Hand-Eye Calibration of SCARA Robots Using Dual Quaternions[1,2]

M. Ulrich and C. Steger

MVTec Software GmbH, Neherstr. 1, 81675 München, Germany
e-mail: ulrich@mvtec.com, steger@mvtec.com

Abstract—In SCARA robots, which are often used in industrial applications, all joint axes are parallel, covering three degrees of freedom in translation and one degree of freedom in rotation. Therefore, conventional approaches for the hand-eye calibration of articulated robots cannot be used for SCARA robots. In this paper, we present a new linear method that is based on dual quaternions and extends the work of Daniilid is 1999 (IJRR) for SCARA robots. To improve the accuracy, a subsequent nonlinear optimization is proposed. We address several practical implementation issues and show the effectiveness of the method by evaluating it on synthetic and real data.

Keywords: calibration, hand-eye, SCARA, robotics, dual quaternions, industry.

DOI: 10.1134/S1054661816010272

1. INTRODUCTION

SCARA (Selectively Compliant Arm for Robotic Assembly) robots [11, 12] are used in many industrial applications. They differ from articulated (antropomorphic) robots in that their movements are more restricted. The arm of an articulated robot typically has 6 rotary joints that cover 6 degrees of freedom (3 translations and 3 rotations). In contrast, SCARA robots have at least 2 parallel rotary joints and 1 parallel prismatic joint that cover only 4 degrees of freedom (3 translations and 1 rotation). Figure 1 shows typical SCARA setups with 3 rotary joints. The camera can either be mounted on the robot's end effector (tool) and is moved to different positions by it or it can be mounted stationary outside the robot. Compared to other robot types, SCARA robots offer faster and more precise performance. They are best suited for pick and place, packaging, and assembly applications, and are often preferred if only limited space is available.

We assume that the (moving or stationary) camera observes the workspace of the robot. Objects are detected and localized in the camera coordinate system. To be able to grasp the object, the object pose must be transformed into the base coordinate system of the robot. The process of determining the required transformation between the camera and the robot base coordinate systems for a stationary camera or between the camera and the robot tool coordinate system for a moving camera is called hand-eye calibration.

[1] The article is published in the original.

[2] This paper uses the materials of the report submitted at the 9th Open German-Russian Workshop on Pattern Recognition and Image Understanding, held in Koblenz, December 1–5, 2014 (OGRW-9-2014).

Received October 14, 2015

Figure 1 shows the transformations that are involved in the hand-eye calibration process for the case of a stationary and a moving camera. ${}^{c2}\mathbf{H}_{c1}$ is a rigid 3D transformation, represented by a 4×4 homogeneous transformation matrix, that transforms 3D points from the coordinate system $c1$ into $c2$. For stationary cameras, the fixed and unknown transformations are ${}^{cam}\mathbf{H}_{base}$ and ${}^{tool}\mathbf{H}_{cal}$. Note that although the calibration object is rigidly attached to the robot tool, in general the exact relative pose of the calibration object with respect to the robot tool cannot be gauged accurately by hand, and hence is unknown. The remaining two transformations ${}^{base}\mathbf{H}_{tool}$ and ${}^{cam}\mathbf{H}_{cal}$ of the closed chain of transformations depend on the robot pose and are known from the robot kinematics and from the algorithm that determines the pose of the calibration object in the camera coordinate system based on a calibration image. For moving cameras, the fixed and unknown transformations are ${}^{cam}\mathbf{H}_{tool}$ and ${}^{base}\mathbf{H}_{cal}$, while the known transformations are the same as for stationary cameras.

In the following, we will concentrate on the case of a stationary camera. The case of a moving camera can be treated in an equivalent way. We assume that the robot is calibrated, i.e., ${}^{base}\mathbf{H}_{tool}$ is known accurately. Furthermore, we assume a calibrated camera, i.e., the interior camera parameters are known, and hence ${}^{cam}\mathbf{H}_{cal}$ can be computed [21, chapter 3.9]. For simplicity reasons, we also assume, without loss of generality, that all joint axes of the SCARA robot are parallel to the z axis of the robot base and tool coordinate systems. In [27], is it shown that if the joint axes are not aligned with the z axes, the coordinate system can simply be rotated to achieve alignment. Aligning the joint axes with the z axis has the advantage that the

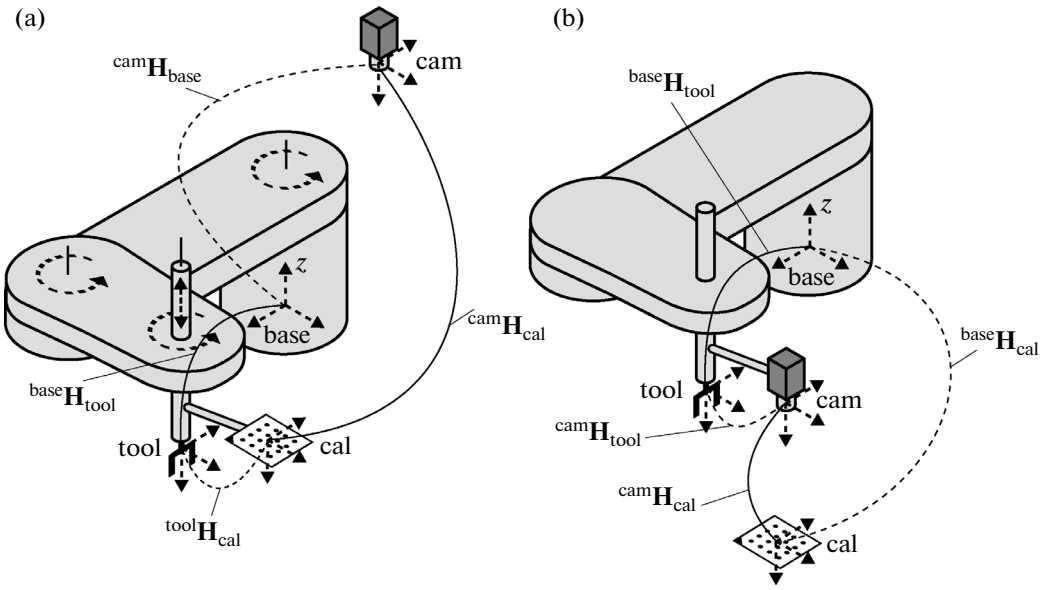

Fig. 1. Coordinate systems and transformations that are involved in the hand-eye calibration for the case of a stationary camera (a) and a moving camera (b). The coordinate systems are: camera (cam), robot base (base), calibration object (cal), robot tool (tool). Unknown transformations that are determined by the hand-eye calibration are visualized by dashed lines.

direction of the indeterminate translation (see below) is well known. This alignment is in accordance with [12], where the z axis of base and tool coordinate system are parallel to the robot joint axes. The input for the calibration is obtained by moving the robot's tool to n different poses and for each pose acquiring an image of the calibration object that is attached to the tool. Thus, the input consists of one pair of calibration poses ${}^{base}\mathbf{H}_{tool}$ and ${}^{cam}\mathbf{H}_{cal}$ for each of the n robot states.

