Machine Vision and Applications (2017) 28:569–590
DOI 10.1007/s00138-017-0841-7

ORIGINAL PAPER

Solving the robot-world hand-eye(s) calibration problem with iterative methods

Amy Tabb[1] · Khalil M. Ahmad Yousef[2]

Received: 28 July 2016 / Revised: 23 March 2017 / Accepted: 1 April 2017 / Published online: 2 May 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 $\mathbf{AX} = \mathbf{ZB}$, where $\mathbf{X}$ and $\mathbf{Z}$ 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 returns 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.

Keywords Robot · Hand-eye · Calibration · Reconstruction

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 ${}^{\mathrm{c}}Z_{\mathrm{h}}$ or simply $\mathbf{Z}$, and the transformation from the robot-base coordinate frame to the world coordinate frame be ${}^{\mathrm{w}}X_{\mathrm{b}}$ or simply $\mathbf{X}$. The transformation of the robot-base frame to the hand coordinate frame is ${}^{\mathrm{h}}B_{\mathrm{b}}$ or simply $\mathbf{B}$ 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 ${}^{\mathrm{c}}A_{\mathrm{w}}$ or $\mathbf{A}$. $\mathbf{A}$ 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 Fig. 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 ($\mathbf{A}, \mathbf{B}$) and the rest of matrices are exchanged ($\mathbf{X}, \mathbf{Z}$). Despite this difference, the linear relationship $\mathbf{AX} = \mathbf{ZB}$ 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 Sect. 2.2.

Mention of trade names or commercial products in this publication is solely for the purpose of providing specific information and does not imply recommendation or endorsement by the US Department of Agriculture. USDA is an equal opportunity provider and employer. A. Tabb acknowledges the support of US National Science Foundation grant number IOS-1339211.

✉ Amy Tabb
amy.tabb@ars.usda.gov

Khalil M. Ahmad Yousef
khalil@hu.edu.jo

1 United States Department of Agriculture, Agricultural Research Service, Appalachian Fruit Research Laboratory, Kearneysville, WV 25430, USA

2 Computer Engineering Department, The Hashemite University, Zarqa 13115, Jordan

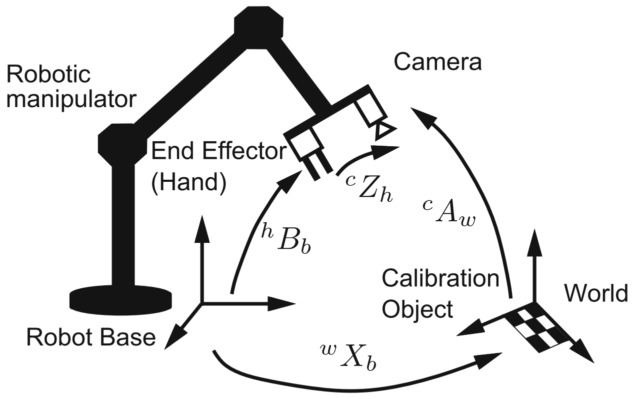


Fig. 1 Hand-eye robot-world calibration: a camera (eye) mounted at the end-effector (hand) of a robot

Given these preliminaries, Zhuang et al. [18] formalized the robot-world hand-eye calibration explicitly as the homogeneous matrix equation $\mathbf{AX} = \mathbf{ZB}$, where all of the matrices are 4×4 matrices, and were the first to provide a method to find solutions for $\mathbf{X}$ and $\mathbf{Z}$. Many different positions of the robot and camera are used to generate multiple relationships $\mathbf{A}_i\mathbf{X} = \mathbf{ZB}_i$, $i \in [0, n-1]$, 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 $\mathbf{AX} = \mathbf{XB}$ by Shiu and Ahmad [11]. In the hand-eye calibration problem, the matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ 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 $\mathbf{A}_i$ and $\mathbf{B}_i$ 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 Sect. 1 and specifically in Sect. 1.1, we provide an overview of existing approaches and recent work. In Sect. 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 Sect. 5, the results on real and simulated datasets are shown and discussed. Finally, in Sect. 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 $\mathbf{AX} = \mathbf{ZB}$ into a purely rotational part and a translational part, where $\mathbf{R}_A$ represents a 3×3 rotation matrix, and $\mathbf{t}_A$ a 3×1 translation vector as shown in Eq. 1.

$$\begin{bmatrix} \mathbf{R}_A & \mathbf{t}_A \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_X & \mathbf{t}_X \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_Z & \mathbf{t}_Z \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_B & \mathbf{t}_B \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{1}$$

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

$$\mathbf{R}_A\mathbf{R}_X = \mathbf{R}_Z\mathbf{R}_B \tag{2}$$

$$\mathbf{R}_A\mathbf{t}_X + \mathbf{t}_A = \mathbf{R}_Z\mathbf{t}_B + \mathbf{t}_Z \tag{3}$$

Since Eq. 3 is linear in $\mathbf{t}_X$ and $\mathbf{t}_Z$ if $\mathbf{R}_Z$ is known, the most frequent approach to estimate $\mathbf{X}$ and $\mathbf{Z}$ is to first find $\mathbf{R}_X$ and $\mathbf{R}_Z$ using Eq. 2, and then use that solution and Eq. 3 to find $\mathbf{t}_X$ and $\mathbf{t}_Z$.

Of those methods that separate the estimation of the rotation and translation parts using Eq. 1, there are several different approaches. In Zhuang et al. [18], a linear solution was proposed for finding the unknowns $\mathbf{R}_X$ and $\mathbf{R}_Z$ by representing the rotation matrices as quaternions, and the translation components were found using linear least squares. Dornaika and Horaud in [2] 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 [3] proposed a separable, iterative approach in which the estimation of $\mathbf{X}$ and $\mathbf{Z}$ is alternated. The method consists of assuming that $\mathbf{Z}$ is known and estimating $\mathbf{X}$ by averaging $\mathbf{X}_i = \mathbf{ZB}_i\mathbf{A}^{-1}$ for all i to generate an estimate for $\mathbf{X}$; this process is repeated in a similar way for $\mathbf{Z}$ using the estimate found for $\mathbf{X}$, and the estimation of $\mathbf{X}$ and $\mathbf{Z}$ 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 is 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 [2], in addition to a closed-form separable solution, Dornaika and Horaud present a formulation of the problem as a nonlinear 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 Horaud, Stobl and Hirzinger in [12] estimate the translation and rotation components through nonlinear 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. [5] proposed a simultaneous solution that is found using dual quaternions and the Kronecker product.

Besides those approaches that deal with the equality $\mathbf{AX} = \mathbf{ZB}$ 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 [4], Horaud and Dornaika incorporated camera calibration parameters into a cost function for the hand-eye calibration problem $\mathbf{AX} = \mathbf{XB}$ 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 [7] 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 $\mathbf{X}$ and $\mathbf{Z}$ HTMs. Unlike Horaud and Dornaika [4], Malti [7] 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* [6] and Ceres [1], in addition to the general interest in bundle adjustment [16], 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 $\mathbf{AX}$ and $\mathbf{ZB}$ over n positions of the robot as given in Eq. 4 below, or in a closely-related version of it, where F denotes the Frobenius norm. This class is discussed in Sect. 2.1.

$$\sum_{i\in[0,n-1]} ||\mathbf{A}_i\mathbf{X} - \mathbf{ZB}_i||_F^2 \tag{4}$$

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

Each of the aforementioned two classes of cost functions contains two or more subclasses 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 $\mathbf{X}$ and $\mathbf{Z}$. It is important to mention here that the above collection of classes and methods easily extends to the problem of calibrating multiple cameras mounted on one robot as shall be discussed in Sect. 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. [3] and the iterative method of Dornaika and Horaud [2]. The work of Malti [7] concerning using camera reprojection error to estimate HTMs $\mathbf{X}$ and $\mathbf{Z}$ is similar to our second class of methods, when using the Euler angle representation for rotations (Sect. 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 $\mathbf{R} = R_Z R_Y R_X$. The axis-angle representation is another three-variable representation, where a three-element vector $\mathbf{v}$ represents the axis of rotation, and $||\mathbf{v}||$ is the angle of rotation about the $\mathbf{v}$ axis such that a corresponding rotation matrix is generated with Rodrigues' rotation formula. Finally, quaternions are a 4-element representation, where a unit vector $\mathbf{q}$ 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 $\mathbf{p}$, with the size of $\mathbf{p}$ 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 $\mathbf{p}$ is $\mathbf{R}(\mathbf{p})$.

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 $\mathbf{A}_i$ and base to end-effector transformations $\mathbf{B}_i$ to estimate the HTMs **X** and **Z**. This is achieved by seeking the minimum of c_1 as given below:

$$c_1 = \sum_{i=0}^{n-1} ||\mathbf{A}_i\mathbf{X} - \mathbf{Z}\mathbf{B}_i||_F^2 \tag{5}$$

In the cost function c_1 in Eq. 5, if we substitute in the rotation representation $\mathbf{R}(\mathbf{p})$ and the translation vectors for the unknowns **X** and **Z**, the minimization problem becomes:

$$\underset{\mathbf{p}_X,\mathbf{t}_X,\mathbf{p}_Z,\mathbf{t}_Z}{\text{argmin}} \sum_{i=0}^{n-1} \left\| \mathbf{A}_i \begin{bmatrix} \mathbf{R}(\mathbf{p}_X) & \mathbf{t}_X \\ \mathbf{0}^T & 1 \end{bmatrix} - \begin{bmatrix} \mathbf{R}(\mathbf{p}_Z) & \mathbf{t}_Z \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{B}_i \right\|_F^2 \tag{6}$$

A solution to the problem in Eq. 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 Eq. 6 into two parts via a separable formulation: one part for rotation and the other part to estimate the translation components as captured by Eqs. 7 and 8, respectively.

$$\underset{\mathbf{p}_X,\mathbf{p}_Z}{\text{argmin}} \sum_{i=0}^{n-1} \left\| \mathbf{R}_{A,i}\mathbf{R}(\mathbf{p}_X) - \mathbf{R}(\mathbf{p}_Z)\mathbf{R}_{B,i} \right\|_F^2 \tag{7}$$

$$\underset{\mathbf{t}_X,\mathbf{t}_Z}{\text{argmin}} \sum_{i=0}^{n-1} \left\| \mathbf{R}_{A,i}\mathbf{t}_X + \mathbf{t}_A - \mathbf{R}(\mathbf{p}_Z)\mathbf{t}_{B,i} - \mathbf{t}_Z \right\|_F^2 \tag{8}$$

