{b}}\alpha_{\mathrm{w}}$ | 0 | $-x_{\mathrm{w}}(\sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}} - \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}})$ <br> $-y_{\mathrm{w}}(\cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} + \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}})$ <br> $-z_{\mathrm{b}} \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}}$ | $x_{\mathrm{w}}(\cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}} + \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}})$ <br> $+y_{\mathrm{w}}(\cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} - \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}})$ <br> $-z_{\mathrm{b}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}}$ |
| $\ldots/\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}$ | $-x_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}}$ <br> $+y_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ <br> $+z_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}}$ | $x_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}}$ <br> $-y_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ <br> $+z_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}}$ | $-x_{\mathrm{w}} \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ <br> $+y_{\mathrm{w}} \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ <br> $-z_{\mathrm{w}} \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}}$ |
| $\ldots/\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ | $-x_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ <br> $-y_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ | $x_{\mathrm{w}}(\cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} - \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}})$ <br> $-y_{\mathrm{w}}(\cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}} + \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}})$ | $x_{\mathrm{w}}(\cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} + \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}})$ <br> $-y_{\mathrm{w}}(\sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}} - \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}})$ |
| $\ldots/\partial {}^{\mathrm{b}}tx_{\mathrm{w}}$ | 1 | 0 | 0 |
| $\ldots/\partial {}^{\mathrm{b}}ty_{\mathrm{w}}$ | 0 | 1 | 0 |
| $\ldots/\partial {}^{\mathrm{b}}tz_{\mathrm{w}}$ | 0 | 0 | 1 |
| $\ldots/\partial x_{\mathrm{w}}$ | $\cos {}^{\mathrm{b}}\beta_{\mathrm{w}}\cos {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ | $\cos {}^{\mathrm{b}}\alpha_{\mathrm{w}}\sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}+ \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}}\sin {}^{\mathrm{b}}\alpha_{\mathrm{w}}\sin {}^{\mathrm{b}}\beta_{\mathrm{w}}$ | $\sin {}^{\mathrm{b}}\alpha_{\mathrm{w}}\sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}- \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}}\cos {}^{\mathrm{b}}\gamma_{\mathrm{w}}\sin {}^{\mathrm{b}}\beta_{\mathrm{w}}$ |
| $\ldots/\partial y_{\mathrm{w}}$ | $-\cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ | $\cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} - \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ | $\cos {}^{\mathrm{b}}\gamma_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}} + \cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \sin {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ |
| $\ldots/\partial z_{\mathrm{w}}$ | $\sin {}^{\mathrm{b}}\beta_{\mathrm{w}}$ | $-\cos {}^{\mathrm{b}}\beta_{\mathrm{w}} \sin {}^{\mathrm{b}}\alpha_{\mathrm{w}}$ | $\cos {}^{\mathrm{b}}\alpha_{\mathrm{w}} \cos {}^{\mathrm{b}}\beta_{\mathrm{w}}$ |

