Solving the Robot-World Hand-Eye(s) Calibration Problem with Iterative Methods

Amy Tabb · Khalil M. Ahmad Yousef

Received: 28 July 2016 / Accepted: 1 April 2017

Abstract Robot-world, hand-eye calibration is the problem of determining the transformation between the robot end-effector and a camera, as well as the transformation between the robot base and the world coordinate system. This relationship has been modeled as $AX = ZB$, where $X$ and $Z$ are unknown homogeneous transformation matrices. The successful execution of many robot manipulation tasks depends on determining these matrices accurately, and we are particularly interested in the use of calibration for use in vision tasks. In this work, we describe a collection of methods consisting of two cost function classes, three different parameterizations of rotation components, and separable versus simultaneous formulations. We explore the behavior of this collection of methods on real datasets and simulated datasets, and compare to seven other state-of-the-art methods. Our collection of methods return greater accuracy on many metrics as compared to the state-of-the-art. The collection of methods is extended to the problem of robot-world hand-multiple-eye calibration, and results are shown with two and three cameras mounted on the same robot.

1 Introduction

The robot-world hand-eye calibration problem consists of determining the homogeneous transformation matrices (HTMs) between the robot hand, or end-effector, to the camera, as well as the transformation of the robot base to the world coordinate system.

The preliminaries of the robot-world hand-eye calibration problem are as follows. Let the transformation from the hand coordinate frame to the camera coordinate frame be $^c Z_h$ or simply $Z_h$, and the transformation from the robot-base coordinate frame to the world coordinate frame be $^w X_b$ or simply $X$. The transformation
of the robot base frame to the hand coordinate frame is $hB_h$ or simply $B$, and is assumed to be known from the robot controller. Finally, the transformation from the world coordinate frame to the camera coordinate frame is represented by HTM $^OA_w$ or $A$. $A$ is calculated using a camera calibration procedure such as Zhang [17], where the world coordinate frame is defined by a calibration object in the workspace. The transformations are illustrated by Figure [1]. We note here that the labeling of the transformations in our version of the problem is different than that used in the traditional robot-world hand-eye calibration problem [18] in that some matrices are inverted ($A, B$) and the rest of matrices are exchanged ($X, Z$). Despite this difference, the linear relationship $AX = ZB$ is still the same in our interpretation of the problem versus others. We interpreted the problem differently because by doing so we are able to simplify the derivation of some of the cost functions that we use in Section 2.

Given these preliminaries, Zhuang et al. [18] formalized the robot-world hand-eye calibration explicitly as the homogeneous matrix equation $AX = ZB$, where all of the matrices are $4 \times 4$ matrices, and were the first to provide a method to find solutions for $X$ and $Z$. Many different positions of the robot and camera are used to generate multiple relationships $A_iX = B_iZ$, $i \in [0, n-1]$, where $n$ is the number of robot poses used for the calibration.

The robot-world hand-eye calibration problem is different, though related, to the hand-eye calibration problem, which was formulated as $AX = XB$ by Shiu and Ahmad [11]. In the hand-eye calibration problem, the matrices $A_i$ and $B_i$ are now considered as the relative transformations from one pose to another. Use of relative transformations is problematic as decisions must be made as to how to convert absolute transformations into relative ones. This work considers instead $A_i$ and $B_i$ as poses with respect to the world coordinate system and robot base coordinate system, respectively, and as a result robot-world hand-eye calibration is needed.

We structure the paper as follows. In the rest of Section 1, we provide an overview of existing approaches and recent work. In Subsection 1.2, we discuss our approach and contributions to the robot-world hand-eye calibration problem. Section 2 introduces two classes of proposed methods and extends those classes to the multiple camera case. Section 3 describes the experiments, metrics, and implementation details. In Section 5, the results on real and simulated datasets are shown and discussed. Finally, in Section 6, we present our conclusions.

1.1 Recent work

Most recent work on the robot-world hand-eye calibration problem makes use of the decomposition of $AX = ZB$ into a purely rotational part and a translational part, where $R_A$ represents a $3 \times 3$ rotation matrix, and $t_A$ a $3 \times 1$ translation vector as shown in Equation (1).

$$\begin{bmatrix} R_A & t_A \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R_X & t_X \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R_Z & t_Z \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R_B & t_B \\ 0^T & 1 \end{bmatrix}$$

Methods utilizing the decomposition of the problem that is shown in Equation (1) into two parts are called separable methods. Separable methods cause Equation (1) to produce two other equations: the rotation part as represented by Equation (2) and the translation part as represented by Equation (3).

$$R_AR_X = R_ZR_B$$

$$R_AT_X + t_A = R_Zt_B + t_Z$$

Since Equation (3) is linear in $t_X$ and $t_Z$ if $R_Z$ is known, the most frequent approach to estimate $X$ and $Z$ is to first find $R_X$ and $R_Z$ using Equation (2) and then use that solution and Equation (3) to find $t_X$ and $t_Z$.

Of those methods that separate the estimation of the rotation and translation parts using Equation (1) there are several different approaches. In Zhuang et al. [18], a linear solution was proposed for finding the unknowns $R_X$ and $R_Z$ by representing the rotation matrices as quaternions, and the translation components were found using linear least squares. Dornaika and Horaud in [3] gave a closed-form method for estimating the rotation components using quaternions, which did not require normalization like in [18]. Translation components were estimated with linear least squares as in [18]. Hirsh et al. in [4] proposed a separable, iterative approach in which the estimation of $X$ and $Z$ is alternated. The method consists of assuming that $Z$ is known and estimating $X$ by averaging $X_i = ZB_iA^{-1}$ for all $i$ to generate an estimate for $X$; this process is repeated in a similar way for $Z$ using the estimate found for $X$, and the estimation of $X$ and $Z$ is continued until termination conditions are met. Like the other methods in this group, estimation of rotation is separated from translation, and rotation is represented by quaternions. In the method of Shah [10], the Kronecker product and singular value decomposition were used to create a closed-form solution.

The second class of methods are typically called simultaneous solutions; they do not decouple rotation and translation error and thus have the advantage that error from the rotation estimation is not propagated to the translation estimation. In [3], in addition to a
closed-form separable solution, Dornaika and Horua present a formulation of the problem as a non-linear least squares problem. The rotation components are represented by matrices, so there are 18 parameters for rotation and 6 for translation. The cost function contains penalty terms that enforce orthonormality of the rotation matrices. Following Dornaika and Horua, Stobol and Hirzinger in 21 estimate the translation and rotation components through non-linear estimation techniques, though in their approach weights for the rotational and translational parts are chosen automatically and a position/orientation precision ratio parameter is required. The work of Li et al. 6 proposed a simultaneous solution that is found using dual quaternions and the Kronecker product.

Besides those approaches that deal with the equality $AX = ZB$ in separable and simultaneous versions, there have been some approaches within the context of the hand-eye calibration problem to refine the estimation of the camera calibration parameters. For example, in 5, Horua and Dornaika incorporated camera calibration parameters into a cost function for the robot-eye calibration problem $AX = XB$ and solved using the Levenberg-Marquardt method when the rotations are represented by quaternions. In this method, the requirement that the quaternion must have unit norm was enforced using penalty terms. Recently, Malti 5 in a work about hand-eye calibration, incorporated a variation of the robot-world hand-eye calibration formulation and used reprojection error together with epipolar constraints to simultaneously refine the camera intrinsic and distortion parameters in addition to the $X$ and $Z$ HTMs. Unlike Horua and Dornaika, 5, Malti in 8 uses Euler angles to represent rotation and as a result his method does not need penalty terms.

1.2. Our approach and contributions

Our work on the robot-world hand-eye calibration problem is motivated by three particular situations. The first is that we use a robot-mounted camera for multi-view high-quality reconstruction of complicated objects such as leafless trees as in Tabb 13; while some extrinsic camera calibration errors can be tolerated, the reconstruction outcomes improve when the camera calibration error is low. The second situation is where we use the same robot-mounted camera in laboratory or field conditions, where it can be the case that non-experts will be involved with performing calibration and acquiring data from a remote location. For this particular situation, our goal was to devise methods for robot-world hand-eye calibration that are not sensitive to the particular sequence of motions used during the calibration and is robust to non-ideal calibration situations. Finally, third, we have some situations in which multiple cameras are mounted on the same robot end-effector. While it is possible to only calibrate one camera using robot-world hand-eye calibration and then perform stereo camera calibration, we instead desired a way to calibrate all components at the same time so that error from one step is not propagated to the final calibration. This approach also allows cameras to have vastly different fields of view and does not require overlapping fields of view for any two cameras, which is more flexible than stereo calibration.

We note here that the robot-world hand-eye calibration problem has been discussed in the literature for more than two decades. Early contributions focused on linear and/or closed-form solutions because of computational efficiency. However, with the advent of open source nonlinear least square solvers, such as levmar 7 and Ceres 2, in addition to the general interest in bundle adjustment 10, iterative methods with previously unconsidered parameterizations of the rotational components offer advantages over the linear and closed-form approaches.

In this paper, we propose a collection of iterative methods for the robot-world hand-eye calibration problem. There are two classes of cost functions in this collection. The first class of cost functions minimizes the sum of squared difference between $AX$ and $ZB$ over $n$ positions of the robot as given in Equation 4 below, or in a closely-related version of it, where $F$ denotes the Frobenius norm. This class is discussed in Subsection 2.1

$$\sum_{i=0}^{n-1} ||AX_i - ZB_i||_F^2 = F$$

However, in the first class of cost functions as given in Equation 4 we observed that artifacts and errors related to the calibration object and camera calibration method are sometimes propagated to the estimation of $X$ and $Z$. The second class of cost functions aims to reduce the influence of these camera calibration artifacts by finding $X$ and $Z$ based on camera reprojection error, without explicitly using $A$ and this is discussed in Subsection 2.2.

Each of the aforementioned two classes of cost functions contains two or more sub-classes or methods, first, to explore different choices of cost functions; separable versus simultaneous solutions, and second, to explore different parameterizations of the rotation components. For the latter, we use three different possible parameterizations namely: Euler angles, axis-angle, and quaternions.