An approximate solution to Eq. 7 is first found and then the translation components are found from Eq. 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 Fig. 1. We also explore the use of a slightly different cost function c_2 as given in Eq. 9:

$$c_2 = \sum_{i=0}^{n-1} ||\mathbf{A}_i - \mathbf{Z}\mathbf{B}_i\mathbf{X}^{-1}||_F^2 \tag{9}$$

We can find an approximate solution to c_2 simultaneously, similar to what we did with the function c_1, by estimating $\mathbf{p}_X$, $\mathbf{t}_X$, $\mathbf{p}_Z$, and $\mathbf{t}_Z$. In order to simplify the notation, we let $\tilde{\mathbf{X}} = \mathbf{X}^{-1}$. Then $\tilde{\mathbf{p}}_X$ and $\tilde{\mathbf{t}}_X$ are the rotation parameters and translation vectors of $\mathbf{X}^{-1}$, respectively. With these substitutions, we have the following minimization problem:

$$\underset{\tilde{\mathbf{p}}_X,\tilde{\mathbf{t}}_X,\mathbf{p}_Z,\mathbf{t}_Z}{\operatorname{argmin}} \sum_{i=0}^{n-1} \left\| \mathbf{A}_i - \begin{bmatrix} \mathbf{R}(\mathbf{p}_Z) & \mathbf{t}_Z \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{B}_i \begin{bmatrix} \mathbf{R}(\tilde{\mathbf{p}}_X) & \tilde{\mathbf{t}}_X \\ \mathbf{0}^T & 1 \end{bmatrix} \right\|_F^2 \tag{10}$$

As with c_1, it is possible to create a separable formulation by first estimating the rotation components in Eq. 11 and then the translation components in Eq. 12.

$$\underset{\tilde{\mathbf{p}}_X,\mathbf{p}_Z}{\operatorname{argmin}} \sum_{i=0}^{n-1} \left\| \mathbf{R}_{A,i} - \mathbf{R}(\mathbf{p}_Z)\mathbf{R}_{B,i}\mathbf{R}(\tilde{\mathbf{p}}_X) \right\|_F^2 \tag{11}$$

$$\underset{\tilde{\mathbf{t}}_X,\mathbf{t}_Z}{\operatorname{argmin}} \sum_{i=0}^{n-1} \left\| \mathbf{t}_{A,i} - \mathbf{R}(\mathbf{p}_Z)\mathbf{R}_{B,i}\tilde{\mathbf{t}}_X - \mathbf{R}(\mathbf{p}_Z)\mathbf{t}_B - \mathbf{t}_Z \right\|_F^2 \tag{12}$$

In both of the minimization formulations (simultaneous and separable) for c_1 and c_2, the number of parameters to be estimated for **X** and **Z** is: six translation components and six or eight rotation parameters, depending on the choice of the rotation representation. Approximate solutions to all of the problems proposed in this paper are found with the Levenberg–Marquardt method for nonlinear least squares [8] and specifically its implementation in *Ceres* [1]. For an initial solution, values are chosen such that $\mathbf{R}(\mathbf{p}_X)$ and $\mathbf{R}(\mathbf{p}_Z)$ are 3×3 identity matrices and $\mathbf{t}_X$ and $\mathbf{t}_Z$ have all elements zero.

2.2 Second class: camera reprojection error

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 [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 Sect. 3.1.

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 $\overrightarrow{\mathcal{X}}$, and because there are multiple m such points we give a subscript $j \in [0, m-1]$, $\overrightarrow{\mathcal{X}}_j$. 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 $\overrightarrow{\mathcal{X}}_j$ is projected to the image from robot position i using the intrinsic camera calibration parameters, we have image point $\overrightarrow{\tilde{\mathbf{x}}}_{ij}$ and a corresponding original image point $\overrightarrow{\mathbf{x}}_{ij}$. We can represent $\overrightarrow{\tilde{\mathbf{x}}}_{ij}$ in the context of the robot-world hand-eye calibration problem as follows:

$$\overrightarrow{\tilde{\mathbf{x}}}_{ij} = f\left(\mathbf{k}, \left[\mathbf{Z}\mathbf{B}_i\mathbf{X}^{-1}\right]_{3\times4} \overrightarrow{\mathcal{X}}_j\right) \tag{13}$$

where $\mathbf{k}$ is a vector containing the intrinsic camera calibration parameters. The bracket notation $[\,]_{3\times4}$ is used to denote the upper 3×4 sub-matrix of what is inside the bracket. $f()$ is the function that transforms $\left[\mathbf{Z}\mathbf{B}_i\mathbf{X}^{-1}\right]_{3\times4} \overrightarrow{\mathcal{X}}_j$ 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 Eq. 10, to simplify the representation of Eq. 13, we substitute for $\mathbf{X}^{-1}$ another matrix $\tilde{\mathbf{X}}$, which is composed of an orthonormal rotation matrix and translation vector.

Given the preliminaries discussed above, the reprojection sum of squares error (rsse) is:

$$\text{rsse} = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} ||\overrightarrow{\mathbf{x}}_{ij} - \overrightarrow{\tilde{\mathbf{x}}}_{ij}||^2 \tag{14}$$

and by substituting $\overrightarrow{\tilde{\mathbf{x}}}_{ij}$ we have:

$$\text{rsse} = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} ||\overrightarrow{\mathbf{x}}_{ij} - f\left(\mathbf{k}, \left[\mathbf{Z}\mathbf{B}_i\tilde{\mathbf{X}}\right]_{3\times4} \overrightarrow{\mathcal{X}}_j\right)||^2 \tag{15}$$

We note that in Eqs. 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 $\tilde{\mathbf{X}}$ and **Z** matrices, the camera reprojection sum of squares error (rsse) minimization problem becomes as given in Eq. 16, which we refer to as rp_1 method:

$$\underset{\tilde{\mathbf{p}}_X, \tilde{\mathbf{t}}_X, \mathbf{p}_Z, \mathbf{t}_Z}{\operatorname{argmin}} \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} || \overrightarrow{\mathbf{x}}_{ij} - f(\mathbf{k}, \begin{bmatrix} \mathbf{R}(\mathbf{p}_Z) & \mathbf{t}_Z \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{B}_i \begin{bmatrix} \mathbf{R}(\tilde{\mathbf{p}}_X) & \tilde{\mathbf{t}}_X \\ \mathbf{0}^T & 1 \end{bmatrix}_{3\times 4} \overrightarrow{\mathcal{X}}_j ||^2 \tag{16}$$

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

Within the simultaneous formulation of the robot-world hand-eye calibration problem as given in Eq. 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 Eq. 16 instead of constants as given in Eq. 17, which we refer to as the rp_2 method.

$$\underset{\tilde{\mathbf{p}}_X, \tilde{\mathbf{t}}_X, \mathbf{p}_Z, \mathbf{t}_Z, \mathbf{k}}{\operatorname{argmin}} \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} || \overrightarrow{\mathbf{x}}_{ij} - f(\mathbf{k}, \begin{bmatrix} \mathbf{R}(\mathbf{p}_Z) & \mathbf{t}_Z \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{B}_i \begin{bmatrix} \mathbf{R}(\tilde{\mathbf{p}}_X) & \tilde{\mathbf{t}}_X \\ \mathbf{0}^T & 1 \end{bmatrix}_{3\times 4} \overrightarrow{\mathcal{X}}_j ||^2 \tag{17}$$

In contrast with the minimized reprojection sum of squared error (rsse) in both rp_1 and rp_2 methods, it is important to mention that in the camera calibration literature, the reprojection root-mean-squared error (rrmse) is more common, as shown in Eq. 18. The 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 rsse also result in a minimum of rrmse, rsse is used as a cost function in this work.

$$\text{rrmse} = \sqrt{\frac{1}{mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} || \overrightarrow{\mathbf{x}}_{ij} - \overrightarrow{\tilde{\mathbf{x}}}_{ij} ||^2} \tag{18}$$

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 0th camera at the ith robot position. Then the relationship between the matrices in the context of robot-world hand-multiple-eyes calibration problem would be formulated as:

$$\mathbf{A}_{i,0}\mathbf{X} = \mathbf{Z}_0\mathbf{B} \tag{19}$$

$$\mathbf{A}_{i,1}\mathbf{X} = \mathbf{Z}_1\mathbf{B} \tag{20}$$

$$\vdots$$

$$\mathbf{A}_{i,q-1}\mathbf{X} = \mathbf{Z}_{q-1}\mathbf{B} \tag{21}$$

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

$$\underset{\mathbf{p}_X, \mathbf{t}_X, \mathbf{p}_{Z,0}, \mathbf{t}_{Z,0}, \mathbf{p}_{Z,1}, \mathbf{t}_{Z,1}, \ldots, \mathbf{p}_{Z,q-1}, \mathbf{t}_{Z,q-1}}{\operatorname{argmin}} \sum_{d=0}^{q-1}\sum_{i=0}^{n-1} \left\| \mathbf{A}_{i,d} \begin{bmatrix} \mathbf{R}(\mathbf{p}_X) & \mathbf{t}_X \\ \mathbf{0}^T & 1 \end{bmatrix} - \begin{bmatrix} \mathbf{R}(\mathbf{p}_{Z,d}) & \mathbf{t}_{Z,d} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{B}_i \right\|_F^2 \tag{22}$$

where $\mathbf{p}_{Z,d}$ and $\mathbf{t}_{Z,d}$ are the rotation parameterization and translation vectors for the dth HTM, $\mathbf{Z}_d$. Separable versions follow using the same pattern, as it is done in the cost function c_1 and the 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 has 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 $\mathbb{S}_d$, and denote $\min_s$ as the minimum size of $\mathbb{S}_d$ determined for all cameras. Then, an individual weight w_d for camera d can be set as $\frac{\min_s}{|\mathbb{S}_d|}$. 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 Eq. 23.

$$\underset{\substack{\mathbf{p}_X, \mathbf{t}_X, \\ \mathbf{p}_{Z,0}, \mathbf{t}_{Z,0}, \\ \mathbf{p}_{Z,1}, \mathbf{t}_{Z,1}, \\ \cdots \\ \mathbf{p}_{Z,q-1}, \mathbf{t}_{Z,q-1}}}{\operatorname{argmin}} \sum_{d=0}^{q-1}\sum_{i\in\mathbb{S}_d} w_d \quad g(\mathbf{p}_X, \mathbf{t}_X, \mathbf{p}_{Z,d}, \mathbf{t}_{Z,d}) \tag{23}$$

