Approximating Displacements in $\mathbb{R}^3$ by Rotations in $\mathbb{R}^4$ and its Application to Pointcloud Registration

Soheil Sarabandi and Federico Thomas

***Abstract*—No proper norm exists to measure the distance between two object poses essentially because a general pose is defined by a rotation and a translation, and thus it involves magnitudes with different units. As a means to solve this dimensional-inhomogeneity problem, the concept of *characteristic length* has been put forward in the area of kinematics. The idea consists in scaling translations according to this characteristic length and then approximating the corresponding displacement defining the object pose in $\mathbb{R}^3$ by a rotation in $\mathbb{R}^4$, for which a norm exists. This paper sheds new light on this kind of approximations which permits simplifying optimization problem whose cost functions involve translations and rotations simultaneously. A good example of this kind of problems is the pointcloud registration problem in which the optimal rotation and translation between two sets of corresponding 3D point data, so that they are aligned/registered, have to be found. As a result, a simple closed-form formula for solving this problem is presented which is shown to be an attractive alternative to the previous approaches.**

***Index Terms*—3D rigid displacements, 4D rotations, 3D pointcloud registration, characteristic length, quaternions, dual quaternions.**

I. Introduction

IT is well-known that there is no proper definition of norm for rigid-body displacements because of the disparity of the units of translations and rotations. Various approaches have been proposed in the literature to provide ways around this inconvenience (see [1] and the references therein). The problem is of particular relevance when performing optimizations of cost functions that involve both translations and rotations.

Norms obviously exist for displacements that are either pure translations or pure rotations. The existence of norms for rotations in $\mathbb{R}^3$ [2] was exploited in [3] to define a norm for a planar displacement by approximating the displacement in $\mathbb{R}^2$ by a rotation in $\mathbb{R}^3$. This work was extended in [4] to the approximation of a displacement in $\mathbb{R}^3$ by a rotation in $\mathbb{R}^4$. Since rigid displacements in $\mathbb{R}^3$ can be represented using dual quaternions, and rotations in $\mathbb{R}^4$ by double quaternions, the problem can be reduced to approximate dual quaternions by double quaternions [5]. This kind of approximation has been successfully used in [6] to solve the inverse kinematics of 6R serial robots by expressing all displacements in terms of double quaternions. An alternative approach, which avoids the use of quaternions, consists in computing the singular-value decomposition, or the polar decomposition, of the homogeneous transformation matrix representing the rigid displacement to approximate it by a rotation [7], [5]. In all these approximations, distances are actually scaled according to a *characteristic length*. This leads to a homogenization of units which allows, in turn, for a consistent addition of translation and rotation terms.

In [1], an exact expression for the error of the approximation as a function of the characteristic length is derived to arrive at the rather obvious conclusion that this approximation is improved monotonically as the characteristic length tends to infinity. Nevertheless, in practice, the value of the characteristic length is limited by the numerical errors that arise if it is set to very high values, a limit that depends on the used floating-point representation. As a consequence, up to the present time, the choice of the characteristic length value is based on application-dependent rules of thumb. For example, in [1], the synthesis of a planar mechanism is solved by taking this value as ten times the root mean square of all involved distances. In [6], the inverse kinematics of a 6R robot is solved taking it as L/ϵ^2, where ϵ is the desired accuracy level and L is the maximum distance from the base to the end-effector of the manipulator. More recently, the characteristic length used in [8], based upon the investigations reported in [3], [9], is $24T/\pi$, where T is the maximum translational component in all involved displacements. This characteristic length is the radius of the hypersphere that approximates the translational terms by angular displacements that are lower than 7.5 degrees. It was shown in [9] that this characteristic length yields a good balance between translational and rotational displacement terms.

Once assuming that we have solved the problem of approximating elements of $SE(3)$ by elements of $SO(4)$, an additional challenge remains: the application of a metric on $SO(4)$ is not well-defined in this case because of the dependence on the choice of the fixed reference frame. To mitigate this problem, the concept of *principal frame* is introduced in [8]. This frame is unique for a finite set of displacements and invariant with respect to the choice of fixed coordinate frame and the system of units. All of the displacements are then expressed with respect to the principal frame and all distances are measured with respect to this same frame.