TABLE VIII
DERIVATIVES OF STEP 2: TRANSFORMATION FROM BCS TO TCS

| | $\partial x_{\mathrm{t}}/\ldots$ | $\partial y_{\mathrm{t}}/\ldots$ | $\partial z_{\mathrm{t}}/\ldots$ |
|---|---|---|---|
| $\ldots/\partial {}^{\mathrm{b}}\alpha_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\alpha_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\alpha_{\mathrm{w}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\alpha_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\alpha_{\mathrm{w}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\alpha_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\alpha_{\mathrm{w}}}$ |
| $\ldots/\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\beta_{\mathrm{w}}}$ |
| $\ldots/\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial {}^{\mathrm{b}}\gamma_{\mathrm{w}}}$ |
| $\ldots/\partial {}^{\mathrm{b}}tx_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial x_{\mathrm{b}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial x_{\mathrm{b}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial x_{\mathrm{b}}}$ |
| $\ldots/\partial {}^{\mathrm{b}}ty_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial y_{\mathrm{b}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial y_{\mathrm{b}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial y_{\mathrm{b}}}$ |
| $\ldots/\partial {}^{\mathrm{b}}tz_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial z_{\mathrm{b}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial z_{\mathrm{b}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial z_{\mathrm{b}}}$ |
| $\ldots/\partial x_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial x_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial x_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial x_{\mathrm{w}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial x_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial x_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial x_{\mathrm{w}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial x_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial x_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial x_{\mathrm{w}}}$ |
| $\ldots/\partial y_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial y_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial y_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial y_{\mathrm{w}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial y_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial y_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial y_{\mathrm{w}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial y_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial y_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial y_{\mathrm{w}}}$ |
| $\ldots/\partial z_{\mathrm{w}}$ | $\frac{\partial x_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial z_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial z_{\mathrm{w}}} + \frac{\partial x_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial z_{\mathrm{w}}}$ | $\frac{\partial y_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial z_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial z_{\mathrm{w}}} + \frac{\partial y_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial z_{\mathrm{w}}}$ | $\frac{\partial z_{\mathrm{t}}}{\partial x_{\mathrm{b}}}\frac{\partial x_{\mathrm{b}}}{\partial z_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial y_{\mathrm{b}}}\frac{\partial y_{\mathrm{b}}}{\partial z_{\mathrm{w}}} + \frac{\partial z_{\mathrm{t}}}{\partial z_{\mathrm{b}}}\frac{\partial z_{\mathrm{b}}}{\partial z_{\mathrm{w}}}$ |

derivatives of $x_{\mathrm{c}}$, $y_{\mathrm{c}}$, and $z_{\mathrm{c}}$ with respect to the parameters of the previous steps (see Table IX).

## *F. Derivatives of Step 4: Projection into the Image Plane*

The partial derivatives of $x_{\mathrm{u}}$ and $y_{\mathrm{u}}$ with respect to $c$, $x_{\mathrm{c}}$, $y_{\mathrm{c}}$ and $z_{\mathrm{c}}$ are shown in Table X.

Then, we apply the chain rule to obtain the partial derivatives of $x_{\mathrm{u}}$ and $y_{\mathrm{u}}$ with respect to the parameters of the previous steps (see Table XI).

## *G. Derivatives of Step 5: Applying Lens Distortions*

The partial derivatives of $x_{\mathrm{d}}$ and $y_{\mathrm{d}}$ with respect to $\kappa$, $x_{\mathrm{u}}$ and $y_{\mathrm{u}}$ are shown in Table XII with $r_{\mathrm{u}} = \sqrt{x_{\mathrm{u}}^2 + y_{\mathrm{u}}^2}$ and $D = \sqrt{1 - 4\kappa r_{\mathrm{u}}^2}(1 + \sqrt{1 - 4\kappa r_{\mathrm{u}}^2})^2$.

Then, we apply the chain rule to obtain the partial derivatives of $x_{\mathrm{d}}$ and $y_{\mathrm{d}}$ with respect to the parameters of the previous steps (see Table XIII).

## *H. Derivatives of Step 6: Transformation into the Image Coordinate System*

The partial derivatives of $x_{\mathrm{i}}$ and $y_{\mathrm{i}}$ with respect to $s_x$, $s_y$, $c_x$, $c_y$, $x_{\mathrm{d}}$ and $y_{\mathrm{d}}$ are shown in Table XIV.

Then, we apply the chain rule to obtain the partial derivatives of $x_{\mathrm{i}}$ and $y_{\mathrm{i}}$ with respect to the parameters of the previous steps (see Table XV). Thus, all partial derivatives that are required to build the Jacobian $\mathbf{A}$ are known.

## ACKNOWLEDGMENT

We thank the MVTec Software GmbH, where the first ideas on this research topic were developed and discussed. Furthermore, we thank our colleague Bastian Erdnüß for his careful review of the Appendix and his valuable suggestions for improvement.

## REFERENCES

[1] A. Hofhauser, C. Steger, and N. Navab, "Perspective planar shape matching," in *Image Processing: Machine Vision Applications II*, ser. Proc. SPIE 7251, K. S. Niel and D. Fofi, Eds., 2009.