The problem in Eq. 23 can be applied to the first class of cost functions, as well as the $rp1$ 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 $\mathbf{k}$ for all q cameras and doing so follows from Eq. 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

Table 1 Dataset descriptions

Dataset	Image Size	Lens focal length (mm)	Robot	n	rrmse (pixels)
1	(640 × 480)	8	Denso VS-6577GM-B	88	0.185242
2	(2456 × 2058)	8	Denso VS-6577GM-B	28	0.199418
3	(2456 × 2058)	6	Denso VM-60BIG	36	0.540056
4	(1600 × 1200)	6	Denso VM-60BIG	20	0.447463
5	(1228 × 1029)	6	Denso VS-6577GM-B	15	0.124774
6	(1228 × 1029)	6	Denso VS-6577GM-B	15	0.124774
	(1228 × 1029)	6			0.118215
7	(1228 × 1029)	6	Denso VS-6577GM-B	42	0.131076
	(1228 × 1029)	6			0.121817
	(640 × 480)	not provided			0.109997
8	(1228 × 1029)	8	Denso VS-6577GM-B	42	0.143151
	(1228 × 1029)	8			0.139296
	(640 × 480)	not provided			0.113589

The number of robot positions is n, and rrmse is the reprojection root-mean-square error from the camera calibration step

field settings.[1] 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 using Zhang's method [17] to estimate the matrices $\mathbf{A}_i$. For datasets 7 and 8, the third camera is a commodity RGB-D (Red-Green-Blue-Depth) camera; we calibrated only the color camera in this work. Figure 2 shows the arrangement of cameras in dataset 7 as well as sample images from one position of the robot. Figure 3 shows both of the robots used in this work; two 6-axis robot arms were used, a Denso VS-6577GM-B robot arm rigidly mounted to the floor and a Denso VM-60BIG robot arm mounted on a mobile platform.

In the rest of this section, we discuss some of the implementation details of our collection of proposed methods, and then discuss the error metrics used in reporting the performance results of the comparison of the proposed methods.

3.1 Implementation details

For the two proposed classes of cost functions, we use the implementation of the Levenberg–Marquardt method found in the software package *Ceres* [1]. 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 [9].

For the first class of cost functions using simultaneous and separable versions of c_1 and c_2, the initial solutions of $\mathbf{p}_X$ and $\mathbf{p}_Z$ are set such that the corresponding directed cosine matrix

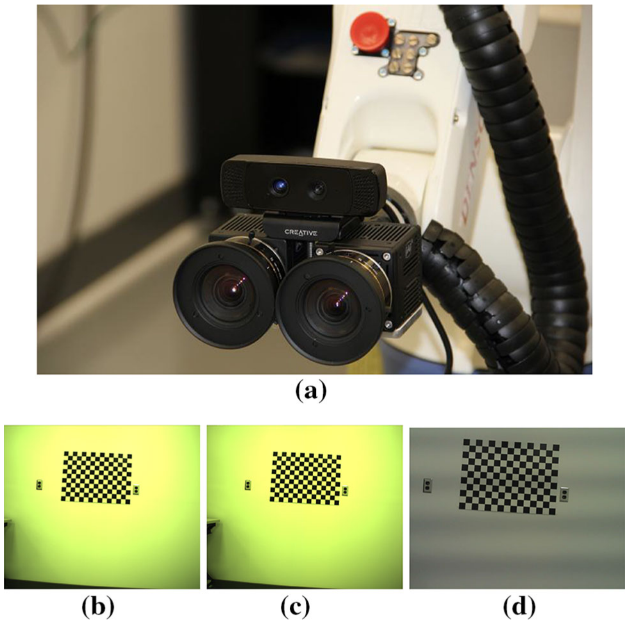
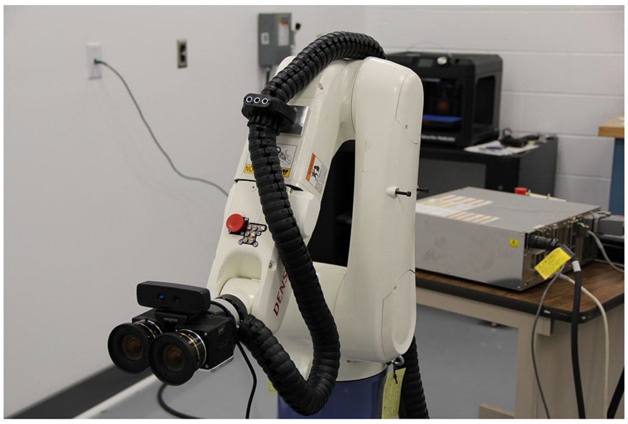

(a) (b) (c) (d)

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. **a** The arrangement of cameras in dataset 7, **b** camera 0, **c** camera 1, and **d** camera 2

is a 3×3 identity matrix, and the translation components are set to zero.[2] As mentioned previously, the second class of cost functions rp_1 and rp_2 is sensitive to initial solutions. To find an approximate solution to the minimization problem of

[1] All of the datasets are available from [14].

[2] Various different initial solutions were tested, and there was small or negligible difference in the solution quality versus using an identity matrix for rotation matrices and translation component with all elements zero. For this reason, we conclude that for the experiments covered in this paper, the first class of methods is not sensitive to initial solutions.


(a)

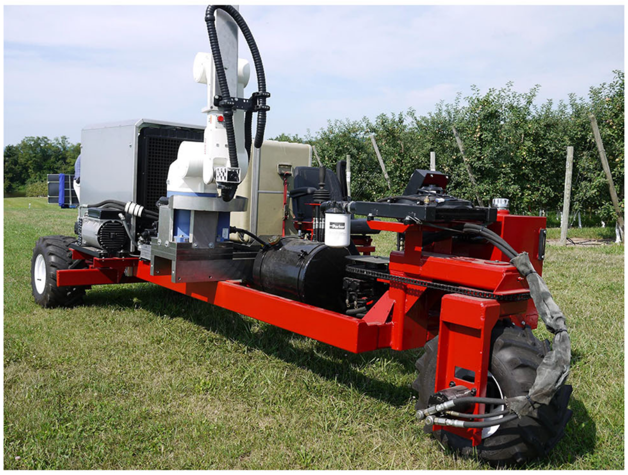

(b)

Fig. 3 Two 6-axis robot arms used to generate the performance evaluation datasets. In Fig. 3a, three cameras are mounted on the end-effector as in dataset 7, and this robot arm is rigidly attached to the laboratory floor. Fig. 3b shows the larger robot arm mounted on a mobile platform with one camera (Fig. 3b image is courtesy of Edwin Winzeler). **a** Denso VS-6577GM-B and **b** DENSO VM-60BIG

rp_1 method that was shown in Eq. 16, we used the simultaneous solution using the c_2 cost function (problem specified in Eq. 10) as an initial solution. To find an approximate solution to the problem of the rp_2 method shown in Eq. 17, the solution from Eq. 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 [2]: closed form and iterative, Hirsh et al. [3] and Li et al. [5] based on the dual quaternion and Kronecker product, and the work of Shah [10]. Concerning the implementation details of Li et al. [5] methods, we draw the reader's attention that the computation of coefficients λ_1 and λ_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 [5] 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 Sect. 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 Eq. 2 and shown in Eq. 24.

$$e_{R1} = \frac{1}{n}\sum_{i=0}^{n-1} ||\mathbf{R}_{A_i}\mathbf{R}_X - \mathbf{R}_Z\mathbf{R}_{B_i}||_F^2 \tag{24}$$

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 Eq. 2. We find the relative rotation between the two sides (Eq. 25) and then compute the angle of the relative rotation using the axis-angle representation, which is represented as angle() in Eq. 26; in the results shown in Sect. 5, its units are degrees.

$$\mathbf{R}_i = (\mathbf{R}_Z\mathbf{R}_{B_i})^T(\mathbf{R}_{A_i}\mathbf{R}_X) \tag{25}$$

$$e_{R2} = \frac{1}{n}\sum_{i=0}^{n-1} \text{angle}(\mathbf{R}_i) \tag{26}$$

3.2.2 The mean translation error

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

$$e_t = \frac{1}{n}\sum_i ||(\mathbf{R}_{A_i}\mathbf{t}_X + \mathbf{t}_{A_i}) - (\mathbf{R}_{Z_i}\mathbf{t}_B + \mathbf{t}_{Z_i})||^2 \tag{27}$$

3.2.3 The mean combined rotation and translation error

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

$$e_C = \frac{1}{n}\sum_i ||\mathbf{A}_i\mathbf{X} - \mathbf{Z}\mathbf{B}_i||_F^2 \tag{28}$$

3.2.4 The reprojection root-mean-squared error (rrmse)

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

$$\text{rrmse}=\sqrt{\frac{1}{mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}||\overrightarrow{\mathbf{x}}_{ij}-f(\mathbf{k},\left[\mathbf{ZB}_i\mathbf{X}^{-1}\right]_{3\times 4}\overrightarrow{\mathcal{X}}_j)||^2} \tag{29}$$

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 Sect. 2.2: $\overrightarrow{\mathbf{x}}_{ij}$ is the jth image point from the ith robot position and corresponds to the three-dimensional point $\overrightarrow{\mathcal{X}}_j$. First, we estimate the most likely three-dimensional point (represented by $\hat{\mathcal{Y}}_j$) that generated the n $\overrightarrow{\mathbf{x}}_{ij}$ image points by solving the minimization problem in Eq. 30.

$$\hat{\mathcal{Y}}_j=\underset{\mathcal{Y}_j}{\text{argmin}}\sum_{i=0}^{n-1}||\overrightarrow{\mathbf{x}}_{ij}-f(\mathbf{k},\left[\mathbf{ZB}_i\mathbf{X}^{-1}\right]_{3\times 4}\mathcal{Y}_j)||^2 \tag{30}$$

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

$$\text{rae}=\frac{1}{m}\sum_{j=0}^{m-1}||\hat{\mathcal{Y}}_j-\overrightarrow{\mathcal{X}}_j||^2 \tag{31}$$

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 $\mathbf{A}_i$, $\mathbf{B}_i$, $\mathbf{X}$, and $\mathbf{Z}$ 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 [10], Sect. 5.1. Briefly, the rotation components of $\mathbf{A}_i$, $\mathbf{X}$, and $\mathbf{Z}$ 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 $i = 1, 2, ..., 25$. Then, the homogeneous matrix $\mathbf{B}_i$ is computed as:

$$\mathbf{B}_i=\mathbf{Z}^{-1}\mathbf{A}_i\mathbf{X} \tag{32}$$

Noise is then introduced to $\mathbf{B}_i$, 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. η is the parameter controlling the magnitude of the random noise added to $\mathbf{B}_i$. $\eta \in (0, 0.25]$ and is evenly spaced over 19 values along that interval. Data were generated for ten trials of the experiment.