Since the two classes of cost functions are nonlinear, we use nonlinear least-square solvers based on the Levenberg-Marquardt method to find the approximate solutions of $X$ and $Z$. It is important to mention here
that the above collection of classes and methods easily extend to the problem of calibrating multiple cameras mounted on one robot as shall be discussed in Subsection 2.3.

Similarities between the state-of-the-art and our proposed collection of methods are as follows. Our methods are iterative, like the method of Hirsh et al. [1] and the iterative method of Dornaika and Horaud [3]. The work of Malti [8] concerning using camera reprojection error to estimate HTMs $X$ and $Z$ is similar to our second class of methods, when using the Euler angle representation for rotations (Subsection 2.2). One difference is that Malti’s work assumes that the motions are relative and as a result uses different definitions of the matrices than we do.

Our contributions to the state-of-the-art in robot-world hand-eye calibration are summarized as follows:

- We provide a comprehensive comparison that explores different choices of cost function, parameter choices, and separable versus simultaneous solutions within our collection of methods, and contrast those methods with the state-of-the-art methods, on real datasets as well as on simulated datasets.
- Our collection of methods is provided to the community as open source code [14].

A portion of the work presented in this paper was previously published as [15]. However, the present paper expands the work of [15] in the following ways:

- This paper evaluates three different parameterizations of the rotation matrices in the experimental results: Euler angles, the axis-angle representation, and quaternions. Also, it discusses the effect of the parameter choice on results. In contrast, the prior work only used Euler angle parameterizations.
- While in [15] only simultaneous methods were proposed, we also created separable formulations of some of the cost functions and as a result were able to demonstrate the differences on the results produced by separability.
- We incorporated a comparison with the work of Shah [10], which was not presented in [15].
- Prior work only considered one camera. In contrast, the methods presented in this paper are generalized to the multiple-eye, one robot problem, with no limit to the number of cameras used.
- Four additional real datasets were acquired.
- Simulated datasets were added to the experiments, which allow comparisons between the estimated and true solutions and additional analysis of the methods.

2 Collection of methods description

Many of the differences among the previous methods concern how orthonormality of the rotation matrices is maintained. In this section, we first describe the parameterizations of the rotational components of $X$ and $Z$ such that orthonormality is preserved, and then discuss in details the two proposed classes of cost functions that we briefly introduced in the previous section followed by the extension to multiple cameras.

We explore three different parameterizations: Euler angle, axis-angle, and quaternions. Since these rotations are commonplace in the robotics literature and practice, in the next paragraph, we will only give a brief sketch of some details relevant to such parameterizations.

The Euler angle rotation representation is composed of three variables, which represent the rotation angles or direction cosines along the $x$, $y$, $z$ directions. In this work, we selected the directional order of the rotations in the following order: $x$, $y$, $z$. Consequently, a rotation matrix can be represented as in the following manner $R = R_Z R_Y R_X$. The axis-angle representation is another three-variable representation, where a three-element vector $v$ represents the axis of rotation, and $|v|$ is the angle of rotation about the $v$ axis such that a corresponding rotation matrix is generated with Rodrigues’ rotation formula. Finally, quaternions are a 4-element representation, where a unit vector $q$ can be converted to an orthonormal matrix.

We now define some notation to help us describe our two classes of cost functions and their extension to multiple cameras. Let the selected rotation representation be represented by a vector $p$, with the size of $p$ equal to 3 for the Euler angles and the axis-angle representations, and equal to 4 for the quaternion representation. Then the directed cosine matrix representation of $p$ is $R(p)$.

With these preliminaries, we will now present our first class of cost functions.

2.1 First class: $AX = ZB$

The first class of cost functions uses the world to camera transformations $A_i$ and base to end-effector transformations $B_i$ to estimate the HTMs $X$ and $Z$. This is achieved by seeking the minimum of $c_1$ as given below:

$$c_1 = \sum_{i=0}^{n-1} ||A_i X - ZB_i||_F^2$$

(5)

In the cost function $c_1$ in Equation 5, if we substitute-in the rotation representation $R(p)$ and the translation vectors for the unknowns $X$ and $Z$, the minimization
A solution to the problem in Equation (6) belongs to the simultaneous category of methods for robot-world hand-eye calibration, because the rotation and translation are solved for at the same time.

We can also break the minimization problem in Equation (6) into two parts via a separable formulation: one part for rotation and the other part to estimate the translation components as captured by Equation (7) and Equation (8), respectively.

An approximate solution to Equation (7) is first found and then the translation components are found from Equation (8).

The left hand (AX) and right hand (ZB) sides of the cost function $c_1$ represent the transformation from robot base to camera via the world coordinate system (left hand side) and from robot base to camera via the end-effector coordinate system (right hand side) as graphically captured in Figure 1. We also explore the use of a slightly different cost function $c_2$ as given in Equation (9).

In this subsection, we present our second class of cost functions and methods, which are based on the idea of minimizing camera reprojection error in order to find $X$ and $Z$. The approach presented in this subsection shares many similarities with the camera calibration approach of Zhang in [17], where the extrinsic camera calibration parameters are estimated by minimizing the camera reprojection error. Unlike the first class of methods, this approach is sensitive to the choice of initial solution; briefly, we use the result from the first class of methods as an initial solution and a more in-depth discussion of initial solution choices can be found in Subsection 3.3.

Before getting into the details of this class of methods, we first mention some preliminaries and notation. Let a three-dimensional point on the calibration object be $\mathbf{X}$, and because there are multiple $m$ such points we give a subscript $j \in [0, m - 1]$. $\mathbf{X}_j$. It is assumed that all of the points are detected in the images used for calibration, so in the case of the chessboard pattern, these points are the corners of the chessboard. When $\mathbf{X}_j$ is projected to the image from robot position $i$ using the intrinsic camera calibration parameters, we have image point $\mathbf{X}_{ij}$ and a corresponding original image point $\mathbf{x}_{ij}$. We can represent $\mathbf{x}_{ij}$ in the context of the robot-world hand-eye calibration problem as follows:

$$\mathbf{x}_{ij} = f(k, [\mathbf{ZB}X^{-1}]_{3 \times 4} \mathbf{X}_j)$$

where $k$ is a vector containing the intrinsic camera calibration parameters. The bracket notation $[\mathbf{ZB}X^{-1}]_{3 \times 4}$ is used to denote the upper $3 \times 4$ sub-matrix of what is
inside the bracket. $f()$ is the function that transforms $[\mathbf{ZB}, \mathbf{X}^{-1}]_{3 \times 4}$ into image points using $\mathbf{k}$.

Concerning $\mathbf{k}$, we use 4 parameters from the intrinsic camera calibration matrix and 8 radial and tangential distortion parameters, so $|\mathbf{k}| = 12$. However, other radial and tangential distortion models may be chosen with no change to the method. As with $c_2$ in Equation 10 to simplify the representation of Equation 13 we substitute for $\mathbf{X}^{-1}$ another matrix $\mathbf{X}$, which is composed of an orthonormal rotation matrix and translation vector.

Given the preliminaries discussed above, the reprojection sum of squares error ($\text{rsse}$) is:

$$\text{rsse} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} ||\mathbf{X}_{ij} - \hat{\mathbf{X}}_{ij}||^2$$

(14)

and by substituting $\hat{\mathbf{X}}_{ij}$ we have:

$$\text{rsse} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} ||\mathbf{X}_{ij} - f(\mathbf{k}, [\mathbf{ZB}, \mathbf{X}]_{3 \times 4})||^2$$

(15)

We note that in Equations 14 and 15 and for the remainder of this paper, we use the L2 norm for vectors.

Finally, if we substitute in the rotation representation $\mathbf{R}(\mathbf{p})$ and the translation vectors for the unknown $\mathbf{X}$ and $\mathbf{Z}$ matrices, the camera reprojection sum of squares error ($\text{rsse}$) minimization problem becomes as given in Equation 16 which we refer to as $r_p1$ method:

$$\arg\min_{\mathbf{X}, \mathbf{p}_X, \mathbf{t}_X} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} ||\mathbf{X}_{ij} - f(\mathbf{k}, [\mathbf{ZB}, \mathbf{X}]_{3 \times 4})||^2$$

(16)

where $\mathbf{p}_X$ and $\mathbf{t}_X$ are again the rotation parameter vector and translation vectors of $\mathbf{X}$ or $\mathbf{X}^{-1}$, respectively.

Within the simultaneous formulation of the robot-world hand-eye calibration problem as given in Equation 16 it is also possible to refine the camera intrinsic parameter vector $\mathbf{k}$ by letting $\mathbf{k}$ be parameters to be estimated in order to produce a local minimum in Equation 16 instead of constants as given in Equation 17 which we refer to as the $r_p2$ method.

$$\arg\min_{\mathbf{X}, \mathbf{p}_X, \mathbf{t}_X, \mathbf{k}} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} ||\mathbf{X}_{ij} - f(\mathbf{k}, [\mathbf{ZB}, \mathbf{X}]_{3 \times 4})||^2$$

(17)

In contrast with the minimized reprojection sum of squared error ($\text{rsse}$) in both $r_p1$ and $r_p2$ methods, it is important to mention that in the camera calibration literature, the reprojection root mean squared error ($\text{rrmse}$) is more common, as shown in Equation 18. The $\text{rrmse}$ represents the average Euclidean distance between detected and reprojected calibration pattern points in the image plane, and its units are pixels. However, since the parameters that result in a minimum of $\text{rsse}$ also result in a minimum of $\text{rrmse}$, $\text{rsse}$ is used as a cost function in this work.

$$\text{rrmse} = \sqrt{\frac{1}{mn} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} ||\mathbf{X}_{ij} - \hat{\mathbf{X}}_{ij}||^2}$$

(18)

2.3 Extending the two proposed classes of methods to multiple cameras

The multiple camera case requires estimating one HTM $\mathbf{X}$ and multiple HTMs $\mathbf{Z}$ as such, if we let the number of cameras be $q$, and $\mathbf{A}_i,0$ be the transformation from the world coordinate frame to the $0^{th}$ camera at the $i^{th}$ robot position. Then the relationship between the matrices in the context of robot-world hand-multiple-eyes calibration problem would be formulated as:

$$\mathbf{A}_i,0 \mathbf{X} = \mathbf{Z}_0 \mathbf{B}$$

(19)

$$\mathbf{A}_i,1 \mathbf{X} = \mathbf{Z}_1 \mathbf{B}$$

(20)

$$\vdots$$

$$\mathbf{A}_i,q-1 \mathbf{X} = \mathbf{Z}_{q-1} \mathbf{B}$$

(21)

for all $i \in [0, n-1]$. Therefore, we should now be able to represent all of the cost functions presented thus far in Subsections 2.1 and 2.2 within this multiple camera context. For instance, Equation 6 in the multiple camera context becomes:

$$\arg\min_{\mathbf{X}, \mathbf{p}_X, \mathbf{p}_Z, \mathbf{t}_Z, \mathbf{k}} \sum_{d=0}^{n-1} ||\mathbf{A}_i,d \begin{bmatrix} \mathbf{R}(\mathbf{p}_X) \mathbf{t}_X \\ \mathbf{0}_T \end{bmatrix} - \begin{bmatrix} \mathbf{R}(\mathbf{p}_Z,d) \mathbf{t}_Z,d \\ \mathbf{0}_T \end{bmatrix} \mathbf{B}_r ||^2$$

(22)

where $\mathbf{p}_Z,d$ and $\mathbf{t}_Z,d$ are the rotation parameterization and translation vectors for the $d^{th}$ HTM, $\mathbf{Z}_d$. Separable versions follow using the same pattern, as it is done in the cost function $c_1$ and the $\text{rsse}$ formulations of the two classes of methods found in the previous subsections.

