IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 11, 2017, accepted January 20, 2018, date of publication January 29, 2018, date of current version March 13, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2799173

Automatic Calculation of a Transformation Matrix Between Two Frames

JASMINE CASHBAUGH[1], (Member, IEEE), AND CHRISTOPHER KITTS[2], (Senior Member, IEEE)

[1]JLLJ Technologies, LLC, West End, NC 27376, USA
[2]Department of Mechanical Engineering, Santa Clara University, Santa Clara, CA 95053, USA

Corresponding author: Jasmine Cashbaugh (jasmine@jllj-tech.com)

ABSTRACT This paper presents a novel strategy for the automatic calculation of a homogeneous transformation matrix between two frames given a set of matched position measurements of objects as observed in both frames. The transformation matrix is calculated by applying a linear regression to matched data in two coordinate frames. This method scales linearly with the number of data points, enabling it to be used in a wide variety of applications without requiring large amounts of processing time. The automatic transformation matrix calculation is demonstrated to work in a real-world application and produce deterministic results. The error on the transformation matrix calculation was found to be low, making this an ideal method for the easy calculation of transformation matrices.

INDEX TERMS Automatic control, kinematics, least squares approximation, linear algebra, mathematical programming, mobile robots.

I. INTRODUCTION

Homogeneous transforms enable position coordinates to be converted between two coordinate frames while preserving the vector lengths. They are widely used in applications such as navigation, robotic vision, rigid body motion, augmented reality, telerobotics, and product assembly lines. They are used wherever data in one coordinate frame needs to be represented in another coordinate frame. This is especially useful if a system coordinate frame is difficult to visualize. A coordinate transformation can translate the hard-to-visualize system position information into a frame that is more easily accessible by the user. The research presented in this article examines this concept by utilizing simple mathematical tools to automatically calculate a transformation matrix between matched coordinate frame data.

Despite their utility, transformation matrices are not always easy to manually calculate. In order to avoid these calculations, cameras and image software are sometimes used to find the coordinate transformation automatically with minimal user input [1]. However, image software, such as OpenCV, often requires special setup and calibration procedures that need to be repeated when the camera is moved or lighting conditions change [2]. In addition, OpenCV determines the transformation matirx using a Levenberg-Marquardt optimization process. This implementation has a small window of convergence and fails to converge when the initial estimate of the camera properties are too poor. Although this is a known issue, it had not yet been corrected at the time this article was written [3]. Further, image-based methods assume the use of cameras or, at least, access to cameras and familiarity with vision processing techniques which may not always be available. Even when cameras are available and users are familiar with vision processing, it is not always possible to place cameras in every location necessary to obtain the desired transformation matrices.

Some publications have even proposed ways of avoiding coordinate transformations altogether and instead apply rotational operators singly to compute rotations rather than compiling the rotational operators into a single transform matrix. These rotational operators are determined based on the difference between the single-axis rotation angles of each frame from a common stationary reference frame [4]. Others have dealt purely with qualitative models [5]. Both of these methods require a deep understanding of the coordinate frames involved. However, this is not always a valid assumption since not all end users are knowledgeable about coordinate transformations.

In response to these issues, the method presented in this article deterministically calculates a homogeneous transform between two frames given a set of matched pairs of

position measurements from the point of view of each of the reference frames. This method represents an improvement over the state of the art because it makes no assumptions about the source of either set of data points or the knowledge of the user. Instead, it applies a simple linear regression to an augmented transformation matrix in order to obtain the deterministic best-fit between the two data sets. If desired, the rotation angles can then be extracted from this transformation matrix using trigonometry and knowledge of the Euler angle sequence used. This step is optional and depends upon the needs of the user. The transformation calculation can also be integrated with the data collection system, producing an accurate transformation matrix with no input from the user. This greatly improves the method's accessibility since it removes the constraint of a knowledgeable end user.

In this article, a review of the methods presented in the literature is given in Section II while a description of the mathematical formulation of the transformation matrix calculations are provided in Section III. Next, the automatic transformation matrix calculation method is verified using experimental data as presented in Section IV and a method comparison is provided in Section V. Finally, Section VI provides the conclusions and future work.

II. LITERATURE REVIEW

The literature provides multiple methods to automatically calculate a transformation matrix. For example, Li *et al.* [6] simultaneously calculated the spatial transformation matrix and super resolution matrices for a hyperspectral image super resolution application. The first step was to obtain an initial input for both the transformation matrix and super resolution matrices. Next, these matrices were calculated using an iterative optimization approach that alternately optimized for the spatial transformation matrix and the super resolution matrices until convergence was reached. Although this method results in a transformation matrix, it is important to note that only the rotation transformation was considered in this method.

In [7], Umeyama calculated the rotation, translation, and scaling between two sets of data points using singular value decomposition and least-squares estimation. In Umeyama's approach, rotation, translation, and scaling were all computed separately. Solutions making use of reflection were rejected, substituting instead the closest transformation matrix solution. The author stated that this method always gives the correct answer, even when the data is corrupted, asserting that a reflection can never be the correct answer.

Arun *et al.* also calculated the translation and rotation between two coordinate frames using singular value decomposition and least-squares estimation [8]. Their method used the determinate of the calculated matrix to determine the method's success. A determinate of $+1$ indicated that the calculated matrix was a rotation matrix and the method was successful. However, a determinate of -1 indicated that the calculated matrix was a reflection and the method had failed.

Another approach, used in [9], made use of an eigenvalue-eigenvector decomposition and least-squares estimation. As in the previous methods, the authors calculated the rotation, translation, and scaling between two reference frames separately. This approach ensured the calculation of orthonormal transformation matrices in order to guarantee that vector lengths and angles between vectors were preserved in the frame transformation.

