Sensor Calibration, Modeling and Simulation

Qianli Ma
Gregory S. Chirikjian
Robot and Protein Kinematics Laboratory
Laboratory for Computational Sensing and Robotics
Department of Mechanical Engineering
The Johns Hopkins University
Baltimore, Maryland, 21218
Email: {mqianli1, gchirik1}@jhu.edu

For humanoid robots, sensors are a critical aspect of the entire robot system because they are in charge of receiving and perceiving the outer environment accurately. Due to the cost of real experiments on humanoids, sensor simulation comes in handy and plays an increasingly important role in robotics research. For applications in the real world, sensor calibration is the prerequisite for sensors to function properly and is the major focus of this chapter. An often used formulation of sensor calibration in robotics and computer vision is "AX=XB", where A, X, and B are rigid-body transformations with A and B given from sensor measurements, and X is the unknown rigid-body transformation to be calibrated. Many methods have been proposed to solve X given data streams of A and B under different scenarios. This chapter presents the most complete picture of the $AX = XB$ solvers up-to-date. First, a brief overview of the various important sensor calibration techniques is given and problems of interest are highlighted. Then, a detailed review of the various "AX=XB" research algorithms is presented. The notations of the selected algorithms are unified to the largest extent in order to show the interconnections between the selected methods in a straightforward way. Finally, the criterion for data selection and various error metrics are introduced, which are of critical importance for evaluating the performance of $AX = XB$ solvers.

Index terms— Sensor Calibration, Hand-Eye Calibration, Humanoid Robot, Review

Nomenclature

$SO(3)$	$SO(3) \doteq \{R \,\vert\, RR^T = R^T R = \mathbb{I}\ \det(R) = 1$ where $R \in \mathbb{R}^{3\times3}\}$
$SE(3)$	$SE(3) \doteq \{H \,\vert\, H = \begin{pmatrix} R\,\mathbf{t}; \mathbf{0}^T 1\end{pmatrix} \in \mathbb{R}^{4\times4}$, where $R \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3\}$
$so(3)$	$so(3) \doteq \{\Omega \,\vert\, R = exp(\Omega)$, where $\Omega \in \mathbb{R}^{3\times3}$ and $R \in SO(3)\ \}$
$se(3)$	$se(3) \doteq \{\Xi \,\vert\, H = exp(\Xi)$, where $\Xi = \begin{pmatrix}\Omega\,\xi;\ \mathbf{0}^T 0\end{pmatrix}$ and $\Omega \in so(3), \xi \in \mathbb{R}^3, H \in SE(3)\}$
$\exp()$	The matrix exponential of a square matrix.
$\log()$	The matrix logarithm of a square matrix
H	A general rigid body transformation ($H \in SE(3)$)
$\mathfrak{H}$	If H is an element of a Lie Group, $\mathfrak{H}$ is the corresponding element in the Lie algebra
$\mathbb{O}_n$	$n \times n$ zero matrix
$\mathbb{I}_n$	$n \times n$ identity matrix
$\{E_i\}$	the set of "natural" basis elements for Lie algebra
$\vee$	For Lie algebra, the "vee" operator is defined such that $\left(\sum_{i=1}^{n} x_i E_i\right)^{\vee} \doteq (x_1, x_2, \ldots, x_n)^T$ where $n = 3$ for $so(3)$ and $n = 6$ for $se(3)$
$\widehat{\ }$	For Lie algebra, the "hat" operator is the inverse of the "vee" operator. $\widehat{(x_1, x_2, \ldots, x_n)^T} \doteq \left(\sum_{i=1}^{n} x_i E_i\right)$; For a vector $\mathbf{n} \in \mathbb{R}^{3\times1}$. $\hat{\mathbf{n}}$ represents the corresponding skew symmetric matrix
$\circ$	The operator defined for the group product
$\odot$	The operator defined for the quaternion product
$\hat{\odot}$	The operator defined for the dual quaternion product
$\otimes$	The operator defined for the Kronecker product
vec	The "vec" operator is defined such that $vec(A) = [a_{11}, \ldots, a_{1m}, a_{21}, \ldots, a_{2m}, \ldots, a_{n1}, \ldots, a_{nm}]^T$ for $A = [a_{ij}] \in \mathbb{R}^{m\times n}$
A_i	A rigid body transformation ($A_i \in SE(3)$), associated with one sensor measurement source
B_i	A rigid body transformation ($B_i \in SE(3)$), usually associated with a different sensor measurement source
X	The rigid body transformation ($X \in SE(3)$) that relates A_i to B_i
R_H	The rotation matrix of any general transformation matrix $H \in SE(3)$
$\mathbf{t}_H$	The translation vector of any general transformation matrix $H \in SE(3)$
$\mathbf{n}_H$	The axis of rotation for R_H
θ_H	The angle of rotation for R_H about $\mathbf{n}_H$
ρ	A probability distribution of $H \in G$, where G $\subset$ SE(3)

M For ρ, M is the mean of the distribution

Σ For ρ, Σ is the covariance of the distribution about the mean, M

Ad For the Lie group G and the Lie algebra $\mathfrak{G}$, the adjoint operator is the transformation
Ad: $G \to \mathrm{GL}(\mathfrak{G})$, defined as
$\mathrm{Ad}(H_1)\mathfrak{H}_2 \doteq \frac{d}{dt}(H_1 \circ e^{t\mathfrak{H}_2} \circ H_1^{-1})$

$Ad(H)$ The adjoint matrix with columns $(HE_iH^{-1})^{\vee}$

1 INTRODUCTION TO SENSOR SIMULATION

With the rapid development of computing hardware, more and more experiments or tests on robots can be performed inside simulation environment. This greatly reduces time, cost and risk of research and production in comparison to real robotic implementations. Among all the parts of a robot system, sensors are the key for a robot to interact with the physical world, and both sensor simulation and sensor calibration are necessary steps for a successful interactive application.

1.1 Background

Sensor simulation has been integrated into many robot simulation systems due to its several advantages. First, sensors and their compatibility with the system can be tested in simulation before implementation, and a variety of scenarios can be set up without physical costs. Second, sensor-dependent algorithms can be adjusted or corrected in an early stage which might lower the burden of debugging in real tests. Last but not the least, sensor simulation provides a low cost, convenient and fast approach for testing new algorithms on either the robot or the sensor itself, boosting the theoretical work of robotics regardless of researchers' access to real hardware. Generally speaking, the more accurate the sensor model is, the better the simulation result will be. In many cases, a good sensor model can be obtained as follows:

a) Design the parametric measurement model without noise.

b) Determine sources of noise.

c) Add noise to parameters.

d) Update parameters to fit model to data.

e) Compare the actual and the expected measurement.

Nonetheless, it is sometimes difficult to develop an accurate sensor model. On the one hand, an almost-real sensor model is extremely computationally intensive. On the other, some state variables are not available for developing accurate sensor models. All of these led to the shortcomings of sensor simulation. The inaccuracy of sensor models make sensor simulation inherently inaccurate so that one can't avoid real experiments under certain circumstances. In addition, sensor simulation is programmed to reflect the performance of the system within a certain scenario, and generalization to other conditions is not automatic. Moreover, physical world is full of details and uncertainties, making it impossible to build an equal virtual environment to test the capability of the target sensor.

1.2 Simulation Software and Libraries

Sensor simulation is closely related to robot simulation and quite a few platforms have emerged which include both commercial and open source software packages. Some of the selected robotics simulators are ARS [1], Gazebo [2, 3], MORSE [4], Webots [5, 6], V-Rep [7], RoboDK [8] and SimSpark [9], etc. Webots is a widely applied commercial robotics simulator which has various features. It provides a rich library of sensors including distance sensors, light sensors, cameras, LIDARs, GPS, gyro, accelerometer, compass, bumpers, position sensors, force sensors, etc. Gazebo, however, is an open source robot simulator which gains a lot of attention recently. It is deeply integrated with another open source framework ROS (Robot Operating System). Gazebo is also well known for serving as the platform for DARPA (Defense Advanced Research Projects Agency) Virtual Robotics Challenge. Its sensor library covers cameras, contact sensors, force-torque sensors, GPS, GPU-ray sensors, IMU, RFID sensors, sonar sensors, and wireless transmitter & receiver. In terms of popularity, V-Rep is also a comprehensive robotics simulator which offers both a commercial version and an educational version. All the platforms above incorporates sensor simulation functionalities and they altogether provide a wide choice of sensor simulation methodologies.

In the field of autonomous driving, a lot of advanced commercial systems are available in the market such as CarMaker [10], PreScan [11], AdasWorks [12], etc. Other companies marching into the autonomous vehicle industry have even integrate the almost autonomous driving system into real cars. Good examples are Google self-driving car project [13], the ConnectedDrive Connect (CDC) Semi-Autonomous Driving System from BMW [14] and the newly released auto-pilot system from Tesla [15]. The effectiveness of sensor simulation depends on not only the sensor models that are used, but also the completeness of the virtual environments. Sophisticated autonomous driving simulation software builds in both different sensor models and a variety of complicated road and traffic environment that can exist in real life. All of these systems set up a good example for tomorrow's humanoid simulation software. In the following section, an introduction of sensor calibration is given, which is also the focus of this chapter.

2 INTRODUCTION TO SENSOR CALIBRATION

In the fields of robotics and computer vision, sensor calibration problems are often codified using the "AX=XB" formulation, (Figs. 1–3). Example applications include camera calibration, Cartesian robot hand calibration, robot eye-to-hand calibration [16], aerial vehicle sensor calibration [17], image-guided therapy (IGT) sensor calibration and endoscopic surgery [18]. An alternative name that describes this system is "hand-eye" calibration. Several methods exist in the literature that can perform hand-eye calibration without directly dealing with the $AX = XB$ formulation such as [19–25]. Another problem, called the robot-world and hand-eye calibration, deals with the case where an additional calibration of the robot pose with respect to the world frame is needed. This calibration technique is formulated as the “AX=YB” problem and multiple methods have been developed such as [26–32].

Concrete examples are the camera-IMU calibration in Fig. (4) and IR camera & drone camera calibrations in Fig. (5). Furthermore, a hand-eye, tool-flange, and robot-robot calibration problem is formulated as the “AXB = YCZ” problem in [33, 34].

Generally speaking, all of the three calibration frameworks above involve the sensor extrinsic calibration, which describes the fixed 3D rigid-body transformation either between a sensor and the mounted body frame or between two sensors fixed on the same rigid body. The sensors that can benefit such calibration techniques include but are not limited to monocular camera, stereo camera, ultrasound (US) probe, and IMU. Once the unknown rigid-body transformations in the system are calibrated, one can determine the pose of the target in the sensor frame more accurately. This is beneficial in many ways such as helping the manipulator to better determine the configuration of the object to be grasped, improving the accuracy in constructing 3D environment and facilitating the localization of a robot. Calibrating the extrinsic parameters of various sensors can also help with the sensor fusion in a system. From the perspective of simulation, employing the calibrated configurations of sensors can enable the simulation to better reflect the implementation in the physical world compared to using artificial sensor frames or those obtained from the spec sheet.