The authors are with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC), ETSEIB, Diagonal 647, Pavelló E, planta 1, 08028 Barcelona, Spain. ssarabandi@iri.upc.edu, f.thomas@csic.es

This work was partially supported by the Spanish Ministry of Economy and Competitiveness through the projects DPI2017-88282-P, PID2020-117509GB-I00, and MDM-2016-0656.

This paper has supplementary multimedia material that can be downloaded from http://www.iri.upc.edu/people/thomas/Soft/code-4R.zip. This material consists of a MATLAB toolbox and some scripts that permit replicating the reported results. Contact F. Thomas for questions about this material.

There are many problems in which we have to optimize a function that involves both translations and rotations. Hence, it is interesting to attempt the introduction of a characteristic length in this kind of problems to verify whether the obtained results outperform those obtained using standard approaches. In this sense, the pointcloud registration problem is an excellent testbed where to perform this attempt. This paper is devoted to this analysis to conclude that the introduction of a characteristic length in the pointcloud registration problem leads to a simple (in the sense that it only involves the four basic arithmetic operations) closed-form formula which is finally independent, curiously enough, of the introduced characteristic length. Moreover, the resulting method is faster than all previously proposed ones.

When dealing with the displacement of 3D rigid bounded object poses, it is possible to define some metrics, also called *object norms*, based only on geometric properties of the object (for example, based on the distances between corresponding points) that avoids the use of characteristic lengths. A good example of this kind of metric can be found in [10], which can be considered as a generalization of the one proposed in [11]. Actually, we can say that all the methods proposed to date to solve the pointcloud registration problem implicitly use an object norm because they are based on minimizing the sum of the distances between corresponding points. This avoids having to weigh translations and rotations differently. Thus, although our formulation of the pointcloud registration is certainly more complicated, the obtained solution is outstandingly simple as it boils down to compute a pseudoinverse (which could be performed off-line because it only depends on the coordinates of the points in the reference poincloud) and a matrix product.

This paper is outlined as follows. Section II reviews the pointcloud registration problem. In this section the three methods proposed in the literature for its resolution are presented in a unified way. Although they differ in the way they represent a spatial displacement, they minimize the same cost function. As a consequence, despite the literature is confusing at this point, it is not surprising that the solutions using them all coincide, as we will see in Section V. In Section III we present the new method based on the introduction of a characteristic distance that allows us to approximate a homogeneous transformation matrix by a rotation matrix. Section IV discusses the main features of the obtained closed-form formula, including the effect of noisy measurements. Section V analyzes the performance of the presented method. This paper concludes in Section VI with a summary of the main findings of this research.

II. The pointcloud registration problem

The goal of pointcloud registration is to find the optimal rigid transformation between two pointclouds. This problem arises in many applications of computer vision and pattern recognition (see [12], [13], [14] for reviews on the different proposed methods to solve it and applications where it arises). We herein assume that the exact correspondence between the points in both pointclouds is known. An acceptable solution in such a case consists in minimizing the square of the sum of the squared Euclidean distances between corresponding points of the two pointclouds. The methods designed following this approach are usually called *analytical methods* [15], [16], [17], [18], [19] (see [13] and [20] for performance comparisons). The first methods for solving the general case in which we do not have the point correspondence were independently proposed during the early nineties in [21], [22] and [23]. The name given in [22] for their method was the Iterative Closest Point (ICP) method. Nowadays, the three methods and those which have evolved from them (see, for example, [24], [12]) are referred collectively in the literature as *ICP methods*. Broadly speaking, these methods iteratively generate hypothetical point correspondences using a local search scheme until an optimal correspondence is obtained. In general, these methods require a good starting estimate, otherwise they could get trapped in local minima of the error function. As a consequence, even if we use an ICP method, we have to rely on a fast analytic method to obtain the rigid transformation from which we can evaluate the error function at each iteration.