Chen *et al.* [10] used a noise-tolerant algorithm to automatically calculate the transformation matrix between the frame of a laser sensor and the positioning device frame. This approach required the use of an elliptical target and calculated the rotation and translation components separately. First, the target's chord was measured at least 100 times before applying ellipse fitting to the chord data. The plane of the ellipse was then calculated and used to find the rotation matrix using singular value decomposition. The translation component was found next using the least squares method. The accuracy of this method was found to improve as more chords were measured.

Gomez *et al.* [11] compared a least-squares approach and an orthogonal approach to see which method yielded a better result for a radiosurgery application. In both approaches, the rotation and translation between the two frames were calculated separately. The least-squares approach made use of a QR-factorization while the orthogonal method made use of a vector approach using three points. For this application, the orthogonal method was found to yield more accurate results based on a comparison of the resulting positioning errors.

III. MATHEMATICAL FORMULATION

The method presented in this article deterministically calculates a homogeneous transform between two frames given a set of matched pairs of position measurements from the point of view of each frame. This homogenous transform is given in the form of an augmented transformation matrix, as shown in (1), which is used to describe the rotation and translation between two coordinate frames. This transformation matrix contains the rotations between frames in the top left 3 by 3 matrix and the translation between the two coordinate frames in the first three rows of the fourth column. The fourth row of the augmented transformation matrix is a scaling and perspective factor [12]. In the work presented here, the perspective is assumed to be orthonormal and the scaling is assumed to be 1:1 since the transform is homogeneous.

$$T_A^B = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The method presented in this article determines the augmented transformation matrix by applying a linear regression to matched and synchronized data points in two coordinate frames. The details of this method will be discussed in the following subsections.

A. LINEAR REGRESSION

The first step in performing a linear regression is to write the transformation matrix as a series of linear equations. This can be done by visualizing the transformation shown in (2).

$$\begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} \tag{2}$$

This corresponds to the set of linear equations defined in (3).

$$\begin{aligned} cx_B &= r_{xx}x_A + r_{xy}y_A + r_{xz}z_A + t_x \\ y_B &= r_{yx}x_A + r_{yy}y_A + r_{yz}z_A + t_y \\ z_B &= r_{zx}x_A + r_{zy}y_A + r_{zz}z_A + t_z \end{aligned} \tag{3}$$

The goal of linear regression is to minimize the square of the residuals. For a linear equation, the square of the residual is defined as:

$$R^2(m, b) = \sum_{i=1}^{n} [y_i - (mx_i + b)]^2 \tag{4}$$

Defining the square of residuals for the system of equations in (3) results in the following series of equations where n is the number of paired samples:

$$\begin{aligned} R^2_{xB} &= \sum\nolimits_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)]^2 \\ R^2_{yB} &= \sum\nolimits_{i=1}^{n} [y_{Bi} - (r_{yx}x_{Ai} + r_{yy}y_{Ai} + r_{yz}z_{Ai} + t_y)]^2 \\ R^2_{zB} &= \sum\nolimits_{i=1}^{n} [z_{Bi} - (r_{zx}x_{Ai} + r_{zy}y_{Ai} + r_{zz}z_{Ai} + t_z)]^2 \end{aligned} \tag{5}$$

The minimum value occurs when the derivative of the square of the residual is equal to zero. Thus, the next step is to take the partial derivatives of the first equation in (5) and set the results equal to zero, as shown in (6) through (9).

$$\begin{aligned} \frac{\partial R^2_{xB}}{\partial r_{xx}} &= 0 \\ &= -2\sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)]x_{Ai} \end{aligned} \tag{6}$$

$$\begin{aligned} \frac{\partial R^2_{xB}}{\partial r_{xy}} &= 0 \\ &= -2\sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)]y_{Ai} \end{aligned} \tag{7}$$

$$\begin{aligned} \frac{\partial R^2_{xB}}{\partial r_{xz}} &= 0 \\ &= -2\sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)]z_{Ai} \end{aligned} \tag{8}$$

$$\begin{aligned} \frac{\partial R^2_{xB}}{\partial t_x} &= 0 \\ &= -2\sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)] \end{aligned} \tag{9}$$

Putting these results into matrix form achieves the equation described in (10). This results in the values for the first row of the transformation matrix defined in (1).

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \\ t_x \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_{Bi}x_{Ai} \\ \sum_{i=1}^{n} x_{Bi}y_{Ai} \\ \sum_{i=1}^{n} x_{Bi}z_{Ai} \\ \sum_{i=1}^{n} x_{Bi} \end{bmatrix} \tag{10}$$

Here, A is defined in (11) where all sums are from $i = 1$ to n:

$$\begin{bmatrix} \sum(x_{Ai}^2) & \sum(x_{Ai}y_{Ai}) & \sum(x_{Ai}z_{Ai}) & \sum(x_{Ai}) \\ \sum(x_{Ai}y_{Ai}) & \sum(y_{Ai}^2) & \sum(y_{Ai}z_{Ai}) & \sum(y_{Ai}) \\ \sum(x_{Ai}z_{Ai}) & \sum(y_{Ai}z_{Ai}) & \sum(z_{Ai}^2) & \sum(z_{Ai}) \\ \sum(x_{Ai}) & \sum(y_{Ai}) & \sum(z_{Ai}) & n \end{bmatrix} \tag{11}$$

This process is then repeated for the second and third rows of the matrix defined in (1), resulting in (12) and (13), respectively, where A is defined by (11).

$$\begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \\ t_y \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_{Bi}x_{Ai} \\ \sum_{i=1}^{n} y_{Bi}y_{Ai} \\ \sum_{i=1}^{n} y_{Bi}z_{Ai} \\ \sum_{i=1}^{n} y_{Bi} \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \\ t_z \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum_{i=1}^{n} z_{Bi}x_{Ai} \\ \sum_{i=1}^{n} z_{Bi}y_{Ai} \\ \sum_{i=1}^{n} z_{Bi}z_{Ai} \\ \sum_{i=1}^{n} z_{Bi} \end{bmatrix} \tag{13}$$

B. REQUIRED DATA

The data requirements for the presented methodology are fairly simple; the only requirement is the use of matched pairs of data in two reference frames. No assumptions are made about the method of data collection, so any type of sensors may be used to obtain the data sets. However, these data sets must be paired in terms of pose in order to establish a relationship between the two frames.