In the “AX=XB” formulation A, X, and B are each homogeneous transformations (i.e., elements of the special Euclidean group, $SE(3)$) with each pair of measurements (A,B) coming from sensors such as cameras, US probes, optical, or electromagnetic pose tracking systems, among others. X is the unknown rigid-body transformation that is found as a result of solving $AX = XB$. It is well known that it is not possible to solve for a unique X from a single pair of exact (A,B), but if there are two instances of independent exact measurements, (A_1,B_1) and (A_2,B_2) satisfying certain constraints, then the problem can be solved for a unique, fixed X. However, in practice, sensor noise is always present, and an exact solution is not possible. The goal, therefore, becomes one of finding an X with the least error given corresponding noisy pairs of data (A_i,B_i) for i = 1,2, ... ,n.

3 THE $AX = XB$ FORMULATION

Any (proper) rigid-body motion in a three-dimensional space can be described as a 4×4 homogeneous transformation matrix of the form:

$$H(R,\mathbf{t}) = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{1}$$

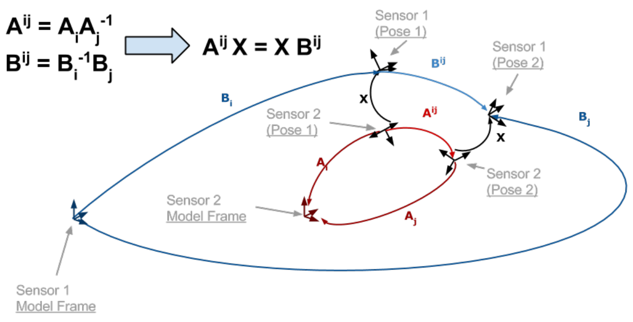

Fig. 1. The $AX = XB$ Sensor Calibration Formulation

where $R \in SO(3)$ is a 3×3 (proper) rotation matrix, and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector. The set of all such matrices can be identified with $SE(3)$, the group of rigid-body motions, where the group law is matrix multiplication.

Given:

$$AX = XB \tag{2}$$

where $A,B,X \in SE(3)$, it is well known that, in non-degenerate cases, there are two unspecified degrees of freedom to the problem for a single pair of sensor measurements, (A,B). This situation is rectified by considering two pairs of exact measurements of the form in Eq. (2), i.e., $A_1X = XB_1$ and $A_2X = XB_2$, provided that some mild conditions are observed for the selection of the pairs (A_1,B_1) and (A_2,B_2) [35–37]. Note that (A_i,B_i) here are relative transformation data pairs, so in real experiments at least 3 absolute transformation data pairs have to be measured. Additionally, if there is sensor measurement error, then it may not be possible to find compatible pairs that reproduce the exact value of X. For this reason, minimization and least squares approaches are often taken over large sets of As and Bs.

Performing the matrix multiplication of homogeneous transformations in Eq. (2) and separating out the rotational and translational parts results in two equations of the form:

$$R_A R_X = R_X R_B \tag{3a}$$
$$R_A \mathbf{t}_X + \mathbf{t}_A = R_X \mathbf{t}_B + \mathbf{t}_X. \tag{3b}$$

One strategy for solving Eq. (2) reduces to first solving Eq. (3a) for only rotation and then rearranging Eq. (3b) to find acceptable values of $\mathbf{t}_X$ satisfying $(R_A - \mathbb{I}_3)\mathbf{t}_X = R_X\mathbf{t}_B - \mathbf{t}_A$. The other strategy is to solve for R_X and $\mathbf{t}_X$ simultaneously based on some cost function $f(R_X,\mathbf{t}_X)$ or reformulating the matrix equation into a dual quaternion representation. However, when there are only two exact sensor measurements, there are some problems with the first approach. As pointed out in [36, 37], in nondegenerate cases, there is a one-parameter set of solutions to Eq. 3a, and the matrix $R_A - \mathbb{I}_3$ in general has a rank of 2. Hence, there are two unspecified degrees of freedom to the problem, and it cannot be solved uniquely unless additional measurements are taken.

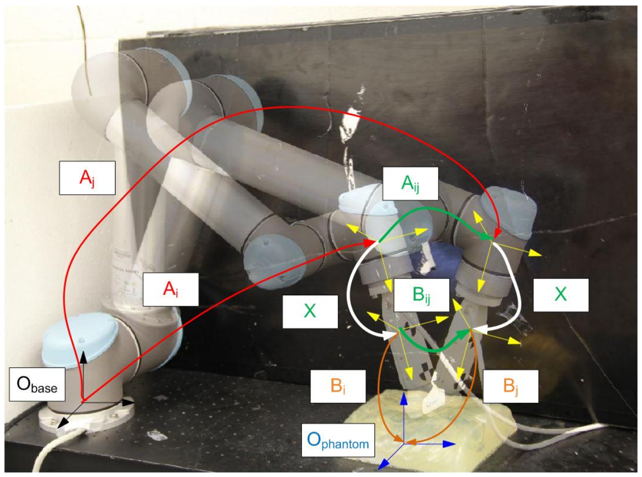

Fig. 2. Application of AX=XB in ultrasound sensor calibration: an ultrasound probe is attached to the end-effector and the calibration phantom is used for ultrasound probe calibration. (The UR5 robot pictured above is owned by Professor Emad Boctor of Johns Hopkins University)

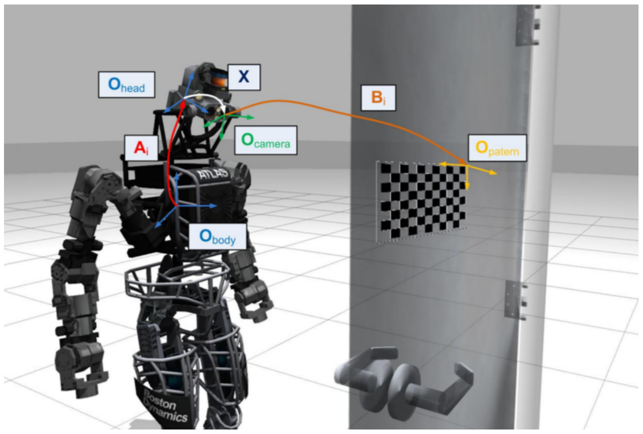

Fig. 3. Application of $AX = XB$ in Humanoid Camera Sensor Calibration: The Humanoid Robot Named Atlas Designed by *Boston Dynamics* Requires head-body calibration before turning the knob. Different pairs of (A_i, B_i) are measured by changing the head pose of the humanoid robot. (This picture was generated using the open-source robot simulation software Gazebo)

Additionally, if there is sensor error, then it may not be possible to find compatible pairs that reproduce the exact value of X. For this reason, minimization approaches are often taken, where for $n > 2$ a cost function:

$$C(X) = \sum_{i=1}^{n} w_i d^2(A_i X, X B_i) \tag{4}$$

is computed for some distance metric $d(\cdot,\cdot)$ on $SE(3)$ and $\{w_i\}$ is a set of weights which can be taken as a partition of unity.

This chapter is organized as follows. In Section 4, a detailed review is given for the existing methods that solve the $AX = XB$ problem using pairs of $\{A_i, B_i\}$ with correspon-

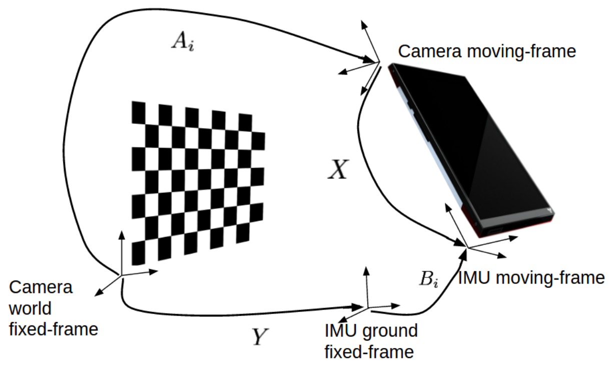

Fig. 4. Application of $AX = YB$ in IMU-camera calibration of a mobile phone using checkerboard.

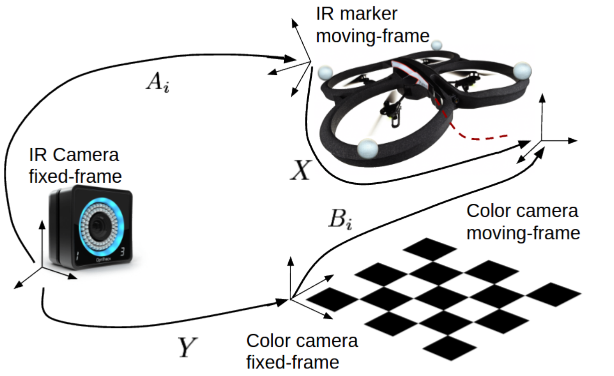

Fig. 5. Application of $AX = YB$ in IR camera and drone camera calibrations.

dence. The notation used throughout the literature is not consistent and the authors strove to standardize the notation used across the reviewed methods to provide the readers with a straightforward and consistent representation of the methods and their relationships. Some approaches are preferable under certain circumstances and their advantages and disadvantages are also highlighted. In Section 5, some important criterion for data selection and different errors metrics used in the literature are presented. Finally, in Section 6 a complete summary of the $AX = XB$ solvers is provided and open problems are noted as well.

4 EXISTING SOLUTION METHODS

The problem of solving Eq. (2) for X when multiple corresponding pairs of As and Bs are present has a history that stretches back more than a quarter of a century [36–38], with the earliest proposed method introduced by Shiu [37, 39] and Tsai [16]. Applications involving this problem still remain active today [23, 40]. Shah [41] overviewed several different AX = XB methods qualitatively. Fassi and Legnani [42] gave a geometric view of $AX = XB$ and discussed the over-constrained and singular conditions. A more complete list of the traditional $AX = XB$ solvers includes: the Shiu method [37], the screw motion method [35, 43], the Euclidean group method [36, 44], the quaternion method [38, 45, 46], the dual quaternion method [47, 48], and the Kronecker method [49,50]. Several new optimization methods emerged recently such as the convex optimization method [51], some global optimization methods [31, 50, 52] and a structure from motion (SfM) approach [53]. The SfM method deals with the case where a calibration target is not applicable and a scaling factor needs to be calibrated in addition to rotation and translation. The methods mentioned previously are all offline methods where X can be calculated given a complete list of data pairs. However, several online methods have been proposed that are more preferable to real-time applications [49, 54, 55]. All the methods above assume $\{A_i, B_i\}$ data pairs with correspondence. Probabilistic methods that deal with data pairs without correspondence are introduced in [56, 57]. In the follow sections, representatives of the above $AX = XB$ solvers will be reviewed in detail.

4.1 Shiu and Ahmad

Shiu and Ahmad [37, 39] use two data pairs (A_i, B_i) to solve for X. The necessary condition for the uniqueness of X is that the rotation axes of R_{A_1} and R_{A_2} are neither parallel nor anti-parallel, and the angles of rotation are neither 0 nor π. Though this method shows tolerance to noise to a certain extent, it is specifically designed to solve for the case where only two sets of (A, B) are given. The rotation matrix R_X is solved for first and the translation is obtained using a least squares technique given a known R_X.

