Review of Methods for Solving AX=XB Sensor Calibration Problem

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

Remember to relate the content to humanoid robot. An often used formulation of sensor calibration in robotics and computer vision is "AX=XB", where A, X, and B are rigid-body motions with A and B given from sensor measurements, and X is the unknown calibration parameter. We compare some of the most effective algorithms that have been recorded in the literature and present new algorithms for solving this problem. We focus on the case when there is noise on the incoming sensor data, and, therefore, multiple sensor readings are needed. Furthermore, in practical problems, it is often the case that the sensor data streams containing the As and Bs, will not only contain measurement error, but they may present at different sample rates, or may be asynchronous. Each stream may contain gaps in information. We, therefore, present a method for calculating the calibration transformation, X, that works for data without any a priori knowledge of the correspondence between the As and Bs. The new method formulates the sensor data as probability distributions and gives the mean solution, X_{solved}. Additionally, we show how the covariance of X_{solved} can be determined, which could provide insight into the magnitude and type of noise on the sensor measurements.

Nomenclature

$SO(3)$ $SO(3) \doteq \{R|RR^T = R^TR = \mathbb{I}$ and $\det(R) = 1$ where $R \in \mathbb{R}^{3\times3}\}$

$SE(3)$ $SE(3) \doteq \{H|H = \left(R\,\mathbf{t}; \mathbf{0}^T 1\right) \in \mathbb{R}^4$, where $R \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3\}$

$so(3)$ $so(3) \doteq \{\Omega|R = exp(\Omega)$, where $\Omega \in \mathbb{R}^{3\times3}$ and $R \in SO(3)\ \}$

$se(3)$ $se(3) \doteq \{\Xi|H = exp(\Xi)$, where $\Xi = \left(\Omega\,\xi; \mathbf{0}^T 0\right)$ 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

$\mathbb{O}^{nxn}$ $n \times n$ zero matrix

$\mathbb{I}_n$ $n \times n$ identity matrix

$\{E_i\}$ the set of "natural" basis elements for Lie algebra

$^\vee$ The "vee" operator is defined such that $\left(\sum_{i=1}^{n} x_i E_i\right)^\vee \doteq (x_1, x_2, ..., x_n)^T$ where $n = 3$ for $so(3)$ and $n = 6$ for $se(3)$

$\hat{}$ The "hat" operator is the inverse of the "vee" operator. $\widehat{(x_1, x_2, ..., x_n)^T} \doteq \left(\sum_{i=1}^{n} x_i E_i\right)$

$\circ$ The operator defined for group product

$\odot$ The operator defined for quaternion product

$\hat{\odot}$ The operator defined for dual quaternion product

vec The "vec" operator is defined such that $vec(A) = [a_{11}, ..., a_{1m}, a_{21}, ..., a_{2m}, ..., a_{n1}, ..., a_{nm}]^T$ for $A = [a_{ij}] \in \mathbb{R}^{m\times n}$

$\mathbf{p}$ For $P \in G$ (where G is a Lie group, e.g. $SE(3)$ or $SO(3)$), $\mathbf{p} = (p_1, p_2, ..., p_n)^T = \log^\vee(P)$

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 one sensor measurement source

X The rigid body transformation ($X_i \in SE(3)$) 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

$\tilde{\mathbf{n}}$ The skew-symmetric representation of the axis of rotation ($\mathbf{n}_H$)

θ_H The angle of rotation for R_H about $\mathbf{n}_H$

ρ A probability distribution of $H \in G$ on SE(3)

M For ρ, M is the mean of the distribution

*Address all correspondence to this author.

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

In the fields of robotics and computer vision, sensor calibration problems are often codified using the "AX=XB" formulation, (Fig.1). Example applications include camera calibration, Cartesian robot hand calibration, robot eye-to-hand calibration [**?**], aerial vehicle sensor calibration [**?**] and image guided therapy (IGT) sensor calibration [**?**]. 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 motion 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), then the problem can be solved. 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 least error given corresponding noisy pairs (A_i,B_i) for i=1,2,...,n.

2 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:

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

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:

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

for $A,B,X \in SE(3)$, 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 (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) [**?**,**?**,**?**]. Additionally, if there is sensor error, then it may not be possible to find

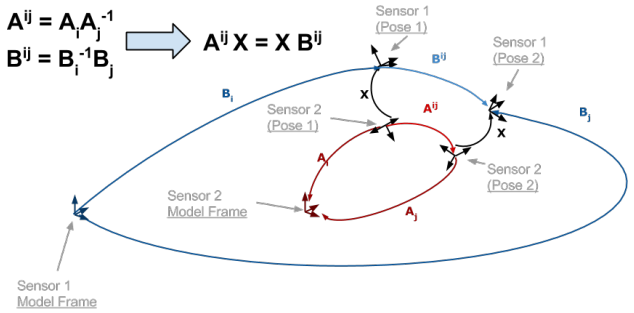

Fig. 1. The AX=XB Sensor Calibration Formulation Can be replaced with some humanoid robot picture

compatible pairs that reproduce the exact value of X. For this reason, minimization and least squared 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:

$$R_AR_X = R_XR_B \quad \text{and} \quad R_A\mathbf{t}_X + \mathbf{t}_A = R_X\mathbf{t}_B + \mathbf{t}_X. \tag{3}$$