The existing analytic methods essentially differ in the way rigid displacements are represented. Three alternatives can be found in the literature that consist in using:

1) A *rotation matrix* and a *translation vector* [25]. In this case, the problem is essentially reduced to compute the nearest rotation matrix to a 3×3 matrix using the singular value decomposition (SVD). The method based on this representation is reviewed in Section II-A.
2) A set of *Euler parameters* and a *translation vector* [16], [22]. In this case, the problem is reduced to compute the dominant eigenvector of a 4×4 matrix. The method based on this representation is reviewed in Section II-B.
3) A set of *screw parameters* [18], [26]. In this case, the problem is also reduced to compute the dominant eigenvector of a 4×4 matrix. The method based on this representation is reviewed in Section II-C.

Euler parameters can be arranged as the elements of a quaternion. Thus, although the algebra of quaternions is not strictly necessary to solve the pointcloud registration problem, the methods using the second kind of representation are usually called quaternion-based methods. Likewise, since screw parameters can be organized as the elements of a dual quaternion, the methods using the third kind of representation are called dual quaternion-based methods, despite the algebra of dual quaternions is not strictly needed in the resolution of the pointcloud registration problem.

The methods based on the first two representations compute the rotation and the translation in a decoupled way. This is accomplished by first centering both pointclouds by subtracting their centroid coordinates and thus reducing the problem to optimize a rotation. Therefore, the original problem is reduced to two parts: (a) obtaining the rotation that minimizes the registration error between the two centered pointclouds; and (b) computing the translation from both centroids and the obtained rotation. To avoid this decoupling, the representation based on screw parameters has been used to encapsulate the rotation and the translation in a single representation. This idea was first presented in [18], and later extended in [26].

Unfortunately, all analytic methods are equivalent after all, they just differ in the way the problem is represented. All experimental comparisons have found that the differences are negligible in practical applications with nondegenerate data [27], [13]. This conclusion is confirmed in the analysis included in this paper.

Finally, it is also worth mentioning here that, motivated by the success of deep learning in high-level vision tasks, various types of deep learning based pointcloud registration methods have been proposed to exploit different aspects of the problem (see [28] for a comprehensive overview of these approaches). Nevertheless, similarly to what happens with ICP methods, it is also important for these approaches to have a rapid way to estimate the sought rigid transformation from which we can evaluate the committed error.

A. The standard approach

Let $\{A_i\}$ and $\{B_i\}$, $i = 1, \ldots, n$, denote two sets of n points in $3D$ whose position vectors are given by $\mathbf{a}_i = (a_{ix}\, a_{iy}\, a_{iz})^T$ and $\mathbf{b}_i = (b_{ix}\, b_{iy}\, b_{iz})^T$, respectively. These point coordinates can be organized in matrix form as

$$\mathbf{A} = (\mathbf{a}_1\, \mathbf{a}_2\, \cdots\, \mathbf{a}_n) \;\text{ and }\; \mathbf{B} = (\mathbf{b}_1\, \mathbf{b}_2\, \cdots\, \mathbf{b}_n) \tag{1}$$

Now, if $\{B_i\}$ is the result of applying a noisy rigid spatial transformation to $\{A_i\}$, we have that

$$\mathbf{b}_i = \mathbf{R}\mathbf{a}_i + \mathbf{t} + \mathbf{n}_i, \tag{2}$$

where $\mathbf{R}$ is a $3{\times}3$ rotation matrix, $\mathbf{t}$, a translation vector, and $\mathbf{n}_i$, a noise vector.

The problem consists in finding $\mathbf{R}$ and $\mathbf{t}$ that minimizes the error function

$$\mathcal{E} = \sum_{i=1}^{n} \|\mathbf{b}_i - (\mathbf{R}\mathbf{a}_i + \mathbf{t})\|^2 . \tag{3}$$

If the rigid transformation is not contaminated by noise, the centroids of $\{A_i\}$ and $\{B_i\}$ (say $\mathbf{a}$ and $\mathbf{b}$, respectively) are clearly governed by the same rotation matrix and translation vector [25]. Then, if we define

$$\bar{\mathbf{a}}_i = \mathbf{a}_i - \mathbf{a}, \quad \text{where} \quad \mathbf{a} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{a}_i, \tag{4}$$

$$\bar{\mathbf{b}}_i = \mathbf{b}_i - \mathbf{b}, \quad \text{where} \quad \mathbf{b} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{b}_i, \tag{5}$$

the error function (3) can be rewritten as

$$\mathcal{E} = \sum_{i=1}^{n} \left\|\bar{\mathbf{b}}_i - \mathbf{R}\bar{\mathbf{a}}_i\right\|^2 . \tag{6}$$