The closed form expression for R_X is:

$$R_X = \exp(\hat{\mathbf{n}}_A \beta) R_{X_P} \tag{5}$$

where

$$\begin{gathered} R_{X_p} = \exp(\widehat{\mathbf{n}}_X \theta_X) \\ \mathbf{n}_X = \mathbf{n}_B \times \mathbf{n}_A \\ \theta_X = \operatorname{atan2}(|\mathbf{n}_\mathrm{B} \times \mathbf{n}_\mathrm{A}|, \mathbf{n}_\mathrm{B} \cdot \mathbf{n}_\mathrm{A}) \end{gathered}$$

and β is an arbitrary angle. Given a vector $\mathbf{n} = (n_1, n_2, n_3)^T \in \mathbb{R}^3$, $\widehat{\mathbf{n}}$ is a skew-symmetric matrix defined as below:

$$\widehat{\mathbf{n}} = \begin{pmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{pmatrix}. \tag{6}$$

Equation (5) shows that R_x has one degree of freedom which is determined by the angle β. Therefore, two relative arm motions are needed to generate two (A_i, B_i) data pairs in order to calculate the unique solution of X. Given two pairs of As and Bs, two equations can be obtained as:

$$A_1 X = X B_1 \tag{7a}$$

$$A_2 X = X B_2. \tag{7b}$$

Instead of giving a closed-form solution, R_X is calculated by solving for β in Eq. (8), which is formulated as a system of linear equations obtained by equating two instances of Eq. (5) that are obtained from data pairs (A_1, B_1) and (A_2, B_2):

$$CY = D. \tag{8}$$

In Eq. (8), $Y = \big(\cos(\beta_1), \sin(\beta_1), \cos(\beta_2), \sin(\beta_2)\big)^T$ where β_1, β_2 correspond to Eq. (7a) and Eq. (7b) respectively, $C \in \mathbb{R}^{9\times 4}$ and $D \in \mathbb{R}^{9\times 1}$ are determined from $\mathbf{n}_{A_1}$, $\mathbf{n}_{B_1}$, $\mathbf{n}_{A_2}$, and $\mathbf{n}_{B_2}$. The explicit expressions for C and D are given in Eq. (44) of [37].

Similarly, with known R_X, $\mathbf{t}_x$ can be calculated using the least squares method:

$$\begin{pmatrix} R_{A_1} - \mathbb{I}_3 \\ R_{A_2} - \mathbb{I}_3 \end{pmatrix} \mathbf{t}_X = \begin{pmatrix} R_X \mathbf{t}_{B_1} - \mathbf{t}_{A_1} \\ R_X \mathbf{t}_{B_2} - \mathbf{t}_{A_2} \end{pmatrix}. \tag{9}$$

This is not a closed-form solution and it is constrained to the case where only two data pairs are provided.

4.2 Lie Group Method

The Lie Group method [36] by Park and Martin is the first method to solve the AX = XB problem from the perspective of Lie groups. It uses the axes of rotation of A_i and B_i to construct R_X and gives both the closed-form solution for the no-noise case and the numerical solution for multiple noisy (A_i, B_i) pairs.

4.2.1 Closed-Form Solution with Two Exact Pairs

The closed-form solution for R_X is as follows:

$$R_X = \mathcal{A}\mathcal{B}^{-1} \tag{10}$$

where

$$\begin{gathered} \mathcal{A} = (\mathbf{n}_{A_1}, \mathbf{n}_{A_2}, \mathbf{n}_{A_1} \times \mathbf{n}_{A_2}) \in \mathbb{R}^{3\times 3} \\ \mathcal{B} = (\mathbf{n}_{B_1}, \mathbf{n}_{B_2}, \mathbf{n}_{B_1} \times \mathbf{n}_{B_2}) \in \mathbb{R}^{3\times 3} \\ \widehat{\mathbf{n}}_{A_1} = \log(R_{A_i}) / \| \log^\vee(R_{A_i}) \| \\ \widehat{\mathbf{n}}_{B_1} = \log(R_{B_i}) / \| \log^\vee(R_{B_i}) \| \end{gathered}$$

The solution for R_X is uniquely determined given two pairs of (A_i, B_i), and the solution for $\mathbf{t}_X$ can be obtained using Eq. (9) once R_X is obtained.

4.2.2 Estimation of X Using Multiple Pairs with Noise

When there are multiple pairs of (A_i, B_i) with noise, rotation matrix R_X is solved for first and then the translation vector $\mathbf{t}_x$ is obtained using a least squares method given known R_X. The closed-form expression for R_X is as follows:

$$R_X = (M^T M)^{-\frac{1}{2}} M^T \tag{11}$$

where

$$M = \sum_{i=1}^{n} \mathbf{n}_{B_i}\mathbf{n}_{A_i}^T.$$

Note that $i \geq 3$ is a necessary condition for M to be a nonsingular matrix, but it does not guarantee M to be nonsingular. Theoretically, the Lie group method is more likely to fail when the number of data pairs is small, i.e., close to 3 pairs, while in real application, failure is rarely seen as long as the data pairs are not specially chosen such that M is degenerate. Given known R_X, $\mathbf{t}_X$ can be calculated using the least squares method shown in Eq. (12):

$$\mathbf{t}_x = (C^TC)^{-1}C^Td \tag{12}$$

where

$$C = \begin{pmatrix} \mathbb{I}_3 - R_{A_1} \\ \mathbb{I}_3 - R_{A_2} \\ \cdot \\ \cdot \\ \cdot \\ \mathbb{I}_3 - R_{A_n} \end{pmatrix} \quad d = \begin{pmatrix} \mathbf{t}_{A_1} - R_X\mathbf{t}_{B_1} \\ \mathbf{t}_{A_2} - R_X\mathbf{t}_{B_2} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{t}_{A_n} - R_X\mathbf{t}_{B_n} \end{pmatrix}.$$

4.3 Quaternion Method

The Quaternion method proposed by Chou and Kamel [38, 45] uses unit quaternions to transform the rotation parts of $A_iX = XB_i$ (i = 1,2) into two linear systems. Then a singular value decomposition (SVD) is performed to obtain a closed-form solution for R_X. In order to estimate X given multiple pairs of (A_i, B_i) with noise, Horaud and Dornaika [46] cast the problem into a nonlinear optimization one. Two different approaches are discussed: (1) estimate the rotation matrix R_X by minimizing an objective function, and solve for the translation $\mathbf{t}_X$ using a least squares method separately and (2) estimate R_X and $\mathbf{t}_X$ simultaneously by minimizing an objective function that incorporates both the rotational and translational information. Method (1) turns out to have a closed-form solution for the unit quaternion q_X representing the rotation R_X, while method (2) is a nonlinear optimization problem which requires an initial guess and will be discussed in Section 4.6.

4.3.1 Closed Form Solution with Two Exact Pairs

First, the rotation equation in Eq. (3) is transformed into the equation of quaternion multiplication as below:

$$R_AR_X = R_XR_B \;\Leftrightarrow\; q_A \odot q_X = q_X \odot q_B \tag{13}$$

where q_A, q_B and q_X are unit quaternions that represent the rotation parts of matrices A, B and X, and $\odot$ denotes quaternion multiplication.

Given two quaternions $q_\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)^T = (\alpha_0, \alpha^\mathbf{T})^T$ and $q_\beta = (\beta_0, \beta_1, \beta_2, \beta_3)^T = (\beta_0, \beta^\mathbf{T})^T$, quaternion multiplication $\odot$ is defined as:

$$q_\alpha \odot q_\beta = \begin{pmatrix} \alpha_0\beta_0 - \alpha^\mathbf{T}\beta \\ \alpha_0\beta + \beta_\mathbf{0}\alpha + \tilde{\alpha}\beta \end{pmatrix}. \tag{14}$$

α_0 is called the scalar component and α is the vector component of the quaternion q_α. In order to solve for q_X, the quaternion equation is transformed into:

$$E\mathbf{q}_X = \mathbf{0} \tag{15}$$

where $E \in \mathbb{R}^{4\times4}$ is obtained by grouping q_A and q_B together, and $\mathbf{q}_X \in \mathbb{R}^4$ is the vector representation of the unit quaternion q_X.

It turns out that the unit quaternion q_X which represents the rotation part of X can be written as:

$$\mathbf{q}_X = V_2\mathbf{y_2}. \tag{16}$$

To obtain the matrix V_2 and vector y_2, E is first written as $E = \sin(\theta_{A|B}/2)E_0$ with $\theta_{A|B} = \theta_A = \theta_B$, which is the constraint that the corresponding transformations A_i and B_i should have the same angle of rotation. Next, the SVD of M is computed as $E_0 = U\Sigma V^T$ where $V = (V_1, V_2)$, $V_1 \in \mathbb{R}^{4\times2}$, $V_2 \in \mathbb{R}^{4\times2}$, $U \in \mathbb{R}^{4\times4}$ and Σ is a diagonal matrix. Vector $\mathbf{y_2}$ is obtained by calculating $\mathbf{y} = V^T\mathbf{q}_x$ where $\mathbf{y} = (\mathbf{y_1^T}, \mathbf{y_2^T})^\mathbf{T}$, $\mathbf{y_1} \in \mathbb{R}^{2\times1}$ and $\mathbf{y_2} \in \mathbb{R}^{2\times1}$. Their expressions are also outlined as follows:

$$\begin{aligned} E &= \sin(\theta_{A|B}/2)E_0 \\ M &= U\Sigma V^T \\ V &= (V_1, V_2) \\ \mathbf{y} &= V^T\mathbf{q}_x \\ \mathbf{y} &= (\mathbf{y_1^T}, \mathbf{y_2^T})^\mathbf{T}. \end{aligned}$$

The translation vector $\mathbf{t}_x$ satisfies the following equation:

$$\left(\cot(\frac{\theta_A}{2})\hat{\mathbf{n}}_A(R_A - \mathbb{I}_3) + R_A + \mathbb{I}_3\right)\mathbf{t}_x = \mathbf{n}_Az \tag{17}$$

where $z \in \mathbb{R}$ is arbitrary. A unique solution can be calculated using Eq. (16) and Eq. (17) given two nondegenerate pairs of (A_i, B_i).

4.3.2 Estimation of X Using Multiple Pairs With Noise

As shown in [46], in the case where there are n pairs of (A_i, B_i), the problem of recovering R_X is converted into minimizing the following error objective function:

$$\begin{aligned} f(R_X) &= \sum_{i=1}^{n} ||n_{A_i} - q_X \odot n_{Bi} \odot \bar{q}_X||^2 \\ &= \mathbf{q}_X{}^T\tilde{K}\mathbf{q}_X \end{aligned} \tag{18}$$

where $n_{A_i} = \left(0, \mathbf{n}_{A_i}{}^T\right)^T$ and $n_{B_i} = \left(0, \mathbf{n}_{B_i}{}^T\right)^T$. $\tilde{K} = \sum_{i=1}^n \tilde{K}_i$ and $\tilde{K}_i \in \mathbb{R}^{4\times 4}$ is a symmetric positive definite matrix determined by $\mathbf{n}_{A_i}$ and $\mathbf{n}_{B_i}$; $\bar{q}_X$ is the conjugate of q_X where $q_X \odot \bar{q}_X = 1$.

To minimize Eq. (18) under the constraint that $\mathbf{q}_X$ is a unit quaternion, the Lagrangian multiplier is introduced as:

$$\min_q f = \min_q (\mathbf{q}_X{}^T \tilde{K} \mathbf{q}_X + \lambda(1 - \mathbf{q}_X{}^T \mathbf{q}_X)). \tag{19}$$

Differentiating the error function with respect to $\mathbf{q}_X$, the 1st order necessary optimality condition is obtained as:

$$\tilde{K}\mathbf{q}_X = \lambda \mathbf{q}_X. \tag{20}$$

It can be shown that the unit quaternion $\mathbf{q}_X$ that minimizes f is the eigenvector of $\tilde{K}$ associated with its smallest positive eigenvalue. After recovering $\mathbf{q}_X$ (or equivalently R_X), $\mathbf{t}_X$ can be recovered using the least square techniques introduced in previous methods.