It should be noted that the hand-eye calibration can also be performed without a calibration object by using approaches that are able to determine the 3D pose of arbitrary objects in a monocular image, e.g., [9, 25]. Furthermore, instead of a camera, a 3D sensor can be used to observe the workspace of the robot. In this case, ${}^{cam}\mathbf{H}_{cal}$ can be obtained by approaches that are able to determine the pose of objects in 3D sensor data, like [6, 7, 13].

1.1. Hand-Eye Calibration of Articulated Robots

The hand-eye calibration of articulated robots is well understood. One of the most common problem formulations is based on closing the chain of transformations [22]:

$$ {}^{tool}\mathbf{H}_{cal} = {}^{tool}\mathbf{H}_{base}\,{}^{base}\mathbf{H}_{cam}\,{}^{cam}\mathbf{H}_{cal}. \quad (1)$$

For each pair of calibration poses we set $\mathbf{A}_i = {}^{tool}\mathbf{H}_{base}$ and $\mathbf{B}_i = {}^{cam}\mathbf{H}_{cal}$, $i = 1 \ldots n$. By setting the unknown poses to $\mathbf{X} = {}^{base}\mathbf{H}_{cam}$ and $\mathbf{Y} = {}^{tool}\mathbf{H}_{cal}$, we obtain $\mathbf{Y} = \mathbf{A}_i\mathbf{X}\mathbf{B}_i$. The pose $\mathbf{Y}$ can be eliminated by combining two pairs of poses from two different states i and j, resulting in $\mathbf{A}_i\mathbf{X}\mathbf{B}_i = \mathbf{A}_j\mathbf{X}\mathbf{B}_j$. Rearranging yields $\mathbf{A}_j^{-1}\mathbf{A}_i\mathbf{X} = \mathbf{X}\mathbf{B}_j\mathbf{B}_i^{-1}$. The relative poses $\mathbf{A} = \mathbf{A}_j^{-1}\mathbf{A}_i$ and $\mathbf{B} = \mathbf{B}_j\mathbf{B}_i^{-1}$ represent the motions between the poses i and j of the tool and of the calibration object, respectively. Finally, the hand-eye calibration must solve

$$\mathbf{A}\mathbf{X} = \mathbf{X}\mathbf{B} \quad (2)$$

with respect to the unknown $\mathbf{X}$. Simple linear algorithms typically handle rotation and translation separately [3, 23]. This results in rotational errors propagating and increasing translational errors. More advanced linear methods, typically based on screw theory (see Section 2.1 for a short introduction to screw theory), avoid this by solving for rotation and translation simultaneously [1, 4, 10]. Based on the results of linear methods, nonlinear optimizations can be used to further improve accuracy [4, 10, 14].

Another class of approaches computes $\mathbf{X}$ and $\mathbf{Y}$ simultaneously [5, 17, 28].

1.2. Hand-Eye Calibration of SCARA Robots

Unfortunately, it is not possible to simply apply the methods for articulated robots to SCARA robots. In [23], the authors state that the error of the hand-eye calibration of articulated robots is inversely proportional to the sine of the angle between the screw axes (robot motions can be represented by screws, see below). Because the rotation axes of SCARA robots are parallel, all screw axes are parallel, too. Consequently, the error would be infinite. In particular, it is shown in [1] that if all screw axes are parallel, one parameter cannot be determined by the hand-eye calibration.

The hand-eye calibration of SCARA robots has drawn considerably less attention compared to articu-

lated robots. In [8, 16], linear methods are presented for solving the rotational part of the hand-eye calibration that work for robots with only one degree of freedom in rotation. Both approaches assume restricted specifically designed robot motions during calibration. In [27], a linear solution is presented that does not impose such restrictions on the robot motion while the calibration data is gathered. The linear solution is then used as an initial condition within an iterative optimization framework to further improve the accuracy. The author emphasizes that only up to five of the six unknown pose parameters can be determined and claims that the missing sixth parameter is irrelevant for the pose measurement of the sensor. Like [3, 23], the linear method proposed in [27] solves for rotation and translation separately, which also results in the disadvantage that rotational errors propagate and increase translational errors.

In this paper, we extend the approach of [4], which is based on dual quaternions and handles rotation and translation simultaneously, for the calibration of SCARA robots. Furthermore, we show that for practical applications it is indeed necessary to know all of the six pose parameters. Therefore, we propose a pragmatic solution to determine the unknown sixth parameter. Finally, experiments on synthetic as well as on real data are presented. This paper extends our previous work [24] by a more thorough mathematical derivation, a slightly improved accuracy, and additional evaluations.

2. HAND-EYE CALIBRATION OF SCARA ROBOTS USING DUAL QUATERNIONS

2.1. Screw Theory

To better understand the relation between the motion of the tool and calibration object, screw theory can be used. It was first proposed for hand-eye calibration in [1]. A screw $(d, \theta, \vec{L})$ describes a 3D rigid transformation (rotation and translation) through a translation d along the screw axis $\vec{L}$ and a rotation by the rotation angle θ around the same axis. It is known from Chasles' theorem that any rigid 3D transformation can be described by a screw [4].

In [1], the motions **A** and **B** in (2) are expressed as screws. This allows to simultaneously solve for rotation and translation during hand-eye calibration. Because the calibration object is rigidly connected to the robot tool, the transformation between both coordinate systems is constant over time. The screws **A** and **B** represent the same rigid 3D transformation seen from different coordinate systems. The hand-eye calibration problem can be interpreted as finding the rigid transformation that aligns both screws [4]. The screw congruence theorem further tells us that the rotation angle θ and the translation d of two screws are identical if one screw is the result of a rigid transformation applied to the other screw [1]. Consequently, the task of aligning screws reduces to aligning their screw axes, i.e., lines, in 3D space.

In [1], it is shown that at least two non-parallel robot motions are necessary to fix all degrees of freedom of the alignment of screw axes. For SCARA robots, all rotation axes, and hence all screw axes, are parallel to the z axis of the robot base coordinate system. Two parallel motions only fix five of the six parameters of the unknown transformation between the base and camera coordinate system. Therefore, for SCARA robots, the translation t_z along the z axis of the robot base coordinate system is undetermined [1, 27].