Therefore, the original least-squares problem is reduced to two subproblems:

(a) obtaining $\widehat{\mathbf{R}}$ that minimizes (6); and
(b) obtaining the translation vector as

$$\hat{\mathbf{t}} = \mathbf{b} - \widehat{\mathbf{R}}\mathbf{a}. \tag{7}$$

The original problem is thus simplified by decoupling the rotation and the translation. Although this is the usual approach, it should be seen as an approximation as it does not necessarily lead to the best solution both in terms of rotational and translational error under the presence of noise, as it is proved in V-A.

The expansion of (6) yields

$$\mathcal{E} = \sum_{i=1}^{n} \left(\bar{\mathbf{a}}_i^T\bar{\mathbf{b}}_i + \bar{\mathbf{b}}_i^T\bar{\mathbf{b}}_i - 2\bar{\mathbf{b}}_i^T\mathbf{R}\bar{\mathbf{a}}_i\right). \tag{8}$$

$\mathcal{E}$ is minimized when the last term is maximized. That is, when

$$\mathcal{E}' = \sum_{i=1}^{n} \bar{\mathbf{b}}_i^T\mathbf{R}\bar{\mathbf{a}}_i = \text{Trace}(\mathbf{R}^T\mathbf{H}), \tag{9}$$

is maximized, where

$$\begin{aligned}\mathbf{H} &= \sum_{i=1}^{n} \bar{\mathbf{b}}_i\bar{\mathbf{a}}_i^T = \sum_{i=1}^{n} (\mathbf{b}_i - \mathbf{b})(\mathbf{a}_i - \mathbf{a})^T \\ &= \sum_{i=1}^{n} \mathbf{b}_i\mathbf{a}_i^T - n\mathbf{b}\mathbf{a}^T = \mathbf{B}\mathbf{A}^T - n\mathbf{b}\mathbf{a}^T.\end{aligned} \tag{10}$$

In other words,

$$\mathbf{H} = \bar{\mathbf{B}}\bar{\mathbf{A}}^T = \mathbf{B}\mathbf{A}^T - n\mathbf{b}\mathbf{a}^T, \tag{11}$$

where

$$\bar{\mathbf{A}} = (\bar{\mathbf{a}}_1\, \bar{\mathbf{a}}_2\, \cdots\, \bar{\mathbf{a}}_n) \;\text{ and }\; \bar{\mathbf{B}} = \left(\bar{\mathbf{b}}_1\, \bar{\mathbf{b}}_2\, \cdots\, \bar{\mathbf{b}}_n\right). \tag{12}$$

$\mathbf{H}$ is defined as the *centered* cross-correlation matrix between both sets of pointclouds. It is important to observe that it is not necessary to explicitly center the pointclouds to obtain $\mathbf{H}$. This is very important when having thousands of points in the pointclouds, a fact that is not taken into account in some implementations (for example, see the recent MATLAB implementation due to Jin Wu [19]).

It can be proved that the matrix $\mathbf{R}$ that maximizes (9) is the nearest rotation matrix, in Frobenius norm, to $\mathbf{H}$ [17]. Analytically, this optimum can be expressed as [29]:

$$\widehat{\mathbf{R}} = \mathbf{H}\left(\mathbf{H}^T\mathbf{H}\right)^{-\frac{1}{2}}. \tag{13}$$

There are many different ways to compute (13) that include iterative and closed-form methods [30], but the standard one is based on the SVD of the cross-correlation matrix $\mathbf{H}$. Let this decomposition be expressed as $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are $3{\times}3$ orthogonal matrices, and $\mathbf{\Lambda}$ is a diagonal matrix with nonnegative elements. Then, it can be proved that $\widehat{\mathbf{R}} = \mathbf{U}\mathbf{V}^T$. This approach was first proposed in [31] and later rediscovered in [15]. It remains as the standard one ever since. A simple later improvement, that has been broadly adopted, was introduced in [32] for better robustness when the point measurements are severely corrupted by noise. This method is usually referenced to as the Kabsch-Umeyana's method in recognition of the authors of [31] and [32].