The protocol above sets the translation components within the interval of (0, 1), which reflects a bias toward 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 $\mathbf{X}$ and $\mathbf{Z}$ are known for the simulated datasets, for each calibration method the difference between the rotation and translation components is 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 $\hat{\mathbf{R}}_X$ be the rotation matrix estimated by a calibration method for a simulated dataset, and $\mathbf{R}_X$ be the ground truth rotation matrix for $\mathbf{X}$. Then, the error in the rotation component is e_{RX}:

$$e_{RX}=||\hat{\mathbf{R}}_X-\mathbf{R}_X||_F \tag{33}$$

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

4.1.2 Translation error

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

$$e_{tX}=||\hat{\mathbf{t}}_X-\mathbf{t}_X|| \tag{34}$$

Table 2 Comparison of methods using the error metrics described in Sect. 3.2 for Dataset 1

Method	Time (s)	e_{R1}	e_{R2} (degrees)	e_t (mm)	e_C	rrmse (pixels)	rae (mm)
Zhuang et al. [18]	0.025	0.000101190	0.336407	1.62805	233.3	4.35341	434.012
Dornaika and Horaud [2]: Closed Form	0.01	0.261051	19.5162	61.24160	330047	762.333	195437.640
Dornaika and Horaud [2]: Iterative	3.464	1.947	58.652	56.7113	283025	272.697	231374732
Hirsh et al. [3]	2.273	0.000098372	0.331641	2.02068	359.3	3.68820	189.545
Li et al. [5]: Dual Quaternion	0.494	0.000174349	0.506214	1.90747	320.2	4.30692	103.749
Li et al. [5]: Kronecker product	0.005	0.000102497	0.347818	2.25149	446.1	5.41810	388.55
Shah [10]	0.001	0.000098372	0.331641	1.63616	235.577	4.10190	403.674
c_1, Euler angles, simultaneous	0.524	0.0002675	0.622864	1.56439	215.365	12.13	772.895
c_1, axis angle, simultaneous	0.435	0.000108493	0.364398	1.56439	215.364	3.62686	268.006
c_1, quaternion, simultaneous	0.458	0.000108522	0.364739	1.56439	215.364	3.62641	266.815
c_2, Euler angles, simultaneous	0.511	0.000111989	0.367257	1.77959	278.692	1.67244	5.82014
c_2, axis angle, simultaneous	0.423	0.00010333	0.350712	1.72619	262.215	1.66791	5.01893
c_2, quaternion, simultaneous	0.556	0.000103324	0.350664	1.72618	262.215	1.66771	5.00157
c_1, Euler angles, separable	1.54	0.0000983715	0.331638	1.63597	235.524	4.09968	403.114
c_1, axis angle, separable	0.981	0.0000983714	0.331641	1.63614	235.573	4.10055	403.335
c_1, quaternion, separable	2.01	0.0000983715	0.331634	1.63566	235.433	4.09923	403.33
c_2, Euler angles, separable	1.331	0.0000983714	0.331642	1.75085	269.761	2.29073	5.64685
c_2, axis angle, separable	0.868	0.0000983714	0.331637	1.75079	269.745	2.29095	5.645626
c_2, quaternion, separable	0.895	0.0000983714	0.331642	1.75099	269.806	2.29097	5.661261
rp_1, Euler angles	14.548	0.000127916	0.384648	1.96216	338.808	1.56905	0.2382108
rp_1, axis angle	15.507	0.000127963	0.384498	1.96183	338.691	1.56905	0.2373199
rp_1, quaternion	15.622	0.000127963	0.384498	1.96183	338.691	1.56905	0.23732
rp_2, Euler angles	63.688	0.0197266	5.6824	8.41622	6233.3	1.36292	0.5235261
rp_2, axis angle	73.939	0.0197274	5.68251	8.41609	6233.11	1.36292	0.5241948
rp_2, quaternion	153.515	0.0198761	5.70398	8.41867	6236.93	1.3629	0.5233601

and the same formula follows to compute the translation error for $\mathbf{Z}$, 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 Sect. 3.2 and the eight real datasets described in Table 1.[3] These comparison results also include the running time for our implementation of each method in seconds.

In Tables 2, 3, 4, 5, and 6, the first seven rows correspond to the seven comparison methods from the literature, listed in chronological order: Zhuang et al. [18], Dornaika and Horaud [2]: closed form and iterative, Hirsh et al. [3] and Li et al. [5] 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.[4] The comparison methods were not evaluated on Datasets 6, 7, and 8 and are not shown in Tables 7, 8, and 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 Figs. 4, 5, 6, and 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.

[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.

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

Method	Time (s)	e_{R1}	e_{R2} (degrees)	e_t (mm)	e_C	rrmse (pixels)	rae (mm)
Zhuang et al. [18]	0.014	0.0000625721	0.288076	1.307	47.8309	11.8458	52.65020
Dornaika and Horaud [2]: Closed Form	0.003	0.404248	24.1399	79.3305	176214	2959.58	5580067.998
Dornaika and Horaud [2]: Iterative	1.206	1.30484	45.2478	74.8997	157080	1329.87	173943689.69
Hirsh et al. [3]	0.35	0.0000199547	0.149794	0.888279	22.0931	4.66805	2.500515
Li et al. [5]: Dual Quaternion	0.07	0.0000251894	0.185176	0.75443	15.9366	5.24936	14.9587
Li et al. [5]: Kronecker product	0.0001	0.0000227466	0.169176	0.972035	26.4559	5.67147	23.201702
Shah [10]	0.0001	0.0000199547	0.149793	0.834843	19.515	4.6186	2.387767
c_1, Euler angles, simultaneous	0.239	0.0000220172	0.164733	0.736446	15.1859	4.35244	5.125558
c_1, axis angle, simultaneous	0.14	0.0000220365	0.164714	0.736446	15.1859	4.34548	4.963828
c_1, quaternion, simultaneous	0.149	0.0000333404	0.196254	0.736446	15.1859	7.43213	15.024401
c_2, Euler angles, simultaneous	0.199	0.000022573	0.168609	0.800019	17.9209	3.96036	0.0863250
c_2, axis angle, simultaneous	0.14	0.0000227808	0.171629	0.804379	18.1168	3.96272	0.095928
c_2, quaternion, simultaneous	0.181	0.000022573	0.168604	0.800019	17.9209	3.96047	0.086529
c_1, Euler angles, separable	0.502	0.0000199547	0.149794	0.834828	19.5143	4.61824	2.38108
c_1, axis angle, separable	0.323	0.0000199547	0.149794	0.834836	19.5146	4.61825	2.38061
c_1, quaternion, separable	0.689	0.0000199547	0.149795	0.834902	19.5177	4.61847	2.37722
c_2, Euler angles, separable	0.301	0.0000199547	0.149793	0.84598	20.0391	4.48836	0.069070
c_2, axis angle, separable	0.28	0.0000199547	0.149795	0.845938	20.0371	4.48847	0.069380
c_2, quaternion, separable	0.325	0.0000199547	0.149794	0.84595	20.0377	4.48849	0.069498
rp_1, Euler angles	5.789	0.0000331533	0.184911	1.00207	28.1162	3.74123	0.0295049
rp_1, axis angle	6.189	0.0000331533	0.184911	1.00207	28.1162	3.74123	0.0295049
rp_1, quaternion	6.22	0.0000331533	0.184911	1.00207	28.1162	3.74123	0.0295049
rp_2, Euler angles	66.702	0.00296179	2.1996	6.31123	1115.29	3.18114	0.44990
rp_2, axis angle	48.242	0.00276524	2.12493	6.11936	1048.51	3.1713	0.44034
rp_2, quaternion	69.996	0.00295924	2.19865	6.31096	1115.19	3.18101	0.46199

5.1 Discussion of comparison methods on real datasets

We mention here that the Dornaika and Horaud iterative method [2] does not converge for any datasets using the *Ceres* solver [1], where derivatives are automatically computed by the software. In comparison, in our previous work [15], where we used the *levmar* solver [6], 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. [3] 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 [2] 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 (rrmse), the Hirsh et al. [3] and Shah [10] methods both perform generally the best, with Shah's method resulting in lower 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 rrmse in Datasets 1 (3.68829 pixels), but large rrmse in Dataset 5 (238.041 pixels), though still lower compared to Shah's method's rrmse value (439.512 pixels).