The strategy to solve Eq.(2) would appear to reduce to first solving the part of Eq.(3) with only rotation and then rearranging the second equation so as to find acceptable values of $\mathbf{t}_X$: $(R_A - \mathbb{I}_3)\mathbf{t}_X = R_X\mathbf{t}_B - \mathbf{t}_A$. However, there are some problems with this naive approach. As pointed out in [?, ?], in nondegenerate cases, there is a one-parameter set of solutions to the first part in Eq.(3), 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.

However, 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:

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

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

The paper is organized as follows. In section 3, we give a detailed review of the existing methods that solve $AX = XB$ problem using $\{A_i,B_i\}$ pairs with correspondence. The notations in the literature are not consistent and we tried our best to standardize the notations across all the reviewed methods here so that readers can have a straight forward and consistent view of the methods and their relationships. All these methods are able to solve for both the noisy and no noise case. Some approaches are preferable under certain circumstance and their advantages and disadvantages are also highlighted. In section 4, numerical simulations are performed on

the selected methods and further conclusions are given based on the results of the simulation. In section 5, a new method called the batch method is presented which does not need a priori knowledge of the correspondence between the sets of measurements $A = \{A_i\}$ and $B = \{B_i\}$.

3 EXISTING SOLUTION METHODS

The problem of solving (2) for X when multiple corresponding pairs of As and Bs are presented has a history that goes back more than a quarter of a century [**?**, **?**, ?, ?], with the earliest proposed by Tsai [?] and Shiu [?], and applications involving this problem remain active today [**?**, **?**, **?**]. Shah [**?**] overviewed several different AX = XB methods qualitatively, which include but are not limited to the Shiu method, the Lie Group method, the quaternion method, the dual quaternion method, and the Kronecker method. Several new methods emerged recently such as the convex optimization method [**?**], the global polynomial optimization method [**?**], and the gradient descent method [**?**] etc. In order to better compare the existing multiple AX = XB methods, we standardize the notations and provide numerical simulations of the above methods. All of the methods mentioned in this section assumes strict correspondence between A_i and B_i.

3.1 Shiu and Ahmad

The Shiu and Ahmad method [?] uses 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 the tolerance of noise to a certain level, 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 least square technique given known R_X.

The closed form expression for R_X is:

$$R_X = e^{\tilde{\mathbf{n}}_A \beta} R_{X_P} \tag{5}$$

where

$$\begin{gathered} R_{X_p} = e^{\tilde{\mathbf{n}}_X \theta_X} \\ \mathbf{n}_X = \mathbf{n}_B \times \mathbf{n}_A \\ \theta_X = \text{atan2}(|\mathbf{n}_\text{B} \times \mathbf{n}_\text{A}|, \mathbf{n}_\text{B} \cdot \mathbf{n}_\text{A}) \end{gathered}$$

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

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

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) in order to calculate the unique solution of X. Given two pairs of As and Bs, we have two equations as:

$$\begin{aligned} A_1 X &= X B_1 \\ A_2 X &= X B_2. \end{aligned} \tag{7}$$

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

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

In Eq.(8), $Y = \big(\ \cos(\beta_1),\ \sin(\beta_1),\ \cos(\beta_2),\ \sin(\beta_2)\ \big)^T$, $C \in \mathbb{R}^{9\times 4}$ and $D \in \mathbb{R}^{9\times 1}$ are determined by $\mathbf{n}_{A_1}$,$\mathbf{n}_{B_1}$, $\mathbf{n}_{A_2}$ and $\mathbf{n}_{B_2}$, and the explicit expressions are given in Eq.(44) of [?].
Similarly, with kwown R_X, $\mathbf{t}_x$ can be calculated using the least square method:

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

However, the method proposed here is constrained by two uniqueness conditions of solution, and the author fails to give an explicit closed-form solution of X (neither rotation part R_X or translation part $\mathbf{t}_X$) based on two pairs of (A_i, B_i).

3.2 Lie Group Method

The Lie Group method [?] is the first method to solve the AX = XB problem from the perspective of Lie group. 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.

3.2.1 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

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

The solution for $\mathbf{t}_X$ can be obtained by Eq.(9) following the same procedure.