B. Using Euler parameters

According to Euler's theorem, any spacial rotation is equivalent to a rotation by some amount about some axis [33]. Then, we can represent an arbitrary rotation using

1) $\mathbf{v} = (v_x\ v_y\ v_z)^T$, a unit vector in the direction of the rotation axis; and

2) θ, a rotation angle.

Then, we can define the following vector whose elements are called Euler parameters:

$$\mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ \\ \sin\left(\frac{\theta}{2}\right)\mathbf{v} \end{pmatrix}. \tag{14}$$

From a given set of Euler parameters, the corresponding rotation matrix can be obtained as

$$\mathbf{R}(\mathbf{e}) = \frac{1}{e_1^2+e_2^2+e_3^2+e_4^2} \begin{pmatrix} e_1^2+e_2^2-e_3^2-e_4^2 & 2(e_2e_3-e_1e_4) & 2(e_2e_4+e_1e_3) \\ 2(e_2e_3+e_1e_4) & e_1^2-e_2^2-e_3^2-e_4^2 & 2(e_3e_4-e_1e_2) \\ 2(e_2e_4-e_1e_3) & 2(e_3e_4+e_1e_2) & e_1^2-e_2^2-e_3^2+e_4^2 \end{pmatrix}. \tag{15}$$

Now, given $\mathbf{H} = (h_{ij})_{1\le i,j\le 3}$, let us define the associated 4×4 symmetric matrix

$$\mathbf{G} = \begin{pmatrix} h_{11}+h_{22}+h_{33} & h_{32}-h_{23} & h_{13}-h_{31} & h_{21}-h_{12} \\ h_{32}-h_{23} & h_{11}-h_{22}-h_{33} & h_{21}+h_{12} & h_{31}+h_{13} \\ h_{13}-h_{31} & h_{21}+h_{12} & h_{22}-h_{11}-h_{33} & h_{32}+h_{23} \\ h_{21}-h_{12} & h_{31}+h_{13} & h_{32}+h_{23} & h_{33}-h_{11}-h_{22} \end{pmatrix}. \tag{16}$$

It was proved in [16], [22], and later independently rediscovered in [34] and [35], that the dominant eigenvector of $\mathbf{G}$ can be interpreted as a vector of Euler parameters, in the form given in (14), whose corresponding rotation matrix is $\widehat{\mathbf{R}}$ (let us remember that the dominant eigenvector is the eigenvector associated with the eigenvalue whose absolute value is maximal).

The computation of the maximal eigenvalue requires computing the roots of a quartic polynomial which can be performed using Ferrari's method, as done in [16]. Alternatively, a numerical method is proposed in [35]. Very recently, a simple closed-form solution has been presented in [19] which is next compactly summarized.

The characteristic polynomial of $\mathbf{G}$ can be expressed as

$$\lambda^4 + \tau_3\lambda^3 + \tau_2\lambda^2 + \tau_1\lambda + \tau_0, \tag{17}$$

where (see [16] for details)

$$\begin{aligned} \tau_3 &= \text{Trace}(\mathbf{G}) = 0, \\ \tau_2 &= -2\sum_{i=1}^{3}\sum_{j=1}^{3} h_{i,j}^2 = -2\,\text{Trace}\left(\mathbf{H}^T\mathbf{H}\right), \\ \tau_1 &= -8\det(\mathbf{H}), \\ \tau_0 &= \det(\mathbf{G}). \end{aligned}$$

The roots of (17) are real because $\mathbf{G}$ is symmetric. Moreover, the application of Ferrari's method to obtained these roots is simplified because τ_3 is identically zero. In [19], it is actually shown how the largest root of (17) can be expressed as

$$\lambda_{\max} = \begin{cases} \sqrt{-\frac{\tau_1}{2}}, \text{ if } |\tau_1| < \zeta \text{ and } |\,k_1| < \zeta, \\ \\ \frac{1}{\sqrt{6}}\left(p_1 + \sqrt{-k_1^2 - 12\tau_2 - \frac{12\sqrt{6}\tau_1}{k_1}}\right), \text{ otherwise,} \end{cases}$$

where

$$\begin{aligned} k_0 &= 2\tau_2^3 + 27\tau_1^2 - 72\tau_2\tau_0, \\ \theta &= \text{atan2}\left(\sqrt{4(\tau_2^2+12\tau_0)^3 - k_0^2},\, k_0\right), \\ k_1 &= 2\sqrt{\left(\sqrt{\tau_2^2+12\tau_0}\right)\cos\frac{\theta}{3} - \tau_2}. \end{aligned}$$

The threshold ζ is typically taken to be 10^{-5} [19].