When using multiple cameras, it is common that in order to gain a wide variety of views of the calibration object, not all images for each camera will view the calibration object, particularly if the cameras are far apart or have different lenses and imaging sensor sizes. We now describe a generic method that we used
in our implementation, to weight the influences from each camera in the cost function. We desired that each camera have an equal influence, but this weighting can be adjusted according to other needs.

We let the set of robot positions where a camera \( d \) can view the calibration object be \( S_d \), and denote \( \min_d \) as the minimum size of \( S_d \) determined for all cameras. Then, an individual weight \( w_d \) for camera \( d \) can be set as \( \frac{1}{|S_d|} \). Furthermore, if we let the cost function that is to be minimized be \( g() \), the formulation of the robot-world hand-multiple-eyes calibration problem such that missing observations are handled is given by Equation 23.

\[
\arg\min_{p_X, p_Z, d} \sum_{d=0}^{q-1} w_d g(p_X, t_X, p_Z, d, t_Z, d) \tag{23}
\]

The problem in Equation 23 can be applied to the first class of cost functions, as well as the \( rpl \) method in the second class of methods. For the \( rp2 \) method in the second class of cost functions, we also estimate intrinsic camera calibration parameters \( k \) for all \( q \) cameras and doing so follows from Equation 23.

3 Performance Evaluation on real datasets

The behavior of the robot-world hand-eye calibration methods is demonstrated on eight datasets in real laboratory and field settings. These datasets represent different combinations of robots, cameras, and lenses; some of the datasets have multiple cameras. Descriptions of the datasets are given in Table 1. The table also lists the number of robot positions (\( n \)) used for the calibration and the \( rrmse \) error (in pixels) from the camera calibration step.

![Image](image-url)

**Fig. 2** An example of the experimental setup for dataset 7, consisting of three cameras and the images from each camera for one stop of the robot.

| Dataset | Image Size | Lens focal length | Robot | \( rrmse \) (pixels) |
|---------|------------|-------------------|-------|--------------------|
| 1       | (1228 × 1024) | 6 mm              | Denso VS-6577GM-B | 0.231813 |
| 2       | (1228 × 1024) | 8 mm              | Denso VS-6577GM-B | 0.124774 |
| 3       | (1228 × 1024) | 6 mm              | Denso VS-6577GM-B | 0.185242  |
| 4       | (1228 × 1024) | 8 mm              | Denso VS-6577GM-B | 0.540056  |
| 5       | (1228 × 1024) | 6 mm              | Denso VS-6577GM-B | 0.044746  |
| 6       | (1228 × 1024) | 8 mm              | Denso VS-6577GM-B | 0.131076  |
| 7       | (1228 × 1024) | 6 mm              | Denso VS-6577GM-B | 0.143151  |
| 8       | (1228 × 1024) | 8 mm              | Denso VS-6577GM-B | 0.139296  |

For the two proposed classes of cost functions, we use the implementation of the Levenberg-Marquardt method found in the software package Ceres 2. All of the results shown in this paper were generated on a workstation with one 12-core processor and 192 GB of RAM. The camera calibration was carried out using the OpenCV library’s camera calibration functions.

For the first class of cost functions using simultaneous and separable versions of \( c_1 \) and \( c_2 \), the initial

\[ c_1 = 
\]

\[ c_2 = 
\]

\[ \text{argmin}_{p_X, q} \sum_{d=0}^{q-1} w_d g(p_X, t_X, p_Z, d, t_Z, d) \tag{23}
\]
solutions of \( p_X \) and \( p_Z \) are set such that the corresponding directed cosine matrix is a 3 \( \times \) 3 identity matrix, and the translation components are set to zero.\(^2\) As mentioned previously, the second class of cost functions \( r_{p1} \) and \( r_{p2} \) are sensitive to initial solutions. To find an approximate solution to the minimization problem of \( r_{p1} \) method that was shown in Equation 16, we used the simultaneous solution using the \( c_2 \) cost function (problem specified in Equation 10) as an initial solution. To find an approximate solution to the problem of the \( r_{p2} \) method shown in Equation 17, the solution from Equation 16 was used as the initial solution.

To present a comparative study against our collection of cost functions and methods, we implemented seven comparison methods: Zhuang et al. [18], Dornaika and Horaud [3]; closed form and iterative, Hirsh et al. [4] and Li et al. [6] based on the dual quaternion and Kronecker product, and the work of Shah [10]. Concerning the implementation details of Li et al. [6] methods, we draw the reader’s attention that the computation of coefficients \( \lambda_1 \) and \( \lambda_2 \) in the dual quaternion method was not specified clearly by the authors, thus we used a Levenberg-Marquardt method to find those values. For the conversion of rotation matrices found by the Kronecker product method [6] into orthonormal matrices, we used the singular value decomposition method.

3.2 Error metrics

In presenting the comparison results of both of our collection of methods and those methods of others, we used the following error metrics: two types of mean rotation error, the mean translation error, the mean combined rotation and translation error, and the reprojection root mean squared error. Since we are interested in using our methods for reconstruction and vision tasks as indicated in Subsection 1.2, we also introduced a sixth metric related to reconstruction accuracy. The following subsections describe each error type.

3.2.1 The Mean Rotation Error

We list two rotation errors. The first, which we denote Rotation Error 1 \( (e_{R1}) \), is derived from Equation 2 and shown in Equation 24

\[
e_{R1} = \frac{1}{n} \sum_{i=0}^{n-1} ||R_{A_i}R_X - R_{Z_i}R_{B_i}||_F^2
\]  

However, the \( e_{R1} \) value may not be particularly meaningful when evaluating the methods, and has no units. For this reason, we have a second rotation error, \( e_{R2} \), that is representative of the difference between the left hand and right hand sides of Equation 2. We find the relative rotation between the two sides (Equation 25) and then compute the angle of the relative rotation using the axis-angle representation, which is represented as \( \text{angle}(\cdot) \) in Equation 26 in the results shown in Section 5 its units are degrees.

\[
R_i = (R_{Z_i}R_{B_i})^T(R_{A_i}R_X)
\]  

\[
e_{R2} = \frac{1}{n} \sum_{i=0}^{n-1} \text{angle}(R_i)
\]
3.2.2 The Mean Translation Error

The translation error is derived from Equation 3 and shown in Equation 27, where the units are in millimeters to be consistent with our selection of units for camera calibration error.

\[ e_t = \frac{1}{n} \sum_{i} \left\| (R_{A_i} t_X + t_{A_i}) - (R_{Z_i} t_B + t_{Z_i}) \right\|^2 \]  (27)

3.2.3 The Mean Combined Rotation and Translation Error

This error is shown in Equation 28 and has no units.

\[ e_C = \frac{1}{n} \sum_{i} \left\| A_i X - Z B_i \right\|^2 \]  (28)

3.2.4 The Reprojection Root Mean Squared Error (rrmse)

This error is shown in Equation 29 and its units are pixels.

\[ rrmse = \sqrt{\frac{1}{mn} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \left\| \hat{X}_{ij} - f(k_i[ZB_iX^{-1}]_{3 \times 4} \hat{Y}_j) \right\|^2} \]  (29)

3.2.5 The Reconstruction Accuracy Error

Since we are also interested in reconstruction accuracy when using a robot to acquire images with associated camera calibration information, our final metric aims to represent the reconstruction accuracy.

The idea behind this metric is given correspondences from different images acquired at different robot positions, estimate the world points that generated those correspondences, and compute the difference between the estimated versus true world points to represent reconstruction accuracy. We borrow some of the notation from Section 2.2; \( \hat{X}_{ij} \) is the jth image point from the ith robot position and corresponds to the three-dimensional point \( \hat{X}_{j} \). First, we estimate the most likely three-dimensional point (represented by \( \hat{Y}_j \)) that generated the n \( \hat{X}_{ij} \) image points by solving the minimization problem in Equation 30

\[ \hat{Y}_j = \arg\min_{Y_j} \sum_{i=0}^{n-1} \left\| \hat{X}_{ij} - f(k_i[ZB_iX^{-1}]_{3 \times 4} Y_j) \right\|^2 \]  (30)

Then, the reconstruction accuracy error (rae) is the average Euclidean distance between the estimated \( \hat{Y}_j \) points and calibration object points \( \hat{X}_{j} \):

\[ rae = \frac{1}{m} \sum_{j=0}^{m-1} ||\hat{Y}_j - \hat{X}_{j}|| \]  (31)

4 Performance Evaluation on simulated datasets

The comparison methods as well as a subset of our proposed methods are evaluated on simulated datasets. For the simulated datasets contained in this paper, only \( A_i \), \( B_i \), \( X \), and \( Z \) are generated. As a result, we do not evaluate our proposed second class of methods, since doing so would also require simulating camera models. The protocol for generating the simulated datasets closely follows that of Shah 2013 \[10\], Section 5.1. Briefly, the rotation components of \( A_i \), \( X \), and \( Z \) are set as random rotation matrices, and the translation components are drawn from the standard uniform distribution \((0,1)\) using a random number generator, for \( i = 1,2,\ldots,25 \). Then, the homogeneous matrix \( B_i \) is computed as:

\[ B_i = Z^{-1} A_i X \]  (32)

Noise is then introduced to \( B_i \), only in the rotation component, by converting the rotation matrix to the quaternion representation and adding random noise, and then converting back to the matrix representation. \( \eta \) is the parameter controlling the magnitude of the random noise added to \( B_i \). \( \eta \in (0,0.25) \) and is evenly spaced over 19 values along that interval. Data was generated for ten trials of the experiment.