C. EULER ANGLES

The rotation portion of the matrix has the form defined in (14) where the values of $m_{i,j}$ are defined by a variety of rotation sequences.

$$R^B_A = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} \tag{14}$$

These rotation sequences are well understood and involve the rotation of the coordinate frame about three distinct axes. The sequence that will be examined in this article is the 3-2-1 sequence, which is the NASA standard for airplane applications [13]. In this case, the rigid body is rotated first about its z-axis, then about its new y-axis (y'), and then about its new x-axis (x''). The transformation

matrix coefficient definitions for the 3-2-1 case can be seen in (15).

$$\begin{aligned} m_{1,1} &= cos(\psi)cos(\theta) \\ m_{1,2} &= sin(\psi)cos(\theta) \\ m_{1,3} &= -sin(\theta) \\ m_{2,1} &= -sin(\psi)cos(\phi) + cos(\psi)sin(\theta)sin(\phi) \\ m_{2,2} &= cos(\psi)cos(\phi) + sin(\psi)sin(\theta)sin(\phi) \\ m_{2,3} &= cos(\theta)sin(\phi) \\ m_{3,1} &= sin(\psi)sin(\phi) + cos(\psi)sin(\theta)cos(\phi) \\ m_{3,2} &= -cos(\psi)sin(\phi) + sin(\psi)sin(\theta)cos(\phi) \\ m_{3,3} &= cos(\theta)cos(\phi) \end{aligned} \tag{15}$$

The Euler angles for the 3-2-1 sequence can be individually calculated using (16) through (18).

$$\phi = atan2(m_{2,3}, m_{3,3}) \tag{16}$$

$$\psi = atan2(m_{1,2}, m_{1,1}) \tag{17}$$

$$\theta = atan2\left(-m_{1,3}, \frac{m_{1,2}}{sin(\psi)}\right) \tag{18}$$

D. WORKED EXAMPLE

To illustrate the application of this method, an example of paired data is shown in Table 1. This data was generated in Matlab 2011a. $Frame_A$ was generated using *rand()*, a random number generator [14], to produce six data points in (x, y, z). $Frame_B$ was generated by rotating $Frame_A$ with a known transformation matrix using the formula $T_A^B * Frame_A$. In this case, the known transformation matrix is given in (19).

$$T_A^B = \begin{bmatrix} 0.3830 & 0.3214 & -0.8660 & 3.0 \\ -0.4492 & 0.8840 & 0.1294 & 7.0 \\ 0.8072 & 0.3394 & 0.4830 & 1.0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

TABLE 1. Sample paired data in two coordinate frames.

	$Frame_A$	$Frame_B$
Data Point 1	(0.5449, 0.1955, 0.9227)	(2.5144, 7.0691, 1.9754)
Data Point 2	(0.6862, 0.7202, 0.8004)	(2.8292, 7.4454, 2.2224)
Data Point 3	(0.8936, 0.7218, 0.2859)	(3.3518, 7.3060, 2.1198)
Data Point 4	(0.0548, 0.8778, 0.5437)	(2.8392, 7.8455, 1.6229)
Data Point 5	(0.3037, 0.5824, 0.9848)	(2.4901, 7.5449, 1.9518)
Data Point 6	(0.0462, 0.0707, 0.7157)	(2.4273, 7.1354, 1.4349)

To illustrate this process, the sample data provided in Table 1 was used to calculate a transformation matrix. First, this data was substituted into (10), (12), and (13) in order to obtain:

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \\ t_x \end{bmatrix} = [A]^{-1} \begin{bmatrix} 7.3307 \\ 9.0623 \\ 11.2758 \\ 16.4519 \end{bmatrix} \tag{20}$$

$$\begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \\ t_y \end{bmatrix} = [A]^{-1} \begin{bmatrix} 18.5407 \\ 23.8025 \\ 31.3733 \\ 44.3464 \end{bmatrix} \tag{21}$$

$$\begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \\ t_z \end{bmatrix} = [A]^{-1} \begin{bmatrix} 5.2436 \\ 6.1794 \\ 8.0390 \\ 11.3271 \end{bmatrix} \tag{22}$$

A was found by substituting the data into (11). The results of this calculation are shown in (23).

$$A = \begin{bmatrix} 1.6638 & 1.4739 & 1.6695 & 2.5294 \\ 1.4739 & 2.1925 & 2.0645 & 3.1683 \\ 1.6695 & 2.0645 & 3.3514 & 4.2532 \\ 2.5294 & 3.1683 & 4.2532 & 6.0000 \end{bmatrix} \tag{23}$$

Using the results of (20) through (23) as described in Subsection III-A, the following transformation matrix was obtained:

$$T_A^B = \begin{bmatrix} 0.4165 & 0.3242 & -0.8206 & 2.9769 \\ -0.4389 & 0.9085 & 0.1435 & 6.9946 \\ 0.8103 & 0.3442 & 0.5115 & 1.0019 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{24}$$

This is very close to the actual transformation given in (19). Further details on evaluating the accuracy of the transformation matrix will be discussed in Subsection III-E.