Finally, it can be proved that all the rows of the cofactor matrix of $(\mathbf{G} - \lambda_{\max}\mathbf{I})$ are proportional to the eigenvector corresponding to $\lambda_{\max}$ [16]. In [19], some computational time is saved by computing only the last row of this cofactor matrix. Unfortunately, all the elements of this row are identically zero for rotations whose rotation axis lies on the xy-plane. Although, at least in theory, rotations whose rotation axes lie on the xy-plane can be seen as a set of measure zero in the space of quaternions, in practice it is enough to be close to this situation to generate large errors. Similar situations arise if we take any other row. Thus, for the sake of robustness, we have to compute all rows and take, for example, the one with the largest norm, say $\hat{\mathbf{e}}$. Then, using (15), we have that $\hat{\mathbf{R}} = \mathbf{R}(\hat{\mathbf{e}})$.

C. Using screw parameters

The use of screw parameters, also known as Study parameters, is based on Chasles' theorem, which states that any rotation and translation can be expressed as a translation along a line, called screw axis, and a rotation around that line [36]. According to this theorem, any rigid displacement can be expressed using

1) $\mathbf{v} = (v_x\ v_y\ v_z)^T$, a unit vector in the direction of the screw axis;
2) $\mathbf{u} = (u_x\ u_y\ u_z)^T$, a vector from the origin of coordinates to any point on the screw axis;
3) θ, a rotation angle; and
4) d, translation distance;

Notice that the sign of d can be set arbitrarily, but once this sign is fixed, the positive sense of angle θ is determined according to the right-hand rule.

Now, let us define $\mathbf{e}$ as in (14) and

$$\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} = \begin{pmatrix} -\frac{d}{2}\sin(\frac{\theta}{2}) \\ \\ \frac{d}{2}\cos(\frac{\theta}{2})\mathbf{v} + \sin(\frac{\theta}{2})(\mathbf{u}\times\mathbf{v}) \end{pmatrix}. \tag{18}$$

Then, the rotation matrix can be obtained from $\mathbf{e}$ using equation (15), and the translation vector as

$$\mathbf{t} = 2\begin{pmatrix} -e_2 & e_1 & -e_4 & e_3 \\ -e_3 & e_4 & e_1 & -e_2 \\ -e_4 & -e_3 & e_2 & e_1 \end{pmatrix}\mathbf{s}. \tag{19}$$

Now, the goal is to obtain $\mathbf{e}$ and $\mathbf{s}$ that minimizes (3). Let us call them $\hat{\mathbf{e}}$ and $\hat{\mathbf{s}}$, respectively.

It can be proved that the cost function given in (3) can be expressed as a quadratic function in terms of $\mathbf{e}$ and $\mathbf{s}$ as follows (see [18] for details):

$$\mathbf{e}^T\mathbf{C}_1\mathbf{e} + \mathbf{s}^T\mathbf{C}_2\mathbf{s} + \mathbf{s}^T\mathbf{C}_3\mathbf{e} + \text{constant} \tag{20}$$

where

$$\begin{aligned}\mathbf{C}_1 &= \frac{1}{2}\sum_{i=1}^{n}\mathbf{Q}_i\mathbf{W}_i,\\ \mathbf{C}_2 &= n\mathbf{I},\\ \mathbf{C}_3 &= \sum_{i=1}^{n}\left(\mathbf{W}_i - \mathbf{Q}_i\right),\end{aligned}$$

and

$$\mathbf{Q}_i = \begin{pmatrix} 0 & -b_{ix} & -b_{iy} & -b_{iz}\\ b_{ix} & 0 & -b_{iz} & b_{iy}\\ b_{iy} & b_{iz} & 0 & -b_{ix}\\ b_{iz} & -b_{iy} & b_{ix} & 0\end{pmatrix}, \tag{21}$$

$$\mathbf{W}_i = \begin{pmatrix} 0 & -a_{ix} & -a_{iy} & -a_{iz}\\ a_{ix} & 0 & a_{iz} & -a_{iy}\\ a_{iy} & -a_{iz} & 0 & a_{ix}\\ a_{iz} & a_{iy} & -a_{ix} & 0\end{pmatrix}. \tag{22}$$