4.4 Dual Quaternion Method

The dual quaternion method proposed by Daniilidis and Bayro-Corrochano [47] (Daniilidis [48]) treats the rotation and translation parts of the matrix X in a unified way and facilitates a new simultaneous solution of X using SVD. To begin, Eq. (2) is transformed into an equation of dual quaternions:

$$AX = XB \;\Leftrightarrow\; \check{a} = \check{q}_X \hat{\odot} \check{b} \hat{\odot} \bar{\check{q}}_X \tag{21}$$

where $\check{a}$, $\check{b}$ and $\check{q}$ are the dual quaternions that represent matrices A, B and X, and $\bar{\check{q}}$ is the conjugate of $\check{q}$.

The dual quaternion that corresponds to a 4x4 rigid transformation matrix is defined as follows:

$$\check{q}_X = \begin{pmatrix} \cos(\frac{\theta + \varepsilon d}{2}) \\ \sin(\frac{\theta+\varepsilon d}{2})(\mathbf{l} + \varepsilon \mathbf{m}) \end{pmatrix} \tag{22}$$

where θ, d, $\mathbf{l}$ and $\mathbf{m}$ are screw parameters and $\varepsilon^2 = 0$. θ is the rotation angle, d is the pitch, $\vec{\mathbf{l}}$ is the direction of the screw, and $\mathbf{m} = \mathbf{p} \times \mathbf{l}$ is the line moment where $\mathbf{p}$ is a point on the line. The six tuple $(\mathbf{l}, \mathbf{m})$ defines a line in 3-D space. Furthermore, by expanding the dual terms in $\check{q}_X$, Eq. (22) can also be written as:

$$\check{q}_X = q_X + \varepsilon q'_X. \tag{23}$$

Both q and q' are quaternions satisfying the following constraints:

$$\mathbf{q}_X^T \mathbf{q}_X = 1 \text{ and } \mathbf{q}_X^T \mathbf{q}'_X = 0 \tag{24}$$

where $\mathbf{q}_X$ and $\mathbf{q}'_X$ are the vector representations of q_X and q'_X. Then equation $A_i X = X B_i$ can be converted into the form below:

$$S_i \underbrace{\begin{pmatrix} \mathbf{q}_X \\ \mathbf{q}'_X \end{pmatrix}}_{\mathbf{x}} = \mathbf{0} \tag{25}$$

where

$$S_i = \begin{pmatrix} \mathbf{a} - \mathbf{b} & (\mathbf{a}+\mathbf{b})^\wedge & \mathbf{0_{3\times 1}} & \mathbf{0_{3\times 3}} \\ \mathbf{a}' - \mathbf{b}' & (\mathbf{a}' + \mathbf{b}')^\wedge & \mathbf{a} - \mathbf{b} & (\mathbf{a}+\mathbf{b})^\wedge \end{pmatrix} \in \mathbb{R}^{6\times 8}. \tag{26}$$

The notation for $\mathbf{x}$ here will be also used in Section To maintain the consistency of notation throughout the chapter as well as preserve the original notation in [47], for a vector $\mathbf{v} \in \mathbb{R}^{3\times 1}$, $\mathbf{v}^\wedge$ is the same as $\tilde{\mathbf{v}}$ which maps a vector into the corresponding skew-symmetric matrix. $\mathbf{a}' = \frac{1}{2}\mathbf{t}_X \times \mathbf{a}$ and $\mathbf{a}$ is the vector part of q_X. Similarly, $\mathbf{b}$ is the vector part of q'. After concatenating S_i, the following matrix T can be obtained and used to solve for X:

$$T = \begin{pmatrix} S_1^T \; S_2^T \; \ldots \; S_n^T \end{pmatrix}^T. \tag{27}$$

By calculating the SVD of $T = U\Sigma V^T$, the dual quaternion for matrix X can be expressed as a linear combination of the last two right-singular vectors ($\mathbf{v_7}$, $\mathbf{v_8}$) of matrix T, which are the last two columns of matrix V, as shown below:

$$\begin{pmatrix} \mathbf{q}_X \\ \mathbf{q}'_X \end{pmatrix} = \lambda_1 \mathbf{v_7} + \lambda_2 \mathbf{v_8} \in \mathbb{R}^8 \text{ where } \lambda_1, \lambda_2 \in \mathbb{R}. \tag{28}$$

Different from the quaternion method in 4.3.1, the dual quaternion method solves the rotational part and translational part in a united way, and it contains all the information to reconstruct matrix X. However, it does not use all of the available information, only the imaginary parts of $\check{a}$ and $\check{b}$. Despite the advantages of the dual quaternion method, its major drawback is the need to filter the data pairs to ensure appropriate solutions when there is noise on A_i and B_i.

4.5 Kronecker Product Method

Inspired by the well known Sylvester equation ($AX + XB = C$) in linear systems, Andreff *et al.* [49] proposed the Kronecker method which converts Eq. (2) into the form of Kronecker products [49]:

$$AX = XB \Leftrightarrow$$
$$\underbrace{\begin{pmatrix} \mathbb{I}_9 - R_B \otimes R_A & 0_{9\times 3} \\ \mathbf{t}_B^T \otimes \mathbb{I}_3 & \mathbb{I}_3 - R_A \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} vec(R_X) \\ \mathbf{t}_X \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} 0_9 \\ \mathbf{t}_A \end{pmatrix}}_{d}. \tag{29}$$

where $C, d, \mathbf{x}$ will be referred to in Section 4.6. Given multiple pairs of As and Bs with noise, the Kronecker product is reformulated as:

$$\begin{pmatrix} \mathbb{I}_9 - R_{B_1} \otimes R_{A_1} \\ \mathbb{I}_9 - R_{B_2} \otimes R_{A_2} \\ \vdots \\ I_9 - R_{B_n} \otimes R_{A_n} \end{pmatrix} vec(R_X) = \mathbf{0}_{9n\times 1}, \tag{30}$$

and

$$\begin{pmatrix} \mathbb{I}_3 - R_{A_1} \\ \mathbb{I}_3 - R_{A_2} \\ \vdots \\ \mathbb{I}_3 - R_{A_n} \end{pmatrix} \mathbf{t}_X = \begin{pmatrix} \mathbf{t}_{A_1} - R_X \mathbf{t}_{B_1} \\ \mathbf{t}_{A_2} - R_X \mathbf{t}_{B_2} \\ \vdots \\ \mathbf{t}_{A_n} - R_X \mathbf{t}_{B_n} \end{pmatrix}. \tag{31}$$

The vectorized version of R_X obtained from Eq. (30) is not an element of $SO(3)$ and orthogonalization on R_X is required to obtain a rotation matrix:

$$R_{X_e} = R_X (R_X^T R_X)^{-1/2} \tag{32}$$

[58] where R_{X_e} denotes the orthogonalized R_X.

The orthogonalized matrix R_{X_e} is further normalized as:

$$R_{X_n} = \frac{sign(\det(R_{X_e}))}{|\det(R_{X_e})|^{\frac{1}{3}}} R_{X_e} \tag{33}$$

where R_{X_n} is the normalized matrix of R_{X_e}. After getting the estimation of R_X, a least squares method is implemented on Eq. (31) to recover $\mathbf{t}_X$. One advantage of the Kronecker product method is its capability of dealing with small motions, because the associated orthogonal matrix R_{X_n} is always defined, while the rotation can be ill-defined when using the axis-angle representation. Moreover, the linear system can also be used to analyze.

the recoverable information in X based on the available type and number of motions. Details are included in Section 5. In addition, an on-line hand-eye calibration method is developed for an unknown scene based on the above algorithm, where the camera translations are estimated up to a scaling factor.

4.6 Optimization Methods

Different optimization methods have been proposed in the literature and most of them are built upon the various parametrizations of the $AX = XB$ equations mentioned previously.

4.6.1 Quaternion Based Simultaneous Approach

When trying to solve for R_X and $\mathbf{t}_X$ simultaneously, it is impossible to find a closed-form solution. Horaud and Dornaika [46] presented an objective function for minimization, which is a sum of squares of nonlinear functions as:

$$\begin{aligned} f(q_X, \mathbf{t}_X) = \lambda_1 \sum_{i=1}^{n} ||vec(q_X \odot t_{B_i} \odot \bar{q}_X) - (R_{A_i} - \mathbb{I})\mathbf{t}_X - \mathbf{t}_{A_i}||^2 \\ + \lambda_2 \sum_{i=1}^{n} ||n_{A_i} - q_X \odot n_{B_i} \odot \bar{q}_X||^2 + \lambda_3 (1 - \mathbf{q}_X^T \mathbf{q}_X)^2 \end{aligned} \tag{34}$$

where $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}$. Note that $vec()$ here represents the vector part of a quaternion such that $vec(q) \in \mathbb{R}^3$. The third term is a penalty function where the modulus of q will approach 1 when λ_3 becomes large. This is a non-convex optimization problem which requires a good initial guess due to the existence of multiple local minima. However, the result can be more accurate than the rest of the solvers for certain motion pairs when the initial guess is “good".

4.6.2 Polynomial Global Optimization

Heller *et al.* [31] developed a polynomial global optimization method which does not require an initial estimate and is also globally optimal in the L_2-norm sense. Defining a certain parametrization of $X \in SE(3)$ as $P(X)$, the previous minimization problem Eq. (4) is formulated as:

$$\begin{aligned} &\min_{X \in SE(3)} \sum_{i=1}^{n} ||A_i X - X B_i||^2 \Leftrightarrow \\ &\text{minimize} \quad f(P(X)) \\ &\text{subject to} \ \ \mathbf{c}(P(X)) \geq \mathbf{0} \end{aligned} \tag{35}$$

where $f(X)$ is the converted multivariate polynomial function using the convex linear matrix inequality (LMI) relaxations technique [59].

When R_X is parametrized using the orthonormal basis as $R_X(\mathbf{u}, \mathbf{v}) = (\mathbf{u}, \mathbf{v}, \mathbf{u} \times \mathbf{v})$ where $\mathbf{v}, \mathbf{u} \in \mathbb{R}^3$, then Eq. (35) becomes:

$$\begin{aligned} \text{minimize} \quad & f_1(\mathbf{u}, \mathbf{v}, \mathbf{t}_X) = \\ & \sum_{i=1}^{n} ||A_i X(\mathbf{u}, \mathbf{v}, \mathbf{t}_X) - X(\mathbf{u}, \mathbf{v}, \mathbf{t}_X) B_i||^2 \\ \text{subject to} \quad & \mathbf{u}^T \mathbf{u} = 1, \mathbf{v}^T \mathbf{v} = 1, \mathbf{u}^T \mathbf{v} = 0. \end{aligned} \tag{36}$$

Similarly, using the quaternion representation of R_X, Eq. (35) becomes:

$$\begin{aligned} \text{minimize} \quad & f_2(\mathbf{q}_X, \mathbf{t}_X) = \\ & \sum_{i=1}^{n} ||A_i X(\mathbf{q}_X, \mathbf{t}_X) - X(\mathbf{q}_X, \mathbf{t}_X) B_i||^2 \\ \text{subject to} \quad & \mathbf{q}_X^T \mathbf{q}_X = 1, \ \ q_{X1} \geq 0. \end{aligned} \tag{37}$$

If A, B and X are parametrized using dual the quaternion representation, then

$$\begin{aligned} \text{minimize} \quad & f_3(\check{q}_X) = \\ & \sum_{i=1}^{n} ||\check{q}_A \hat{\odot} \check{q}_X - \check{q}_X \hat{\odot} \check{q}_B||^2 \\ \text{subject to} \quad & \mathbf{q}_X^T \mathbf{q}_X = 1, \; q_{X1} \geq 0 \\ & q_{X1} q_{X5} + q_{X2} q_{X6} + q_{X3} q_{X7} + q_{X4} q_{X8} = 0. \end{aligned} \tag{38}$$

As pointed out in [31], the polynomial global optimization method as described in Eq. (38) can give better solutions than Park [36], Eq. (34), Eq. (36) and Eq. (37). For those who are also interested in robot-world and hand-eye calibration, several solvers for $AX = YB$ using the LMI technique are also given; however, they fail to give better results than the traditional methods.