[2] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Computer Vision – ACCV 2012*, K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562.

[3] M. Ulrich, C. Wiedemann, and C. Steger, "Combining scale-space and similarity-based aspect graphs for fast 3D object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1902–1914, Oct. 2012.

[4] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic, "CosyPose: Consistent multi-view multi-object 6D pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[5] C. Steger, M. Ulrich, and C. Wiedemann, *Machine Vision Algorithms and Applications*, 2nd ed. Weinheim: Wiley-VCH, 2018.

TABLE IX
DERIVATIVES OF STEP 3: TRANSFORMATION FROM TCS TO CCS

| | $\partial x_\mathrm{c}/\dots$ | $\partial y_\mathrm{c}/\dots$ | $\partial z_\mathrm{c}/\dots$ |
|---|---|---|---|
| $\dots/\partial^\mathrm{b}\alpha_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}\beta_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}\gamma_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}tx_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}tx_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}ty_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}ty_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}tz_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{b}tz_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{t}\alpha_\mathrm{b}$ | $\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\alpha_\mathrm{b}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ | $\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\alpha_\mathrm{b}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ | $\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\alpha_\mathrm{b}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{t}\beta_\mathrm{b}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{t}\gamma_\mathrm{b}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{t}tx_\mathrm{b}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}$ |
| $\dots/\partial^\mathrm{t}ty_\mathrm{b}$ | $\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}$ | $\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}$ | $\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}$ |
| $\dots/\partial^\mathrm{t}tz_\mathrm{b}$ | $\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}$ | $\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}$ | $\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}$ |
| $\dots/\partial x_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial x_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial x_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial x_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial x_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial x_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial x_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial x_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial x_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial x_\mathrm{w}}$ |
| $\dots/\partial y_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial y_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial y_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial y_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial y_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial y_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial y_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial y_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial y_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial y_\mathrm{w}}$ |
| $\dots/\partial z_\mathrm{w}$ | $\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial z_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial z_\mathrm{w}}+\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial z_\mathrm{w}}$ | $\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial z_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial z_\mathrm{w}}+\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial z_\mathrm{w}}$ | $\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{t}}\frac{\partial x_\mathrm{t}}{\partial z_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{t}}\frac{\partial y_\mathrm{t}}{\partial z_\mathrm{w}}+\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{t}}\frac{\partial z_\mathrm{t}}{\partial z_\mathrm{w}}$ |

TABLE X
DERIVATIVES OF STEP 4: PROJECTION INTO THE IMAGE PLANE (I)

| | $\partial x_\mathrm{u}/\dots$ | $\partial y_\mathrm{u}/\dots$ |
|---|---|---|
| $\dots/\partial c$ | $\frac{x_\mathrm{c}}{z_\mathrm{c}}$ | $\frac{y_\mathrm{c}}{z_\mathrm{c}}$ |
| $\dots/\partial x_\mathrm{c}$ | $\frac{c}{z_\mathrm{c}}$ | 0 |
| $\dots/\partial y_\mathrm{c}$ | 0 | $\frac{c}{z_\mathrm{c}}$ |
| $\dots/\partial z_\mathrm{c}$ | $\frac{-cx_\mathrm{c}}{z_\mathrm{c}^2}$ | $\frac{-cy_\mathrm{c}}{z_\mathrm{c}^2}$ |

[6] ISO 9283:1998, "Manipulating industrial robots — performance criteria and related test methods," 1998.

[7] P. Shiakolas, K. Conrad, and T. Yih, "On the accuracy, repeatability, and degree of influence of kinematics parameters for industrial robots," *International Journal of Modelling and Simulation*, vol. 22, no. 4, pp. 245–254, 2002.

[8] M. Abderrahim, A. Khamis, S. Garrido, and L. Moreno, "Accuracy and calibration issues of industrial manipulators," in *Industrial Robotics: Programming, Simulation and Application*. IntechOpen, 2006, pp. 131–146.