The protocol above sets the translation components within the interval of \((0,1)\), which reflects a bias towards a particular use of units. Depending on the particular experiment setup and choices made, this interval may or may not be representative of that experimental setup. Consequently, we denote the simulated datasets using Shah’s protocol \[10\] as Simulated Dataset I. We generated another dataset using the same protocol, except that the translation components were within the interval of \((0,1000)\), and refer to this as Simulated Dataset II. We chose the interval \((0,1000)\) since in our real experiments we use millimeters for translation components to be consistent with camera calibration; of course, the use of meters would be another choice and this choice is reflected in Simulated Dataset I.
4.1 Error metrics for simulated datasets

Since the ground truth $X$ and $Z$ are known for the simulated datasets, for each calibration method the difference between the rotation and translation components are computed. Again, we follow closely Shah’s presentation \[10\] in our description of the error metrics for the simulated experiments.

4.1.1 Rotation error

Let $\hat{R}_X$ be the rotation matrix estimated by a calibration method for a simulated dataset, and $R_X$ be the ground truth rotation matrix for $X$. Then, the error in the rotation component is $e_{RX}$:

$$ e_{RX} = \|\hat{R}_X - R_X\|_F $$

(33)

and the same formula follows to compute the rotation error for $Z$, $e_{RZ}$.

4.1.2 Translation error

Let $\hat{t}_X$ be the translation vector estimated by a calibration method for a simulated dataset, and $t_X$ be the ground truth translation vector for $X$. Then, the error in the translation component is $e_{tX}$:

$$ e_{tX} = \|\hat{t}_X - t_X\| $$

(34)

and the same formula follows to compute the translation error for $Z$, $e_{tZ}$.

5 Experimental Results

In this section, we first show and discuss comparison results between our methods and the seven methods we referred to in the previous section on real datasets. Then, we show and discuss comparison results using the two simulated datasets.

Tables 2 through 9 show the results using the five error metrics described in Subsection 4.2 and the eight real datasets described in Table 1. These comparison results also include the running time for our implementation of each method in seconds.

In Tables 2-9, the first seven rows correspond to the seven comparison methods from the literature, listed in chronological order: Zhuang et al. \[18\], Dornaika and Horaud \[3\]: closed form and iterative, Hirsh et al. \[4\] and Li et al. \[2\] based on the dual quaternion and Kronecker product, and the method of Shah \[10\], whereas the remainder of the rows correspond to our proposed methods. The comparison methods were not evaluated on Datasets 6, 7, and 8 and are not shown in Tables 7-9 because those datasets have more than one camera, and to the best of our knowledge, there are currently no other robot-world hand-eye calibration methods for multiple cameras.

Results for Simulated Datasets I and II are shown graphically in Figures 4-7. The seven comparison methods were evaluated as well as the first class from our proposed collection of methods. Within the first class of our proposed collection of methods, the results were very similar for different choices of rotation parameterization. Consequently, we only listed one rotation parameterization choice, that of Euler angles, to allow better readability.

5.1 Discussion of comparison methods on real datasets

We mention here that the Dornaika and Horaud iterative method \[3\] does not converge for any datasets using the Ceres solver \[2\], where derivatives are automatically computed by the software. In comparison, in our previous work \[15\], where we used the levmar solver \[7\], the Dornaika and Horaud iterative method converged after only 2-3 iterations after termination conditions were met, however, the results were very similar to that produced with the authors’ closed form method (which served as the iterative method’s initial solution). In summary, the behavior of different solvers may give different results, but the main conclusion we have made about this method is that the large penalty terms used to enforce orthonormality of the rotation matrices result in either very little change from the initial solution, or non-convergent behavior.

Another similarity among the comparison methods is the Hirsh et al. \[4\] and Shah \[10\] methods. Both of these methods consistently produce the lowest values of rotation errors $e_{R1}$ and $e_{R2}$ that are usually the same up to 5 digits of precision. In contrast, the Dornaika and Horaud closed-form method \[3\] usually has one of the highest rotation errors. With regard to the mean translation error ($e_t$) and the combined rotation and translation error ($e_C$), Shah’s method in general has a good performance relative to other comparisons methods as it results in one of the lowest values among the datasets until Dataset 5 (Table 6). With respect to the reprojection mean square error ($\text{rrmse}$), the Hirsh et al. \[4\] and Shah \[10\] methods both perform generally

---

3 While use of tables is perhaps not the easiest for the reader, we note that the huge range of values for each of the six error metrics made other types of representation (graphs, etc.) infeasible.

4 Only the first author is mentioned in the tables’ text for brevity.
the best, with Shah’s method resulting in lower \( \text{rrmse} \) in general as compared to the method of Hirsh et al., though this relationship is inverted in Datasets 1 and 5. For example, the method of Hirsh et al. has the lowest \( \text{rrmse} \) in Datasets 1 (3.68829 pixels), but large \( \text{rrmse} \) in Dataset 5 (238.041 pixels), though still lower compared to Shah’s method’s \( \text{rrmse} \) value (439.512 pixels).

The first conclusion drawn from Tables 2–9 which correspond to datasets with a single camera, is that the behavior of many methods is dataset dependent. This dataset dependency may result from many factors, including the positions chosen for calibration, the robot model, and the way in which the robot is mounted, which may introduce new errors. Recall from Section 3 that the DENSO VM-60BIG from Datasets 3 and 4 is mounted on a mobile unit with tires, so as the robot moves, there might be some movement in the robot base.

The dataset dependency of the comparison methods is illustrated with a selection of examples. For instance, the combined rotation and translation error \( (e_C) \) of Zhuang et al. [18] is the lowest of the group of comparison methods in Dataset 1 (233.3), in the middle range in Dataset 2, and greater than 170,000 in Datasets 3, 4, and 5. The Kronecker product method from Li et al. [6] performed well relative to the comparison methods for Dataset 5 (Table 6), where the combined rotation and translation error was \( e_C = 329.24 \) and the other methods had combined rotation and translation error \( e_C \geq 13,000 \). However, for no other single dataset that has one camera does the Li et al. Kronecker product method have the lowest combined rotation and translation error relative to the other comparison methods.

5.2 Discussion of the proposed collection of methods on real datasets

5.2.1 First class of cost functions

Within the first class of cost functions, the methods based on minimizing \( c_1 \) (i.e. simultaneous and separable under different choices of parameterization of the rotational components) tend to have a lower value of combined rotation and translation error \( (e_C) \), while methods based on minimizing \( c_2 \) have lower values of \( \text{rrmse} \) as compared to \( c_1 \) methods. Exceptions are Datasets 7 and 8, which have very similar values of \( e_C \) and \( \text{rrmse} \) for \( c_1 \) and \( c_2 \) separable versions (see Tables 8 and 9).

Specifically, within the first class of cost functions, the separable versions of the methods tended to have lower rotation errors \( e_{R1}, e_{R2} \), but higher translation errors \( e_t \) and combined rotation and translation error \( e_C \), as well as reprojection error \( \text{rrmse} \). This is particularly obvious in Dataset 5 (Table 6), where the reprojection root mean square errors of the separable versions are all greater than 48 pixels, while the simultaneous versions have a maximum \( \text{rrmse} \) of 3.89 pixels. The separable methods have a much greater \( \text{rrmse} \) than the simultaneous versions in particular in Dataset 6, Table 7.

5.2.2 Second class of cost functions

As expected when using reprojection error as a cost function versus the relationship \( AX = ZB \) (and its variations), the second class of methods consistently gives results with lower values of \( \text{rrmse} \) than the first class of methods or the comparison methods. On the other hand, the rotation errors, translation errors and combined rotation and translation errors, when using the second class of methods, are greater than the first class of methods as well as some of the comparison methods. In general, the change in \( \text{rrmse} \) using \( r_{p1} \) (where intrinsic camera calibration parameters are treated as constants) versus \( r_{p2} \) (where intrinsic camera calibration values are parameters) is small.

5.2.3 Effect of rotation representation

It seems that the choice of rotation representation – Euler angles, axis-angle, or quaternion – has only a small effect on the results, and those differences may be dependent on the particular solver used. For instance, in our prior work [15], the simultaneous version of the cost function \( c_1 \) was used based on the Euler angles parameterization (in that work, labeled as Euler Parameterization I) using the \( \text{levmar} \) solver [14] and a provided Jacobian matrix. In that work, when tested on Dataset 1, the \( \text{rrmse} \) was 3.62709 pixels. Whereas using the \( \text{Ceres} \) solver [2] with automatic Jacobian matrices for the same dataset, the \( \text{rrmse} \) values were 12.13, 3.62686 and 3.62641 pixels, respectively for the Euler angles, axis-angle and quaternion versions (Table 2). This difference in \( \text{rrmse} \) occurs even in the cases when the combined rotation and translation error \( (e_C) \) between the three rotation representations, using \( \text{Ceres} \), differs by only 0.001. Usually, the differences between the rotation representations on all of the metrics are relatively small, though exceptions are Datasets 5 and 6 (Tables 6 and 7), the separable \( c_1 \) and \( c_2 \) methods.