Screw theory is the foundation of the calibration method for articulated robots presented in [4], which simultaneously solves for rotation and translation. It will be described and extended for SCARA robots in the next section.

2.2. Dual Quaternions

Quaternions $\mathbf{q} = (q, \vec{q})$ with scalar part q and vector part $\vec{q}$ are an extension of complex numbers to $\mathbf{R}^4$. For every rotation about an axis $\vec{n}$ ($\|\vec{n}\| = 1$) with an angle θ, there exist two corresponding unit quaternions $\mathbf{q} = (\cos(\theta/2), \vec{n}\sin(\theta/2))$ and $-\mathbf{q}$ that each map a vector $\vec{p} \in \mathbf{R}^3$ to the rotated vector $\mathbf{q}(0, \vec{p})\overline{\mathbf{q}}$, where $\overline{\mathbf{q}}$ is the conjugate quaternion $(q, -\vec{q})$.

Dual numbers are defined as $\hat{z} = a + \varepsilon b$ with the real part a, the dual part b, and the dual unit $\varepsilon^2 = 0$. Dual 3-vectors $\hat{\vec{z}}$ with orthogonal real and dual parts are a representation of lines in $\mathbf{R}^3$, known as Plücker coordinates, where the real part is the direction of the line and the dual part is its moment. The line with direction $\vec{l}$ through the point $\vec{p}$ is $\hat{\vec{z}} = \vec{l} + \varepsilon(\vec{p} \times \vec{l})$ with $\|\vec{l}\| = 1$.

Dual quaternions are an extension of quaternions to dual numbers. A general introduction to dual quaternions can be found in [15], a brief outline is included in [4]. Similar to how unit quaternions are a tool to easily manipulate rotations in 3D space, dual unit quaternions can be used as a representation of rigid 3D transformations. Dual quaternions directly encode screws and have advantages over homogeneous transformation matrices when transforming lines in 3D [19].

A dual quaternion $\hat{\mathbf{q}} = \mathbf{q} + \varepsilon\mathbf{q}'$ consists of the quaternions $\mathbf{q}$ (real part) and $\mathbf{q}'$ (dual part) with 4 elements each and is also often written as a single 8-vector. For unit dual quaternions, the following two conditions hold:

$$\mathbf{q}\overline{\mathbf{q}} = 1 \Rightarrow \mathbf{q}^\top\overline{\mathbf{q}} = 1 \tag{3}$$

and

$$\overline{\mathbf{q}}\mathbf{q}' + \mathbf{q}\overline{\mathbf{q}}' = 0 \Rightarrow \mathbf{q}^\top\mathbf{q}' = 0. \tag{4}$$

2.3. Hand-Eye Calibration

Dual vectors (lines) can be written as pure dual quaternions (scalar part 0). The rigid transformations of the screw axes can be written concisely by using dual quaternions [19]:

$$\hat{\mathbf{a}} = \hat{\mathbf{x}}\hat{\mathbf{b}}\bar{\hat{\mathbf{x}}}, \tag{5}$$

where $\hat{\mathbf{a}} = \mathbf{a} + \varepsilon\mathbf{a}'$ and $\hat{\mathbf{b}} = \mathbf{b} + \varepsilon\mathbf{b}'$ are unit dual quaternions that represent the screws for the tool and the calibration object, respectively, for a single robot motion. $\hat{\mathbf{x}} = \mathbf{x} + \varepsilon\mathbf{x}'$ is a unit dual quaternion that represents the unknown transformation $\mathbf{X}$. In [4], the definition of dual quaternion multiplication is used to rewrite (5) as

$$\mathbf{S}\begin{pmatrix}\mathbf{x}\\ \mathbf{x}'\end{pmatrix} = 0 \tag{6}$$

with the 6×8 matrix

$$\mathbf{S} = \begin{pmatrix}\vec{a}-\vec{b} & [\vec{a}+\vec{b}]_\times & \vec{0} & \vec{0}\\ \vec{a}'-\vec{b}' & [\vec{a}'+\vec{b}']_\times & \vec{a}-\vec{b} & [\vec{a}+\vec{b}]_\times\end{pmatrix}, \tag{7}$$

encoding the result of the dual quaternion multiplication, where $\mathbf{a} = (0, \vec{a})$, $\mathbf{a}' = (0, \vec{a}')$, $\mathbf{b} = (0, \vec{b})$, and $\mathbf{b}' = (0, \vec{b}')$ represent the screw axes. Stacking the matrices $\mathbf{S}_i$ from all m motions results in the $6m \times 8$ matrix $\mathbf{T}^\top = (\mathbf{S}_1^\top \mathbf{S}_2^\top \ldots \mathbf{S}_n^\top)^\top$, which can be decomposed by the SVD as:

$$\mathbf{T} = \mathbf{U}\Sigma\mathbf{V}^\top. \tag{8}$$

For articulated robots, $\mathbf{T}$ has rank 6, and hence results in two vanishing singular values with the two corresponding right singular vectors $\vec{v}_7$ and $\vec{v}_8$. The set of solutions is the null space of $\mathbf{T}$:

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{x}'\end{pmatrix} = \lambda_1\vec{v}_7 + \lambda_2\vec{v}_8. \tag{9}$$

The remaining two degrees of freedom can be fixed by substituting the two unity constraints (3) and (4) into (9). This leads to a system of two quadratic polynomials in λ_1 and λ_2, which can be solved directly for $\hat{\mathbf{x}}$ [4].

For SCARA robots, the rank of the matrix $\mathbf{T}$ is 5 in the noise-free case. This results in three vanishing singular values and three matching right singular vectors $\vec{v}_6$, $\vec{v}_7$, and $\vec{v}_8$, yielding the set of solutions:

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{x}'\end{pmatrix} = \lambda_1\vec{v}_6 + \lambda_2\vec{v}_7 + \lambda_3\vec{v}_8. \tag{10}$$