[9] M. Morozov, J. Riise, R. Summan, S. G. Pierce, C. Mineo, C. N. MacLeod, and R. H. Brown, "Assessing the accuracy of industrial robots through metrology for the enhancement of automated non-destructive testing," in *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2016, pp. 335–340.

[10] M. Płaczek and Ł. Piszczek, "Testing of an industrial robot's accuracy and repeatability in off and online environments," *Eksploatacja i Niezawodnosc — Maintenance and Reliability*, vol. 20, no. 3, pp. 455–464, 2018.

[11] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 22, no. 2, pp. 215–221, Jun. 1955.

[12] M. Ulrich and M. Hillemann, "Generic hand–eye calibration of uncertain robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 060–11 066.

[13] MVTec Software GmbH, "HALCON/HDevelop Reference Manual," HALCON Version 23.05, May 2023.

[14] K. H. Strobl and G. Hirzinger, "Optimal hand-eye calibration," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 4647–4653.

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

[16] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous

TABLE XI
DERIVATIVES OF STEP 4: PROJECTION INTO THE IMAGE PLANE (II)

| | $\partial x_\mathrm{u}/\dots$ | $\partial y_\mathrm{u}/\dots$ |
|---|---|---|
| $\dots/\partial^\mathrm{b}\alpha_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}\beta_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}\gamma_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}tx_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}tx_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}tx_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}ty_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}ty_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}ty_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{b}tz_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}tz_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{b}tz_\mathrm{w}}$ |
| $\dots/\partial^\mathrm{t}\alpha_\mathrm{b}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{t}\alpha_\mathrm{b}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{t}\alpha_\mathrm{b}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{t}\beta_\mathrm{b}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{t}\gamma_\mathrm{b}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{t}tx_\mathrm{b}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{t}tx_\mathrm{b}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}tx_\mathrm{b}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{t}tx_\mathrm{b}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}tx_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{t}ty_\mathrm{b}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{t}ty_\mathrm{b}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}ty_\mathrm{b}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{t}ty_\mathrm{b}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}ty_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{t}tz_\mathrm{b}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{t}tz_\mathrm{b}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}tz_\mathrm{b}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{t}tz_\mathrm{b}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{t}tz_\mathrm{b}}$ |
| $\dots/\partial^\mathrm{c}\alpha_\mathrm{t}$ | $\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{c}\alpha_\mathrm{t}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{c}\alpha_\mathrm{t}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{c}\alpha_\mathrm{t}}$ |
| $\dots/\partial^\mathrm{c}\beta_\mathrm{t}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{c}\beta_\mathrm{t}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{c}\beta_\mathrm{t}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{c}\beta_\mathrm{t}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{c}\beta_\mathrm{t}}$ |
| $\dots/\partial^\mathrm{c}\gamma_\mathrm{t}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial^\mathrm{c}\gamma_\mathrm{t}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{c}\gamma_\mathrm{t}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial^\mathrm{c}\gamma_\mathrm{t}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial^\mathrm{c}\gamma_\mathrm{t}}$ |
| $\dots/\partial^\mathrm{c}tx_\mathrm{t}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}$ | 0 |
| $\dots/\partial^\mathrm{c}ty_\mathrm{t}$ | 0 | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}$ |
| $\dots/\partial^\mathrm{c}tz_\mathrm{t}$ | $\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}$ | $\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}$ |
| $\dots/\partial x_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial x_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial x_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial x_\mathrm{w}}$ |
| $\dots/\partial y_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial y_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial y_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial y_\mathrm{w}}$ |
| $\dots/\partial z_\mathrm{w}$ | $\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{c}}\frac{\partial x_\mathrm{c}}{\partial z_\mathrm{w}}+\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{w}}$ | $\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{c}}\frac{\partial y_\mathrm{c}}{\partial z_\mathrm{w}}+\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{c}}\frac{\partial z_\mathrm{c}}{\partial z_\mathrm{w}}$ |