The next step in the analysis is to calculate the Euler angles. As this example used a 3-2-1 rotation sequence, the method described in Subsection III-C was applied. Equations (16) through (18) were used in order to obtain the following values in radians:

$$\begin{aligned} \theta_{calculated} &= 1.00 \\ \phi_{calculated} &= 0.27 \\ \psi_{calculated} &= 0.66 \end{aligned} \tag{25}$$

For comparison, the ideal angle values used in the known transformation matrix, shown in (19), are given in (26). These values are also provided in radians.

$$\begin{aligned} \theta_{ideal} &= 1.05 \\ \phi_{ideal} &= 0.26 \\ \psi_{ideal} &= 0.70 \end{aligned} \tag{26}$$

The calculated values for these angles of rotation are very close to the actual angle measurements. This greatly adds to the versatility of the method presented here as rotation angles can be difficult to measure on a physical system, necessitating their calculation. These angles are natively calculated in the method discussed in this article.

E. ERROR ANALYSIS

The accuracy of the method presented in this article was evaluated by first calculating the difference between the actual position in $Frame_B$, p_B, and the calculated position in $Frame_B$, $T_A^B p_A$, for each position pair.

Here, $p_i = [x_i, y_i, z_i, 1]^T$. The mathematical formulation of this can be seen in (27) and (28).

$$\vec{e} = \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} - \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} \quad (27)$$

$$e_i = \sqrt{e_{i,1}^2 + e_{i,2}^2 + e_{i,3}^2} \quad (28)$$

The mean error was then found using the following formula from [15]:

$$\mu_e = \frac{1}{n} \sum_{i=1}^{n} e_i \quad (29)$$

Finally, the standard deviation was found using (30) from [16].

$$e = \sqrt{\frac{1}{n} \sum_{i=1}^{n} ((e_i - \mu_e)^2)} \quad (30)$$

The goal of this analysis was to characterize the error of the transformation matrix. A successful transformation matrix has a mean error similar to the error on the measurement system and a low standard deviation.

F. CORNER CASES

The method presented in this paper requires an invertible A matrix, as defined in (11). However, A is not invertible when all data points lie in the same plane. If A is not invertible, (10), (12), and (13) cannot be solved [17]. In this situation, the pseudoinverse is used instead of an exact inverse of the A matrix. This pseudoinverse is found using the Moore-Penrose method which makes uses of singular value decomposition to find an approximate inverse [18]. This allows (10), (12), and (13) to be solved and an accurate transformation matrix to be found for cases where only planar data is available. However, the authors would like to note that a pseudoinverse is an approximation and three dimensional data should be used where possible in order to obtain the most accurate results.

IV. EXPERIMENTAL EXAMPLE

The effectiveness of the automatic transformation matrix calculations presented in this article was verified using experimental data obtained from a multi-robot experiment. In this experiment, an aerial robot flew through an indoor test area while a land robot remained stationary at a position of (1.28, −4.78, 0.38) m in the global coordinate frame, $\{G\}$. The global coordinate frame is a standard right-hand ground reference frame. The coordinate frame of the stationary land robot, $\{R\}$, was aligned with the global coordinate frame; that is, it was translated but not rotated. This relationship is visualized in Fig. 1. The ideal transformation matrix for this system is shown in (31) below.

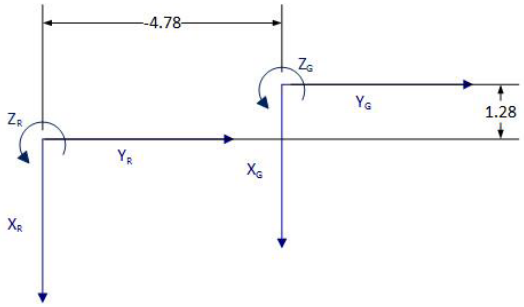

FIGURE 1. An overhead view of the global and robot frames used in the physical experiment.

$$T_G^R = \begin{bmatrix} 1 & 0 & 0 & -1.28 \\ 0 & 1 & 0 & 4.78 \\ 0 & 0 & 1 & -0.38 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (31)$$

The position of each robot was measured using an Ultra-Wide Band system with a position error of $(x, y, z) + (\pm 0.05, \pm 0.07, \pm 0.39)$ m. For reference, the experimental setup can be seen in Fig. 2. In addition to the global coordinate frame, the position of the aerial robot was also tracked with respect to the stationary land robot using a single camera vision system which used calibrated pixel data to determine the distance between the two robots. Thus, the camera positioning system had a position error of $(x, y, z) + (\pm 0.04, \pm 0.40, \pm 0.40)$ m. This yielded paired position coordinates of the aerial robot in two reference frames: the Ultra-Wide Band frame ($\{G\}$) and the frame of the stationary robot ($\{R\}$).

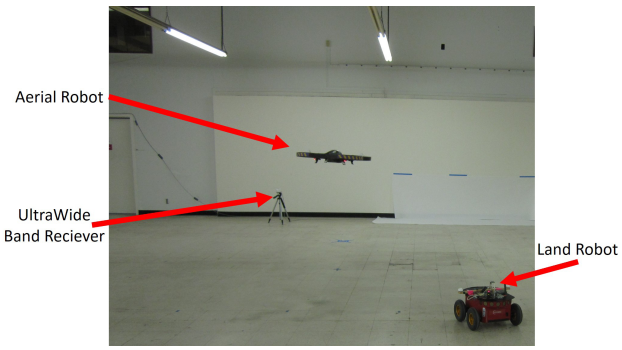

FIGURE 2. The experimental setup used to verify the automatic transformation calculation.