5.3 Reconstruction accuracy performance for the proposed collection of methods and comparison methods

While reprojection root mean square error is a good indicator of relative reconstruction accuracy error \( (\text{rae}) \),
there is not a monotonic relationship of $rrmse$ to $rae$. We observed that the $rp_2$ version from the second class in our collections returns slightly lower values of $rrmse$ than $rp_1$. Concerning reconstruction accuracy error of the $rp_2$ method versus the $rp_1$ method, $rae$ increases slightly in datasets with better selections of robot positions (Datasets 1, 2, 3, 4, 7, 8) using $rp_1$ and decreases slightly in datasets with poorer selections of robot positions (Datasets 5 and 6) when using $rp_2$ method as compared to the $rp_1$ method. Concerning our first class of cost functions ($c_1$ and $c_2$), we observed that using the $c_2$ simultaneous method always gives a lower $rae$. In particular, the $c_2$ simultaneous version in Dataset 2 has $rae \leq 0.1$ mm for all three rotation representations, and this pattern ($rae \leq 0.1$ mm) is repeated in the separable versions of $c_2$ as compared to $c_1$ ($rae \leq 15$ mm, $rae \leq 2.38$ mm for separable and simultaneous, respectively). Additionally, it is observed that the separable versions of both $c_1$ and $c_2$ perform well in some datasets with respect to $rae$ and worse in others (e.g. perform well in Dataset 2 ($rae \leq 2.381$ mm), and perform poorly in Dataset 5 ($rae \geq 33.338.9$ mm)).

Within the group of comparison methods, the methods of Hirsh et al. [4] and Shah [10] consistently have the lowest values of rotation error, and both of these methods perform generally well with respect to $rrmse$. In general, Shah’s computation of translation vectors results in lower translation error ($e_t$) and $rrmse$ as compared to the method of Hirsh et al., though this relationship is inverted in Dataset 5. However, none of the comparison methods perform as well as our proposed $rp_1$ and $rp_2$ methods with respect to reconstruction accuracy error, though for some datasets the methods of Hirsh et al. and Shah are comparable to the $rae$ of our first class of proposed methods.

5.4 Discussion of the simulated experiments

The examination of the Simulated Dataset I and II results offer some additional insights into the behavior of the comparison and proposed methods, since the ground truth $X$ and $Z$ are known. The Simulated Dataset I results for the rotation error for $X$ and $Z$, show that the Shah and Li et al. Kronecker product methods consistently producing the rotation matrices closest to the ground truth, while the rest of the methods have very similar error values ranging from 2 and 2.6. The Shah and Li et al. Kronecker product methods also have the lowest translation error for $X$ and $Z$ in Simulated Dataset I, while the rest of the methods have similar error values. One exception is the Li et al. dual quaternion method, which has the largest error than the rest of the methods for the translation components of $X$ and $Z$.

Simulated Dataset II is different from Simulated Dataset I in the scaling of the translation components, and this difference affects the performance of the methods. Concerning rotation error, as before the Shah and Li et al. Kronecker product methods produce the lowest values, but in Simulated Dataset II the simultaneous versions from our first class of proposed methods also produce comparable values. For translation error, the Li et al. Kronecker product and simultaneous versions from our first class of proposed methods produce very low error values ($\leq 0.061$), while the remainder have values which vary considerably.

From these experiments, it is clear that some methods are sensitive to the scaling and distribution of the translation components, and a different method for dataset generation may yield different observations. The Li et al. Kronecker product method seems to be a robust choice in both of these settings, the Shah method for the Simulated Dataset I setting, and our first class of proposed methods for the Simulated Dataset II setting.

5.5 Recommendations and Summary

When choosing a robot-world, hand-eye calibration method to use, we hope that our experiments can aid researchers choose the method best suited to a particular task or application. For instance, the methods of Hirsh et al. [4] and Shah [10], and the $c_1$ and $c_2$ separable versions from the first class of our proposed methods consistently give the lowest rotation errors, whereas the $c_1$ simultaneous method returns the lowest combined rotation and translation error $e_C$. For the best $rrmse$ and $rae$ without using camera reprojection error in the cost function, the $c_2$ simultaneous method is the best choice. When using camera reprojection error as the cost function, $rp_1$ is generally better on $rae$ with a higher quality camera calibration and $rp_2$ is better than $rp_1$ with a poorer quality intrinsic camera calibration. Within the collection of methods we propose, the choice of rotation representation does not greatly affect the results when using a modern solver such as Ceres [2].

6 Conclusions

We presented a collection of methods for robot-world hand-eye calibration (including the case of multiple cameras) with accompanying code. Within this collection, we explored cost functions related to the equality $AX = ZB$ (including simultaneous and separable versions) and camera reprojection error. We also implemented these
Table 2: Comparison of methods using the error metrics described in Section 3.2 for Dataset 1.

| Method                  | Time (s)  | $r_{R1}$ | $r_{R2}$ (degrees) | $c_{e}$ (mm) | $c_{e}$ | $r_{Rmse}$ (pixels) | $r_{Rmse}$ (mm) |
|-------------------------|-----------|----------|--------------------|--------------|--------|---------------------|----------------|
| Zhuang [13]             | 0.025     | 0.000101190 | 0.336407          | 1.62805      | 233.3  | 4.36341             | 434.012        |
| Dornaika [14] Closed Form | 0.01     | 0.201501    | 19.5162           | 61.24106     | 330.01 | 762.353             | 195.634.016    |
| Dornaika [14] Iterative  | 3.464     | 1.947      | 58.6522           | 56.7113      | 283.025| 27.687              | 231.57432      |
| Hirsh [15]              | 2.734     | 0.000198832 | 0.331641          | 2.02968      | 350.3  | 3.68620             | 189.545        |
| Li [16] Dual Quaternion  | 0.494     | 0.000174349 | 0.506214          | 1.90974      | 320.2  | 4.36992             | 101.479        |
| Li [16] Kronecker product| 0.085     | 0.000172047 | 0.347818          | 2.25149      | 446.1  | 5.41810             | 388.55         |
| Shaq [17]               | 0.081     | 0.000198832 | 0.331641          | 1.63616      | 235.574| 4.10190             | 403.614        |

$c_{1}$, Euler angles, simultaneous
$c_{11}$, axis angle, simultaneous
$c_{12}$, quaternion, simultaneous
$c_{2}$, Euler angles, simultaneous
$c_{21}$, axis angle, simultaneous
$c_{22}$, quaternion, simultaneous
$r_{R1}$, Euler angles
$r_{R2}$, axis angle
$r_{R3}$, quaternion
$c_{1}$, Euler angles, separable
$c_{11}$, axis angle, separable
$c_{12}$, quaternion, separable
$r_{R1}$, Euler angles
$r_{R2}$, axis angle
$r_{R3}$, quaternion
$c_{2}$, Euler angles, separable
$c_{21}$, axis angle, separable
$c_{22}$, quaternion, separable
$r_{R1}$, Euler angles
$r_{R2}$, axis angle
$r_{R3}$, quaternion

Table 3: Comparison of methods using the error metrics described in Section 3.2 for Dataset 2.

| Method                  | Time (s)  | $r_{R1}$ | $r_{R2}$ (degrees) | $c_{e}$ (mm) | $c_{e}$ | $r_{Rmse}$ (pixels) | $r_{Rmse}$ (mm) |
|-------------------------|-----------|----------|--------------------|--------------|--------|---------------------|----------------|
| Zhuang [13]             | 0.014     | 0.000102627 | 0.288676          | 1.3071      | 47.809 | 11.5458             | 24.56602       |
| Dornaika [14] Closed Form | 0.003     | 0.404278   | 24.1399           | 79.3305     | 176.214| 2959.88             | 5580861.998    |
| Dornaika [14] Iterative  | 1.206     | 1.30184   | 45.2478           | 74.8997     | 157.080| 1329.87             | 173841689.69   |
| Hirsh [15]              | 0.35      | 0.000197954 | 0.149794          | 0.886279    | 22.093 | 4.06050             | 23.00051       |
| Li [16] Dual Quaternion  | 0.05      | 0.000197954 | 0.149794          | 0.886279    | 22.093 | 4.06050             | 23.00051       |
| Li [16] Ktonecker product| 0.0001    | 0.000274766 | 0.149794          | 0.886279    | 22.093 | 4.06050             | 23.00051       |
| Shaq [17]               | 0.0001    | 0.000197954 | 0.149794          | 0.886279    | 22.093 | 4.06050             | 23.00051       |

$c_{1}$, Euler angles, simultaneous
$c_{11}$, axis angle, simultaneous
$c_{12}$, quaternion, simultaneous
$c_{2}$, Euler angles, separable
$c_{21}$, axis angle, separable
$c_{22}$, quaternion, separable
$r_{R1}$, Euler angles
$r_{R2}$, axis angle
$r_{R3}$, quaternion
$c_{2}$, Euler angles, separable
$c_{21}$, axis angle, separable
$c_{22}$, quaternion, separable
$r_{R1}$, Euler angles
$r_{R2}$, axis angle
$r_{R3}$, quaternion

Solving the Robot-World Hand-Eye(s) Calibration Problem with Iterative Methods
### Table 4: Comparison of methods using the error metrics described in Section 3.2 for Dataset 3.

| Method                  | Time (s) | $e_{R1}$ | $e_{R2}$ (degrees) | $e_3$ (mm) | $e_C$   | $rrms$ (pixels) | $rse$ (mm) |
|-------------------------|----------|----------|---------------------|------------|---------|----------------|------------|
| Zhuang [18]             | 0.013    | 0.765969 | 35.537              | 84.21      | 255.288 | 1752.74        | 1.08832 x 10^4 |
| Dornaika [3], Closed Form | 0.005    | 0.076832 | 8.90491             | 97.4986    | 342.26  | 1512.71        | 8.2868 x 10^4  |
| Dornaika [3], Iterative | 3.083    | 7.03167  | 139.069             | 75.6174    | 265.854 | 1719.24        | 1.6618 x 10^4  |
| Hirsh [4], Dual Quaternion | 0.3281   | 0.001146461 | 0.400666   | 2.856167   | 295.872 | 9.10238        | 123.061    |
| Li [6], Kroncker product | 0.002    | 0.01125253 | 0.120413    | 4.82104    | 836.727 | 3174.67         | 1532.47    |
| Shah [10]               | 0.0001   | 0.001146461 | 0.400666   | 2.856167   | 295.872 | 9.10238        | 123.061    |

- **$e_{R1}$**: Error in translation
- **$e_{R2}$**: Error in rotation (degrees)
- **$e_3$**: Error in translation (mm)
- **$e_C$**: Error in orientation
- **$rrms$**: Root mean square error (pixels)
- **$rse$**: Root mean square error (mm)