4.6.3 Convex Optimization

Based on different ways of formulating the rotation part of the rigid body transformation, Zhao [51] gives two formulations that use a L_∞ optimization technique. L_∞ optimization is the minimax problem:

$$\min_{\mathbf{x}} \; \max_{i} f_i(\mathbf{x}) \;\; \mathbf{i} = \mathbf{1}, \mathbf{2}, ..., \mathbf{n} \tag{39}$$

where $\mathbf{x}$ represents all the unknown transformation parameters and $f_i(\mathbf{x})$ is the error function corresponding to (A_i, B_i). Eq. (39) can be converted into a convex optimization problem if $f_i(\mathbf{x})$ is a quasi-convex function on a convex domain on which it is to be minimized.

Using the Kronecker formulation as in Eq. (29) and introducing an additional variable δ, the equivalent form of the L_∞ optimization problem is:

$$\begin{aligned} & \min_{\delta, \mathbf{x}} \; \delta \\ & \text{subject to} \;\; ||C_i \mathbf{x} - \mathbf{d}_i||_2 \leq \delta \\ & \text{where} \;\; i = 1, 2, ..., n. \end{aligned} \tag{40}$$

The matrix $\mathbf{C}_i$, vector $\mathbf{x}$, and $\mathbf{d}_i$ correspond to those in Eq. (29). Above is a convex optimization problem that can be solved using a *second-order cone program*, which can be solved using toolboxes available online.

When the $AX = XB$ problem is formulated as a dual quaternion representation as in Eq. (25), the equivalent L_∞ optimization problem can be written as:

$$\begin{aligned} & \min_{\delta, \mathbf{x}} \; \delta \\ & \text{subject to} \;\; ||S_i \mathbf{x}||_2 \leq \delta \\ & \text{where} \;\; i = 1, 2, ..., n \;\; \text{with} \;\; \mathbf{Dx} \geq \mathbf{f}. \end{aligned} \tag{41}$$

The matrix S_i and vector $\mathbf{x}$ correspond to those in Eq. (25). The inequality constraint $\mathbf{Dx} \geq \mathbf{f}$ is added manually in order to prevent $\mathbf{x}$ from reaching zero, which is a meaningless solution for the program. $\mathbf{x}$ must also satisfy two additional constraints: $\|\mathbf{q}_X\| = 1$ and $\|\mathbf{q}'_X\| = 1$.

The proposed methods need no initial guess and are less time consuming compared to the simultaneous optimization method given in Eq. (34). However, the errors for both of the convex optimization methods are larger than the latter.

4.7 Gradient Descent Method

Except for the Kronecker product method, all the methods mentioned above are only able to solve for matrix X offline, which means $\{A_i, B_i\}$ data pairs should be fully collected before being put into the algorithm. The gradient descent method [55] by Ackerman *et al.* is an online sensor calibration method which uses a gradient descent optimization on the Euclidean group (SE(3)) given an appropriate cost function.

To begin, define $\mathcal{X} \in se(3)$ as the Lie algebra corresponding to $G = SE(3)$, and let $f : G \to \mathbb{R}$ be an analytic function and $g \in G$. As defined in [60], the concept of directional derivatives in $\mathbb{R}^n$ is extended to functions on a Lie group as:

$$(\hat{\mathcal{X}}^r f)(g) \doteq \frac{d}{dt} f(g \circ \exp(t\mathcal{X}))\bigg|_{t=0}. \tag{42}$$

Note that t is just a scalar denoting the time, while $\mathbf{t}$ represents the translation part of a homogeneous transformation. Eq. (42) is the "right" Lie derivative and the "left" Lie derivative can be defined in a similar form. A gradient on $SE(3)$ is then defined using the right Lie derivative with the "natural" basis of Lie algebra $\{E_i\}$ where $i = 1, 2, ..., 6$. Therefore, the gradient of the function on Lie group $f(g)$ is as follows:

$$\nabla f(g) = \begin{pmatrix} \frac{d}{dt} f(g \circ \exp(tE_1))|_{t=0} \\ \frac{d}{dt} f(g \circ \exp(tE_2))|_{t=0} \\ \vdots \\ \frac{d}{dt} f(g \circ \exp(tE_6))|_{t=0} \end{pmatrix}. \tag{43}$$

In order to update g, the rigid body velocity is introduced where $V_g^r = g^{-1}\dot{g}$, and the update law is written as below:

$$g_{s+1} = g_s \exp(\Delta t V_g^r) \tag{44}$$

where $t_{s+1} = t_s + \Delta t$ is the discrete time step corresponding to g_{s+1}. To prevent steps from becoming too small, the update law of g_s is modified by defining V_g^r as:

$$V_g^r = g^{-1}\dot{g} = -\alpha \widehat{\nabla f(g)}. \tag{45}$$

where α is a scaling factor. After choosing the cost function as:

$$C(X) = \sum_{i=1}^{n} ||A_i X - X B_i|| \tag{46}$$

X can be optimized using Eq. (45). The gradient descent method updates the calibrations parameters online based on new incoming data. The initial guess of X will converge to the true X; however, the rate of convergence depends on how "good" the initial guesses are.

4.8 Batch Method

This section presents a probabilistic method [56] by Ackerman and Chirikjian to solve for X in the absence of a priori knowledge of the correspondence between the exact sets of measurements $A = \{A_i\}$ and $B = \{B_j\}$. In other words, the sets A and B each can be given as unordered "batches" without knowing how each A_i matches to a specific B_j.

A commonality of all the methods in the previous sections is that exact knowledge of the correspondence between $\{A_i\}$ and $\{B_j\}$ is assumed; however, this is not always the case. There are many instances in the literature when the sensor data used in calibration becomes "unsynchronized". Different attempts have been implemented to solve this problem, such as time stamping the data, developing dedicated software modules for syncing the data, and analyzing components of the sensor data stream to determine a correlation [17], to varying effects. The Batch method bypasses these issues altogether without tracking, or recomputing, correspondence. By modelling the set of As and Bs as probability distributions on $SE(3)$, the data can be taken as an unordered, uncorrelated "batch" and a solution for X can be generated.

4.8.1 The Batch Method Formulation

Given a large set of pairs $(A_i, B_i) \in SE(3) \times SE(3)$ for $i = 1, ..., n$ that exactly satisfy the equation:

$$A_i X = X B_i, \tag{47}$$

a new algorithm is developed to find $X \in SE(3)$. The Batch method addresses a generalization of the standard problem in which the sets $A = \{A_i\}$ and $B = \{B_j\}$ are provided with elements written in any order and it is known that a correspondence exists between the elements of these sets such that Eq. (47) holds, but no a priori knowledge of this correspondence is known between each A_i and B_j.

Define a Gaussian probability distribution on $SE(3)$ (assuming the norm $\|\Sigma\|$ is small) as:

$$\rho(H; M, \Sigma) = \frac{1}{(2\pi)^3 |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}F(M^{-1}H)}$$

where $|\Sigma|$ denotes the determinant of Σ and

$$F(H) = [\log^{\vee}(H)]^T \Sigma^{-1} [\log^{\vee}(H)].$$

When H is parameterized with exponential coordinates, $H = \exp Z$, this means that $F(\exp Z) = \mathbf{z}^T \Sigma^{-1} \mathbf{z}$ where $\mathbf{z} = Z^{\vee}$ and $\rho(\exp Z; \mathbb{I}_4, \Sigma)$ becomes exactly a zero-mean Gaussian distribution on the Lie algebra $se(3)$, with covariance Σ, that is 'lifted up' to the Lie group $SE(3)$. This definition is valid when $\|\Sigma\|$ is small enough that the tails of the distribution decay rapidly enough such that the value of ρ becomes negligible before they "wrap around" due to the topology of $SE(3)$.

Using formulations of probability theory on SE(3), Eq. (47) can be thought as:

$$(\delta_{A_i} * \delta_X)(H) = (\delta_X * \delta_{B_i})(H) \tag{48}$$

where $*$ denotes the convolution of functions on $SE(3)$, as defined in the Appendix.

Because convolution is a linear operation on functions, the following equation can be obtained by adding all n instances of Eq. (48):

$$(f_A * \delta_X)(H) = (\delta_X * f_B)(H) \tag{49}$$

where

$$f_A(H) = \frac{1}{n}\sum_{i=1}^{n} \delta(A_i^{-1}H) \text{ and } f_B(H) = \frac{1}{n}\sum_{i=1}^{n} \delta(B_i^{-1}H).$$

The above functions can be normalized to be probability densities:

$$\int_{SE(3)} f_A(H)dH = \int_{SE(3)} f_B(H)dH = 1.$$

See the Appendix for a review of the properties of integration on $SE(3)$.

Let the mean and covariance of a probability density function $f(H)$ be defined by the following conditions:

$$\begin{aligned} &\int_{SE(3)} \log(M^{-1}H) f(H) dH = \mathbb{O} \text{ and} \\ &\Sigma = \int_{SE(3)} \log^{\vee}(M^{-1}H)[\log^{\vee}(M^{-1}H)]^T f(H) dH. \end{aligned} \tag{50}$$

If $f(H)$ is of the form of $f_A(H)$ given above, then

$$\begin{aligned} &\sum_{i=1}^{n} \log(M_A^{-1}A_i) = \mathbb{O} \text{ and} \\ &\Sigma_A = \frac{1}{n}\sum_{i=1}^{n} \log^{\vee}(M_A^{-1}A_i)[\log^{\vee}(M_A^{-1}A_i)]^T. \end{aligned} \tag{51}$$

It can be shown that if these quantities are computed for two highly-focused functions, f_1 and f_2, that the same quantities for the convolution of these functions can be closely approximated as [61]:

$$M_{1*2} = M_1 M_2 \text{ and } \Sigma_{1*2} = Ad(M_2^{-1})\Sigma_1 Ad^T(M_2^{-1}) + \Sigma_2 \tag{52}$$

where

$$Ad(H) = \begin{pmatrix} R & \mathbb{O} \\ \hat{\mathbf{x}}R & R \end{pmatrix} \tag{53}$$

and $\widehat{\mathbf{a}}$ is the skew-symmetric matrix such that $\widehat{\mathbf{a}}\mathbf{b} = \mathbf{a} \times \mathbf{b}$.

The mean of $\delta_X(H)$ is $M_X = X$, and its covariance is the zero matrix. Therefore, Eq. (49) together with Eq. (52) yields two “Batch Method" equations:

$$\boxed{M_A X = X M_B} \tag{54}$$

and

$$\boxed{Ad(X^{-1})\,\Sigma_A\, Ad^T(X^{-1}) = \Sigma_B} \tag{55}$$

In Section 4.8.2, it will be shown how X can be recovered using Eq. (54) and Eq. (55).