The aerial robot was flown for 62.25 seconds, collecting 499 paired position points for the aerial robot. Upon conclusion of the flight, the paired position coordinates were used in the analysis described by (10) through (13) in order to obtain the transformation matrix shown in (32).

$$T_G^R = \begin{bmatrix} 1.00 & 0.04 & 0.13 & -1.10 \\ 0.02 & 1.00 & -0.09 & 4.89 \\ 0.01 & 0.00 & 0.93 & -0.36 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (32)$$

This transformation matrix is from the global coordinate frame to the frame of the stationary land robot and is very close to the ideal transformation matrix given in (31). The rotation portion of (32) is nearly an identity matrix, as in the ideal transformation matrix, and the translation column shows an error of $(x, y, z) + (0.18, 0.11, 0.02)$ m compared to the ideal matrix. Such an error is well within the measurement errors of the UltraWide Band and camera systems. This error is also much smaller than the application's allowable maximum error of ± 1.0 m, half the width of the sensor's field of view at the average distance between the sensor and the robot. This allows the robot to remain in the sensor's field of view at all times.

The experimental data, as well as its comparison with the calculated data, can be seen in Fig. 3 and Fig. 4.

Using a 3-2-1 sequence, the angles of rotation for this transformation matrix were calculated. The results are shown in radians in (33).

$$\begin{aligned} \theta &= -0.12 \\ \phi &= -0.10 \\ \psi &= 0.04 \end{aligned} \tag{33}$$

In the ideal transformation matrix, each of these angles are equal to 0 radians. However, the angles were calculated from the measured position data which exhibited errors of between 0.04 and 0.40 m. Thus, small variations are expected in the calculation of the angles of rotation.

Using (27) through (30), this transformation was found to have a mean error of 0.10 m and a standard deviation

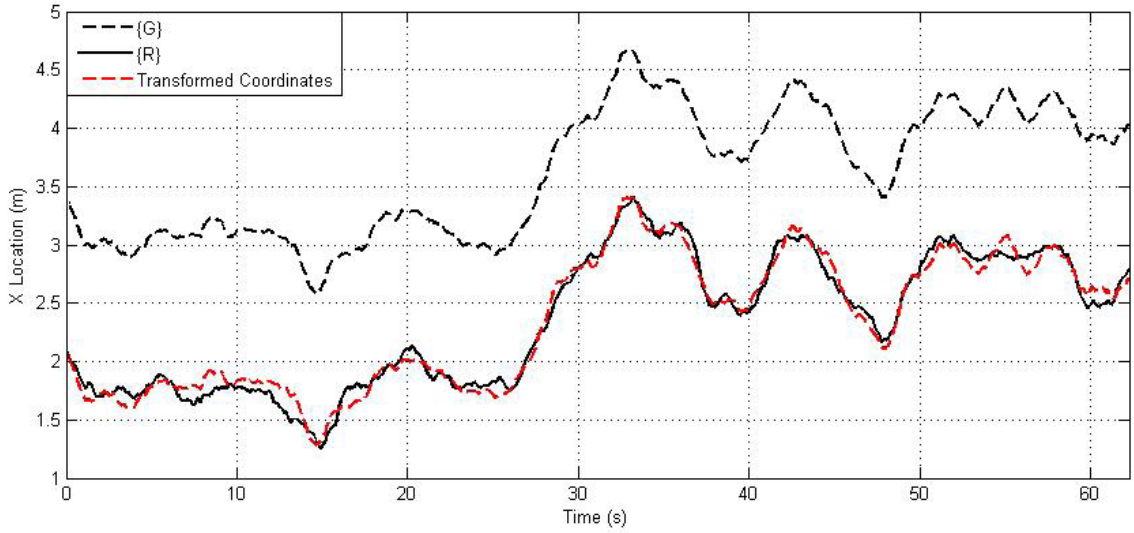

FIGURE 3. The x position of the aerial robot in $\{G\}$, $\{R\}$, and the transformed coordinates.

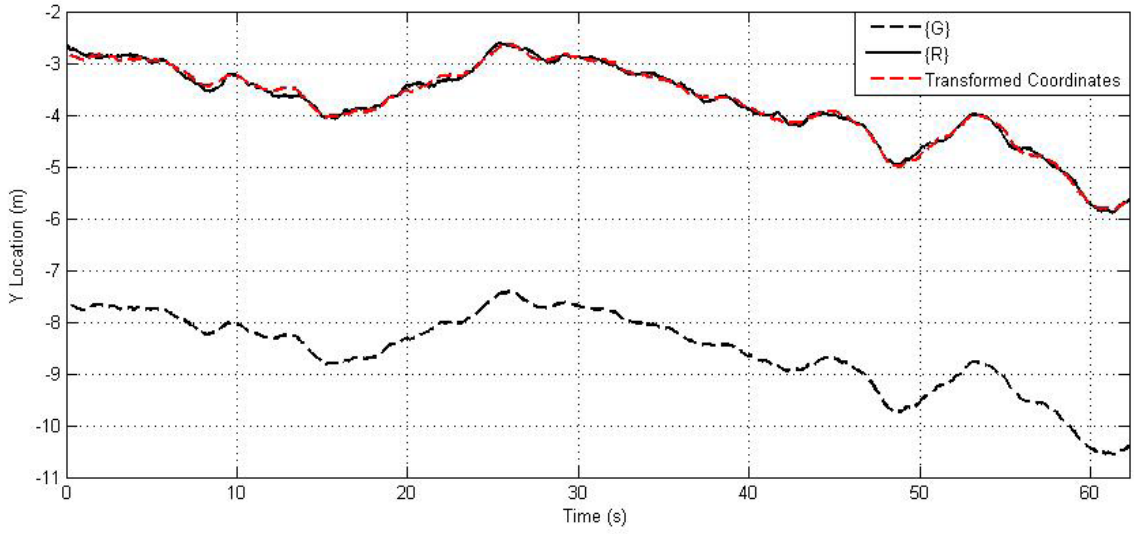

FIGURE 4. The y position of the aerial robot in $\{G\}$, $\{R\}$, and the transformed coordinates.

of 0.05 m. This error is on par with the UltraWide Band system error of $(x, y, z)+(\pm 0.05, \pm 0.07, \pm 0.39)$ m and with the camera system error of $(x, y, z) + (\pm 0.04, \pm 0.40, \pm 0.40)$ m. Since the standard deviation was also low, this transformation matrix calculation was found to be successful using the same evaluation method as performed in [11].