From [27] we know that the translation component t_z of the unknown transformation $\mathbf{X}$ cannot be determined. Hence, there exists a 1D manifold of equivalent solutions, i.e., solutions with identical algebraic error. Therefore, it is sufficient to obtain one arbitrary solution out of this set (i.e., a solution for an arbitrary t_z). To determine t_z we require $t_z = 0$. In Section 3.5, methods to determine the real t_z, which is essential for most practical applications. Let $\vec{v}_6^\top = (\vec{u}_1^\top, \vec{v}_1^\top)^\top$, $\vec{v}_7^\top = (\vec{u}_2^\top, \vec{v}_2^\top)^\top$, and $\vec{v}_8^\top = (\vec{u}_3^\top, \vec{v}_3^\top)^\top$. Then, the two unity constraints yield:

$$\begin{aligned}&\lambda_1^2\vec{u}_1^\top\vec{u}_1 + \lambda_2^2\vec{u}_2^\top\vec{u}_2 + \lambda_3^2\vec{u}_3^\top\vec{u}_3\\ &+ 2\lambda_1\lambda_2\vec{u}_1^\top\vec{u}_2 + 2\lambda_1\lambda_3\vec{u}_1^\top\vec{u}_3 + 2\lambda_2\lambda_3\vec{u}_2^\top\vec{u}_3 = 1,\end{aligned} \tag{11}$$

$$\begin{aligned}&\lambda_1^2\vec{u}_1^\top\vec{v}_1 + \lambda_2^2\vec{u}_2^\top\vec{v}_2 + \lambda_3^2\vec{u}_3^\top\vec{v}_3\\ &+ 2\lambda_1\lambda_2(\vec{u}_1^\top\vec{v}_2 + \vec{u}_2^\top\vec{v}_1) + 2\lambda_1\lambda_3(\vec{u}_1^\top\vec{v}_3 + \vec{u}_3^\top\vec{v}_1)\\ &+ 2\lambda_2\lambda_3(\vec{u}_2^\top\vec{v}_3 + \vec{u}_3^\top\vec{v}_2) = 0.\end{aligned} \tag{12}$$

To apply the third constraint $t_z = 0$, we compute the translation vector $\vec{t}$ of $\hat{\mathbf{x}}$ as $(0, \vec{t}) = 2\mathbf{x}\bar{\mathbf{x}}'$. Let $x_1 \ldots x_4$ and $x_1' \ldots x_4'$ represent the elements of $\mathbf{x}$ and $\mathbf{x}'$. Then, $t_z = 0$ is equivalent to

$$x_1x_4' + x_2x_3' - x_3x_2' - x_4x_1' = 0. \tag{13}$$

By also denoting the elements of $\vec{u}_1$, $\vec{v}_1$, $\vec{u}_2$, $\vec{v}_2$, $\vec{u}_3$, and $\vec{v}_3$ as $u_{1,1} \ldots u_{1,4}$, $v_{1,1} \ldots v_{1,4}$, etc., constraint (13) expands to

$$\begin{aligned}&\lambda_1^2(u_{1,1}v_{1,4} + u_{1,2}v_{1,3} - u_{1,3}v_{1,2} - u_{1,4}v_{1,1})\\ &+ \lambda_2^2(u_{2,1}v_{2,4} + u_{2,2}v_{2,3} - u_{2,3}v_{2,2} - u_{2,4}v_{2,1})\\ &+ \lambda_3^2(u_{3,1}v_{3,4} + u_{3,2}v_{3,3} - u_{3,3}v_{3,2} - u_{3,4}v_{3,1})\\ &+ \lambda_1\lambda_2(u_{1,1}v_{2,4} + u_{2,1}v_{1,4} + u_{1,2}v_{2,3} + u_{2,2}v_{1,3}\\ &- u_{1,3}v_{2,2} - u_{2,3}v_{1,2} - u_{1,4}v_{2,1} - u_{2,4}v_{1,1})\\ &+ \lambda_1\lambda_3(u_{1,1}v_{3,4} + u_{3,1}v_{1,4} + u_{1,2}v_{3,3} + u_{3,2}v_{1,3}\\ &- u_{1,3}v_{3,2} - u_{3,3}v_{1,2} - u_{1,4}v_{3,1} - u_{3,4}v_{1,1})\\ &+ \lambda_2\lambda_3(u_{2,1}v_{3,4} + u_{3,1}v_{2,4} + u_{2,2}v_{3,3} + u_{3,2}v_{2,3}\\ &- u_{2,3}v_{3,2} - u_{3,3}v_{2,2} - u_{2,4}v_{3,1} - u_{3,4}v_{2,1}) = 0.\end{aligned} \tag{14}$$

Equations (11), (12), and (14) are three quadratic equations in the three unknowns λ_1, λ_2, and λ_3, which represent three concentric quadrics centered at the origin. Equation (11) represents an ellipsoid, while (12) and (14) represent elliptic cones. Computing the up to eight solutions corresponds to computing the intersection points of the three quadrics. Methods for intersecting three general quadric surfaces can be found in [2, 26], for example. The downside of their generality is their algorithmic as well as their computational complexity. Therefore, to simplify matters, we exploit the fact that the quadrics are concentric and that their type is known: First, by intersecting the two elliptic cones with the plane at infinity we obtain two conics. The intersection points of the conics represent the directions of up to four 3D lines emanating from the origin. See [18, chapter 11] for further details about the computation of the intersections. The lines can be intersected with the ellipsoid, resulting in up to eight solutions. For each solution $(\lambda_1, \lambda_2, \lambda_3)$ there exists a solution $(-\lambda_1, -\lambda_2, -\lambda_3)$ that would end up in a negated dual quaternion, which represents the same rigid transformation. From the remaining up to four solutions, we select the one with minimum Euclidean norm. This approach was justified by extensive empirical evaluations that have shown that all other solu-

tions would end up in rigid transformations with implausibly large translation components. It should be noted that this algorithm also works if the quadrics are degenerated (e.g., into planes) and also if (11) does not represent an ellipsoid.

Still, the exact computation of the intersection points is cumbersome and computationally expensive. Therefore, we propose an alternative method that approximates the quadrics by planes. It is faster to compute and showed to be perfectly adequate for practical applications. Each of the three quadratic equations can be written as

$$a\lambda_1^2 + b\lambda_2^2 + c\lambda_3^2 + d\lambda_1\lambda_2 + e\lambda_1\lambda_3 + f\lambda_2\lambda_3 + g = 0 \quad (15)$$

with $a \ldots g$ denoting the coefficients of the constraints (11), (12), and (14), respectively. In matrix form, we can rewrite (15) as $\vec{\lambda}^\top \mathbf{A}\vec{\lambda} + g = 0$ with

$$\mathbf{A} = \begin{pmatrix} a & d/2 & e/2 \\ d/2 & b & f/2 \\ e/2 & f/2 & c \end{pmatrix} \quad \text{and} \quad \vec{\lambda} = (\lambda_1, \lambda_2, \lambda_3)^\top. \quad (16)$$

For the first constraint (11), two eigenvalues of $\mathbf{A}$ are close to 0 (or exactly 0 in the noise-free case). Hence, the ellipsoid degenerates into two parallel planes with distances $\pm 1/\sqrt{l}$ and normal vectors $\pm\vec{m}$, where l is the largest eigenvalue of $\mathbf{A}$ and $\vec{m}$ its corresponding eigenvector. Writing the planes in normal form yields

$$\vec{m}^\top\vec{\lambda} = \pm\frac{1}{\sqrt{l}}. \quad (17)$$

From the two planes, we arbitrarily select one since the other one would again result in the negated dual quaternion.