---

### Table 5: Comparison of methods using the error metrics described in Section 3.2 for Dataset 4.

| Method                  | Time (s) | $e_{R1}$ | $e_{R2}$ (degrees) | $e_3$ (mm) | $e_C$   | $rrms$ (pixels) | $rse$ (mm) |
|-------------------------|----------|----------|---------------------|------------|---------|----------------|------------|
| Zhuang [18]             | 0.003    | 0.0077559 | 2.41835             | 92.201     | 170021  | 288.166        | 30389.5    |
| Dornaika [3], Closed Form | 0.003    | 0.010101 | 2.59799             | 122.624    | 300733  | 129.053        | 659311     |
| Dornaika [3], Iterative | 1.692    | 7.97192  | 173.278             | 8.0541     | 130449  | 1435           | 1.1189 x 10^4 |
| Hirsh [4], Dual Quaternion | 3.099    | 1.0015445 | 1.254749   | 2.353655   | 293.165 | 6.6553         | 1.96219    |
| Li [6], Kroncker product | 0.018    | 0.0015883 | 0.133885   | 2.92055    | 307.066 | 2.89018        | 1.45321    |
| Shah [10]               | 0.0001   | 0.00120994 | 0.415269    | 7.2863     | 1001.8  | 21.69          | 4198.79    |

- **$e_{R1}$**: Error in translation
- **$e_{R2}$**: Error in rotation (degrees)
- **$e_3$**: Error in translation (mm)
- **$e_C$**: Error in orientation
- **$rrms$**: Root mean square error (pixels)
- **$rse$**: Root mean square error (mm)

---

- **$c_1$**: Euler angles, simultaneous
- **$c_1$, axis angle, separable**: Euler angles, separable
- **$c_2$, quaternion, simultaneous**: Quaternion, simultaneous
- **$c_2$, quaternion, separable**: Quaternion, separable
- **$r_{p1}$**: Euler angles
- **$r_{p1}$, axis angle**: Axis angle
- **$r_{p1}$, quaternion**: Quaternion
- **$r_{p2}$**: Euler angles
- **$r_{p2}$, axis angle**: Axis angle
- **$r_{p2}$, quaternion**: Quaternion

- **c1, Euler angles, separable**: Euler angles, separable
- **c1, axis angle, separable**: Axis angle, separable
- **c2, quaternion, separable**: Quaternion, separable
- **c2, quaternion, simultaneous**: Quaternion, simultaneous
- **r_{p1}, Euler angles**: Euler angles
- **r_{p1}, axis angle**: Axis angle
- **r_{p1}, quaternion**: Quaternion
- **r_{p2}, Euler angles**: Euler angles
- **r_{p2}, axis angle**: Axis angle
- **r_{p2}, quaternion**: Quaternion

---

The tables compare the performance of various methods in terms of their accuracy and efficiency, as measured by the error metrics listed above. The strategies and algorithms for estimating orientation and translation in image sequences are evaluated for their accuracy and efficiency, highlighting the differences between simultaneous and separable methods.
This collection of methods was extended to be used relative to existing robot-world hand-eye calibration methods. Our collection of methods was evaluated to assess a particular method’s suitability to reconstruction problems.

Our collections of methods, particularly the $c_2$ simultaneous method from the first class and both methods in the second class perform consistently well with respect to reconstruction accuracy error, generating $rae \leq 2 \text{mm}$, which is the average distance between a calibration pattern point and the point generated with the robot-world, hand-eye calibration information. This value of $rae \leq 2 \text{mm}$ resulted from evaluating the collection of minimization problems using three different rotation representations. Our collection of methods was evaluated relative to existing robot-world hand-eye calibration methods.

| Method | Time (s) | $\rho_{p1}$ | $\rho_{p2}$ | $C_1$ (degrees) | $C_2$ (mm) | $C_3$ (mm) | $C_4$ (mm) | $C_5$ (mm) | $C_6$ (mm) | $C_7$ (mm) |
|--------|----------|-------------|-------------|----------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Zhuang [18] | 0.004 | 0.01138914 | 3.614164 | 113.639 | 180.792 | 1190 | 86207.3 |
| Dornak [13] | 0.000205729 | 7.380841 | 36.60263 | 48.1088 | 211.081 | 190597 |
| Dornak [13] | 1.116 | 4.36192 | 95.1924 | 31.289 | 13171.4 | 1206 | 1.36577 $\times 10^{10}$ |
| Hirt [14] | 0.354 | 0.000266174 | 0.199534 | 132.563 | 326.170 | 238.041 | 1.01182 $\times 10^{10}$ |
| Li [12] | 0.008 | 0.000265726 | 0.203764 | 1.84945 | 329.24 | 7.16294 | 236.416 |
| Li [12] | 0.001 | 0.000258029 | 0.203396 | 1.84945 | 329.24 | 7.16294 | 236.416 |
| Shah [19] | 0.0001 | 0.000265854 | 0.199265 | 210.092 | 61794.3 | 419.512 | 2.12246 $\times 10^{10}$ |

Table 6: Comparison of methods using the error metrics described in Section 3.2 for Dataset 5.

$$
\begin{align*}
\text{Method} & \quad |c_1, \text{Euler angles, simultaneous}| \\
\rho_{p1}, \text{Euler angles} & \quad 3.445 \times 10^{-2} \quad 0.393463 \quad 4.43836 \quad 275.785 \quad 0.646864 \quad 1.21858 \\
\rho_{p1}, \text{axis angle} & \quad 3.158 \times 10^{-2} \quad 0.393463 \quad 4.43836 \quad 275.785 \quad 0.646864 \quad 1.21858 \\
\rho_{p1}, \text{quaternion} & \quad 4.593 \times 10^{-2} \quad 0.393463 \quad 4.43836 \quad 275.785 \quad 0.646864 \quad 1.21858 \\
\rho_{p2}, \text{Euler angles} & \quad 22.517 \times 10^{-2} \quad 0.203543 \quad 0.203543 \quad 0.203543 \quad 0.203543 \quad 0.203543 \\
\rho_{p2}, \text{axis angle} & \quad 19.252 \times 10^{-2} \quad 0.203543 \quad 0.203543 \quad 0.203543 \quad 0.203543 \quad 0.203543 \\
\rho_{p2}, \text{quaternion} & \quad 21.251 \times 10^{-2} \quad 0.203543 \quad 0.203543 \quad 0.203543 \quad 0.203543 \quad 0.203543 \\
\end{align*}
$$

Table 7: Comparison of methods using the error metrics described in Section 3.2 for Dataset 6. $rmse$ is computed for each camera.
methods on a range of real datasets representing different mounting configurations and robot positions relative to the world coordinate frame.

Acknowledgements We also would like to thank anonymous reviewers of our previous paper and this paper for their helpful comments.

References
1. OpenCV. http://opencv.org/. Version 2.4.9
2. Agarwal, S., Merte, K., Other: Core solver. http://ceres-solver.org
3. Dornaika, F., Horaud, R.: Simultaneous robot-world and hand-eye calibration. Robotics and Automation, IEEE Transactions on 14(4), 617–622 (1998)
4. Hirsh, R.L., DeSouza, G.N., Kak, A.C.: An iterative approach to the hand-eye and base-world calibration problem.
5. Horaud, R., Dornaika, F.: Hand-eye calibration. The international journal of robotics research 14(3), 195–210 (1995)
6. Li, A., Lin, W., Defeng, W.: Simultaneous robot-world and hand-eye calibration using dual-queroniens and kronecker product. International journal of Physical Sciences 5(10), 1530–1536 (2010)
7. Lourakis, M.: levan: Levensberg-marquardt nonlinear least squares algorithms in C/C++. [web page] http://www.ics.forth.gr/~lourakis/levan/ (Jul. 2004). [Accessed on 18 Nov. 2014.]
8. Malti, A.: Hand–eye calibration with epipolar constraints: Application to endoscopy. Robotics and Autonomous Systems 61(2), 161–169 (2013)
9. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial & Applied Mathematics 11(2), 431–441 (1963)
Fig. 4 Best viewed in color. Rotation error results for Simulated Dataset I, as compared to the ground truth $X$ and $Z$, using the four metrics of Subsection 4.1.

10. Shah, M.: Solving the robot-world/hand-eye calibration problem using the kronecker product. ASME Journal of Mechanisms and Robotics 5(3), 031,007–031,007–7 (2013)
11. Shiu, Y.C., Ahmad, S.: Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $ax=xb$. Robotics and Automation, IEEE Transactions on 5(1), 16–29 (1989)
12. Strobl, K.H., Hirzinger, G.: Optimal hand-eye calibration. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4647–4653 (2006). DOI 10.1109/IROS.2006.282250
13. Tabb, A.: Shape from silhouette probability maps: reconstruction of thin objects in the presence of silhouette extraction and calibration error. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on (2013)
14. Tabb, A.: Data from: Solving the robot-world hand-eye(s) calibration problem with iterative methods (2017). DOI 10.15482/USDA.ADC/1340592. URL http://dx.doi.org/10.15482/USDA.ADC/1340592
Translation error results for Simulated Dataset I, as compared to the ground truth $X$ and $Z$, using the four metrics of Subsection 4.1.

15. Tabb, A., Ahmad Yousef, K.: Parameterizations for reducing camera reprojection error for robot-world hand-eye calibration. In: IEEE RSJ International Conference on Intelligent Robots and Systems, pp. 3030–3037 (2015)

16. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment - a modern synthesis. In: Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, ICCV 99, pp. 298–372. Springer-Verlag, London, UK, UK (2000). URL http://dl.acm.org/citation.cfm?id=648271.685629

17. Zhang, Z.: A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on 22(11), 1330–1334 (2000). DOI 10.1109/34.888718

18. Zhuang, H., Roth, Z.S., Sudhakar, R.: Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form $ax=yb$. Robotics and Automation, IEEE Transactions on 10(4), 549–554 (1994). DOI 10.1109/70.313105
Fig. 6  
Best viewed in color. Rotation error results for Simulated Dataset II, as compared to the ground truth X and Z, using the four metrics of Subsection 4.1.
Fig. 7 Best viewed in color. Translation error results for Simulated Dataset II, as compared to the ground truth $X$ and $Z$, using the four metrics of Subsection 4.1.
7 Erratum

We regret that there were some errors in the original paper, concerned mostly with the computation of the mean translation error $e_t$ (Section 3.2.2 Equation 27).