Further, this calculation used 499 pairs of data points and ran in $5e^{-4}$ seconds. The transformation matrix was calculated in Matlab 2011a on a computer with 4.00 GB of RAM and a 2.10 GHz processor running Windows 7. This computation time scales linearly with the number of data points, $O(n)$.

V. METHOD COMPARISON

The method discussed in this article makes no assumptions about the types of sensors used to collect the required paired position data, giving it an advantage over methods that require specific equipment such as image-based approaches. The presented method also makes no assumptions about the shape of the target used to determine the transformation matrix, giving it an advantage over methods that do assume a target shape, as in [10]. The elimination of both of these assumptions gives the user more flexibility in their implementation. In addition, the approach presented in this article can easily be integrated with the data collection software to directly record the paired data. This eliminates the need for any user input to create the transformation matrix. If no user input is required, the system may be used accurately even when the user is not familiar with transformation matrices, greatly increasing the number of people who are able to use the system.

The method presented in this paper is also more robust to measurement errors than that found in [11]. For the method found by Gomez *et al.* to yield the lowest errors, only three paired locations were used. Measurement errors have a much larger impact on a small number of measurements than on a larger data set and the method presented in this paper minimizes this risk by using a larger selection of paired data.

Another important distinction of the approach presented in this article is its generality. The methods presented in [7] and [8] both assume that a reflection is never the right answer and find a best-fit rotation matrix instead, even when this results in a larger error. The method presented in this article is a more general approach that allows the user to find the best transformation between two frames regardless of what form this transformation takes. This is especially relevant for vision applications as cameras often use left-handed coordinate systems.

Furthermore, one of the main differences between the approach described in this article and those discussed in the literature review section is that the approach discussed in this article calculates the rotation and translation simultaneously without user input. In the approaches discussed in Section II, [6] does not calculate the translation at all while the remaining methods calculate the rotation and translation separately [7]–[11]. Separating out these calculations can add to the necessary computing time as well as create an additional source of error if the results of one calculation are used to obtain the results of the second calculation.

Thus, only a comparison of the rotation calculations will be performed between the method presented in this article and the method performed by Horn, *et al.* The rotation calculation was chosen for comparison because this was the first calculation performed in [9]. This method was chosen for comparison because it was found to be the most similar to the method presented in this article. The same experimental data used in Section IV will be used in this comparison.

The method described by [9] used the following two equations to find the rotation matrix:

$$M = \begin{bmatrix} \sum_{i=1}^{n} x_A x_B & \sum_{i=1}^{n} x_A y_B & \sum_{i=1}^{n} x_A z_B \\ \sum_{i=1}^{n} y_A x_B & \sum_{i=1}^{n} y_A y_B & -\sum_{i=1}^{n} y_A z_B \\ \sum_{i=1}^{n} z_A x_B & \sum_{i=1}^{n} z_A y_B & \sum_{i=1}^{n} z_A z_B \end{bmatrix} \quad (34)$$

$$R = M(M^T M)^{-\frac{1}{2}} \quad (35)$$

Applying (34) and (35) to the experimental data yielded the following rotation matrix:

$$R = \begin{bmatrix} 0.99 & -0.15 & -0.02 \\ 0.15 & 0.99 & 0.06 \\ 0.01 & -0.06 & 1.00 \end{bmatrix} \quad (36)$$

Compared to the rotation portion of the ideal matrix in (31), this rotation matrix has a higher error than that obtained from the rotation portion of the matrix in (32). The rotation matrix in (36) has a larger error in five out of the nine values and exhibits an identical error in one value compared to the rotation matrix found in (32). Thus, the method presented in this article was found to yield a more accurate transformation matrix as it was closer to the ideal transformation matrix.

Although the angles of rotation were never directly calculated in [9], the angles will be calculated here for comparison purposes using (16) through (18). The results of these calculation are given in radians in (37).

$$\begin{aligned} \theta_{comparison} &= 0.02 \\ \phi_{comparison} &= 0.06 \\ \psi_{comparison} &= -0.15 \end{aligned} \quad (37)$$

These angles are about the same distance from the ideal values as those provided in (33) using the method discussed in this article. However, the method presented in [9] does not explicitly calculate the Euler angles, an advantage provided by the method presented in this article.

Additional testing was also performed using simulation data where there was no translation between reference frames in order to further examine the relative accuracy of these two methods. Exclusion of a translation component between the two frames allowed for the calculation of the error produced by each transformation matrix, resulting in a more accurate comparison. This data was generated using a similar method to the worked example. $Frame_A$ was generated in Matlab 2011a using *rand()* to produce 200 data points in (x, y, z). $Frame_B$ was generated with the formula

TABLE 2. Summary of results for four rotation-only test cases using three dimensional data as inputs.