3.2.2 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 translation vector $\mathbf{t}_x$ is obtained using least square 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

$$M = \sum \mathbf{n}_{B_i} \mathbf{n}_{A_i}^T.$$

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

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

where

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

3.3 Quaternion Method

The Quaternion Method proposed by Chou [?] uses normalized quaternions to transform the rotation parts of $A_i X = X B_i$ (i = 1,2) into two linear systems. Then singular value decomposition method is performed to obtain a closed form solution of R_X. In order to estimate X given multiple pairs of (A_i, B_i) with noise, Horaud and Dornaika [**?**] 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 least square method separately and (2) estimate R_X and $\mathbf{t}_X$ simultaneously by minimizing an objective function that incorporates the information of both rotation and translation. Method (2) is a nonlinear optimization problem which requires an initial guess. In this section, we introduce only the first approach, and the corresponding numerical simulation will be given in section 4.

3.3.1 Closed Form Solution with Two Exact Pairs

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

$$R_A R_X = R_X R_B \;\Leftrightarrow\; q_A \odot q_X = q_X \odot q_B \tag{13}$$

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 the quaternion multiplication.

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

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

α_0 is 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 as

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

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.

For the unit quaternion representing the rotation part of X, the resulting expression is:

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

To obtain V_2 and y_2, E is first written as $E = \sin(\theta_{A|B}/2)M$ with $\theta_{A|B} = \theta_A = \theta_B$, which is the constraint that corresponding A and B should have the same angle of rotation resulted from the equation AX = XB. Then the singular value decomposition of M is computed as $M = U\Sigma V^T$ with $V = (V_1, V_2)$, $V_1 \in \mathbb{R}^{4\times2}$ and $V_2 \in \mathbb{R}^{4\times2}$. 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 $\mathbf{y_2} \in \mathbb{R}^{2\times1}$.

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

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

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

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

3.3.2 Estimation of X Using Multiple Pairs With Noise

In the case where there are n pairs of (A_i, B_i), the problem of recovering R_X is converted into minimizing an error objective function as in Eq.(18):

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

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 $\tilde{K}_i \in \mathbb{R}^{4\times4}$ 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 unit quaternion, the Lagrangian multiplier is introduced in Eq.(19) as:

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

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

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

A closed form solution can be easily found, and 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 technique as introduced in previous methods.

3.4 Dual Quaternion Method

The dual quaternion method [**?**] treats the rotation and translation parts of the matrix X in a unified way and facilitates a new simultaneous solution of X using the singular value decomposition method. To begin with, Eq.(2) is transformed into an equation in dual quaternion form as follows:

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

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 4 by 4 rigid transformation matrix is defined as follows:

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

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:

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

Similar to Section 3, $\mathbf{q}_X$ and $\mathbf{q}'_X$ are the vector representations of q_X and q'_X. Then equation $A_iX = XB_i$ can be converted into the form as below:

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

where

$$S_i = \begin{pmatrix} \mathbf{a}-\mathbf{b} & (\mathbf{a}+\mathbf{b})^\wedge & \mathbf{0_{3\times1}} & \mathbf{0_{3\times3}} \\ \mathbf{a}'-\mathbf{b}' & (\mathbf{a}'+\mathbf{b}')^\wedge & \mathbf{a}-\mathbf{b} & (\mathbf{a}+\mathbf{b})^\wedge \end{pmatrix} \in \mathbb{R}^{6\times8}. \tag{26}$$

$\mathbf{a}' = \frac{1}{2}\mathbf{t}_X \times \mathbf{a}$ and $\mathbf{a}$ is the vector part of q_X. After concatenating S_i, 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 singular value decomposition on $T = U\Sigma V^T$, the dual quaternion for matrix X can be expressed as the linear combination of the last two right-singular vectors ($\mathbf{v_7}$, $\mathbf{v_8}$) of matrix T , which are just the last two columns of matrix V, as shown in Eq.(28):

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

Different from the quaternion method (13), 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. Despite the advantages of the dual quaternion method, it has the drawback of having to filter the data pairs to provide inappropriate solutions when there is noise on A_i and B_i.

3.5 Kronecker Product Method

Inspired by the well known Sylvester equation ($AX + XB = C$) in linear systems, the Kronecker method converts Eq.(2) into the form of Kronecker products [**?**] as in Eq.(29):

$$AX = XB \Leftrightarrow \begin{pmatrix} \mathbb{I}_9 - R_B \otimes R_A & 0_{9\times3} \\ \mathbf{t}_B^T \otimes \mathbb{I}_3 & \mathbb{I}_3 - R_A \end{pmatrix} \begin{pmatrix} vec(R_X) \\ \mathbf{t}_X \end{pmatrix} = \begin{pmatrix} 0_9 \\ \mathbf{t}_A \end{pmatrix}. \tag{29}$$

Given multiple pairs of As and Bs with noise, the Kroneker product is reformulated as Eq.(30) and Eq.(31):

$$\begin{pmatrix} \mathbb{I}_9 - R_{B_1} \otimes R_{A_1} \\ \mathbb{I}_9 - R_{B_2} \otimes R_{A_2} \\ \vdots \\ I_9 - R_{B_n} \otimes R_{A_n} \end{pmatrix} vec(R_X) = \mathbf{0}_{9n\times1}, \tag{30}$$

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

Orthogonalization is implemented on R_X to make it a rotation matrix.