Specifically,

1. The computation of $e_t$ was incorrect in our software used to generate the results in tables 2-9 in the original paper.
2. The labeling of units of $e_t$ was incorrect; it should have been mm$^2$.
3. Tables with corrected results are included in this erratum section, in tables 10-17.
4. The software was released with the paper at [14]; this software has been upgraded to present-day libraries and the error in computing $e_t$ noted here has been corrected.
5. Since the underlying libraries supporting the software has undergone revision since we prepared the results for the original paper, there are slight differences in the numerical results despite using the same input. Consequently, we included completely refreshed results in tables 10-17.
Table 10: Erratum: Comparison of methods using the error metrics described in Section 3.2 for Dataset 1.

| Method                | Time (s) | $e_{R1}$ | $e_{R2}$ (degrees) | $e_{C}$ (mm$^2$) | $e_C$ | rmse (pixels) | rae (mm) |
|-----------------------|----------|----------|--------------------|------------------|-------|---------------|----------|
| Zhuang [18]           | 0.0210   | 0.000101 | 0.3641             | 233.243          | 233.243 | 4.35294       | 434.01527|
| Dornaika [13] Closed Form | 0.0059   | 0.261051 | 19.5167            | 380046.790       | 390147.011 | 761.59201     | 158329.92262|
| Dornaika [13] Iterative| 3.0130   | 1.947004 | 58.65204           | 283023.041       | 288024.988 | 272.07926     | 2.3137 × 10^5|
| Hirsh [19]           | 1.841    | 0.000098 | 0.3316             | 359.328          | 359.328 | 3.68832       | 189331.53  |
| Li [22] Dual Quaternion | 0.479    | 0.001174 | 0.90621            | 320.174          | 320.174 | 1.30732       | 103375.65  |
| Li [22] Kronecker product | 0.002   | 0.000102 | 0.34782            | 446.079          | 446.080  | 5.41762       | 384564.32  |
| Shah [10]            | 0.002    | 0.000108 | 0.3316             | 235.577          | 235.577 | 1.01818       | 403656.87  |

$c_{1,}$ Euler angles, simultaneous
$c_{1,}$ axis angle, simultaneous
$c_{1,}$ quaternion, simultaneous
$c_{2,}$ Euler angles, simultaneous
$c_{2,}$ axis angle, simultaneous
$c_{2,}$ quaternion, simultaneous

$c_{1,}$ Euler angles, separable
$c_{1,}$ axis angle, separable
$c_{1,}$ quaternion, separable
$c_{2,}$ Euler angles, separable
$c_{2,}$ axis angle, separable
$c_{2,}$ quaternion, separable

$r_{p1,}$ Euler angles
$r_{p1,}$ axis angle
$r_{p1,}$ quaternion

$r_{p2,}$ Euler angles
$r_{p2,}$ axis angle
$r_{p2,}$ quaternion

Table 11: Erratum: Comparison of methods using the error metrics described in Section 3.2 for Dataset 2.

| Method                | Time (s) | $e_{R1}$ | $e_{R2}$ (degrees) | $e_{C}$ (mm$^2$) | $e_C$ | rmse (pixels) | rae (mm) |
|-----------------------|----------|----------|--------------------|------------------|-------|---------------|----------|
| Zhuang [18]           | 0.014    | 0.000963 | 0.28809            | 47.827           | 47.827 | 11.84704      | 526609   |
| Dornaika [13] Closed Form | 0.003   | 0.041248 | 24.13993           | 176213.072       | 176213.476 | 2900.4505     | 5.580249 × 10^5 |
| Dornaika [13] Iterative| 1.056    | 1.304333 | 43.2477            | 157678.853       | 157080.158 | 1329.87157    | 1.7394 × 10^4 |
| Hirsh [19]           | 0.353    | 0.000002 | 0.14979            | 22.096           | 22.096  | 4.66845       | 2.90364   |
| Li [22] Dual Quaternion | 0.098   | 0.000226 | 0.18517            | 15.591           | 15.591  | 1.24205       | 1.86466   |
| Li [22] Kronecker product | 0.001 | 0.000023 | 0.16918            | 26.456           | 26.456  | 6.47126       | 233157    |
| Shah [10]            | 0.001    | 0.000020 | 0.14979            | 19.516           | 19.516  | 4.61892       | 238297    |

$c_{1,}$ Euler angles, simultaneous
$c_{1,}$ axis angle, simultaneous
$c_{1,}$ quaternion, simultaneous
$c_{2,}$ Euler angles, simultaneous
$c_{2,}$ axis angle, simultaneous
$c_{2,}$ quaternion, simultaneous
$c_{1,}$ Euler angles, separable
$c_{1,}$ axis angle, separable
$c_{1,}$ quaternion, separable
$c_{2,}$ Euler angles, separable
$c_{2,}$ axis angle, separable
$c_{2,}$ quaternion, separable

$r_{p1,}$ Euler angles
$r_{p1,}$ axis angle
$r_{p1,}$ quaternion

$r_{p2,}$ Euler angles
$r_{p2,}$ axis angle
$r_{p2,}$ quaternion

Amy Tabb, Khalil M. Ahmad Yousef
Table 12 Erratum: Comparison of methods using the error metrics described in Section 3.2 for Dataset 3.

| Method                        | Time (s) | $e_{R1}$ | $e_{R2}$ (degrees) | $e_{C}$ (mm) | $e_{C}$(mm) | $\text{rms}$ (pixels) | rae (mm) |
|-------------------------------|----------|----------|-------------------|--------------|------------|------------------------|-----------|
| Zhuang [10]                   | 0.012    | 0.748187 | 35.29844          | 255.631854   | 255.632662 | 1738.74388             | 1.1469 x 10^6 |
| Dornaika [7] Closed Form      | 0.001   | 0.076838 | 8.90484           | 342216.217   | 342216.293 | 1512.72147             | 8.2868 x 10^6 |
| Dornaika [7] Iterative        | 3.027    | 5.733835 | 130.42109         | 242755.069   | 242751.643 | 1082.19641             | 7.5994 x 10^6 |
| Hirsh [9]                     | 2.824    | 0.001146 | 0.40870           | 1372.0563    | 1372.0563  | 39.06440               | 328.99175  |
| Li [8] Quaternion             | 0.132    | 0.000229 | 0.55140           | 1029.034     | 1029.035   | 20.73818               | 189.51141  |
| Li [8] Kronecker product      | 0.002    | 0.000153 | 0.42043           | 836.726      | 836.723    | 31.74299               | 1832.37089 |
| Shah [10]                     | 0.001    | 0.000146 | 0.40967           | 275.857      | 275.856    | 9.10223                | 123.67957  |

$c_1$, Euler angles, simultaneous
$c_1$, axis angle, simultaneous
$c_1$, quaternion, simultaneous
$c_2$, Euler angles, simultaneous
$c_2$, axis angle, simultaneous
$c_2$, quaternion, simultaneous
$c_1$, Euler angles, separable
$c_1$, axis angle, separable
$c_1$, quaternion, separable
$c_2$, Euler angles, separable
$c_2$, axis angle, separable
$c_2$, quaternion, separable
rp$_1$, Euler angles
rp$_1$, axis angle
rp$_1$, quaternion
rp$_2$, Euler angles
rp$_2$, axis angle
rp$_2$, quaternion

Table 13 Erratum: Comparison of methods using the error metrics described in Section 3.2 for Dataset 4.

| Method                        | Time (s) | $e_{R1}$ | $e_{R2}$ (degrees) | $e_{C}$ (mm) | $e_{C}$(mm) | $\text{rms}$ (pixels) | rae (mm) |
|-------------------------------|----------|----------|-------------------|--------------|------------|------------------------|-----------|
| Zhuang [10]                   | 0.004    | 0.00756  | 2.41826           | 1.10620.300  | 1.10620.307 | 288.10520              | 360.85751  |
| Dornaika [7] Closed Form      | 0.002    | 0.01016  | 2.57919           | 3007.342.854 | 3007.342.856 | 420.054.32             | 650.309.8483 |
| Dornaika [7] Iterative        | 1.638    | 0.977799 | 173.35861         | 1270.043     | 1278.021   | 1247.83275             | 1.0814 x 10^7 |
| Hirsh [9]                     | 2.337    | 0.000035 | 0.39031           | 723.043      | 723.043    | 6.222124               | 76.19298   |
| Li [8] Quaternion             | 0.025    | 0.007638 | 10.50628          | 81069.728    | 81069.806  | 189.191725             | 232176.8668 |
| Li [8] Kronecker product      | 0.001    | 0.000121 | 0.41527           | 1001.804     | 1001.804   | 21.89892               | 4198.79638 |
| Shah [10]                     | 0.001    | 0.000153 | 0.30531           | 195.932      | 195.932    | 2.74710                | 22.70185   |

$c_1$, Euler angles, simultaneous
$c_1$, axis angle, simultaneous
$c_1$, quaternion, simultaneous
$c_2$, Euler angles, simultaneous
$c_2$, axis angle, simultaneous
$c_2$, quaternion, simultaneous
$c_1$, Euler angles, separable
$c_1$, axis angle, separable
$c_1$, quaternion, separable
$c_2$, Euler angles, separable
$c_2$, axis angle, separable
$c_2$, quaternion, separable
rp$_1$, Euler angles
rp$_1$, axis angle
rp$_1$, quaternion
rp$_2$, Euler angles
rp$_2$, axis angle
rp$_2$, quaternion
Table 14 Error: Comparison of methods using the error metrics described in Section 3.2 for Dataset 5.

| Method                  | Time (s)  | $e_{R1}$ | $e_{R2}$ (degrees) | $e_{C}$ ($mm^2$) | $e_{C}$ | $rmmse$ (pixels) | $rac$ (mm) |
|-------------------------|-----------|----------|-------------------|------------------|--------|------------------|------------|
| Zeisler [15]            | 0.002     | 0.011901 | 3.41644           | 180.929.08      | 180.929.071 | 1189.98270     | 5862.638952 |
| Dornaka [14]            | 0.001     | 0.050325 | 7.86042           | 1290.2687       | 1290.26871298 | 265.36865     | 6672.22156 |
| Dornaka [14] Iterative  | 0.915     | 3.793803 | 87.44746          | 4497.5653       | 4497.56437   | 1663.73498     | 24452.10646 |
| Shah [19]               | 0.255     | 0.000226 | 0.19955           | 326.216272      | 326.216272236 | 236.669317     | 20238.10686 |
| Li [21] Dual Quaternion  | 0.006     | 0.010035 | 4.05088           | 10313.6892      | 10313.68102  | 65.58040       | 62872.52472 |
| Li [21] Kronecker Product| 0.001     | 0.000029 | 0.20336           | 203.3290        | 203.2407     | 7.16277        | 236.40669    |