The first conclusion drawn from Tables 2, 3, 4, 5, and 6, 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

Table 4 Comparison of methods using the error metrics described in Sect. 3.2 for Dataset 3

Method	Time (s)	e_{R1}	e_{R2} (degrees)	e_t (mm)	e_C	rrmse (pixels)	rae (mm)
Zhuang et al. [18]	0.013	0.765909	35.737	84.21	255288	1752.74	$1.08832 \times 10_8$
Dornaika and Horaud [2]: Closed Form	0.005	0.0760832	8.90491	97.4986	342216	1512.71	$8.28684 \times 10_{16}$
Dornaika and Horaud [2]: Iterative	3.083	7.01367	139.069	75.6174	205854	1179.24	$1.66168 \times 10_7$
Hirsh et al. [3]	3.281	0.000146461	0.400659	6.17446	1372.46	39.071	3729.96
Li et al. [5]: Dual Quaternion	0.125	0.000229668	0.55199	5.34571	1028.76	20.7324	1896.01
Li et al. [5]: Kronecker product	0.002	00.000152553	0.42043	4.82104	836.727	31.7476	1832.47
Shah [10]	0.0001	0.000146461	0.400666	2.76823	275.872	9.10238	123.061
c_1, Euler angles, simultaneous	0.472	0.000151291	0.416122	2.652	253.191	7.58227	144.659
c_1, axis angle, simultaneous	0.225	0.000151289	0.416115	2.652	253.191	7.58583	144.482
c_1, quaternion, simultaneous	0.195	0.00015248	0.426517	2.652	253.191	7.83721	143.142
c_2, Euler angles, simultaneous	0.256	0.000153451	0.425433	2.8537	293.17	3.55362	1.92736
c_2, axis angle, simultaneous	0.221	0.000153445	0.424749	2.85365	293.16	3.5553	1.96219
c_2, quaternion, simultaneous	0.187	0.000153914	0.42986	2.85616	293.676	3.5587	1.74438
c_1, Euler angles, separable	0.843	0.000146461	0.400655	2.76854	275.933	9.11944	123.812
c_1, axis angle, separable	0.504	0.000146461	0.400679	2.76841	275.908	9.11817	123.833
c_1, quaternion, separable	1.293	0.000146461	0.400658	2.76854	275.933	9.11947	123.798
c_2, Euler angles, separable	0.458	0.000146461	0.400658	2.85224	292.871	7.37805	1.5171
c_2, axis angle, separable	0.447	0.000146461	0.400658	2.85223	292.869	7.37791	1.51745
c_2, quaternion, separable	0.506	0.000146461	0.400658	2.85222	292.866	7.3776	1.51769
rp_1, Euler angles	10.706	0.000158848	0.433885	2.92055	307.066	2.89018	1.45425
rp_1, axis angle	9.917	0.000158848	0.433885	2.92055	307.066	2.89018	1.45426
rp_1, quaternion	10.521	0.000158848	0.433885	2.92055	307.066	2.89018	1.45426
rp_2, Euler angles	46.056	0.000900894	1.17365	3.12839	352.326	2.67963	1.5875
rp_2, axis angle	49.188	0.000900894	1.17365	3.12839	352.326	2.67963	1.5875
rp_2, quaternion	87.846	0.000900897	1.17365	3.12839	352.327	2.67963	1.58896

the positions chosen for calibration, the robot model, and the way in which the robot is mounted, which may introduce new errors. Recall from Sect. 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. [5] 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 rrmse as compared to c_1 methods. Exceptions are Datasets 7 and 8, which have very similar values of e_C and 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 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

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

Method	Time (s)	e_{R1}	e_{R2} (degrees)	e_t (mm)	e_C	rrmse (pixels))	rae (mm)
Zhuang et al. [18]	0.003	0.00775597	2.41835	92.201	170021	288.166	36039.5
Dornaika and Horaud [2]: Closed Form	0.003	0.010101	2.57979	122.624	300733	420.053	659311
Dornaika and Horaud [2]: Iterative	1.692	7.97719	173.278	8.05142	1304.49	1235	$1.11892 \times 10_{17}$
Hirsh et al. [3]	3.409	0.0000954823	0.350316	6.06421	735.493	6.22611	76.2172
Li et al. [5]: Dual Quaternion	0.018	0.0675828	10.5063	63.6596	81050.9	189.191	232276
Li et al. [5]: Kronecker product	0.001	0.000120951	0.415269	7.2863	1061.8	21.69	4198.79
Shah [10]	0.0001	0.0000954823	0.350316	3.12994	195.931	2.27361	22.7013
c_1, Euler angles, simultaneous	0.195	0.000124344	0.422193	3.01615	181.943	2.85969	9.30925
c_1, axis angle, simultaneous	0.129	0.000124321	0.42244	3.01615	181.943	2.85827	9.15721
c_1, quaternion, simultaneous	0.111	0.000124662	0.422864	3.01615	181.943	2.95739	9.59481
c_2, Euler angles, simultaneous	0.142	0.0000973168	0.35385	3.12497	195.309	2.38787	1.7146
c_2, axis angle, simultaneous	0.12	0.00009721	0.353469	3.12309	195.073	2.38756	1.72392
c_2, quaternion, simultaneous	0.107	0.0000973516	0.354033	3.12428	195.222	2.38988	1.72006
c_1, Euler angles, separable	0.531	0.0000954823	0.350316	3.13017	195.959	2.27045	22.7467
c_1, axis angle, separable	0.429	0.0000954823	0.350316	3.13001	195.939	2.27319	22.7232
c_1, quaternion, separable	0.843	0.0000954824	0.35032	3.13012	195.953	2.27069	22.7303
c_2, Euler angles, separable	0.255	0.0000954823	0.350316	3.15556	199.151	1.83788	2.11546
c_2, axis angle, separable	0.255	0.0000954823	0.350316	3.15555	199.151	1.8379	2.1149
c_2, quaternion, separable	0.313	0.0000954823	0.350316	3.15555	199.151	1.83791	2.11489
rp_1, Euler angles	6.454	0.000103036	0.367112	3.33431	222.352	1.60321	1.43699
rp_1, axis angle	6.843	0.000103036	0.367112	3.33431	222.352	1.60321	1.43699
rp_1, quaternion	11.683	0.000103036	0.367113	3.3343	222.351	1.60321	1.43714
rp_2, Euler angles	18.414	0.000743268	1.08192	4.22232	356.561	1.50483	1.48145
rp_2, axis angle	22.174	0.000743268	1.08192	4.22232	356.561	1.50483	1.48145
rp_2, quaternion	46.835	0.00074373	1.08227	4.22268	356.622	1.50549	1.48562

the simultaneous versions have a maximum rrmse of 3.89 pixels. The separable methods have a much greater 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 $\mathbf{AX} = \mathbf{ZB}$ (and its variations), the second class of methods consistently gives results with lower values of 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 rrmse using rp_1 (where intrinsic camera calibration parameters are treated as constants) versus rp_2 (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 *levmar* solver [6] and a provided Jacobian matrix. In that work, when tested on Dataset 1, the rrmse was 3.62709 pixels. Whereas using the *Ceres* solver [1] with automatic Jacobian matrices for the same dataset, the 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 rrmse occurs even in the cases when the combined rotation and translation error (e_C) between the three rotation representations, using *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.

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

Method	Time (s)	e_{R1}	e_{R2} (degrees)	e_t (mm)	e_C	rrmse (pixels)	rae (mm)
Zhuang et al. [18]	0.004	0.0118914	3.61164	113.639	180792	1190	862073
Dornaika and Horaud [2]: Closed Form	0.002	0.0505229	7.58041	58.6203	48108.8	211.081	150077
Dornaika and Horaud [2]: Iterative	1.116	4.36192	95.1924	31.289	13710.4	1206	$1.36857 \times 10_{10}$
Hirsh et al. [3]	0.354	0.0000261473	0.199554	152.636	326170	238.041	$1.01182 \times 10_{11}$
Li et al. [5]: Dual Quaternion	0.008	0.0100287	4.05685	85.8321	103140	65.5218	62885.7
Li et al. [5]: Kronecker product	0.001	0.0000285929	0.203356	4.84945	329.24	7.16294	236.416
Shah [10]	0.0001	0.0000261584	0.199265	210.092	617943	439.512	$2.12246 \times 10_{17}$
c_1, Euler angles, simultaneous	0.085	0.0000338659	0.226466	1.28684	23.1834	3.88777	31.421
c_1, axis angle, simultaneous	0.093	0.0000287485	0.206804	1.28684	23.1833	3.32417	17.362
c_1, quaternion, simultaneous	0.08	0.0000313139	0.203908	1.28684	23.1834	3.47606	20.3247
c_2, Euler angles, simultaneous	0.089	0.0000285658	0.210174	1.99459	55.6977	1.0838	16.4967
c_2, axis angle, simultaneous	0.092	0.0000265726	0.202214	1.93993	52.6869	1.05499	13.9331
c_2, quaternion, simultaneous	0.095	0.0000265728	0.202212	1.93994	52.6872	1.05499	13.9361
c_1, Euler angles, separable	0.25	0.0000261473	0.199555	68.3986	65497.1	97.0019	180519
c_1, axis angle, separable	0.265	0.0000261473	0.19954	58.444	47819.8	81.2711	125430
c_1, quaternion, separable	0.362	0.0000261473	0.199555	34.2755	16447.3	47.188	33338.9
c_2, Euler angles, separable	0.154	0.0000261473	0.199546	83.4457	97484.7	80.4444	124079
c_2, axis angle, separable	0.159	0.0000261473	0.199555	56.1978	44214.7	48.1819	35443.8
c_2, quaternion, separable	0.173	0.0000261473	0.199555	56.4971	44687	48.4882	36033.6
rp_1, Euler angles	3.445	0.000112519	0.393463	4.43837	275.788	0.648684	1.21898
rp_1, axis angle	3.158	0.000112519	0.393463	4.43836	275.787	0.648684	1.21894
rp_1, quaternion	4.593	0.000112517	0.393459	4.43833	275.783	0.648684	1.21906
rp_2, Euler angles	22.517	0.000647345	1.02564	9.02427	1140.13	0.605089	0.933008
rp_2, axis angle	19.253	0.00064732	1.02562	9.02516	1140.35	0.605088	0.928356
rp_2, quaternion	32.277	0.000643362	1.02226	7.30616	747.319	0.605207	0.955197

Table 7 Comparison of methods using the error metrics described in Sect. 3.2 for Dataset 6. rrmse is computed for each camera