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

[?] where R_{X_e} denotes the orthogonalized R_X.
The orthogonalized matrix R_{X_e} is further normalized as in Eq.(33), then the rotation matrix $R_X \in SO(3)$ is obtained.

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

where R_{X_n} is the normalized matrix of R_{X_e}. After obtaining the estimation of R_X, least square method is implemented on Eq.(31) to recover the value of $\mathbf{t}_X$.

3.6 Global Polynomial Optimization

To be added or not?

3.7 Convex Optimization

To be added or not?

3.8 Gradient Descent 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 [?] is an online sensor calibration method which uses a gradient descent optimization on the Euclidean group (SE(3)) given an appropriate cost function.

In the first place, 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 [**?**], the concept of directional derivatives in $\mathbb{R}^n$ is extended to functions on a Lie group as:

$$(\hat{\mathcal{X}}^r f)(g) \doteq \frac{d}{dt} f(g \circ \exp(t\mathcal{X}))\Big|_{t=0}. \tag{34}$$

Note that t is just a scalar, while $\mathbf{t}$ represents the translation part of a homogeneous transformation. Eq.(34) is the "right" Lie derivative and the "left" Lie derivative is defined in a similar form. A gradient on $SE(3)$ can be 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:

$$\nabla f(g) = \begin{pmatrix} \frac{d}{dt} f(g \circ \exp(tE_1))|_{t=0} \\ \frac{d}{dt} f(g \circ \exp(tE_2))|_{t=0} \\ \vdots \\ \frac{d}{dt} f(g \circ \exp(tE_6))|_{t=0}. \end{pmatrix} \tag{35}$$

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

$$g_{s+1} = g_s \exp(\Delta t V_g^r) \tag{36}$$

where $t_{s+1} = t_s + \Delta t$ is the discrete time step. To complete the update update law, the rigid body velocity is defined as:

$$V_g^r = g^{-1}\dot{g} = -\alpha \widehat{\nabla f(g)}. \tag{37}$$

The cost function can be chosen as:

$$C(X) = \sum_{i=1}^{n} ||A_i X - X B_i||. \tag{38}$$

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 NUMERICAL SIMULATION AND ERROR COMPARISON

In this section, we numerically simulated the Lie group method, the Kronecker product method, the quaternion method, and the dual quaternion method in Matlab to compare the errors in both R_X and t_x with respect to different noise levels on $\{A_i, B_i\}$.

4.1 Error Metrics and Covariance Matrix

There are multiple ways to define the errors of rigid body transformation. One approach is to measure the errors of R_x and $\mathbf{t}_X$ simultaneously. The other approach is to measure the rotation error and translation error separately.

4.1.1 Metrics for Rotation Error

Various metrics for rotation error have been used in the literature. For one time of simulation, in [?] and [?], the matrix error norm is used as:

$$E_{rot} \doteq \|R_{X_{true}} - R_{X_{calc}}\|, \tag{39}$$

In [?] and [?], the quaternionic error norm is used as:

$$E_{rot} \doteq \|\mathbf{q}_{X_{true}} - \mathbf{q}_{X_{calc}}\|, \tag{40}$$

Another way is to calculate the Frobenius norm of the relative rotation between X_{true} and X_{calc} as:

$$E_{rot} \doteq \|R_{X_{true}}^T R_{X_{calc}}\|_F, \tag{41}$$

We further propose the Lie algebra error for the relative rotation $R_{X_{true}}^T R_{X_{calc}}$ as:

$$E_{rot} \doteq \|\log^\vee(R_{X_{true}}^T R_{X_{calc}})\|_2. \tag{42}$$

By using metrics in Lie algebra, it is possible to avoid any weighting biasing that might appear in metrics involving a norm.

4.1.2 Metrics for Translation Error and Error Covariance Matrix

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

$$E_{tran} \doteq \frac{\|\mathbf{t}_{X_{true}} - \mathbf{t}_{X_{calc}}\|_2}{\|\mathbf{t}_{X_{true}}\|_2}. \tag{43}$$

Furthermore, we compared the accuracy of the 4 methods by calculating their error covariance matrices ${}^e\Sigma$, which is defined as:

$${}^e\Sigma \doteq \sum_{i=1}^{n} K_i K_i^T \in \mathbb{R}^{6\times 6}. \tag{44}$$

where $K_i = \log^\vee(A_i) - Ad(X_{calc})\log^\vee(B_i)$. The error covariance matrix ${}^e\Sigma$ can provide information about the variances of the independent and correlated error on each basis of Lie algebra.