$c_4$, Euler angles, simultaneous
$c_4$, axis angle, simultaneous
$c_4$, quaternion, simultaneous
$c_4$, Euler angles, separable
$c_4$, axis angle, separable
$c_4$, quaternion, separable
$c_5$, Euler angles, separable
$c_5$, axis angle, separable
$c_5$, quaternion, separable
$c_6$, Euler angles
$c_6$, axis angle
$c_6$, quaternion
$r_{p1}$, Euler angles
$r_{p1}$, axis angle
$r_{p1}$, quaternion
$r_{p2}$, Euler angles
$r_{p2}$, axis angle
$r_{p2}$, quaternion

Table 15 Error: Comparison of methods using the error metrics described in Section 3.2 for Dataset 6. $rmmse$ is computed for each camera.

| Method                  | Time (s)  | $e_{R1}$ | $e_{R2}$ (degrees) | $e_{C}$ ($mm^2$) | $e_{C}$ | $rmmse$ (pixels) | $rac$ (mm) |
|-------------------------|-----------|----------|-------------------|------------------|--------|------------------|------------|
| $c_1$, Euler angles, simultaneous | 0.379     | 0.000026 | 0.18394           | 21.058           | 21.058 | 2.31750         | 2.00713     | 16.68061 |
| $c_1$, axis angle, simultaneous | 0.325     | 0.000026 | 0.18394           | 21.058           | 21.058 | 2.31750         | 2.00713     | 16.68061 |
| $c_1$, quaternion, simultaneous | 0.325     | 0.000026 | 0.18394           | 21.058           | 21.058 | 2.31750         | 2.00713     | 16.68061 |
| $c_2$, Euler angles, separable | 0.188     | 0.000028 | 0.20966           | 54.429           | 54.429 | 0.78668         | 0.74838     | 15.62375 |
| $c_2$, axis angle, separable | 0.249     | 0.000025 | 0.18394           | 49.196           | 49.196 | 0.78668         | 0.74838     | 15.62375 |
| $c_2$, quaternion, separable | 0.258     | 0.000025 | 0.18394           | 49.196           | 49.196 | 0.78668         | 0.74838     | 15.62375 |
| $c_3$, Euler angles, separable | 0.551     | 0.000023 | 0.18634           | 60.893700        | 60.893700   | 68.71199       | 62.04198    | 141.86919 |
| $c_3$, axis angle, separable | 0.092     | 0.000023 | 0.18634           | 41.640592        | 41.640592   | 46.28519       | 42.01570    | 109.11472 |
| $c_3$, quaternion, separable | 0.907     | 0.000023 | 0.18634           | 41.640592        | 41.640592   | 46.28519       | 42.01570    | 109.11472 |
| $c_4$, Euler angles, separable | 0.335     | 0.000023 | 0.18634           | 91.211109        | 91.211109   | 51.71400       | 51.71400    | 102.25648 |
| $c_4$, axis angle, separable | 0.355     | 0.000023 | 0.18634           | 41.52222241      | 41.52222241 | 34.20819       | 30.76904    | 29535.0446 |
| $c_4$, quaternion, separable | 0.365     | 0.000023 | 0.18634           | 42.033024        | 42.033024   | 34.30518       | 30.99755    | 30503.85925 |
| $r_{p1}$, Euler angles | 8.865     | 0.000115 | 0.40121           | 284.0308        | 284.0308    | 4.04176        | 4.04176     | 1.49913 |
| $r_{p1}$, axis angle | 9.324     | 0.000115 | 0.40121           | 284.0308        | 284.0308    | 4.04176        | 4.04176     | 1.49913 |
| $r_{p1}$, quaternion | 10.287    | 0.000115 | 0.40121           | 284.0308        | 284.0308    | 4.04176        | 4.04176     | 1.49913 |
| $r_{p2}$, Euler angles | 90.490    | 0.000176 | 1.07923           | 1479.389        | 1479.389    | 4.29860        | 4.29860     | 1.19342 |
| $r_{p2}$, axis angle | 92.689    | 0.000176 | 1.07923           | 147.3897        | 147.3897    | 4.29860        | 4.29860     | 1.19342 |
| $r_{p2}$, quaternion | 109.332   | 0.000176 | 1.07923           | 1479.389        | 1479.389    | 4.29860        | 4.29860     | 1.19342 |
Table 16 Erratum: Comparison of methods using the error metrics described in Section 3.2 for Dataset 7. \( \text{rmmse} \) is computed for each camera

| Method                | Time (s) | \( \epsilon_{R1} \) | \( \epsilon_{R2} \) (degrees) | \( \epsilon_C \) | \( \text{rmmse}_{0} \) (pixels) | \( \text{rmmse}_{1} \) (pixels) | \( \text{rmmse}_{2} \) (pixels) | \( \text{ror}(\text{mm}) \) |
|-----------------------|----------|----------------------|-------------------------------|----------------|--------------------------------|--------------------------------|--------------------------------|--------------------------|
| \( c_1 \), Euler angles, simultaneous | 1.239    | 0.0000388           | 0.22362                       | 37.896         | 37.896                        | 1.70848                        | 1.70482                        | 2.03210                  | 25.5733                  |
| \( c_1 \), axis angle, simultaneous | 1.974    | 0.0000341           | 0.21638                       | 37.896         | 37.896                        | 1.69745                        | 1.59725                        | 1.59716                  | 27.2914                  |
| \( c_1 \), quaternion, simultaneous | 0.940    | 0.0000371           | 0.21982                       | 37.896         | 37.896                        | 1.59013                        | 1.59645                        | 1.88415                  | 21.54825                 |
| \( c_2 \), Euler angles, simultaneous | 1.266    | 0.0000388           | 0.22349                       | 68.522         | 68.522                        | 0.71714                        | 0.68944                        | 0.64628                  | 8.89738                   |
| \( c_2 \), axis angle, simultaneous | 1.086    | 0.0000341           | 0.21444                       | 63.860         | 63.860                        | 0.74081                        | 0.67274                        | 0.64707                  | 7.12096                   |
| \( c_2 \), quaternion, simultaneous | 1.657    | 0.0000341           | 0.21444                       | 63.860         | 63.860                        | 0.74081                        | 0.67274                        | 0.64707                  | 7.11999                   |

Table 17 Erratum: Comparison of methods using the error metrics described in Section 3.2 for Dataset 8. \( \text{rmmse} \) is computed for each camera

| Method                | Time (s) | \( \epsilon_{R1} \) | \( \epsilon_{R2} \) (degrees) | \( \epsilon_C \) | \( \text{rmmse}_{0} \) (pixels) | \( \text{rmmse}_{1} \) (pixels) | \( \text{rmmse}_{2} \) (pixels) | \( \text{ror}(\text{mm}) \) |
|-----------------------|----------|----------------------|-------------------------------|----------------|--------------------------------|--------------------------------|--------------------------------|--------------------------|
| \( c_1 \), Euler angles, simultaneous | 1.380    | 0.0000333           | 0.20999                       | 34.731         | 34.731                        | 1.54184                        | 1.54184                        | 1.89608                  | 35.34167                  |
| \( c_1 \), axis angle, simultaneous | 1.164    | 0.0000221           | 0.17233                       | 34.731         | 34.731                        | 1.47581                        | 1.47581                        | 1.83055                  | 35.79977                  |
| \( c_1 \), quaternion, simultaneous | 1.000    | 0.0000255           | 0.17233                       | 34.731         | 34.731                        | 1.47117                        | 1.47117                        | 1.82199                  | 35.34195                  |
| \( c_2 \), Euler angles, simultaneous | 1.123    | 0.0000333           | 0.22152                       | 65.159         | 65.159                        | 0.80524                        | 0.80524                        | 0.98146                  | 3.79585                   |
| \( c_2 \), axis angle, simultaneous | 1.169    | 0.0000205           | 0.19755                       | 60.197         | 60.197                        | 0.76779                        | 0.76779                        | 0.86147                  | 2.18646                   |
| \( c_2 \), quaternion, simultaneous | 1.161    | 0.0000205           | 0.19755                       | 60.197         | 60.197                        | 0.77681                        | 0.77681                        | 0.88115                  | 2.18844                   |
| \( c_1 \), Euler angles, separable | 2.256    | 0.0000195           | 0.15764                       | 39.414         | 39.414                        | 1.28480                        | 1.28480                        | 1.58336                  | 9.52354                   |
| \( c_1 \), axis angle, separable | 2.907    | 0.0000195           | 0.15764                       | 39.421         | 39.421                        | 1.28478                        | 1.28478                        | 1.58336                  | 9.50195                   |
| \( c_1 \), quaternion, separable | 3.440    | 0.0000195           | 0.15764                       | 39.422         | 39.422                        | 1.28477                        | 1.28477                        | 1.58336                  | 9.50182                   |
| \( c_2 \), Euler angles, separable | 2.256    | 0.0000195           | 0.15763                       | 39.704         | 39.704                        | 1.28115                        | 1.28115                        | 1.58128                  | 9.52374                   |
| \( c_2 \), axis angle, separable | 2.395    | 0.0000195           | 0.15764                       | 39.698         | 39.698                        | 1.28112                        | 1.28112                        | 1.58125                  | 9.52629                   |
| \( c_2 \), quaternion, separable | 2.486    | 0.0000195           | 0.15764                       | 39.699         | 39.699                        | 1.28113                        | 1.28113                        | 1.58125                  | 9.52629                   |

\[ \text{rp}_1 \], Euler angles
\[ \text{rp}_1 \], axis angle
\[ \text{rp}_1 \], quaternion
\[ \text{rp}_2 \], Euler angles
\[ \text{rp}_2 \], axis angle
\[ \text{rp}_2 \], quaternion