4.8.2 A Batch Method Solution

Starting with Eq. (54), the solution space of X can be defined as a cylinder. Specifically Eq. (54) can be rewritten as:

$$\log^\vee(M_A) = Ad(X)\,\log^\vee(M_B). \tag{56}$$

In the case of general M_A and M_B (i.e. not degenerate cases in which the rotation angle[1] is outside of the range $(0,\pi)$), the solution space of all possible Xs that satisfy this equation is known to be two-dimensional. This can be seen by defining

$$\log^\vee(M) = \begin{pmatrix} \omega \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \theta\mathbf{n} \\ \mathbf{v} \end{pmatrix},$$

where $\omega \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$ are the elements of the Lie algebra, and writing the rotation and translation parts of Eq. (56) separately,

$$\mathbf{n}_A = R_X \mathbf{n}_B \quad \text{and} \tag{57}$$

$$\mathbf{v}_A = \theta_B \widehat{\mathbf{t}_X} R_X \mathbf{n}_B + R_X \mathbf{v}_B. \tag{58}$$

The first of these equations has a one-dimensional solution space of the form $R_X = R(\mathbf{n}_A,\mathbf{n}_B)R(\mathbf{n}_B,\phi)$ where $\phi \in [0,2\pi)$ is free and $R(\mathbf{n}_A,\mathbf{n}_B)$ is any rotation matrix that rotates the vector $\mathbf{n}_B$ into $\mathbf{n}_A$. In particular, choose

$$R(\mathbf{n}_A,\mathbf{n}_B) = \mathbb{I} + \widehat{\mathbf{n}_B \times \mathbf{n}_A} + \frac{(1-\mathbf{n}_B\cdot\mathbf{n}_A)}{\|\mathbf{n}_B \times \mathbf{n}_A\|^2}\left(\widehat{\mathbf{n}_B \times \mathbf{n}_A}\right)^2. \tag{59}$$

The rotation $R(\mathbf{n}_B,\phi)$ is given by Rodrigues' rotation formula:

$$R(\mathbf{n}_B,\phi) = \mathbb{I} + \sin\phi\,\widehat{\mathbf{n}_B} + (1-\cos\phi)\,(\widehat{\mathbf{n}_B})^2.$$

Substituting $R_X = R(\mathbf{n}_A,\mathbf{n}_B)R(\mathbf{n}_B,\phi)$ into Eq. (58) and rearranging terms, we obtain:

$$\frac{R(\mathbf{n}_A,\mathbf{n}_B)R(\mathbf{n}_B,\phi)\mathbf{v}_B - \mathbf{v}_A}{\theta_B} = \widehat{\mathbf{n}_A}\mathbf{t}_X. \tag{60}$$

The skew-symmetric matrix $\widehat{\mathbf{n}_A}$ has a rank of 2, so a free translational degree of freedom exists in $\mathbf{t}_X$ along the $\mathbf{n}_A$ direction. $\mathbf{t}_X$ can thus be described as:

$$\mathbf{t}_X = \mathbf{t}(s) = s\,\mathbf{n}_A + a\,\mathbf{m}_A + b\,\mathbf{m}_A \times \mathbf{n}_A \tag{61}$$

where $s \in \mathbb{R}$ is a second free parameter, $\mathbf{m}_A$ and $\mathbf{m}_A \times \mathbf{n}_A$ are defined to be orthogonal to $\mathbf{n}_A$ by construction. If $\mathbf{n}_A = [n_1,n_2,n_3]^T$ and n_1,n_2 are not simultaneously zero, then one can define[2]

$$\mathbf{m}_A \doteq \frac{1}{\sqrt{n_1^2+n_2^2}}\begin{pmatrix} -n_2 \\ n_1 \\ 0 \end{pmatrix}.$$

The coefficients a and b are then computed by substituting Eq. (61) into Eq. (60) and using the fact that $\{\mathbf{n}_A,\mathbf{m}_A,\mathbf{n}_A \times \mathbf{m}_A\}$ is an orthonormal basis for $\mathbb{R}^3$. Explicitly,

$$a = -\left(\frac{R(\mathbf{n}_A,\mathbf{n}_B)R(\mathbf{n}_B,\phi)\mathbf{v}_B - \mathbf{v}_A}{\theta_B}\right)\cdot(\mathbf{m}_A \times \mathbf{n}_A) \quad \text{and}$$

$$b = \left(\frac{R(\mathbf{n}_A,\mathbf{n}_B)R(\mathbf{n}_B,\phi)\mathbf{v}_B - \mathbf{v}_A}{\theta_B}\right)\cdot\mathbf{m}_A.$$

This means that the feasible solutions can be completely parameterized as:

$$X(\phi,s) = H(R(\mathbf{n}_A,\mathbf{n}_B)R(\mathbf{n}_B,\phi),\mathbf{t}(s)) \tag{62}$$

where $(\phi,s) \in [0,2\pi) \times \mathbb{R}$.

[1]This angle is computed from the Frobenius norm $\theta_A = \|\frac{1}{2}\log R_A\| = \|\frac{1}{2}\log R_B\| = \theta_B$.

[2]The special case when they are simultaneously zero is a set of measure zero, and hence is a rare event. Nevertheless, it is easy to handle, since in this case R_A is necessarily a rotation around $\mathbf{e}_3$.

To provide the additional constraints needed to solve for X, decompose Σ_A and Σ_B into blocks as:

$$\Sigma_i = \begin{pmatrix} \Sigma_i^1 & \Sigma_i^2 \\ \Sigma_i^3 & \Sigma_i^4 \end{pmatrix}$$

where $\Sigma_i^3 = (\Sigma_i^2)^T$, then take the first two blocks of Eq. (55) and write:

$$\Sigma_{M_B}^1 = R_X^T \Sigma_{M_A}^1 R_X \tag{63a}$$

$$\Sigma_{M_B}^2 = R_X^T \Sigma_{M_A}^1 R_X (\widehat{R_X^T t_x}) + R_X^T \Sigma_{M_A}^2 R_X \tag{63b}$$

Calculate the eigendecomposition as $\Sigma_i = Q_i \Lambda Q_i^T$, where Q_i is the square matrix whose ith column is the eigenvector of Σ_i and Λ is the diagonal matrix with corresponding eigenvalues as diagonal entries. Write the first block of Eq. (63b) as [56,62]:

$$\Lambda = Q_{M_B}^T R_X^T Q_{M_A} \Lambda Q_{M_A}^T R_X Q_{M_B} = \mathcal{Q} \Lambda \mathcal{Q}^T. \tag{64}$$

The set of $\mathcal{Q}$s that satisfy this equation are given as:

$$\mathcal{Q} = \left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \right\} \tag{65}$$

This means that the rotation component of X is given by:

$$R_x = Q_{M_A} \mathcal{Q} Q_{M_B}^T. \tag{66}$$

When Λ has repeated entries, a continuum of symmetries result. For example, in the extreme case when $\Lambda = \lambda \mathbb{I}_3$, the finite set $\mathcal{Q}$ would be replaced by $SO(3)$, which is the same as saying that Eq. (63a) imposes no constraint. In general, it is possible to construct trajectories that define the sets $\{A_i\}$ and $\{B_j\}$ such that this does not happen, and so the Batch method limits the discussion to the case where Eq. (65) holds.

Once the four possibilities of R_X are found in this manner, the corresponding possible $\mathbf{t}_X$ can be found easily from blocks 2 and 4 of in Eq. (63b).

The correct solution, from the set of 4 possibilities (given Eq. (66)), can be found by applying the cylindrical constraints as in Eq. (57) and Eq. (58). This is achieved by choosing the Q that minimizes a cost function, such as $\|\mathbf{n}_{M_A} - R_X \mathbf{n}_{M_B}\| + w\|\mathbf{v}_{M_A} - \theta_{M_B} \widehat{\mathbf{t}_X} R_X \mathbf{n}_{M_B} + R_X \mathbf{v}_{M_B}\|$ where w is an appropriately chosen weighting factor, and in this case, it is chosen to be unity.

5 DATA SELECTION AND ERROR METRICS

When performing hand-eye calibration in experiments, the accuracy of the calibrated X is highly dependent on the data that is obtained in the process. Tsai and Lenz [16] proposed several principles on designing the movement of the robot. This is useful when the motion planning under these constraints is practical. However, data selection has to be considered when such movement is not applicable, such as when there is a lack of free space or a hand-held sensor is used. In this section, several principles and methods for data selection are reviewed. In addition, different error metrics are also discussed to give a more complete picture of the $AX = XB$ problem.

5.1 Data Selection

Selection of well defined (A_i, B_i) is very important for the $AX = XB$ solvers. Data selection methods for off-line application have been proposed in [62–64], and corresponding selection techniques for on-line solvers are introduced in [65, 66]. For probabilistic methods, data sets $\{A_i\}$ and $\{B_i\}$ must be highly concentrated, which means small and relative motions are preferable [56]. This is opposite to the data selection criterion of other non-probabilistic approaches.

To determine the hand-eye transformation, at least 2 non-parallel rotation axes from the data pairs are needed (which is also referred at the non-parallelism criterion), and further data selection algorithms are all built on top of this.

In the error analysis of [16], four observations are given to show the relationships between the errors in rotation and translation and the features of the robot motions. In addition, seven steps are suggested to improve the calibration accuracy. Shi *et al.* [65] developed a motion selection algorithm based on three out of the four observations in [16]. However, the thresholds in [65] are chosen in a heuristic manner. To fix this problem, Zhang *et al.* [66] proposed an adaptive selection method which can update the thresholds online. All of the above approaches share some common standards such that small relative rotations between (A_i, B_i) and (A_{i+1}, B_{i+1}) should be avoided, and the rotation angles for both $\{A_i\}$ and $\{B_i\}$ should be large enough to avoid the singularity of the representation. The Kronecker product method, however, has a tolerance on small robot motions as pointed out in Section 4.5. It also offers an algebraic analysis to show what information of X can be obtained using certain types and numbers of motions. Interesting results are: (1) three independent pure translations can fully define R_X but not $\mathbf{t}_X$; (2) with two or more independent pure rotations, both R_X and $\mathbf{t}_X$ can be recovered. A more detailed summary can be seen in Table 1 of [49]. Schmidt *et al.* [18] discussed data selection for the dual quaternion hand-eye calibration algorithm based on a RANSAC approach for filtering, which shows that the dual quaternion method can yield a better X after data selection; without filtering the method can either fail or perform worse. However, the computation of all possible relative movements from the data set results in a long computational time.

For a set of continuous robot motions, a sequence of images or sensor recorded information will be obtained. But due to the small differences between consecutive images or sensor recordings, it is often undesirable to process the data in the temporal order because this can yield high errors. Schmidt *et al.* [63] then proposed a vector quantization based data selection technique which selects a globally consistent set of motions that optimizes the non-parallelism criterion. The main idea is to select a subset of the given rotation axes of (A_i, B_i) using clustering algorithms. It is shown that compared to the above two approaches, the algorithm presented is both fast and accurate. Ackerman *et al.* [62] also proposed a method which uses the Euclidean group invariants in the structure of $\{A_i\}$ and $\{B_i\}$ to realign asynchronous data streams.