4.2 $\{A_i, B_i\}$ Pairs Generation

There are various ways of generating noisy $\{A_i, B_i\}$ pairs and different data sets can cause certain methods to perform badly or even to crash. To simulate different $AX = XB$ solvers with synchronous noisy $\{A_i, B_i\}$ pairs, we first generate B_i by randomly sampling the Lie algebra of matrix B_i, and then recover B_i using the matrix exponential as shown below:

$$B_i = \exp\begin{pmatrix} \Omega_i & \mathbf{v_i} \\ \mathbf{0}^T & 0 \end{pmatrix}. \tag{45}$$

Next, the noisy A_i is obtained by $A_i = {}^1X_i\, B_i\, {}^2X_i^{-1}$ where ${}^1X_i$ and ${}^2X_i$ are calculated by applying small disturbances on the ground truth X_{true}:

$${}^mX_i = X_{true}\exp\begin{pmatrix} {}^m\Omega_{X_i} & {}^m\mathbf{v}_{X_i} \\ \mathbf{0}^T & 0 \end{pmatrix} \text{ where } m = 1,2 \tag{46}$$

where ω_{X_i} and $\mathbf{v}_{X_i}$ are drawn from independent Gaussian distributions. In order to test the sensitivity to different noise levels for each $AX = XB$ solver, we choose the mean of $\left(\omega_{X_i}^T, \mathbf{v}_{X_i}^T\right)^T$ to be $\mathbf{0}_{6\times 1}$ and its covariance matrix to be $\Sigma_{X_i} = \sigma^2\mathbb{I}_6$ with σ ranging from 0 to 0.1.

4.3 Numerical Simulation Result

The numerical simulation results are presented in Fig.2, Fig.3, and Fig.4. Figure 2 shows that as the noise level σ increases, Lie group method shows the least increase in rotation error, while the dual quaternion method's increase in the rotation error is the largest. The rotation errors of the Kronecker method (light blue line) and the quaternion method (dark blue line) are pretty close and they are even closer in terms of translational error (figure 3).

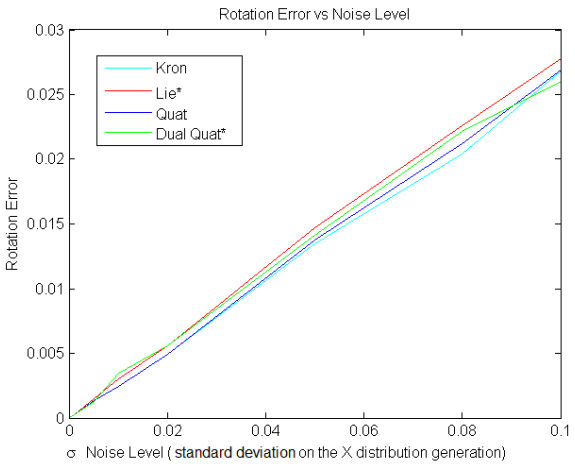

Fig. 2. Rotation Error v.s. Noise Level using Lie algebra Error Metric

Lie* is an improved version of the original Lie group method where ill-conditioned $\{A_i, B_i\}$ pairs are eliminated solving for the estimated X. This is due to the limitation of algorithm that A_i or B_i whose trace is equal or very close to -1 will cause the singularity of numerical computation. Different ways of generating $\{A_i, B_i\}$ can affect the number of filtered pairs and certain ways can render the Lie group method giving R_X with $\det(R_X) = -1$.

Dual Quat* also eliminates certain $\{A_i, B_i\}$ pairs by checking the proximity of the scalar parts of the dual quaternions of A_i and B_i under certain boundary. Similar to the Lie group method, the way of generating $\{A_i, B_i\}$ pairs can affect the number of filtered data, and this will affect the accuracy of the estimated X. In addition, certain ways of generating

the data pairs can cause the dual quaternion method failing to give a valid result from fixed filtered data pairs. This problem can possibly be solved by extracting a smaller portion of the given $\{A_i, B_i\}$ pairs but this also can degrade the accuracy of the result.

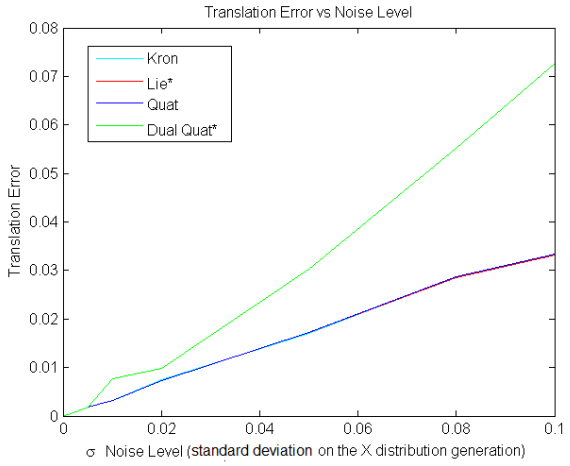

Fig. 3. Translation Error with Randomly Generated AB Pairs (with the improved Lie and Dual Quaternion methods)