For the second and third constraint (12) and (14), one eigenvalue of $\mathbf{A}$ is close to 0 (or exactly 0 in the noise-free case). Hence, the elliptic cones degenerate into pairs of intersecting planes through the origin. The normal vectors of the planes can be derived easily to be $\vec{n}_{1,2} = -1/\sqrt{|l_1|}\vec{m}_1 \pm 1/\sqrt{|l_2|}\vec{m}_2$. Here, l_1 and l_2 are the eigenvalues of $\mathbf{A}$ with largest norm and $\vec{m}_1$ and $\vec{m}_2$ their corresponding eigenvectors. Writing the planes in normal form yields

$$\vec{n}_{1,2}^\top\vec{\lambda} = 0. \quad (18)$$

Evaluations have shown that one of the two planes $\vec{n}_{1,2}$ is always almost parallel to the plane in (17), while the other one intersects at a significantly larger angle. To avoid solutions close to infinity, we select the plane in (18) that forms the largest dihedral angle to the plane (17).

Equations (17) for the first constraint and (18) for the second and third constraint result in three linear equations in the three unknowns in $\vec{\lambda}$. Solving the system corresponds to intersecting the three planes. The evaluations in Section 4 will show that the loss of accuracy that is induced by approximating the quadrics by planes is negligible for practical applications.

3. IMPLEMENTATION

3.1. Selection of Robot Motions

In [20], it is pointed out that combining poses into motions in their chronological order is not the best choice in terms of numerical stability. Instead, it is proposed to select combinations of poses such that the orientation of the rotation axes of both poses are maximally different. While this criterion is not applicable for the calibration of SCARA robots, the basic idea is still useful.

The first criterion that we apply is the rotation angle of the screws. If the rotation angle is small, the screw axis is not well-defined and might be unstable in noisy conditions. Therefore, robot motions with larger rotation angles should be preferred.

The second criterion is again based on the screw congruence theorem, which for dual quaternions requires that the scalar parts of $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ are equal. Because of measurement errors, inaccuracies of the robot, and noise, this condition is not perfectly fulfilled. Therefore, motion pairs $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ with smaller differences in their scalar parts should be preferred.

In [20], a score is computed for all possible pose pairs. Then, the pairs with highest scores are selected for calibration. This may result in an unfavorable weighting, where some robot poses are represented multiple times while others are completely ignored. We propose the following approach instead: To apply the first criterion, for each pose $\mathbf{A}_i$ and $\mathbf{B}_i$ of the n robot poses, find the robot poses $\mathbf{A}_j$ and $\mathbf{B}_j$ with maximum rotation angle $r_a + r_b$, where r_a and r_b are the screw angles of the corresponding robot motions $\mathbf{A} = \mathbf{A}_j^{-1}\mathbf{A}_i$ and $\mathbf{B} = \mathbf{B}_j\mathbf{B}_i^{-1}$. This results in n robot motions. To apply the second criterion, for each pose pair $\mathbf{A}_i$ and $\mathbf{B}_i$, find the pose pair $\mathbf{A}_j$ and $\mathbf{B}_j$ for which the rotation and translation components of the corresponding screws $\mathbf{A}$ and $\mathbf{B}$ differ the least, i.e., for which the scalar parts of $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ differ the least. Hence, after eliminating duplicates, up to $2n$ robot motions are selected for calibration.

3.2. Antiparallel Screw Axes

If the camera is mounted such that its z axis is parallel to the z axis of the base coordinate system, all vectors $\vec{a}$ are either $(0,0,s)^\top$ or $(0,0,-s)^\top$ (with $s = \sin(\theta/2)$) depending on whether the z axes point in the same orin opposite directions. Furthermore, all vectors $\vec{b}$ are always either $(0,0,s)^\top$ or $(0,0,-s)^\top$. Equation (7) involves a vector product with the vector $\vec{a} + \vec{b}$. Therefore, it might happen that all $\vec{a}$ are antiparallel to the corresponding $\vec{b}$, and hence their sum vanishes. In this case, $\mathbf{T}$ degrades to rank 4. For articulated robots, in general the screw axes of most robot motions are not parallel, even if the z axes of the coordinate systems

are. For SCARA robots, if all screw axes $\vec{a}$ and $\vec{b}$ point in opposite directions ($\vec{a}^\top\vec{b} < 0$), we propose to apply a rotation $\hat{\mathbf{r}}$ of 180° around the x axis to both sides of (5), yielding $\hat{\mathbf{r}}\hat{\mathbf{a}}\bar{\hat{\mathbf{r}}} = \hat{\mathbf{r}}\hat{\mathbf{x}}\hat{\mathbf{b}}\bar{\hat{\mathbf{x}}}\bar{\hat{\mathbf{r}}}$. By substituting $\hat{\mathbf{a}}^\star := \hat{\mathbf{r}}\hat{\mathbf{a}}\bar{\hat{\mathbf{r}}}$ and $\hat{\mathbf{x}}^\star := \hat{\mathbf{r}}\hat{\mathbf{x}}$, we obtain a modified problem formulation $\hat{\mathbf{a}}^\star = \hat{\mathbf{x}}^\star\hat{\mathbf{b}}\bar{\hat{\mathbf{x}}}^\star$, with $\hat{\mathbf{a}}^\star$ and $\hat{\mathbf{b}}$ now pointing to the same half space. After performing the hand-eye calibration, the solution of the original formulation can be obtained by $\hat{\mathbf{x}} = \bar{\hat{\mathbf{r}}}\hat{\mathbf{x}}^\star$.

3.3. Sign Ambiguity of Screws

In the same way that two unit quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation, every rigid transformation can be described by two screws $\hat{\mathbf{q}}$ and $-\hat{\mathbf{q}}$. Extracting the translation and rotation components from both screws results in $d^+ = -d^-$ and $\theta^+ = 2\pi - \theta^-$. The screw congruence theorem that is assumed in (5) requires $d_a = d_b$ and $\theta_a = \theta_b$. Consequently, it is important to choose the signs of the screws $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ consistently.

In [1], two approaches for solving the sign ambiguity are proposed. The first approach orients the screw axis such that $d > 0$. For d close to 0, this method becomes unstable. The second approach constrains the rotation angle θ to $0 \le \theta < \pi$, which becomes unstable for θ close to 0 or π. In [4], the scalar parts, which represent both θ and d, are checked for equality for all robot motions. Because for SCARA robots all screw axes are parallel, it is sufficient to determine the sign only for a single robot motion and fix the signs of all other motions accordingly. For this, we select the most unambiguous motion pair as the reference motion, i.e., the motion pair for which θ differs most from 0 and π. Then, the screw axes of all other motions are compared to the axis of the reference motion and flipped if necessary.