Take the ultrasound sensor calibration in Fig. (2) as an example. The data stream $\{A_i\}$ is calculated from the joint angles of the robot arm, which is very accurate due to the high sensitivity of the encoder on each motor. However, due to the sensing accuracy of 2D US probe, $\{B_i\}$ obtained from the ultrasound sensor might not be as accurate and sometimes one can get B_k that is far from the "correct" measurement . In order to get X that is closer to the ground truth, the above data selection algorithms can be used to detect the mismatched data pair (A_k, B_k) caused by the overbig measurement noise on B_k.

5.2 Calibration Verification

Normally, the verification of a calibration algorithm is consisted of two parts: synthetic data and real experiments. For the verification with synthetic data, ground truth X_{true} is usually given in advance. The data stream $\{A_i\}$ (or $\{B_i\}$) then is generated according to a predetermined trajectory or distribution on $SE(3)$ and the other stream is obtained by $B_i = X_{true}^{-1} A_i X_{true}$ (or $A_i = X_{true} B_i X_{true}^{-1}$). Till this point, both $\{A_i\}$ and $\{B_i\}$ are noise-free which can be regarded as the exact sensor measurements in an ideal experiment. However, different types and levels of noise exist in real experiments, and noise is usually artificially applied onto either one or both of the data streams. There are a lot of ways to apply noise onto the data stream and one can choose based on the properties of the sensor measurement in the specific experiment setup. At last, noisy data streams are fed into the calibration solvers to compute X_{calc}, which will be compared with X_{true} using error metrics. The verification of calibration solver with synthetic data is relatively consistent in the literature, because one can artificially provide the ground truth X_{true}. However, in the real experiment, it is quite often impossible to get the ground truth of X and the validation process usually uses the calibrated X_{calc} for other tasks, e.g. 3D reconstruction, to evaluate its accuracy. The validation procedure in real experiments varies from platform to platform.

5.3 Error Metrics

There are multiple ways to define the errors of rigid body transformations, and some methods rely heavily on the metric that is chosen [67]. One approach is to measure the errors of R_x and $\mathbf{t}_X$ simultaneously which is rarely seen in more recent literature. The other approach is to measure the rotation error and translation error separately.

5.3.1 Metrics for Rotation and Translation Errors

Various metrics for rotation error have been used in the literature. In [16] and [46], the matrix error metric is defined as:

$$E_{rot} \doteq \| R_{X_{true}} - R_{X_{calc}} \|. \tag{67}$$

However, this is less preferable because rotation matrices lie in $SO(3)$ and the deduction operation is not defined for $SO(3)$. In [48] and [49], the quaternion error metric is used as:

$$E_{rot} \doteq \| \mathbf{q}_{X_{true}} - \mathbf{q}_{X_{calc}} \|. \tag{68}$$

As noted in [47], $\| R_{X_{true}} - R_{X_{calc}} \|_F$ is $2\sqrt{2}$ times larger than $\| \mathbf{q}_{X_{true}} - \mathbf{q}_{X_{calc}} \|$ so it is important to maintain a consistent error metric especially for result comparisons. Another rotational error metric is to calculate the norm of the relative rotation between X_{true} and X_{calc} as:

$$E_{rot} \doteq \| R_{X_{true}}^T R_{X_{calc}} \|. \tag{69}$$

In [56], the Lie algebra error metric is defined for the relative rotation $R_{X_{true}}^T R_{X_{calc}}$ as:

$$E_{rot} \doteq \| \log^{\vee}(R_{X_{true}}^T R_{X_{calc}}) \|. \tag{70}$$

Metrics for the translation error are relatively simple because translation lies in Euclidean space. A common method is to use the relative translation error to eliminate the influence of the translation unit:

$$E_{tran} \doteq \frac{\| \mathbf{t}_{X_{true}} - \mathbf{t}_{X_{calc}} \|_2}{\| \mathbf{t}_{X_{true}} \|_2}. \tag{71}$$

Conventionally, multiple trials are performed at each fixed noise level in numerical simulation, and the "averaged" errors in rotation and translation are defined in [46] as:

$$e_{rot} \doteq \sqrt{\frac{1}{N} \sum_{i=1}^{N} E_{rot}^2}, \tag{72}$$

$$e_{tran} \doteq \sqrt{\frac{1}{N} \sum_{i=1}^{N} E_{tran}^2}. \tag{73}$$

It should be noted due to the diversity of rotation error metrics, it is yet to be seen which metric is better or whether different metrics make a difference at all.

6 CONCLUSION AND FUTURE DIRECTIONS

In this paper, the "AX=XB" formulation of the sensor calibration problem is examined, which is widely used in camera calibration, humanoid head-eye calibration, robot eye-to-hand calibration, aerial vehicle sensor calibration, and IGT sensor calibration. A review of some of the most influential and effective methods was presented and their positive and negative traits were discussed. For the various $AX = XB$ solvers, the focus is put on the case where there is noise on the incoming sensor data, and therefore multiple sensor readings are needed. It was clear that each algorithm has strengths and weaknesses in different contexts, and it is important to use the appropriate method for different circumstances.

In addition to measurement error contributing to noise, it was emphasized that the sensor data streams containing the As and Bs may be present at different sampling rates, may be asynchronous, and/or each stream may contain gaps in information. Therefore, a probabilistic method is reviewed in detail which is used for calculating the calibration transformation that works for data without any a priori knowledge of the correspondence between the As and Bs. Data selection is of critical importance to $AX = XB$ solvers. Depending on the quality of the data pairs, the usage of data selection techniques can either greatly improve the final result or prevent the solver from failing.

Though many algorithms are available for hand-eye calibration, which algorithm is preferable for the given type and number of motions is still unclear. It will be helpful to categorize the type of motions and their corresponding preferable solvers.

Appendix A: Integration and Convolution on SE(3)

This appendix reviews the features of integration and convolution on the group $SE(3)$ that are relevant to the formulation in this paper. For more detailed treatments see [60, 68].

Integration

$SE(3)$ is a six-dimensional matrix Lie group. If $H = H(\mathbf{q})$ where $\mathbf{q} = [q_1, ..., q_6]^T$ is a global set of coordinates, then functions $f : SE(3) \to \mathbb{R}$ can be integrated as

$$\int_{SE(3)} f(H)dH \doteq \int_{\mathbf{q}\in D} f(H(\mathbf{q}))|J(\mathbf{q})|d\mathbf{q}$$

where D is the domain of integration in the parameter space and $d\mathbf{q} = dq_1 dq_2 \cdots dq_6$. The Jacobian determinant $|J(\mathbf{q})|$ is computed from the Jacobian matrix:

$$J(\mathbf{q}) = \left[\left(H^{-1}\frac{\partial H}{\partial q_1}\right)^{\vee}; \left(H^{-1}\frac{\partial H}{\partial q_2}\right)^{\vee}; \cdots \left(H^{-1}\frac{\partial H}{\partial q_6}\right)^{\vee} \right].$$

For example, if Cartesian coordinates are used for the translation vector and ZXZ Euler angles are used for rotations, then $\mathbf{q} = [x, y, z, \alpha, \beta, \gamma]^T$, $D = \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times [0, 2\pi] \times [0, \pi] \times [0, 2\pi]$ and $|J(\mathbf{q})| = \sin\beta$. While D and $J(\mathbf{q})$ will change depending on which parametrization is used, the value of the integral itself does not as it is a property of the Lie group. The integral depends on the function themselves but not a matter of how the function is expressed or the coordinates of the integral.

$SE(3)$ is unimodular, which means that the integation measure, $dH = |J(\mathbf{q})|d\mathbf{q}$, has the property that for any fixed $H_0 \in SE(3)$ and "well-behaved fuction[3]" $f : SE(3) \to \mathbb{R}$, [60]

$$\int_{SE(3)} f(H_0 \circ H)dH = \int_{SE(3)} f(H \circ H_0)dH = \int_{SE(3)} f(H)dH. \tag{74}$$

In addition, it can be shown that when these conditions hold, so too does

$$\int_{SE(3)} f(H^{-1})dH = \int_{SE(3)} f(H)dH. \tag{75}$$

A common source of confusion is that many books on Lie groups are concerned with compact Lie groups, which possess both bi-invariant metrics and bi-invariant integration measures. When discussing the noncompact case, bi-invariant metrics generally do not exist (except for special cases such as products of tori and Euclidean spaces), and they do not exist for $SE(3)$. Though bi-invariant integration measures also do not exist in general, they do exist for a broader class of special noncompact Lie groups, and this includes $SE(3)$.

Convolution

Given two functions, $f_1, f_2 \in (L^1 \cap L^2)(SE(3))$, the convolution is defined as

$$(f_1 * f_2)(H) \doteq \int_{SE(3)} f_1(K) f_2(K^{-1}H)dK. \tag{76}$$

This integral can be rewitten in a number of equivalent forms using Eq. (74) and Eq. (75). Convolution inherits the associative property from the underlying group, which is written as

$$(f_1 * f_2) * f_3 = f_1 * (f_2 * f_3)$$

where the dependence of these functions on H has been temporarily suppressed. Analogous with the way convolution inherits associativity, it also inherits noncommutativity for general functions, with the exception of special functions called "class functions".

[3]Here a well-behaved function means a function for which the integral exists, and hence $f \in L^1(SE(3))$, and later that the convolution integral exists, which is guaranteed by further requiring that $f \in L^2(SE(3))$. And so, with the notable exception of the Dirac delta function, the discussion is restricted to $f \in (L^1 \cap L^2)(SE(3))$.

The Dirac delta function on $SE(3)$ is defined as follows:

$$\delta(H) = \begin{cases} +\infty, & H = I \\ 0, & H \neq I \end{cases} \tag{77}$$

which satisfies the constraint that:

$$\int_{SE(3)} \delta(H)dH = 1. \tag{78}$$

A slightly further expansion of allowable functions to include shifted delta functions of the form:

$$\delta_X(H) \doteq \delta(X^{-1}H) = \delta(HX^{-1}). \tag{79}$$

The unshifted delta function is an example of a symmetric function, in that $\delta(H) = \delta(H^{-1})$.

If the class of functions is expanded to consider beyond $(L^1 \cap L^2)(SE(3))$ to include Dirac delta functions, then the following is true for every $f \in (L^1 \cap L^2)(SE(3))$:

$$(f * \delta)(H) = (\delta * f)(H) = f(H).$$

Using the properties of the invariant integral on $SE(3)$, convolving a shifted delta function with an arbitrary function transfers the shift:

$$\begin{aligned}(\delta_X * f)(H) &= \int_{SE(3)} \delta(X^{-1}K) f(K^{-1}H) dK \\ &= \int_{SE(3)} \delta(J) f((XJ)^{-1}H) dK = f(X^{-1}H)\end{aligned} \tag{80}$$

where the change of variables $J = X^{-1}K$ and the invariance of integration have been used.

References

[1] ARS. https://ars-project.readthedocs.org. Accessed: 2015-12-19.

[2] Koenig, N., and Howard, A., 2004. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2149–2154.

[3] Aguero, C., Koenig, N., Chen, I., Boyer, H., Peters, S., Hsu, J., Gerkey, B., Paepcke, S., Rivero, J., Manzo, J., Krotkov, E., and Pratt, G., 2015. "Inside the virtual robotics challenge: Simulating real-time robotic disaster response". *IEEE Transactions on Automation Science and Engineering,* **12**(2), April, pp. 494–506.

[4] Echeverria, G., Lemaignan, S., Degroote, A., Lacroix, S., Karg, M., Koch, P., Lesire, C., and Stinckwich, S., 2012. "Simulating complex robotic scenarios with morse". In SIMPAR, pp. 197–208.