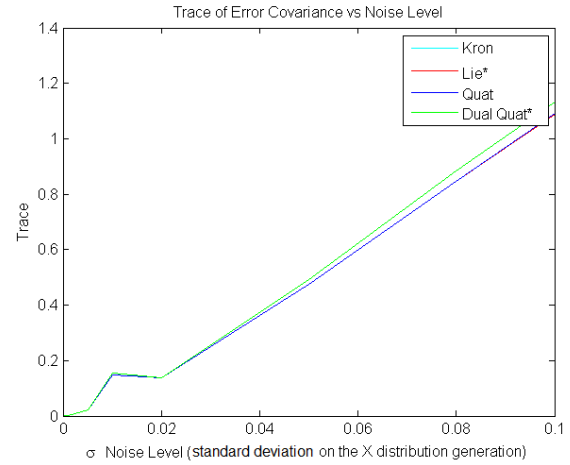

Fig. 4. Trace of the Error Covariance Matrix

In general, all of the traditional methods work well with near random Gaussian distributed noisy $\{A_i, B_i\}$ pairs. However, it should be noted that in many applications the information of $\{A_i, B_i\}$ is obtained by moving the calibration sensor along a trajectory. Depending on the specifics of the trajectory, as well as the sample rate along the trajectory, the data may become ill conditioned for these methods.

A commonality of all the methods in the previous section is that exact knowledge of $\{A_i\}$ and $\{B_j\}$ correspondence 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 [?], to varying effects. The solution methodology presented in this paper bypasses these issues altogether without tracking, or recomputing, correspondence. By modeling 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. Additionally, this new formulation can be expanded to more explicitly model the error in X associated with the noise in As and Bs. The error can then be minimized to further refine X.

In the following section we present a new solution method to the $AX = XB$ formulation that does not require the correspondence to be known between (A, B) pairs.

5 THE BATCH METHOD (THE NOISE-FREE CASE)

This section presents new methods to solve for X wherein there does not need to be any 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 B_j.

5.1 The Batch Method Formulation

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

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

a new algorithm is developed to find $X \in SE(3)$. We address 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 we do not know a priori of this correspondence between each A_i and B_j.

We begin by defining 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 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), we can think of Eq.(47) 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.

Whereas the addition (as opposed to multiplication) of homogeneous transformation matrices is nonsensical, the addition of real-valued functions $f_1(H) + f_2(H)$ is a perfectly reasonable operation and since convolution is a linear operation on functions, we can write all n instances of Eq.(48) into a single equation of the form

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

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 $f(H)$ be defined by the 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 [**?**]:

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

where

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

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) gives two"Batch Method" equations:

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

and

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

For the problem discussed in this paper, there is no loss of generality in assuming that $\|\Sigma_A\|$ and $\|\Sigma_B\|$ are small because the constraint Eq.(55) is linear in Σ_A and Σ_B, and if they are not small, they can both be normalized as $\Sigma'_A = \Sigma_A/(\|\Sigma_A\|)$ and likewise $\Sigma'_B = \Sigma_B/(\|\Sigma_A\|)$. Note that here we have normalized by the same quantity on both sides. We cannot use $\|\Sigma_A\|$ on one side of the equation and $\|\Sigma_B\|$ on the other because the Frobenius norm is not Ad-invariant for $SE(3)$.

Moreover, standard tests from multivariate statistical analysis such as q-q plots can be used to assess whether or not the data are Gaussian. If they are not, they can be made Gaussian without loss of information or by introducing changes to the original mean and covariance in a simple way. With $A_i = XB_iX^{-1}$, it follows that $A_i^p = XB_i^pX^{-1}$ for any power $p \in \mathbb{R}$. This means that each measured data point can be replaced with a continuum of equivalent data points parameterized by p. Practically speaking, we can sample p at fractional powers in the range $p \in [-1, 1]$ and introduce multiple instances of samples with a Gaussian weighting that depends on p. This would cause the resulting augmented data set to behave as a Gaussian. But since the mean and covariance would be unchanged, there is no need to implement this thought experiment – Gaussians can be used in place of data that are not even Gaussian.

Having said all of this, the following theorem indicates that we do not have to limit the discussion to highly concentrated pdfs. Double check whether this has been published in previous papers. If not, remove completely; otherwise, briefly mention the idea and remove the detailed equations.

Theorem 5.1. *Unlike Eq.(52), Eq.(54) and Eq.(55) are exact (rather than approximate) expressions that do not depend on the smallness of* Σ *or any assumptions about the form of the functions* f_A *or* f_B.