	Test 1	Test 2	Test 3	Test 4
Actual Rotation	$\begin{bmatrix} 0.00 & 0.71 & -0.71 \\ -0.50 & 0.61 & 0.61 \\ 0.87 & 0.35 & 0.35 \end{bmatrix}$	$\begin{bmatrix} 0.20 & 0.24 & -0.95 \\ -0.58 & 0.81 & 0.08 \\ 0.79 & 0.54 & 0.30 \end{bmatrix}$	$\begin{bmatrix} 0.98 & 0.09 & -0.17 \\ 0.02 & 0.85 & 0.52 \\ 0.19 & -0.52 & 0.84 \end{bmatrix}$	$\begin{bmatrix} 0.50 & 0 & -0.87 \\ 0.15 & 0.98 & 0.09 \\ 0.85 & -0.17 & 0.49 \end{bmatrix}$
Actual Angles	$\theta = 0.79$ $\phi = 1.05$ $\psi = 1.57$	$\theta = 1.26$ $\phi = 0.26$ $\psi = 0.87$	$\theta = 0.17$ $\phi = 0.56$ $\psi = 0.09$	$\theta = 1.05$ $\phi = 0.17$ $\psi = 0$
Proposed Rotation	$\begin{bmatrix} 0.00 & 0.71 & -0.70 \\ -0.50 & 0.62 & 0.61 \\ 0.87 & 0.35 & 0.35 \end{bmatrix}$	$\begin{bmatrix} 0.20 & 0.23 & -0.95 \\ -0.58 & 0.80 & 0.08 \\ 0.79 & 0.53 & 0.30 \end{bmatrix}$	$\begin{bmatrix} 0.98 & 0.09 & -0.17 \\ 0.02 & 0.85 & 0.52 \\ 0.20 & -0.51 & 0.83 \end{bmatrix}$	$\begin{bmatrix} 0.50 & 0.00 & -0.86 \\ 0.15 & 0.98 & 0.08 \\ 0.85 & -0.17 & 0.49 \end{bmatrix}$
Proposed Angles	$\theta = 0.78$ $\phi = 1.05$ $\psi = 1.57$	$\theta = 1.26$ $\phi = 0.25$ $\psi = 0.85$	$\theta = 0.18$ $\phi = 0.56$ $\psi = 0.09$	$\theta = 1.04$ $\phi = 0.16$ $\psi = 0.00$
Proposed Error	0.02 ± 0.01	0.02 ± 0.01	0.02 ± 0.01	0.02 ± 0.01
Comparison Rotation	$\begin{bmatrix} 0.01 & 0.72 & -0.69 \\ -0.49 & 0.61 & 0.62 \\ 0.87 & 0.33 & 0.36 \end{bmatrix}$	$\begin{bmatrix} 0.22 & 0.26 & -0.94 \\ -0.57 & 0.82 & 0.09 \\ 0.79 & 0.52 & 0.32 \end{bmatrix}$	$\begin{bmatrix} 0.98 & 0.09 & -0.18 \\ 0.01 & 0.86 & 0.52 \\ 0.20 & -0.51 & 0.84 \end{bmatrix}$	$\begin{bmatrix} 0.52 & 0.02 & -0.85 \\ 0.14 & 0.98 & 0.11 \\ 0.84 & -0.17 & 0.51 \end{bmatrix}$
Comparison Angles	$\theta = 0.77$ $\phi = 1.04$ $\psi = 1.55$	$\theta = 1.23$ $\phi = 0.28$ $\psi = 0.87$	$\theta = 0.18$ $\phi = 0.55$ $\psi = 0.10$	$\theta = 1.02$ $\phi = 0.20$ $\psi = 0.03$
Comparison Error	0.04 ± 0.01	0.03 ± 0.01	0.05 ± 0.01	0.04 ± 0.01

TABLE 3. Summary of results for two rotation-only test cases using two dimensional data as inputs.

	Test 5	Test 6
Actual Rotation	$\begin{bmatrix} 0.91 & 0.11 & -0.39 \\ 0.25 & 0.59 & 0.76 \\ 0.32 & -0.80 & 0.51 \end{bmatrix}$	$\begin{bmatrix} 0.84 & 0.48 & -0.26 \\ -0.50 & 0.87 & 0 \\ 0.22 & 0.13 & 0.97 \end{bmatrix}$
Actual Angles	$\theta = 0.40$ $\phi = 0.98$ $\psi = 0.12$	$\theta = 0.26$ $\phi = 0$ $\psi = 0.52$
Proposed Rotation	$\begin{bmatrix} 0.91 & 0.11 & -0.37 \\ 0.25 & 0.59 & 0.74 \\ 0.32 & -0.79 & 0.50 \end{bmatrix}$	$\begin{bmatrix} 0.84 & 0.49 & -0.24 \\ -0.50 & 0.87 & 0.00 \\ 0.23 & 0.13 & 0.93 \end{bmatrix}$
Proposed Angles	$\theta = 0.38$ $\phi = 0.98$ $\psi = 0.12$	$\theta = 0.25$ $\phi = 0.00$ $\psi = 0.53$
Proposed Error	0.02 ± 0.01	0.02 ± 0.01
Comparison Rotation	$\begin{bmatrix} 0.92 & 0.11 & -0.38 \\ 0.25 & 0.60 & 0.76 \\ 0.32 & -0.80 & 0.52 \end{bmatrix}$	$\begin{bmatrix} 0.84 & 0.48 & -0.25 \\ -0.50 & 0.87 & 0.00 \\ 0.22 & 0.12 & 0.97 \end{bmatrix}$
Comparison Angles	$\theta = 0.40$ $\phi = 0.98$ $\psi = 0.12$	$\theta = 0.26$ $\phi = 0.01$ $\psi = 0.52$
Comparison Error	0.03 ± 0.01	0.03 ± 0.01

$T_A^B * Frame_A$. To more accurately model real-world conditions, noise was added separately to each data point in $Frame_B$ using the formula $(x, y, z) + 0.05 * (rand(), rand(), rand())$ where rand() was a single random value. The results with three dimensional data points are shown in Table 2 while results with two dimensional data points are shown in Table 3. In these tables, the error is written as (error) $\pm$ (standard deviation of the error) and was calculated using (27) through (30) for both methods.