[5] Webots. http://www.cyberbotics.com. Commercial Mobile Robot Simulation Software.

[6] Michel, O., 2004. "Webots: Professional mobile robot simulation". *Journal of Advanced Robotics Systems,* **1**(1), pp. 39–42.

[7] E. Rohmer, S. P. N. Singh, M. F., 2013. "V-rep: a versatile and scalable robot simulation framework". In Proc. of The International Conference on Intelligent Robots and Systems (IROS).

[8] RoboDK. http://www.robodk.com. Accessed: 2015-12-19.

[9] SimSpark. http://simspark.sourceforge.net/. Accessed: 2015-12-19.

[10] CarMaker. http://ipg.de/simulationsolutions/carmaker. Accessed: 2015-12-19.

[11] PreScan. https://www.tassinternational.com/prescan. Accessed: 2015-12-19.

[12] AdasWorks. https://adasworks.com. Accessed: 2015-12-19.

[13] Google. https://www.google.com/selfdrivingcar. Accessed: 2015-12-19.

[14] BMW. http://www.bmw.com. Accessed: 2015-12-19.

[15] Tesla. https://www.teslamotors.com/presskit/autopilot. Accessed: 2015-12-19.

[16] Tsai, R., and Lenz, Y., 1989. "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration". *IEEE Transactions on Robotics and Automation,* **5**(3), pp. 345–358.

[17] Mair, E., Fleps, M., Suppa, M., and Burschka, D., 2011. "Spatio-temporal initialization for imu to camera registration". In IEEE International Conference on Robotics and Biomimetics, IEEE, pp. 557–564.

[18] Schmidt, J., Vogt, F., and Niemann, H., 2003. "Robust hand–eye calibration of an endoscopic surgery robot using dual quaternions". In *Pattern Recognition*. Springer, pp. 548–556.

[19] Malm, H., and Heyden, A., 2000. "A new approach to hand-eye calibration". In International Conference on Pattern Recognition, Vol. 1, IEEE, pp. 525–529.

[20] Heller, J., Havlena, M., and Pajdla, T., 2012. "A branch-and-bound algorithm for globally optimal hand-eye calibration". In IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 1608–1615.

[21] Ruland, T., Pajdla, T., and Kruger, L., 2012. "Globally optimal hand-eye calibration". In IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 1035–1042.

[22] Wu, H., Tizzano, W., Andersen, T. T., Andersen, N. A., and Ravn, O., 2014. "Hand-eye calibration and inverse kinematics of robot arm using neural network". In *Robot Intelligence Technology and Applications 2*. Springer, pp. 581–591.

[23] Kim, S.-J., Jeong, M., Lee, J., Lee, J., Kim, K., You, B., and Oh, S. "Robot head-eye calibration using the minimum variance method". pp. 1446 – 1451.

[24] Liu, Y., Wang, Q., and Li, Y., 2015. "Calibration of a robot hand-eye system with a concentric circles target". In International Bhurban Conference on Applied

Sciences and Technology, IEEE, pp. 204–209.

[25] Ma, Q., Li, H., and Chirikjian, G. "New probabilistic approaches to the ax = xb hand-eye calibration without correspondence". In International Conference on Robotics and Automation, IEEE.

[26] Zhuang, H., Roth, Z. S., and Sudhakar, R., 1994. "Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form AX = YB". *IEEE Transactions on Robotics and Automation,* **10**(4), pp. 549–554.

[27] Dornaika, F., and Horaud, R., 1998. "Simultaneous robot-world and hand-eye calibration". *IEEE Transactions on Robotics and Automation,* **14**(4), pp. 617–622.

[28] Hirsh, R. L., DeSouza, G. N., and Kak, A. C., 2001. "An iterative approach to the hand-eye and base-world calibration problem". In IEEE International Conference on Robotics and Automation, Vol. 3, IEEE, pp. 2171–2176.

[29] Li, A., Wang, L., and Wu, D., 2010. "Simultaneous robot-world and hand-eye calibration using dual-quaternions and kronecker product". *International Journal of Physical Science,* **5**(10), pp. 1530–1536.

[30] Ernst, F., Richter, L., Matthäus, L., Martens, V., Bruder, R., Schlaefer, A., and Schweikard, A., 2012. "Non-orthogonal tool/flange and robot/world calibration". *The International Journal of Medical Robotics and Computer Assisted Surgery,* **8**(4), pp. 407–420.

[31] Heller, J., Henrion, D., and Pajdla, T., 2014. "Hand-eye and robot-world calibration by global polynomial optimization". In IEEE International Conference on Robotics and Automation, IEEE, pp. 3157–3164.

[32] Li, H., Ma, Q., Wang, T., and Chirikjian, G. "Simultaneous hand-eye and robot-world calibration by solving the ax= yb problem without correspondence".

[33] Wang, J., Wu, L., Meng, M. Q.-H., and Ren, H., 2014. "Towards simultaneous coordinate calibrations for cooperative multiple robots". In IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 410–415.

[34] Ma, Q., Goh, Z., and Chirikjian, G. "New probabilistic approaches to the ax = xb hand-eye calibration without correspondence". In Robotics: Science and Systems.

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

[36] Park, F., and Martin, B., 1994. "Robot sensor calibration: solving AX = XB on the euclidean group". *IEEE Transactions on Robotics and Automation,* **10**(5), pp. 717–721.

[37] Shiu, Y., and Ahmad, S., 1989. "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX = XB". *IEEE Transactions on Robotics and Automation,* **5**(1), pp. 16–29.

[38] Chou, J., and Kamel, M., 1991. "Finding the position and orientation of a sensor on a robot manipulator using quaternions". *The International Journal of Robotics Research,* **10**(3), pp. 240–254.

[39] Shiu, Y., and Ahmad, S., 1987. "Finding the mounting position of a sensor by solving a homogeneous transform equation of the form AX = XB". In IEEE International Conference on Robotics and Automation, Vol. 4, IEEE, pp. 1666–1671.

[40] Dai, Y., Trumpf, J., Li, H., Barnes, N., and Hartley, R., 2010. "Rotation averaging with application to camera-rig calibration". In *Computer Vision–ACCV 2009*. Springer, pp. 335–346.

[41] Shah, M., Eastman, R., and Hong, T., 2012. "An overview of robot-sensor calibration methods for evaluation of perception systems". In Proceedings of the Workshop on Performance Metrics for Intelligent Systems, ACM, pp. 15–20.

[42] Fassi, I., and Legnani, G., 2005. "Hand to sensor calibration: A geometrical interpretation of the matrix equation AX = XB". *Journal of Robotic Systems,* **22**(9), pp. 497–506.

[43] Zhao, Z., and Liu, Y., 2006. "Hand-eye calibration based on screw motions". In International Conference on Pattern Recognition, Vol. 3, IEEE, pp. 1022–1026.

[44] Gwak, S., Kim, J., and Park, F. C., 2003. "Numerical optimization on the euclidean group with applications to camera calibration". *IEEE Transactions on Robotics and Automation,* **19**(1), pp. 65–74.

[45] Chou, J. C., and Kamel, M., 1988. "Quaternions approach to solve the kinematic equation of rotation, AX = XB, of a sensor-mounted robotic manipulator". In IEEE International Conference on Robotics and Automation, IEEE, pp. 656–662.

[46] Horaud, R., and Dornaika, F., 1995. "Hand-eye calibration". *The International Journal of Robotics Research,* **14**(3), pp. 195–210.

[47] Daniilidis, K., and Bayro-Corrochano, E., 1996. "The dual quaternion approach to hand-eye calibration". In International Conference on Pattern Recognitoin, Vol. 1, IEEE, pp. 318–322.

[48] Daniilidis, K., 1999. "Hand-eye calibration using dual quaternions". *The International Journal of Robotics Research,* **18**(3), pp. 286–298.

[49] Andreff, N., Horaud, R., and Espiau, B., 1999. "On-line hand-eye calibration". In International Conference on 3D Digital Imaging and Modeling, IEEE, pp. 430–436.

[50] Andreff, N., Horaud, R., and Espiau, B., 2001. "Robot hand-eye calibration using structure-from-motion". *The International Journal of Robotics Research,* **20**(3), pp. 228–248.

[51] Zhao, Z., 2011. "Hand-eye calibration using convex optimization". In IEEE International Conference on Robotics and Automation, IEEE, pp. 2947–2952.

[52] Seo, Y., Choi, Y.-J., and Lee, S. W., 2009. "A branch-and-bound algorithm for globally optimal calibration of a camera-and-rotation-sensor system". In International Conference on Computer Vision, IEEE, pp. 1173–1178.

[53] Schmidt, J., Vogt, F., and Niemann, H., 2005.

"Calibration–free hand–eye calibration: a structure–from–motion approach". In *Pattern Recognition*. Springer, pp. 67–74.

[54] Angeles, J., Soucy, G., and Ferrie, F. P., 2000. "The on-line solution of the hand-eye problem". *IEEE Transactions on Robotics and Automation,* **16**(6), pp. 720–731.

[55] Ackerman, M. K., Cheng, A., Boctor, E., and Chirikjian, G., 2014. "Online ultrasound sensor calibration using gradient descent on the euclidean group". In IEEE International Conference on Robotics and Automation, IEEE, pp. 4900–4905.

[56] Ackerman, M. K., and Chirikjian, G. S., 2013. "A probabilistic solution to the AX = XB problem: Sensor calibration without correspondence". In *Geometric Science of Information*. Springer, pp. 693–701.

[57] Ackerman, M. K., Cheng, A., and Chirikjian, G., 2014. "An information-theoretic approach to the correspondence-free AX = XB sensor calibration problem". In IEEE International Conference on Robotics and Automation, IEEE, pp. 4893–4899.

[58] Horn, B., 1986. "Robot vision".

[59] Lepetit, V., Moreno-Noguer, F., and Fua, P., 2009. "Epnp: An accurate o (n) solution to the pnp problem". *International Journal of Computer Vision,* **81**(2), pp. 155–166.

[60] Chirikjian, G. S., and Kyatkin, A. B., 2001. *Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups*. CRC Press.

[61] Wang, Y., and Chirikjian, G. S., 2008. "Nonparametric second-order theory of error propagation on motion groups". *International Journal of Robotics Research*, p. 1258âĂŞ1273.

[62] Ackerman, M., Cheng, A., Shiffman, B., Boctor, E., and Chirikjian, G. S., 2013. "Sensor calibration with unknown correspondence: Solving $AX = XB$ using euclidean-group invariants". In IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 1308–1313.

[63] Vogt, J. S. F., and Niemann, H., 2004. "Vector quantization based data selection for hand-eye calibration". In Vision, Modeling, and Visualization 2004: Proceedings, November 16-18, 2004, Standford, USA, IOS Press, p. 21.

[64] Schmidt, J., and Niemann, H., 2008. "Data selection for hand-eye calibration: a vector quantization approach". *The International Journal of Robotics Research,* **27**(9), pp. 1027–1053.

[65] Shi, F., Wang, J., and Liu, Y., 2005. "An approach to improve online hand-eye calibration". In *Pattern Recognition and Image Analysis*. Springer, pp. 647–655.

[66] Zhang, J., Shi, F., and Liu, Y., 2005. "An adaptive selection of motion for online hand-eye calibration". In *Advances in Artificial Intelligence*. Springer, pp. 520–529.

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

[68] Chirikjian, G. S., 2011. *Stochastic Models, Information Theory, and Lie Groups: Vol. 2*. Birkauser, Boston.