Method	Time (s)	e_{R1}	e_{R2} (degrees)	e_t (mm)	e_C	rrmse$_0$ (pixels)	rrmse$_1$ (pixels)	rae (mm)
c_1, Euler angles, simultaneous	0.386	0.0000257326	0.193937	1.24552	21.0586	2.31762	2.00706	16.6109
c_1, axis angle, simultaneous	0.407	0.0000257326	0.193936	1.24552	21.0586	2.31762	2.00706	16.6109
c_1, quaternion, simultaneous	0.424	0.0000257326	0.193936	1.24552	21.0586	2.31762	2.00706	16.6109
c_2, Euler angles, simultaneous	0.242	0.0000280977	0.206962	1.9874	53.4269	0.766459	0.774432	15.6238
c_2, axis angle, simultaneous	0.248	0.0000249891	0.193788	1.90496	49.1943	0.750998	0.734177	12.6656
c_2, quaternion, simultaneous	0.262	0.000024989	0.193789	1.90496	49.194	0.750993	0.734177	12.6686
c_1, Euler angles, separable	0.708	0.0000234965	0.186335	67.016	60837.1	68.7115	62.0413	141154
c_1, axis angle, separable	0.823	0.0000234965	0.186335	57.4164	44653.1	57.634	52.0727	100740
c_1, quaternion, separable	1.043	0.0000234965	0.186335	33.8565	15522.1	33.5017	30.1025	27709.6
c_2, Euler angles, separable	0.428	0.0000234965	0.186322	81.9702	91006.5	57.0089	51.6298	101915
c_2, axis angle, separable	0.46	0.0000234965	0.186335	55.2702	41375.6	34.1357	30.7028	29771.8
c_2, quaternion, separable	0.474	0.0000234965	0.186335	55.4786	41688.2	34.2896	30.8437	30120.1
rp_1, Euler angles	9.117	0.000114934	0.401383	4.58804	284.219	0.461123	0.470814	1.49774
rp_1, axis angle	9.785	0.000114934	0.401383	4.58803	284.218	0.461123	0.470814	1.49768
rp_1, quaternion	16.697	0.000114932	0.401379	4.588	284.214	0.461122	0.470813	1.49852
rp_2, Euler angles	83.205	0.000583753	0.973613	9.75874	1296.52	0.430003	0.435863	1.22988
rp_2, axis angle	89.531	0.000583745	0.973605	9.75883	1296.54	0.430003	0.435863	1.2324
rp_2, quaternion	88.379	0.00069386	1.06199	10.71	1551.12	0.429859	0.44217	1.25663

Table 8 Comparison of methods using the error metrics described in Sect. 3.2 for Dataset 7. rrmse is computed for each camera

Method	Time (s)	e_{R1}	e_{R2} (degrees)	e_t (mm)	e_C	rrmse$_0$ (pixels)	rrmse$_1$ (pixels)	rrmse$_2$ (pixels)	rae (mm)
c_1, Euler angles, simultaneous	1.548	0.000037808	0.223619	1.01891	37.8961	1.70065	1.70483	2.05211	25.5861
c_1, axis angle, simultaneous	1.349	0.0000344302	0.216382	1.01891	37.8961	1.76765	1.86726	1.57613	27.7002
c_1, quaternion, simultaneous	1.13	0.00003705	0.219817	1.01891	37.8961	1.59012	1.59645	1.89843	21.5528
c_2, Euler angles, simultaneous	1.273	0.0000380638	0.234888	1.3724	68.5225	0.777057	0.689395	0.464227	9.86823
c_2, axis angle, simultaneous	1.32	0.0000340799	0.214434	1.32509	63.8607	0.740757	0.672692	0.464665	7.11799
c_2, quaternion, simultaneous	1.656	0.0000340798	0.214439	1.32509	63.8605	0.740759	0.672692	0.464664	7.12014
c_1, Euler angles, separable	3.31	0.0000255085	0.18337	1.11447	45.1969	1.63226	1.62237	0.928388	14.4895
c_1, axis angle, separable	3.18	0.0000255086	0.183363	1.11471	45.2157	1.63213	1.62221	0.928315	14.4774
c_1, quaternion, separable	6.124	0.0000255086	0.183362	1.11481	45.224	1.63209	1.62216	0.9283	14.4771
c_2, Euler angles, separable	2.506	0.0000255086	0.18335	1.11991	45.6158	1.62662	1.61233	0.924561	9.16612
c_2, axis angle, separable	2.669	0.0000255086	0.183363	1.11979	45.6056	1.62658	1.61229	0.924541	9.16691
c_2, quaternion, separable	2.578	0.0000255086	0.183363	1.11979	45.6061	1.62659	1.6123	0.92455	9.16742
rp_1, Euler angles	55.371	0.000106375	0.383085	2.72123	281.057	0.56	0.562183	0.414597	1.12422
rp_1, axis angle	61.656	0.000106375	0.383085	2.72123	281.057	0.56	0.562183	0.414597	1.12423
rp_1, quaternion	67.778	0.000106363	0.38307	2.72111	281.024	0.559993	0.562184	0.414596	1.12366
rp_2, Euler angles	299.438	0.000449848	0.845571	6.79131	1753	0.536306	0.524921	0.393133	1.17076
rp_2, axis angle	359.832	0.000449848	0.845571	6.79131	1753	0.536306	0.524921	0.393133	1.17062
rp_2, quaternion	503.785	0.000450395	0.845975	6.7963	1756.08	0.536307	0.524912	0.393125	1.17389

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 (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 ≤ 0.1 mm for all three rotation representations, and this pattern (rae ≤ 0.1 mm) is repeated in the separable versions of c_2 as compared to c_1 (rae ≤ 15 mm, rae ≤ 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 ≤ 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. [3] 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 performs as well as our proposed rp_1 and rp_2 methods with respect to reconstruction accuracy error, though for some datasets the methods of

Table 9 Comparison of methods using the error metrics described in Sect. 3.2 for Dataset 8. rrmse is computed for each camera

Method	Time (s)	e_{R1}	e_{R2} (degrees)	e_t (mm)	e_C	rrmse$_0$ (pixels)	rrmse$_1$ (pixels)	rrmse$_2$ (pixels)	rae (mm)
c_1, Euler angles, simultaneous	1.372	0.0000332003	0.208999	1.03046	34.7303	2.51556	3.04886	1.38591	35.3555
c_1, axis angle, simultaneous	1.191	0.0000215955	0.172318	1.03046	34.7303	1.47545	1.74632	0.830485	9.76088
c_1, quaternion, simultaneous	0.998	0.0000247851	0.177514	1.03046	34.7303	1.67143	1.81211	0.984151	10.5367
c_2, Euler angles, simultaneous	1.114	0.0000330719	0.221522	1.42353	65.1598	0.805084	0.885025	0.429181	3.74888
c_2, axis angle, simultaneous	1.164	0.0000288111	0.197555	1.36356	60.1969	0.776654	0.861649	0.422205	2.18689
c_2, quaternion, simultaneous	1.452	0.0000288111	0.197561	1.36355	60.1956	0.776668	0.861652	0.422208	2.18551
c_1, Euler angles, separable	2.792	0.0000189285	0.157639	1.10155	39.413	1.82487	1.83353	0.954764	9.52457
c_1, axis angle, separable	2.878	0.0000189285	0.157637	1.10165	39.4201	1.82476	1.83337	0.954722	9.50306
c_1, quaternion, separable	4.815	0.0000189285	0.157637	1.10166	39.4209	1.82475	1.83337	0.954719	9.50292
c_2, Euler angles, separable	2.241	0.0000189285	0.157627	1.10651	39.7034	1.82112	1.81528	0.952705	4.28782
c_2, axis angle, separable	2.33	0.0000189285	0.157637	1.10644	39.6979	1.8211	1.81525	0.952688	4.28961
c_2, quaternion, separable	2.457	0.0000189285	0.157637	1.10644	39.6982	1.8211	1.81526	0.952697	4.28607
rp_1, Euler angles	44.95	0.0000769972	0.336918	2.50928	203.51	0.667398	0.745895	0.381425	0.901561
rp_1, axis angle	47.474	0.0000769971	0.336919	2.50928	203.509	0.667398	0.745895	0.381425	0.901595
rp_1, quaternion	47.888	0.0000769964	0.336917	2.50927	203.509	0.667399	0.745895	0.381425	0.901831
rp_2, Euler angles	249.902	0.000611282	0.980505	6.85716	1591.65	0.651493	0.722737	0.361992	1.00248
rp_2, axis angle	264.648	0.000611284	0.980507	6.85717	1591.66	0.651493	0.722737	0.361992	1.00167
rp_2, quaternion	267.476	0.000612111	0.981022	6.84548	1585.09	0.651488	0.722743	0.361669	1.01235

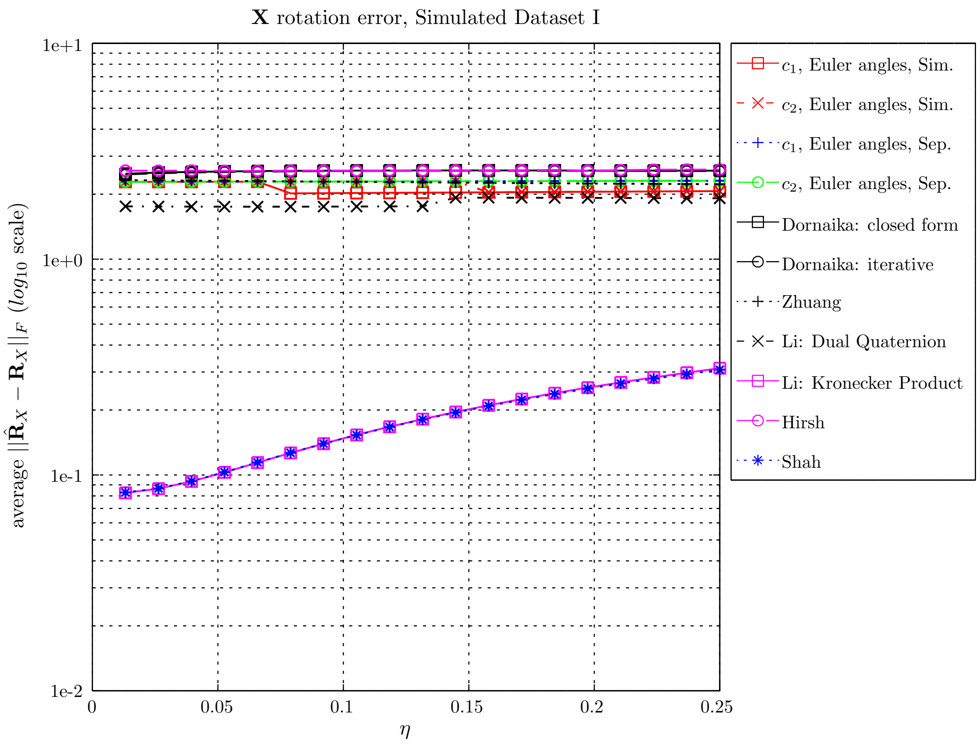
Hirsh et al. and Shah are comparable to the rae of our first class of proposed methods.

5.4 Discussion of the simulated experiments

The examination of the Simulated Dataset I and II results offers 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 (≤ 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.