TABLE XII
DERIVATIVES OF STEP 5: APPLYING LENS DISTORTIONS (I)

| | $\partial x_\mathrm{d}/\dots$ | $\partial y_\mathrm{d}/\dots$ |
|---|---|---|
| $\dots/\partial\kappa$ | $\frac{4x_\mathrm{u}r_\mathrm{u}^2}{D}$ | $\frac{4y_\mathrm{u}r_\mathrm{u}^2}{D}$ |
| $\dots/\partial x_\mathrm{u}$ | $\frac{2}{1+\sqrt{1-4\kappa r_\mathrm{u}^2}}+\frac{8\kappa x_\mathrm{u}^2}{D}$ | $\frac{8\kappa x_\mathrm{u}y_\mathrm{u}}{D}$ |
| $\dots/\partial y_\mathrm{u}$ | $\frac{8\kappa x_\mathrm{u}y_\mathrm{u}}{D}$ | $\frac{2}{1+\sqrt{1-4\kappa r_\mathrm{u}^2}}+\frac{8\kappa y_\mathrm{u}^2}{D}$ |

TABLE XIII
DERIVATIVES OF STEP 5: APPLYING LENS DISTORTIONS (II)

| | $\partial x_\mathrm{d}/\ldots$ | $\partial y_\mathrm{d}/\ldots$ |
|---|---|---|
| $\ldots/\partial^\mathrm{b}\alpha_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}\alpha_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}\beta_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}\beta_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}\gamma_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}\gamma_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}tx_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}tx_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}tx_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}tx_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}ty_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}ty_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}ty_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}ty_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}tz_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}tz_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{b}tz_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{b}tz_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{t}\alpha_\mathrm{b}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}\alpha_\mathrm{b}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}\alpha_\mathrm{b}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}\beta_\mathrm{b}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}\beta_\mathrm{b}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}\gamma_\mathrm{b}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}\gamma_\mathrm{b}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}tx_\mathrm{b}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}tx_\mathrm{b}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}tx_\mathrm{b}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}tx_\mathrm{b}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}tx_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}ty_\mathrm{b}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}ty_\mathrm{b}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}ty_\mathrm{b}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}ty_\mathrm{b}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}ty_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}tz_\mathrm{b}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}tz_\mathrm{b}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}tz_\mathrm{b}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{t}tz_\mathrm{b}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{t}tz_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{c}\alpha_\mathrm{t}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}\alpha_\mathrm{t}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}\alpha_\mathrm{t}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}\alpha_\mathrm{t}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}\alpha_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}\beta_\mathrm{t}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}\beta_\mathrm{t}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}\beta_\mathrm{t}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}\beta_\mathrm{t}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}\beta_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}\gamma_\mathrm{t}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}\gamma_\mathrm{t}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}\gamma_\mathrm{t}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}\gamma_\mathrm{t}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}\gamma_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}tx_\mathrm{t}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}tx_\mathrm{t}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}tx_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}ty_\mathrm{t}$ | $\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}ty_\mathrm{t}}$ | $\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}ty_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}tz_\mathrm{t}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}tz_\mathrm{t}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}tz_\mathrm{t}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial^\mathrm{c}tz_\mathrm{t}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial^\mathrm{c}tz_\mathrm{t}}$ |
| $\ldots/\partial c$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial c}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial c}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial c}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial c}$ |
| $\ldots/\partial x_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial x_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial x_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial x_\mathrm{w}}$ |
| $\ldots/\partial y_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial y_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial y_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial y_\mathrm{w}}$ |
| $\ldots/\partial z_\mathrm{w}$ | $\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{w}}+\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{w}}$ | $\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{u}}\frac{\partial x_\mathrm{u}}{\partial z_\mathrm{w}}+\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{u}}\frac{\partial y_\mathrm{u}}{\partial z_\mathrm{w}}$ |