In order to minimize (20), the two constraints $\mathbf{e}^T\mathbf{e} = 1$ and $\mathbf{s}^T\mathbf{e} = 0$ are incorporated using Lagrange multipliers. Then, it can be concluded that $\hat{\mathbf{e}}$ is the dominant eigenvector of

$$\mathbf{A} = \frac{1}{2n}\mathbf{C}_3^T\mathbf{C}_3 - \mathbf{C}_1 - \mathbf{C}_1^T, \tag{23}$$

and

$$\hat{\mathbf{s}} = -\frac{1}{2n}\mathbf{C}_3\hat{\mathbf{e}}. \tag{24}$$

We have given here a simplified version of the extensive formulation presented in [18]. The cost function in the original presentation included not only the distances between corresponding points, but also the error between unit vectors representing the direction of edges or the normal to faces of the model. This formulation was even later extended to deal with scale factors in [37].

III. The Proposed Method

If point coordinates are represented using homogeneous coordinates, rotations and translations can be more compactly expressed using homogeneous transformations. In this case, equation (3) can be rewritten as:

$$\mathcal{E} = \sum_{i=1}^{n}\left\|\delta\mathbf{b}_i - \delta\left(\mathbf{R}\mathbf{a}_i + \mathbf{t}\right)\right\|^2 = \sum_{i=1}^{n}\left\|\mathbf{p}_i - \mathbf{D}\mathbf{q}_i\right\|^2, \tag{25}$$

where

$$\mathbf{D} = \begin{pmatrix}\mathbf{R} & \delta\mathbf{t}\\ \mathbf{0}^T & 1\end{pmatrix}, \tag{26}$$

and

$$\mathbf{q}_i = \begin{pmatrix}\delta\mathbf{a}_i\\ 1\end{pmatrix}, \qquad \mathbf{p}_i = \begin{pmatrix}\delta\mathbf{b}_i\\ 1\end{pmatrix}. \tag{27}$$

We have introduced the scale factor δ which clearly does not modify the optimum location and whose reciprocal, $1/\delta$, plays the role of a characteristic distance.

Now, let us suppose that we want to approximate $\mathbf{D}$ by a 4×4 rotation matrix (see [38] for an introduction to rotations in four dimensions). It can be proved that the value of $\mathbf{R}$ that minimizes the Frobenius norm of $(\mathbf{R}-\mathbf{D})$ is given by [39], [30]:

$$\widetilde{\mathbf{R}} = \mathbf{D}(\mathbf{I} + \mathbf{E})^{-\frac{1}{2}}, \tag{28}$$

where

$$\mathbf{E} = \mathbf{D}^T\mathbf{D} - \mathbf{I} = \delta\begin{pmatrix}\mathbf{0} & \mathbf{R}^T\mathbf{t}\\ \mathbf{t}^T\mathbf{R} & \delta\mathbf{t}^T\mathbf{t}\end{pmatrix}. \tag{29}$$

In [1], an exact expression for $(\mathbf{I} + \mathbf{E})^{\frac{1}{2}}$ is obtained using the polar decomposition. Thus, by inverting it and substituting the result in (28), the exact expression for $\widetilde{\mathbf{R}}$, as a function of δ, is obtained. This permits deriving an exact expression for the error of the approximation as a function of δ to conclude that this approximation is improved monotonically, at least in theory, as $\frac{1}{\delta}\to\infty$. Nevertheless, this is limited by numerical errors that arise for very low values of δ that depend on the used floating-point representation. Moreover, larger characteristic lengths result in an increase in the weight on the rotational terms whereas smaller ones result in an increase in weight on the translational terms. The used metric is therefore dependent on the choice of characteristic length which is not a desirable property. As a consequence, here we follow a different approach: we use δ as a symbol that represents a very small number. Therefore, it is reasonable to keep, in all algebraic expressions involving δ, only the term of lowest degree in it. We will see how the resulting approximation is finally independent of δ. This should not be surprising. It is like introducing reference frames to solve a geometric problem whose solution is independent of them after all.