3.4. Refinement by Nonlinear Optimization

The resulting poses of the described linear approach can be used as initial values in a nonlinear optimization framework to further increase the accuracy. For this, we compute the error matrix

$$\mathbf{E} = {}^{tool}\mathbf{H}_{cal} - {}^{tool}\mathbf{H}_{base}\,{}^{base}\mathbf{H}_{cam}\,{}^{cam}\mathbf{H}_{cal} \qquad (19)$$

and minimize

$$e = \sum_{i=0}^{n} \mathrm{tr}(\mathbf{E}_i\mathbf{W}\mathbf{E}_i^\top), \quad \text{where} \quad \mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 9 \end{pmatrix}, \qquad (20)$$

over all n robot poses by using the Levenberg-Marquardt algorithm. The matrix $\mathbf{W}$ balances the different number of entries in $\mathbf{E}$ for rotation and translation (9 and 3, respectively). Furthermore, the translation part of all input matrices is scaled by $1/D$, where D is the maximum extent of the workspace defined by the robot poses. This ensures that the results are scale-invariant and errors in rotation and translation are weighted appropriately. Because t_z cannot be determined, the minimization is performed by varying only 11 of the 12 pose parameters and leaving t_z fixed.

3.5. Determination of the Real t_z

For most practical applications, it is essential to know t_z. Otherwise, objects for which the pose was determined in the camera coordinate system cannot be grasped by the robot. The following approach works well in practice:

Let us assume that ${}^{tool}\mathbf{H}_{cal}$ and ${}^{base}\mathbf{H}_{cam}$ (see (1)) are known from the calibration up to the z component of the translation part of ${}^{tool}\mathbf{H}_{cal}$. To determine t_z, the calibration object is detached from the robot and placed at an arbitrary position such that it can be observed by the camera. The pose of the calibration object is then automatically determined in the camera coordinate system to obtain ${}^{cam}\tilde{\mathbf{H}}_{cal}$. From this, we can compute the z component of the translation of ${}^{base}\tilde{\mathbf{H}}_{cal} = {}^{base}\mathbf{H}_{cam}\,{}^{cam}\tilde{\mathbf{H}}_{cal}$, which we will denote z_{calib}. Then, the tool of the robot is manually moved to the origin of the calibration object, which gives us ${}^{tool}\tilde{\mathbf{H}}_{base}$. The z component of the translation of ${}^{base}\tilde{\mathbf{H}}_{tool}$ represents the true translation and is denoted as z_{true}. z_{true} and z_{calib} represent the same physical distance, and hence must be identical. This can be achieved by modifying the z component of ${}^{tool}\mathbf{H}_{cal}$ by $z_{true} - z_{calib}$. Finally, ${}^{base}\mathbf{H}_{cam}$ can be computed by closing the chain of transformations (1).

The case of a moving camera can be treated similarly. However, for some setups it is not possible for the camera to observe the calibration object if the tool is moved to the origin of the calibration object. Let us assume that ${}^{base}\mathbf{H}_{cal}$ and ${}^{cam}\mathbf{H}_{tool}$ are known from the calibration up to the z component of the translation part of ${}^{base}\mathbf{H}_{cal}$. In this case, the robot is manually moved to two poses. First, the tool is moved such that the camera can observe the calibration object. Now, an image of the calibration object is acquired and the tool pose is queried, which gives us ${}^{cam}\tilde{\mathbf{H}}_{cal}$ and ${}^{base}\tilde{\mathbf{H}}_{tool}$. Second, the tool of the robot is moved to the origin of the calibration object, yielding ${}^{base}\tilde{\tilde{\mathbf{H}}}_{tool}$. z_{true} is the z component of the translation of ${}^{tool}\tilde{\mathbf{H}}_{base}\,{}^{base}\tilde{\tilde{\mathbf{H}}}_{tool}$. z_{calib} is the z component of the translation of ${}^{tool}\mathbf{H}_{cam}\,{}^{cam}\tilde{\mathbf{H}}_{cal}$. Again, $z_{true} - z_{calib}$ can be used to correct the z component of ${}^{base}\mathbf{H}_{cal}$.

Actually, it is sufficient in the above approaches to move the tool to a point with the same z coordinate in

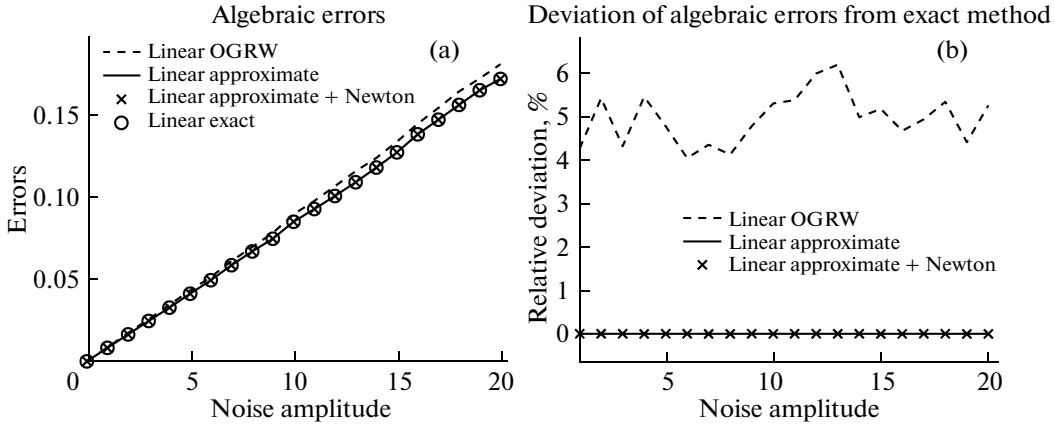

Fig. 2. (a) Algebraic errors for increasing noise levels of the linear approach presented in [24] (Linear OGRW), the linear approximate method (Linear approximate), the linear approximate method refined by Newton iterations (Linear approximate + newton), and the exact method (Linear exact). The noise level was increased up to a maximum amplitude of 3 mm and 2° . (b) Difference (%) of the errors with respect to the errors of the exact method.

the base coordinate system as the origin of the calibration object. Sometimes, however, the origin or even a point with the same z coordinate cannot be reached by the tool. In this case, the tool should be moved to a point with known height (i.e., vertical distance in the z direction of the base coordinate system) above or below the origin. The z component of the transformation must additionally be corrected by this height.