Tables 2 and 3 illustrate that the method presented in this article is more accurate than the method presented by Horn *et al.* in each examined case. Although both methods exhibited errors on the same order of magnitude as the system noise, ± 0.05, the method discussed here resulted in a lower error in every test case.

VI. CONCLUSION

This article presented a novel strategy for the automatic calculation of a transformation matrix. The transformation matrix was calculated without user input from paired data in two coordinate frames, and was able to do so in a very short time, even with large data sets. This method scales

linearly with the number of data points, making it suitable for a wide variety of applications without requiring a prohibitive amount of processing time. The mathematical formulation of this process was described in Section III. This section also described how the individual Euler angles could be calculated from the resultant transformation matrix. The process described in this article was then verified using experimental data as detailed in Section IV. This method was demonstrated to work accurately and produce deterministic results. The error on the transformation matrix calculation was found to be low, making it an ideal method for automatic calculation of transformation matrices. Finally, this method was compared to a method that is currently in use in Section V and was found to yield more accurate results for the examined test cases.

As this method represents a new approach, future work is planned to assess this technique under a wider range of experimental conditions to verify its success in disparate use cases. The testing will include an examination of its sensitivity to the locations used for the transformation matrix calculations. Further, the automatic matrix method will be further compared with current methods to more accurately assess how its accuracy and speed compares to a variety of methods currently in use.

ACKNOWLEDGMENT

The authors would like to acknowledge the help of Thomas Adamek, Anne Mahacek, Mike Rasay, Alicia Sherban, Mike Vlahos, and Christian Zempel.

REFERENCES

[1] K. Hosoi and M. Sugimoto, "Shepherd: An interface for overcoming reference frame transformations in robot control," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Kunming, China, Dec. 2006, pp. 908–913.
[2] OpenCV Dev Team. (2011). *Camera Calibration 3D Reconstruction*. [Online]. Available: http://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=composert#calibratecamera
[3] OpenCV Dev Team. (2017). *Levenberg-Marquardt in Calibratecamera Has a Small Basin of Convergence*. [Online]. Available: https://github.com/opencv/opencv/issues/4339
[4] C. Sao and P. W. Lehn, "A block diagram approach to reference frame transformation of converter dynamic models," in *Proc. Can. Conf. Elect. Comput. Eng.*, Ottawa, ON, Canada, May 2006, pp. 2270–2274.
[5] A. Mukerjee and N. Mittal, "Qualitative models of 3-D frame-transformation motions," in *Proc. IEEE Int. Conf. Robot. Autom.*, Nagoya, Japan, May 1995, pp. 2279–2284.
[6] Y. Li, L. Zhang, C. Tian, C. Ding, Y. Zhang, and W. Wei, "Hyperspectral image Super-resolution extending: An effective fusion based method without knowing the spatial transformation matrix," in *Proc. IEEE Int. Conf. Multimedia Expo*, Hong Kong, Jul. 2017, pp. 1117–1122.
[7] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, Apr. 1991.
[8] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987.
[9] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 5, no. 7, pp. 1127–1135, Jul. 1988.
[10] W. Chen *et al.*, "A noise-tolerant algorithm for robot-sensor calibration using a planar disk of arbitrary 3-D orientation," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 251–263, Jan. 2018.
[11] E. Gomez, Y. Karant, V. Malkoc, M. R. Neupane, K. E. Schubert, and R. W. Schulte, "Orthogonal and least-squares based coordinate transforms for optical alignment verification in radiosurgery," in *Proc. Int. Conf. Inf. Technol. Coding Comput.*, Las Vegas, NV, USA, Apr. 2005, pp. 83–88.
[12] J. J. Craig, "Mappings: Changing descriptions from frame to frame," in *Introduction to Robotics: Mechanics and Control*. 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2005, ch. 2, sec. 3, pp. 24–29.
[13] H. Baruh, *Analytical Dynamics*, 1st ed. New York, NY, USA: McGraw-Hill, 1999, ch. 7, pp. 419–420.
[14] The Mathworks, Inc. (2017). *Rand*. [Online]. Available: https://www.mathworks.com/help/matlab/ref/rand.html
[15] M. H. DeGroot and M. J. Schervish, "The sample mean," in *Probability and Statistics*, 3rd ed. Boston, MA, USA: Addison-Wesley, 2002, ch. 4, sec. 8, pp. 229–236.
[16] M. H. DeGroot and M. J. Schervish, "Variance," in *Probability and Statistics*, 3rd ed. Boston, MA, USA: Addison-Wesley, 2002, ch. 4, sec. 3, pp. 197–203.
[17] G. Strang, "Triangular factors and row exchanges," in *Linear Algebra and Its Applications*, 4th ed. Belmont, CA, USA: Thomson Higher Education, 2006, ch. 1, sec. 5, pp. 32–45.
[18] R. Penrose, "A generalized inverse for matrices," *Math. Proc. Cambridge Philos. Soc.*, vol. 51, no. 3, pp. 406–413, Jul. 1955.

JASMINE CASHBAUGH (M'15) received the B.S. and M.S. degrees in aerospace engineering from Purdue University in 2006 and 2008, respectively, and the Ph.D. degree in mechanical engineering from Santa Clara University in 2016. She was a Systems Engineer with Lockheed Martin Aerospace Company's Advanced Development Programs and worked as a Consultant in the robotics field.

CHRISTOPHER KITTS (SM'16) received the B.S.E. degree in mechanical and aerospace engineering from Princeton University in 1987, the M.S. degree in aerospace engineering from Stanford University in 1992, the M.P.A degree in international defense and policy from the University of Colorado in 1996, and the Ph.D. degree in mechanical engineering from Stanford University in 2006. He is currently an Associate Professor of mechanical engineering, the Associate Dean of Research and Faculty Development for the School of Engineering, Santa Clara University, and a Fellow of the American Society of Mechanical Engineers.