Proof. Starting with Eq.(48), performing a convolution on the left of both sides of the equation with $\delta_{X^{-1}}(H)$, and using the associativity of convolution and the properties of delta

functions reviewed in the Appendix, Eq.(48) can be replaced with:

$$(\delta_{X^{-1}} * \delta_{A_i} * \delta_X)(H) = \delta_{B_i}(H),$$

and summing both sides over i and dividing by n gives

$$(\delta_{X^{-1}} * f_A * \delta_X)(H) = f_B(H). \tag{56}$$

Let M_A be the mean of f_A and M_B be the mean of f_B. Again using the properties of the delta function given in the Appendix, and using Eq.(56),

$$\int_{SE(3)} \log(M_B^{-1}H)(\delta_{X^{-1}} * f_A * \delta_X)(H)dH =$$

$$\int_{SE(3)} \log(M_B^{-1}H) f_A(XHX^{-1})dH = \mathbb{O}.$$

Changing variables as $K = XHX^{-1}$ and using the invariance of integration (also given in the Appendix),

$$\int_{SE(3)} \log(M_B^{-1}X^{-1}KX) f_A(K)dK = \mathbb{O}.$$

Multiplying on the left by X and right by X^{-1} and using the fact that $X[\log(M_B^{-1}X^{-1}KX)]X^{-1} = \log(XM_B^{-1}X^{-1}K)$ gives $XM_B^{-1}X^{-1} = M_A^{-1}$, which is the same as Eq.(54). The proof for Eq.(55) follows in a similar way.

The generality of this result, which follows from convolution of one function on either side by a function and a version of itself with inverted argument, motivates much of the theoretical developments that occur later in the paper. But first, we address how to solve Eq.(54) and Eq.(55) for X.

5.2 A Batch Method Solution

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

$$\log^\vee(M_A) = Ad(X)\log^\vee(M_B). \tag{57}$$

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.(57) separately,

$$\mathbf{n}_A = R_X\mathbf{n}_B \quad \text{and} \tag{58}$$

$$\mathbf{v}_A = \theta_B\widehat{\mathbf{t}_X}R_X\mathbf{n}_B + R_X\mathbf{v}_B. \tag{59}$$

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, we can choose

$$R(\mathbf{n}_A,\mathbf{n}_B) = \mathbb{I} + \widehat{\mathbf{n}_B\times\mathbf{n}_A} + \frac{(1-\mathbf{n}_B\cdot\mathbf{n}_A)}{\|\mathbf{n}_B\times\mathbf{n}_A\|^2}\left(\widehat{\mathbf{n}_B\times\mathbf{n}_A}\right)^2. \tag{60}$$

The rotation $R(\mathbf{n}_B,\phi)$ is given by Euler's 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.(59) and rearranging terms, we get:

$$\frac{R(\mathbf{n}_A,\mathbf{n}_B)R(\mathbf{n}_B,\phi)\mathbf{v}_B - \mathbf{v}_A}{\theta_B} = \widehat{\mathbf{n}_A}\mathbf{t}_X. \tag{61}$$

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:

$$\mathbf{t}_X = \mathbf{t}(s) = s\,\mathbf{n}_A + a\,\mathbf{m}_A + b\,\mathbf{m}_A\times\mathbf{n}_A \tag{62}$$

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 we 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.(62) into Eq.(61) 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,

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

[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 $\mathbf{e}_3$.

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

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

$$X(\phi, s) = H(R(\mathbf{n}_A, \mathbf{n}_B) R(\mathbf{n}_B, \phi), \mathbf{t}(s)) \tag{63}$$

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

To provide the additional constraints needed to solve for X, we can use Eq.(55). If we decompose Σ_{M_A} and Σ_{M_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 we can take the first two blocks of Eq.(55) and write:

$$\begin{aligned} \Sigma_{M_B}^1 &= R_X^T \Sigma_{M_A}^1 R_X \text{ and} \\ \Sigma_{M_B}^2 &= R_X^T \Sigma_{M_A}^1 R_X (\widehat{R_X^T t_x}) + R_X^T \Sigma_{M_A}^2 R_X. \end{aligned} \tag{64}$$

We can then find the eigendecomposition, $\Sigma_i = Q_i \Lambda Q_i^T$, 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 and write the first block of Eq.(64) as [**?**, **?**]:

$$\Lambda = Q_{M_B}^T R_X^T Q_{M_A} \Lambda Q_{M_A}^T R_X Q_{M_B} = Q \Lambda Q^T. \tag{65}$$

This is something new that can be served as a complement in the review of batch method. When Q is constrained to be a rotation matrix. Eq.(65) can be further be written as:

$$\Lambda Q = Q \Lambda. \tag{66}$$

This is the same to say that Q commutes with a diagonal matrix Λ, which forces Q to be block conformal to Λ. If we further assume that Λ does not have repeated diagonal entries, then Q will be constrained to be a diagonal matrix in order to be block conformal to Λ. The set of Qs that satisfy this equation will be given as:

$$\begin{aligned} Q = \Bigg\{ &\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\ &\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \Bigg\} \end{aligned} \tag{67}$$

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

$$R_x = Q_{M_A} Q Q_{M_B}^T. \tag{68}$$

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 Q would be replaced by $SO(3)$, which is the same as saying that the first equation in Eq.(64) 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 we limit the discussion to the case where Eq.(67) holds.