TABLE XIV
DERIVATIVES OF STEP 6: TRANSFORMATION INTO THE IMAGE COORDINATE SYSTEM (I)

| | $\partial x_\mathrm{i}/\ldots$ | $\partial y_\mathrm{i}/\ldots$ |
|---|---|---|
| $\ldots/\partial s_x$ | $-x_\mathrm{d}/s_x^2$ | $0$ |
| $\ldots/\partial s_y$ | $0$ | $-y_\mathrm{d}/s_y^2$ |
| $\ldots/\partial c_x$ | $1$ | $0$ |
| $\ldots/\partial c_y$ | $0$ | $1$ |
| $\ldots/\partial x_\mathrm{d}$ | $1/s_x$ | $0$ |
| $\ldots/\partial y_\mathrm{d}$ | $0$ | $1/s_y$ |

TABLE XV
DERIVATIVES OF STEP 6: TRANSFORMATION INTO THE IMAGE COORDINATE SYSTEM (II)

| | $\partial x_\mathrm{i}/\ldots$ | $\partial y_\mathrm{i}/\ldots$ |
|---|---|---|
| $\ldots/\partial^\mathrm{b}\alpha_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{b}\alpha_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}\beta_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{b}\beta_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}\gamma_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{b}\gamma_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}tx_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{b}tx_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{b}tx_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}ty_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{b}ty_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{b}ty_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{b}tz_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{b}tz_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{b}tz_\mathrm{w}}$ |
| $\ldots/\partial^\mathrm{t}\alpha_\mathrm{b}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{t}\alpha_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}\beta_\mathrm{b}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{t}\beta_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}\gamma_\mathrm{b}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{t}\gamma_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}tx_\mathrm{b}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{t}tx_\mathrm{b}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{t}tx_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}ty_\mathrm{b}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{t}ty_\mathrm{b}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{t}ty_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{t}tz_\mathrm{b}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{t}tz_\mathrm{b}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{t}tz_\mathrm{b}}$ |
| $\ldots/\partial^\mathrm{c}\alpha_\mathrm{t}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{c}\alpha_\mathrm{t}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{c}\alpha_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}\beta_\mathrm{t}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{c}\beta_\mathrm{t}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{c}\beta_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}\gamma_\mathrm{t}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{c}\gamma_\mathrm{t}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{c}\gamma_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}tx_\mathrm{t}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{c}tx_\mathrm{t}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{c}tx_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}ty_\mathrm{t}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{c}ty_\mathrm{t}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{c}ty_\mathrm{t}}$ |
| $\ldots/\partial^\mathrm{c}tz_\mathrm{t}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial^\mathrm{c}tz_\mathrm{t}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial^\mathrm{c}tz_\mathrm{t}}$ |
| $\ldots/\partial c$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial c}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial c}$ |
| $\ldots/\partial \kappa$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial \kappa}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial \kappa}$ |
| $\ldots/\partial x_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial x_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial x_\mathrm{w}}$ |
| $\ldots/\partial y_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial y_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial y_\mathrm{w}}$ |
| $\ldots/\partial z_\mathrm{w}$ | $\frac{\partial x_\mathrm{i}}{\partial x_\mathrm{d}}\frac{\partial x_\mathrm{d}}{\partial z_\mathrm{w}}$ | $\frac{\partial y_\mathrm{i}}{\partial y_\mathrm{d}}\frac{\partial y_\mathrm{d}}{\partial z_\mathrm{w}}$ |

and efficient 3D robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, Jun. 1989.

[17] J. C. K. Chou and M. Kamel, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," *The International Journal of Robotics Research*, vol. 10, no. 3, pp. 240–254, 1991.

[18] H. H. Chen, "A screw motion approach to uniqueness analysis of head-eye geometry," in *Computer Vision and Pattern Recognition*, 1991, pp. 145–151.

[19] R. Horaud and F. Dornaika, "Hand-eye calibration," *International Journal of Robotics Research*, vol. 14, no. 3, pp. 195–210, 1995.

[20] F. Dornaika and R. Horaud, "Simultaneous robot-world and hand-eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 617–622, 1998.