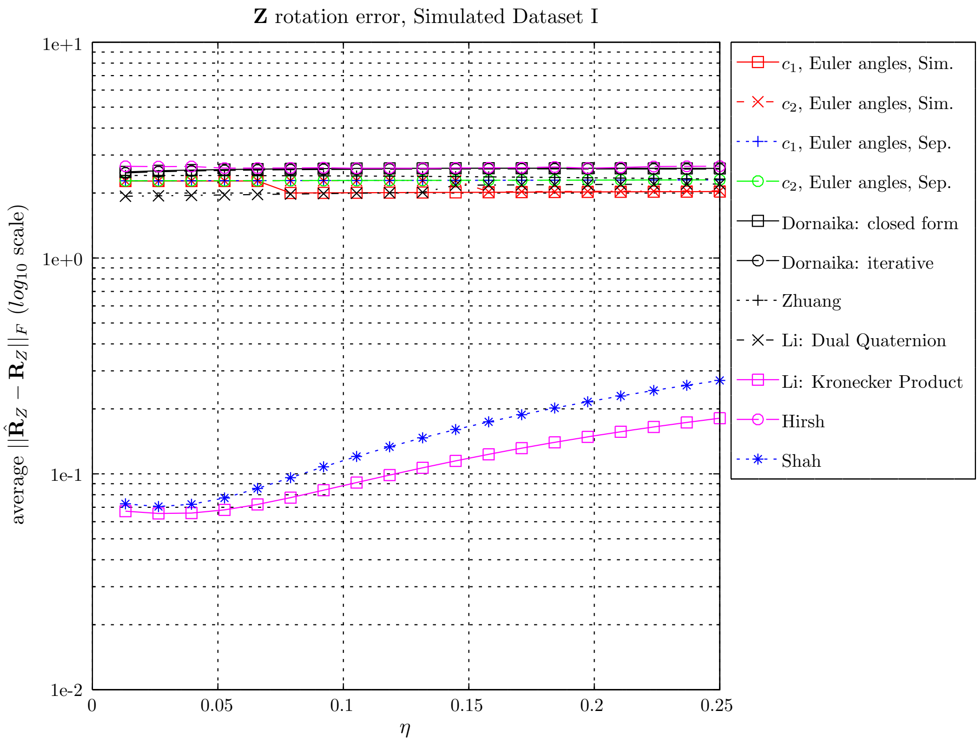
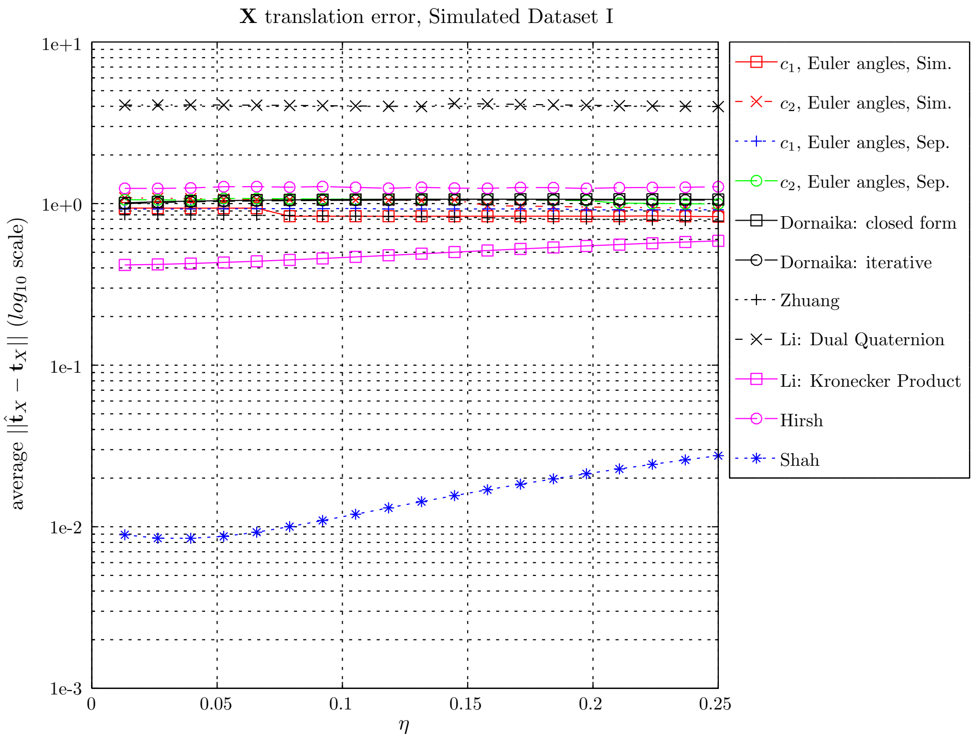


Fig. 4 Best viewed in color. Rotation error results for Simulated Dataset I, as compared to the ground truth **X** and **Z**, using the four metrics of Sect. 4.1

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. [3] and Shah [10], and the c_1 and c_2 separable versions from the first class of our proposed methods consistently give



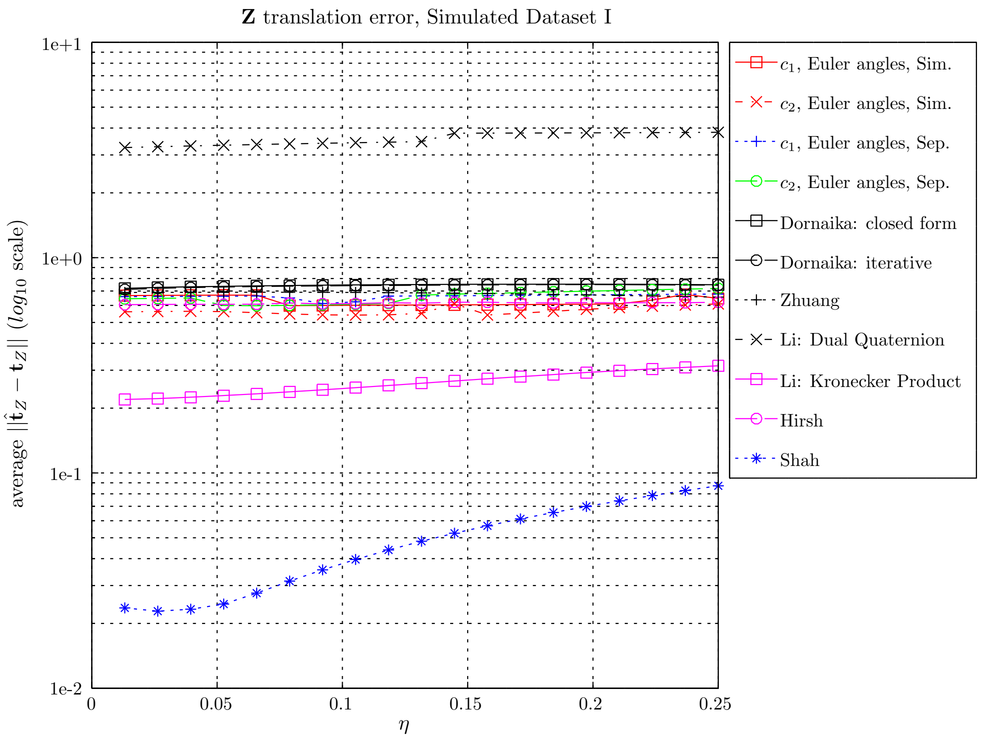
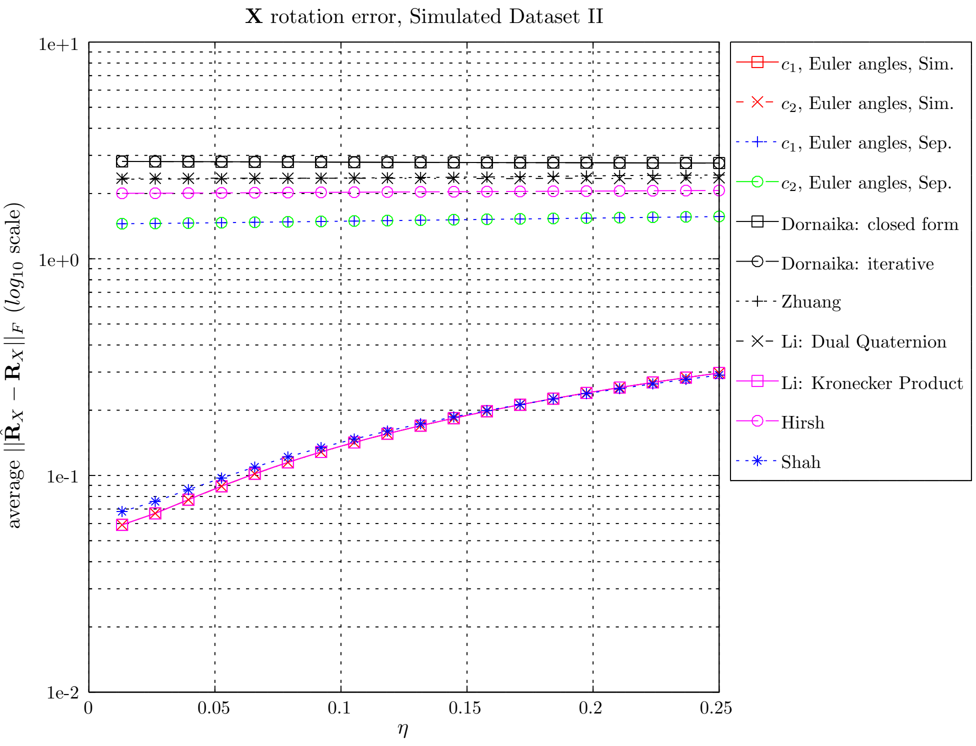


Fig. 5 Best viewed in color. Translation error results for Simulated Dataset I, as compared to the ground truth **X** and **Z**, using the four metrics of Sect. 4.1