Once the four possibilities of R_X are found in this manner, the corresponding possible $\mathbf{t}_X$ can be found easily from the block 2 and 4 of in Eq.(64)(b).

The correct solution, from the set of 4 possibilities (given Eq.(68)), can be found by applying the cylinder constraints as in Eq.(58) and Eq.(59). We do this by choosing the possibility 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 our case it is chosen to be unity.

6 CONCLUSION AND FUTURE WORK

In this paper we rexamined the "AX=XB" formulation of the sensor calibration problem used in camera calibration, Cartesian robot hand calibration, robot eye-to-hand calibration, aerial vehicle sensor calibration and image guided therapy (IGT) sensor calibration. A summary of some of the most influential and effective previous methods was presented and their positive and negative traits were discussed. We focused 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 in different contexts and problems and it is important to use the appropriate method for the circumstance.

In addition to measurement error contributing to noise, we emphasized the fact that the sensor data streams containing the As and Bs may present at different sample rates, be asynchronous, and each stream may contain gaps in information. We therefore presented a method for calculating the calibration transformation that works for data without any a priori knowledge of the correspondence between the As and Bs. The new method formulates the sensor data as probability distributions and gives the mean solution, X_{calc}. Additionally, we show how the covariance of X_{calc} can be determined which could give insight into the magnitude and type of noise on the sensor measurements.

In future works, we would like to continue to examine, in more detail, the usefulness of Σ_X. We will further explore the relationship between the X covariance and the error on the calibration in experimental results, hypothetically using Σ_X to identify ideal sensor placements in the environment and path plan along minimal error regions. Additionally, we hope to harness the knowledge of Σ_X to formulate a cost function for minimizing the error on our initial solution.

Acknowledgements

The authors would like to recognize NSF Grant RI-Medium: IIS-1162095, which served to support this work.

We would also like to acknowledge the support of Alexis Cheng and Emad M. Boctor, colleagues in this endeavor at Johns Hopkins University.

Appendix A: Integration and Convolution on SE(3)

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

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 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 $J(\mathbf{q}) = \sin\beta$. And while D and $J(\mathbf{q})$ will change depending on what parameterization is used, the value of the integral itself does not, as it is a property of the Lie group and the function, and not how the function is expressed and the integral is computed in coordinates.

$SE(3)$ 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]" $f : SE(3) \to \mathbb{R}$, [?]

$$\int_{SE(3)} f(H_0 \circ H)dH = \int_{SE(3)} f(H \circ H_0)dH = \int_{SE(3)} f(H)dH. \tag{69}$$

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

$$\int_{SE(3)} f(H^{-1})dH = \int_{SE(3)} f(H)dH. \tag{70}$$

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 going to 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 than those that have bi-invariant metrics, 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

$$(f_1 * f_2)(H) \doteq \int_{SE(3)} f_1(K) f_2(K^{-1}H)dK. \tag{71}$$

This integral can be rewitten in a number of equivalent forms using Eq.(69) and Eq.(70). 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). In analogy with the way it inherit associativity, convolution also inherits noncommutativity for general functions, with the exception of special functions called "class functions".

If we expand the class of functions that are considered beyond $(L^1 \cap L^2)(SE(3)$ to include Dirac delta functions[4], which are the unique functions such that for every $f \in (L^1 \cap L^2)(SE(3)$

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

then the result is the "group algebra" consisting of functions as elements, and the two operations of convolution and addition of functions: $f_1 * f_2$ and $f_1 + f_2$. A slightly further expansion of allowable functions to include shifted delta functions of the form:

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

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

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

$$\begin{aligned}(\delta_X * f)(H) &= \int_{SE(3)} \delta(X^{-1}K) f(K^{-1}H)dK \\ &= \int_{SE(3)} \delta(J) f((XJ)^{-1}H)dK = f(X^{-1}H)\end{aligned} \tag{72}$$

[3] Here we mean 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, we restrict the discussion to $f \in (L^1 \cap L^2)(SE(3))$.

[4] As in $\mathbb{R}^n$, these can be thought of as spikes of infinite height and infinitesimal width centered on the identity

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

$$(\delta_X * f * \delta_{X^{-1}})(H) = f(X^{-1}HX).$$

Using the associative property,

$$\begin{aligned}
&(\delta_X * f_1 * \delta_{X^{-1}}) * (\delta_X * f_2 * \delta_{X^{-1}}) \\
&= \delta_X * f_1 * (\delta_{X^{-1}} * \delta_X) * f_2 * \delta_{X^{-1}} \\
&= \delta_X * (f_1 * f_2) * \delta_{X^{-1}}.
\end{aligned} \tag{73}$$