[21] K. Daniilidis, "Hand-eye calibration using dual quaternions," *International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[22] N. Andreff, R. Horaud, and B. Espiau, "Robot hand-eye calibration using structure-from-motion," *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 228–248, 2001.

[23] J. Schmidt, F. Vogt, and H. Niemann, "Robust hand–eye calibration of an endoscopic surgery robot using dual quaternions," in *Pattern Recognition*, B. Michaelis and G. Krell, Eds. Springer Berlin Heidelberg, 2003, pp. 548–556.

[24] A. Li, L. Wang, and D. Wu, "Simultaneous robot-world and hand-eye calibration using dual-quaternions and Kronecker product," *International Journal of Physical Sciences*, vol. 5, no. 10, pp. 1530–1536, 2010.

[25] M. Shah, "Solving the robot-world/hand-eye calibration problem using the Kronecker product," *Journal of Mechanisms and Robotics*, vol. 5, no. 3: 031007, 2013.

[26] M. Ulrich and C. Steger, "Hand-eye calibration of SCARA robots using dual quaternions," *Pattern Recognition and Image Analysis*, vol. 26, no. 1, pp. 231–239, 2016.

[27] J. Schmidt and H. Niemann, "Data selection for hand-eye calibration: A vector quantization approach," *The International Journal of Robotics Research*, vol. 27, no. 9, pp. 1027–1053, 2008.

[28] F. C. Park and B. J. Martin, "Robot sensor calibration: solving AX=XB on the Euclidean group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717–721, 1994.

[29] S. Rémy, M. Dhome, J. M. Lavest, and N. Daucher, "Hand-eye calibration," in *International Conference on Intelligent Robots and Systems*, 1997, pp. 1057–1065.

[30] G.-Q. Wei, K. Arbter, and G. Hirzinger, "Active self-calibration of robotic eyes and hand-eye relationships with model identification," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 158–166, Feb. 1998.

[31] J. Schmidt, F. Vogt, and H. Niemann, "Calibration–free hand–eye calibration: A structure–from–motion approach," in *Pattern Recognition*, W. G. Kropatsch, R. Sablatnig, and A. Hanbury, Eds. Springer Berlin Heidelberg, 2005, pp. 67–74.

[32] J. Heller, M. Havlena, and T. Pajdla, "Globally optimal hand-eye calibration using branch-and-bound," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 5, pp. 1027–1033, 2015.

[33] I. Ali, O. Suominen, A. Gotchev, and E. R. Morales, "Methods for simultaneous robot-world-hand-eye calibration: A comparative study," *Sensors*, vol. 19, no. 12, 2837, 2019.

[34] I. Enebuse, M. Foo, B. S. K. K. Ibrahim, H. Ahmed, F. Supmak, and

O. S. Eyobu, "A comparative review of hand-eye calibration techniques for vision guided robots," *IEEE Access*, vol. 9, pp. 113 143–113 155, 2021.

[35] H. Nguyen and Q. Pham, "On the covariance of $\boldsymbol{X}$ in $\boldsymbol{AX} = \boldsymbol{XB}$," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1651–1658, 2018.

[36] J. Brookshire and S. Teller, "Extrinsic calibration from per-sensor egomotion," in *Robotics: Science and Systems VIII*. Cambridge, MA, USA: MIT Press, 2013, pp. 504–512.

[37] J. Wu, Y. Sun, M. Wang, and M. Liu, "Hand-eye calibration: 4-D procrustes analysis approach," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 2966–2981, 2020.

[38] J. Wu, M. Liu, Y. Huang, C. Jin, Y. Wu, and C. Yu, "SE(n)++: An efficient solution to multiple pose estimation problems," *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3829–3840, 2022.

[39] J. H. D. Henrion and T. Pajdla, "Hand-eye and robot-world calibration by global polynomial optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3157–3164.

[40] E. Wise, M. Giamou, S. Khoubyarian, A. Grover, and J. Kelly, "Certifiably optimal monocular hand-eye calibration," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2020, pp. 271–278.

[41] A. Malti, "Hand-eye calibration with epipolar contraints: Application to endoscopy," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 161–169, 2013.