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



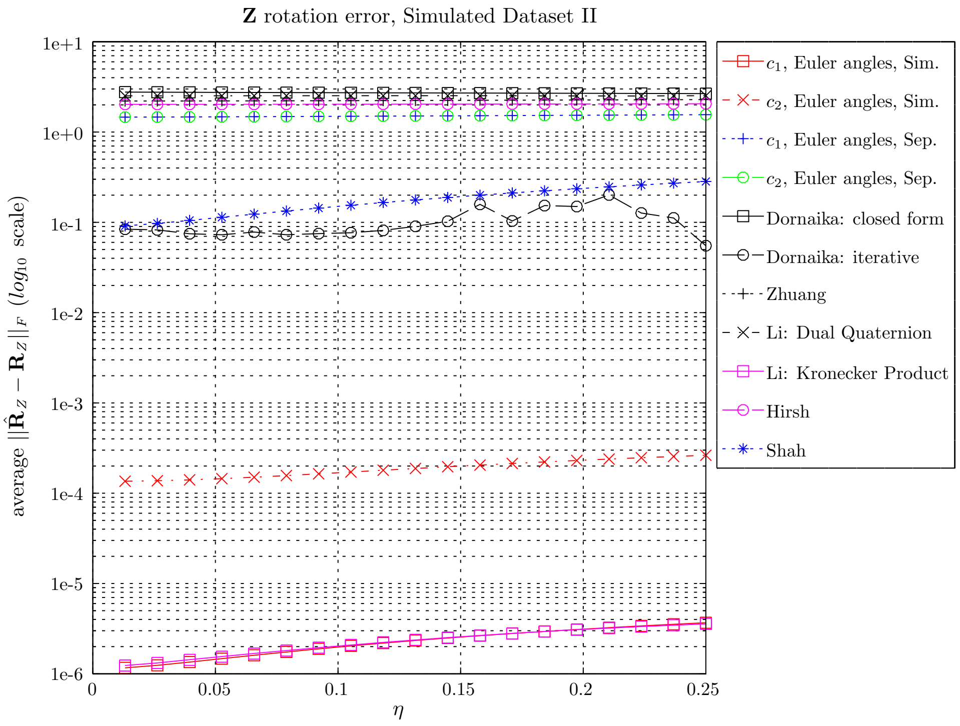
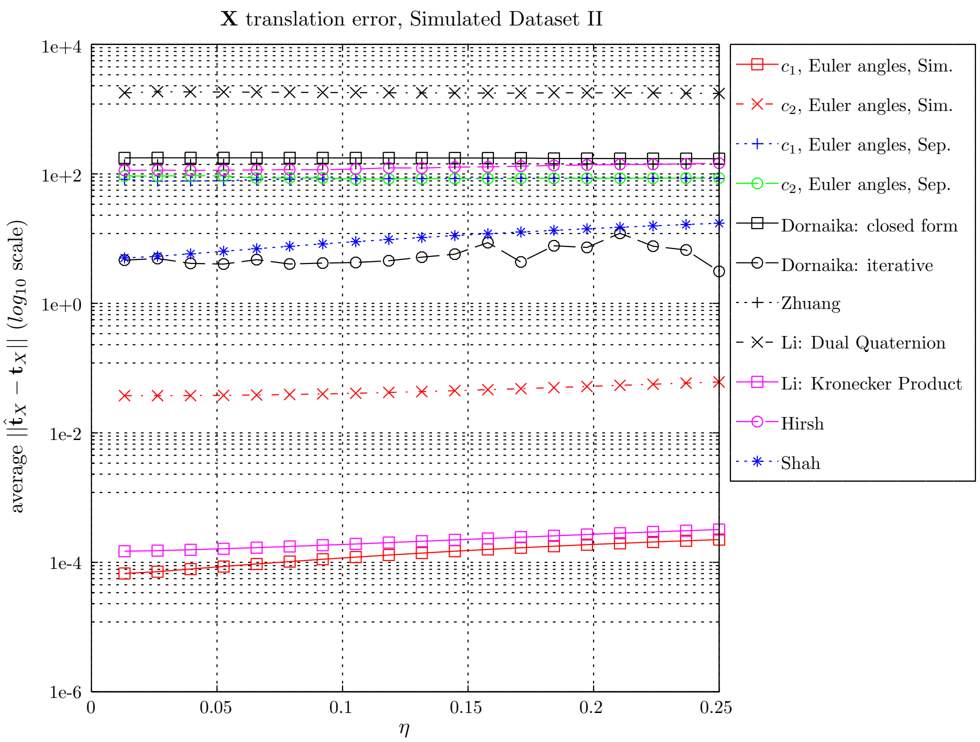


Fig. 6 Best viewed in color. Rotation error results for Simulated Dataset II, as compared to the ground truth **X** and **Z**, using the four metrics of Sect. 4.1

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 [1].



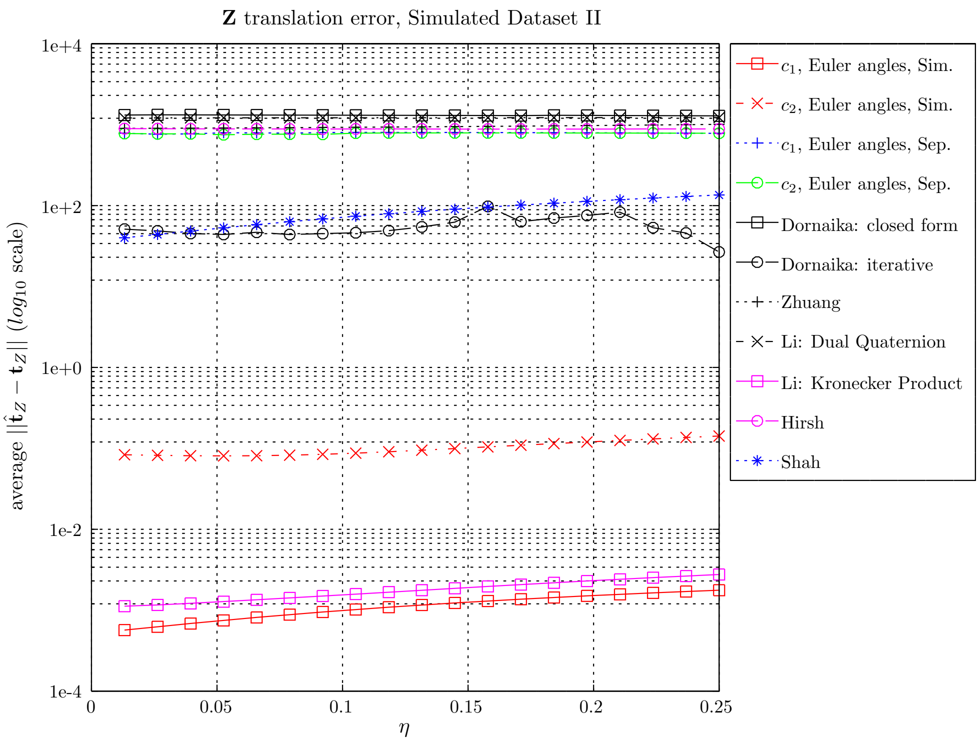


Fig. 7 Best viewed in color. Translation error results for Simulated Dataset II, as compared to the ground truth **X** and **Z**, using the four metrics of Sect. 4.1

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 minimization problems using three different rotation representations. Our collection of methods was evaluated relative to existing robot-world hand-eye calibration methods.

This collection of methods was extended to be used to calibrate multiple cameras mounted on one robot, and this was demonstrated on datasets with one to three cameras. To the best of our knowledge, there are currently no other robot-world hand-eye calibration methods for multiple cameras, and consequently we believe this work is the first attempt to solve such a problem. We also introduced a metric, called Reconstruction Accuracy Error, or rae, 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 $\leq$ 2mm, 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 $\leq$ 2mm resulted from evaluating the collection of 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 [15] and this paper for their helpful comments.

Compliance with ethical standards

Conflict of interest A. Tabb received a grant from the US National Science Foundation, grant number IOS-1339211.

References

1. Agarwal, S., Mierle, K., et al.: Ceres solver. http://ceres-solver.org
2. Dornaika, F., Horaud, R.: Simultaneous robot-world and hand-eye calibration. IEEE Trans. Rob. Autom. **14**(4), 617–622 (1998)
3. Hirsh, R.L., DeSouza, G.N., Kak, A.C.: An iterative approach to the hand-eye and base-world calibration problem. In: Robotics and automation, 2001. Proceedings 2001 ICRA. IEEE international conference on, vol. 3, pp. 2171–2176. IEEE (2001)
4. Horaud, R., Dornaika, F.: Hand-eye calibration. Int. J. Rob. Res. **14**(3), 195–210 (1995)
5. Li, A., Lin, W., Defeng, W.: Simultaneous robot-world and hand-eye calibration using dual-quaternions and Kronecker product. Int. J. Phys. Sci. **5**(10), 1530–1536 (2010)
6. Lourakis, M.: levmar: Levenberg–Marquardt nonlinear least squares algorithms in C/C++. http://www.ics.forth.gr/~lourakis/levmar/ (July 2004). Accessed 18 Nov 2014
7. Malti, A.: Hand-eye calibration with epipolar constraints: application to endoscopy. Rob. Auton. Syst. **61**(2), 161–169 (2013)
8. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11**(2), 431–441 (1963)
9. Opencv. http://opencv.org/. Version 2.4.9
10. Shah, M.: Solving the robot-world/hand-eye calibration problem using the Kronecker product. ASME J. Mech. Robot. **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. IEEE Trans. Rob. Autom. **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: 2013 IEEE conference on computer vision and pattern recognition (CVPR) (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
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, London (2000). http://dl.acm.org/citation.cfm?id=646271.685629
17. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **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. IEEE Trans. Rob. Autom. **10**(4), 549–554 (1994). doi:10.1109/70.313105

Amy Tabb Ph.D., is a Research Agricultural Engineer at the United States Department of Agriculture, Agricultural Research Service, Appalachian Fruit Research Laboratory. Her program focuses on the development of systems that operate in field conditions, particularly for determining shape features of plants and other objects. Consequently, her research interests include robust computer vision and robotics in field applications. Her education consisted of Ph.D. and M.S. degrees (2014 and 2012, respectively) in Electrical and Computer Engineering from Purdue University, a M.A. in music from Duke University in 2003, and a B.A. from Sweet Briar College in 2001 with double majors in Mathematics-Computer Science and Music. Dr. Tabb is a member of IEEE, IEEE's robotics and automation society as well as the computer society, and ASABE.

Khalil M. Ahmad Yousef Ph.D., is currently an Assistant Professor of the Department of Computer Engineering at the Hashemite University, Jordan. He is an IEEE senior member and the adviser of the IEEE Student Computer Society Chapter at the Hashemite University. Dr. Ahmad Yousef is a certified professional engineer from the Jordanian Engineering Association (JEA). He received his Ph.D. in Electrical and Computer Engineering from Purdue University, West Lafayette, USA, in 2013, his M.S. in Computer Engineering from Jordan University of Science and Technology, Jordan, in 2008, and his B.S. in Electrical and Computer Engineering from the Hashemite University in 2005. His research interests include computer vision, sensor fusion, image processing, optimization, and robotics.