4. EXPERIMENTAL RESULTS

We tested our algorithms on synthetic and real data. For the experiments with synthetic data, we simulated different random setups with a stationary camera. For this, the pose of the camera with respect to the robot base, the pose of the calibration object with respect to the robot tool, and the robot poses were randomly created within appropriate bounds.

In the first experiment, we investigated the difference between the exact method that intersects the original quadrics and the approximate method that approximates the quadrics by planes. We also improved the results of the approximate method by a subsequent Newton root finding algorithm to verify whether the results converge to the results of the exact method. Finally, we also show the benefit over the method presented in [24].

We added noise of increasing amplitudes to the position and rotation components of ${}^{cam}\mathbf{H}_{cal}$ and ${}^{tool}\mathbf{H}_{base}$. The step size of the noise amplitude was 0.15 mm in translation and 0.1° in rotation. With 20 different noise levels, we thus end up with a maximum noise amplitude of 3 mm and 2° for this evaluation. For each noise amplitude, we created 15 random robot poses, performed the calibration, and repeated the experiment 500 times.

Figure 2a shows the algebraic errors according to Eq. (6) of the linear approach presented in [24], the linear approximate method, the linear approximate method refined by Newton iterations, and the exact method. The errors of all methods increase linearly with the noise amplitude. The three methods presented in this paper are visually equivalent, while a slight improvement over the method presented in [24] is visible. Figure 2b shows the difference of the errors with respect to the errors of the exact method. The improvement over [24] is about 5%, independent of the noise level. The maximum deviation of the linear approximate method from the exact method was 4×10^{-8}. When applying the Newton iterations, the maximum deviation further decreased to 2×10^{-16}, which is in the order of the stopping criterion for the iteration. This shows that the approximate method refined by Newton is equivalent to the exact method and that the loss of accuracy that is induced by approximating the quadrics by planes is negligible for practical applications.

In the next experiment, we investigated the influence of noise on the accuracy of the resulting pose ${}^{base}\mathbf{H}_{cam}$. Figures 3a and 3b show the mean position and rotation errors for the linear approximate approach and for the nonlinear optimization. All errors again increase linearly with the noise amplitude. Furthermore, the advantage of the subsequent nonlinear optimization is clearly visible.

We also investigated the influence of the number of robot poses n on the accuracy of the calibration. For this, we kept the noise magnitude fixed at 0.4 mm and 0.1° and repeated the experiment 500 times. Figures 3c and 3d show the corresponding mean errors. It should be noted that the errors rapidly decrease for the first 10 images. Further increasing the number of poses only slightly improves the accuracy. The errors of the linear approximate approach decrease roughly with $O(1/n^{0.75})$, the errors of the nonlinear optimization roughly with $O(1/n^{0.8})$, which both is slightly better than the statistically expected value of $O(1/n^{0.5})$.

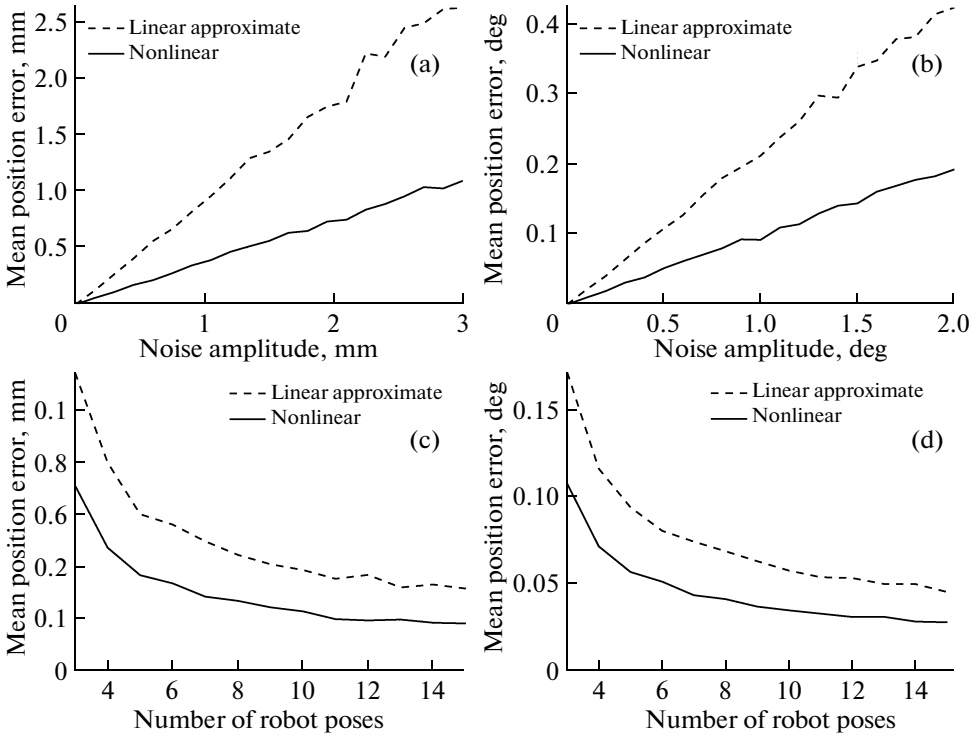

Fig. 3. Mean error in position (a) and rotation (b) of the hand-eye calibration for different noise levels and 15 random robot poses when using the linear approximate approach (dashed line) and when performing a subsequent nonlinear optimization (solid line); Mean error in position (c) and rotation (d) for different number of robot poses and a constant noise level.

For the experiments with real data, we attached a HALCON calibration plate to the tool of a DENSO robot HS-45452E/GM. A calibrated camera (IDS uEye UI-2240-M, 1/2", 1280×1024, $f = 16$ mm) was mounted stationary 0.9 m above the robot. At each of 12 calibration poses, we acquired an image of the calibration plate and determined ${}^{cam}\mathbf{H}_{cal}$ by using HALCON [21, chapter 3.9]. The hand-eye calibration using the linear approximate approach results in RMS/maximum errors of 0.24 mm/0.40 mm in translation and 0.06°/0.10° in rotation. The nonlinear optimization further decreases the errors to 0.20 mm/0.31 mm in translation and 0.05°/0.10° in rotation. Note that in contrast to the evaluations in [24], where error-free robot poses ${}^{tool}\mathbf{H}_{base}$ were assumed, we added noise also to the robot poses to obtain a more realistic behavior. Therefore, the absolute error values cannot be compared directly to the values in [24].