[42] A. Tabb and K. M. A. Yousef, "Solving the robot-world hand-eye(s) calibration problem with iterative methods," *Machine Vision and Applications*, vol. 28, no. 5, pp. 569–590, Aug. 2017.

[43] K. Koide and E. Menegatti, "General hand–eye calibration based on reprojection error minimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1021–1028, Apr. 2019.

[44] R. Lenz and D. Fritsch, "Accuracy of videometry with CCD sensors," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 45, no. 2, pp. 90–110, 1990.

[45] D. C. Brown, "Close-range camera calibration," *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866, Aug. 1971.

[46] W. Förstner and B. P. Wrobel, *Photogrammetric Computer Vision: Statistics, Geometry, Orientation and Reconstruction*. Springer International Publishing, 2016.

[47] H. Zhuang, "Hand/eye calibration for electronic assembly robots," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 612–616, 1998.

[48] K.-R. Koch, *Parameter Estimation and Hypothesis Testing in Linear Models*. Berlin Heidelberg: Springer, 1999.

[49] ——, *Introduction to Bayesian Statistics*, 2nd ed. Heidelberg: Springer, 2007.

[50] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[51] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2020.

[52] J. Mallon and P. F. Whelan, "Which pattern? Biasing aspects of planar calibration patterns and detection methods," *Pattern Recognition Letters*, vol. 28, no. 9, pp. 921–930, 2007.

[53] C. Steger, "A comprehensive and versatile camera model for cameras with tilt lenses," *International Journal of Computer Vision*, vol. 123, no. 2, pp. 121–159, 2017.

[54] DLR, "CalLab," http://www.robotic.dlr.de/callab/, 2005, accessed: 2023-10-17.

[55] Blender Online Community, *Blender - a 3D modelling and rendering package*, http://www.blender.org, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022, accessed: 2022-02-10.

[56] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[57] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct point, corners and centres of circular features," in *Proceedings of the ISPRS Conference on Fast Processing of Photogrammetric Data*, 1987, pp. 281–305.

[58] F. C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *Journal of Mechanical Design*, vol. 117, no. 1, pp. 48–54, Mar. 1995.

[59] B. Erdnuess, "Proper synchronization of geospatial metadata in motion imagery and its evaluation," in *Geospatial Informatics X*, P. J. Doucette, J. D. Harguess, K. Palaniappan, and G. Seetharaman, Eds., vol. 11398, International Society for Optics and Photonics. SPIE, 2020, pp. 85 – 105.

**Markus Ulrich** studied Geodesy at the Technical University of Munich (TUM) where he also received his doctorate at the Chair of Photogrammetry and Remote Sensing in 2003. Subsequently, he joined the Research and Development department at MVTec Software GmbH as a software engineer where he became head of the research team in 2008. From 2005 to 2020, he was also a guest lecturer at TUM, where he taught close-range photogrammetry, and from 2013 to 2020, he was a guest lecturer at the Karlsruhe Institute of Technology (KIT), where he taught machine vision. In 2017, he completed his habilitation and was appointed a Privatdozent (lecturer) at the KIT department of Civil Engineering, Geo and Environmental Sciences.

Since 2020, he is Professor of Machine Vision Metrology with the KIT department of Civil Engineering, Geo and Environmental Sciences at the Institute of Photogrammetry and Remote Sensing. His research areas include machine vision, close-range photogrammetry, image processing, machine learning and their applications in industry.

**Markus Hillemann** studied Geodesy and Geoinformatics at Karlsruhe Institute of Technology (KIT). From 2016 to 2020 he was research associate and PhD student at the Institute of Photogrammetry and Remote Sensing (IPF) at KIT as well as external employee at the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB) in Ettlingen. In 2020 he received his doctorate at the KIT department of Civil Engeneering, Geo and Environmental Sciences. Subsequently, he joined the Machine Vision Metrology group at IPF as a PostDoc. His research interests include methodology development and applications in machine vision, image and point cloud processing, machine learning and photogrammetry.