5. CONCLUSIONS AND OUTLOOK

In [4] a method for hand-eye calibration of articulated robots based on dual-quaternions is proposed, which shows advantages over methods that handle rotation and translation separately. In this paper, we extended the work of [4] to SCARA robots. To fix the additional degree of freedom in (10) we added a third constraint $t_z = 0$ to the unity constraints proposed in [4]. To solve for the unknowns, we described an exact solution and proposed a simple linear method that approximates the exact solution sufficiently well for practical applications.

To further improve the accuracy, we proposed a subsequent nonlinear optimization. We addressed several practical implementation issues and showed the effectiveness of the method by evaluating it on synthetic and real data. It was already shown by [4] that the dual-quaternion-based method is superior to the separate estimation of rotation and translation. Nevertheless, in the future we will extend our evaluation by comparing our dual-quaternion-based approach to the linear approach proposed in [27].

REFERENCES

1. H.H. Chen, "A Screw Motion Approach to Uniqueness Analysis of Head-Eye Geometry," in Computer Vision and Pattern Recognition, 145–151, 1991.
2. E.-W. Chionh, R.N. Goldman, J.R. Miller, "Using multivariate resultants to find the intersection of three quadratic surfaces," ACM Transactions on Graphics **10**(4), 378–400 (1991).
3. J.C.K. Chou, M. Kamel, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," International Journal of Robotics Research **10**(3), 240–254 (1991).

4. K. Daniilidis, "Hand-eye calibration using dual quaternions," International Journal of Robotics Research **18**(3), 286–298 (1999).
5. F. Dornaika, R. Horaud, "Simultaneous robot-world and hand-eye calibration." IEEE Transactions on Robotics and Automation **14**(4), 617–622 (1998).
6. B. Drost, S. Ilic, 3D Object Detection and Localization Using Multimodal Point Pair Featurs, in International Conference on 3D Imaging, Modelling, Processing, Visualization and Transmission (3DIMPTV), 9–16 (2012).
7. B. Drost, M. Ulrich, N. Navab, S. Ilic, Model Globally, "Match Locally: Efficient and Robust 3D Object Recognition," in Computer Vision and Pattern Recognition, 998–1005 (2010).
8. L.J. Everett, M.S. Burnett, "Experimentally Registering Position Sensors," in IEEE International Conference on Robotics and Automation, 641–646 (1996).
9. S. Hinterstoisser, S. Benhimane, N. Navab, "N3M: Natural 3D Markers for Real-Time Object Detection and Pose Estimation," in IEEE International Conference on Computer Vision, 1–7 (2007).
10. R. Horaud, F. Dornaika, "Hand-eye calibration," International Journal of Robotics Research **14**(3), 195–210 (1995).
11. ISO 8373:1994, "Manipulating industrial robots," Vocabulary, 1994.
12. ISO 9787:1999, "Manipulating industrial robots," Coordinate systems and motion nomenclatures, 1999.
13. A.E. Johnson, M. Hebert, "Using spin images for eficient object recognition in cluttered 3d scenes," IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (5), 433–449 (1999).
14. B. Kaiser, R.A. Tauro, H. Wörn, "Extrinsic calibration of a robot mounted 3d imaging sensor," International Journal of Intelligent Systems Technologies and Applications **5** (3/4), 374–379 (2008).
15. B. Kenwright, Dual-Quaternions From Classical Mechanics to Computer Graphics and Beyond, 2012. Accessed 2014-05-08, 11 pages. www.xbdev.net/misc_demos/demos/dual_quaternions_beyond/paper.pdf
16. S.D. Ma, "A self-calibration technique for active vision systems," IEEE Transactions on Robotics and Automation **12**(1), 114–120 (1996).
17. S. Rémy, M. Dhome, J.M. Lavest, N. Daucher, "Hand-Eye Calibration," in International Conference on Intelligent Robots and Systems, 1057–1065 (1997).
18. J. Richter-Gebert, Perspectives on Projective Geometry (Springer, Berlin Heidelberg, 2011).
19. J. Rooney, A comparison of representations of general spatial screw displacement. Environment and Planning B **5**(1), 45–88 (1978).
20. J. Schmidt, F. Vogt, H. Niemann, Robust Hand-Eye Calibration of an Endoscopic Surgery Robot Using Dual Quaternions, in Pattern Recognition. Lecture Notes in Computer Science (Springer, Berlin, 548–556 (2003).
21. C. Steger, M. Ulrich, C. Wiedemann, Machine Vision Algorithms and Applications (Wiley-VCH, Weinheim, 2007).
22. K.H. Strobl, G. Hirzinger, Optimal Hand-Eye Calibration,in IEEE/RSJ International Conference on Intelligent Robots and Systems, 4647–4653, 2006.
23. R.Y. Tsai, R.K. Lenz, A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. IEEE Transactions on Robotics and Automation **5**(3), 345–358 (1989).
24. M. Ulrich, A. Heider, C. Steger, Hand-Eye Calibration of SCARA Robots, in 9th Open German-Russian Workshop on Pattern Recognition and Image Understanding (OGRW 2014), Electronic on-site Proceedings, ed. by D.Paulus, C. Fuchs, D. Droege, 2014.
25. M. Ulrich, C. Wiedemann, C. Steger, "Combining scale-space and similarity-based aspect graphs for fast 3D object recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence 34(10), 1902–1914 (2012).
26. Z. Xu, X. Wang, X. Chen, J. Sun, "A robust algorithm for finding the real intersections of three quadratic surfaces," Computer Aided Geometric Design 22(6), 515–530 (2005).
27. H. Zhuang, "Hand/eye calibration for electronic assembly robots," IEEE Transactions on Robotics and Automation 14(4), 612–616 (1998).
28. H. Zhuang, Z.S. Roth, R. Sudhakar, "Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form AX = YB," IEEE Transactions on Robotics and Automation **10**(4), 549–554 (1994).

Markus Ulrich studied Geodesy and Remote Sensing at the Technische Universität München (TUM) and received his PhD from TUM in 2003. In the same year, he joined the Research and Development department at MVTec Software GmbH as a software engineer and became manager for Research and Development in 2008 heading the Research Team. Since 2005, Markus Ulrich is also a guest lecturer at TUM, where he teaches close-range photogrammetry. Since 2013, he is a guest lecturer at Karlsruhe Institute of Technology (KIT) as well, where he teaches machine vision.

Carsten Steger studied computer science at the Technische Universität München (TUM) and received the PhD degree from TUM in 1998. In 1996, he cofounded the company MVTec Software GmbH, where he heads the Research Department. He has authored and coauthored more than 80 scientific publications in the fields of computer and machine vision, including several textbooks on machine vision. In 2011, he was appointed a TUM adjunct professor for the field of computer vision. Since 2013, he is a member of the Technical Committee of the German Association for Pattern Recognition (DAGM).