First of all, observe that applying Newton's generalized binomial theorem on matrices to (28) yields

$$\widetilde{\mathbf{R}} = \mathbf{D}\left(\mathbf{I} - \frac{1}{2}\mathbf{E} + \frac{3}{8}\mathbf{E}^2 - \frac{5}{16}\mathbf{E}^3 + \frac{35}{128}\mathbf{E}^4 - \dots\right). \tag{30}$$

Now, the substitution of (29) in (30) yields

$$\begin{aligned}\widetilde{\mathbf{R}}\Big|_{\delta\to0} &= \begin{pmatrix}\mathbf{R} & \delta\mathbf{t}\\ \mathbf{0}^T & 1\end{pmatrix}\left[\mathbf{I} - \frac{\delta}{2}\begin{pmatrix}\mathbf{0} & \mathbf{R}^T\mathbf{t}\\ \mathbf{t}^T\mathbf{R} & \delta\mathbf{t}^T\mathbf{t}\end{pmatrix}\right]\\ &= \begin{pmatrix}\mathbf{R} & \frac{\delta}{2}\mathbf{t}\\ -\frac{\delta}{2}\mathbf{t}^T\mathbf{R} & 1\end{pmatrix},\end{aligned} \tag{31}$$

where the higher-order terms in δ have been neglected. This formula was already presented, without proof, in [5]. Now, we have that the nearest 4D rotation to the 3D homogeneous transformation in (26), in Frobenius norm and for $\delta\to0$, is given by (31). It is important to realize that the converse is not true: the nearest 3D homogeneous transformation to the 4D rotation in (31) is not given by (26). Indeed, it is given by

$$\widetilde{\mathbf{D}} = \begin{pmatrix}\mathbf{R} & \frac{\delta}{2}\mathbf{t}\\ \mathbf{0}^T & 1\end{pmatrix} \tag{32}$$

because its last row must necessarily be $(\mathbf{0}^T\ 1)$ (to represent a rigid displacement), and the other rows must be equal to those of $\widetilde{\mathbf{D}}$ (to minimize its Frobenius distance to it).

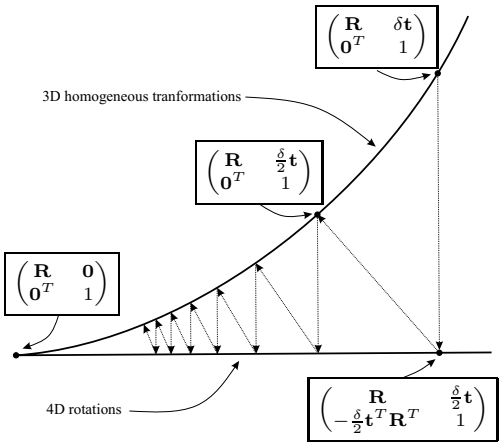


Fig. 1. The approximation of a 3D homogeneous transformation by a 4D rotation matrix, or vice versa, can be seen as a projection between two smooth manifolds embedded in $\mathbb{R}^{4\times 4}$. The result of iteratively repeating this operation leads to an alternating projection process that converges to a 3D rotation.

We can actually repeat the process. That is, we can approximate the 3D homogeneous transformation in (32) by the 4D rotation

$$\widetilde{\widetilde{\mathbf{R}}} = \begin{pmatrix}\mathbf{R} & \frac{\delta}{4}\mathbf{t}\\ -\frac{\delta}{4}\mathbf{t}^T\mathbf{R} & 1\end{pmatrix}, \tag{33}$$

whose nearest 3D homogeneous transformation is, in turn,

$$\widetilde{\widetilde{\mathbf{D}}} = \begin{pmatrix}\mathbf{R} & \frac{\delta}{4}\mathbf{t}\\ \mathbf{0}^T & 1\end{pmatrix}. \tag{34}$$

If this is infinitely repeated, the result is an alternating projection process [40] converging to

$$\begin{pmatrix}\mathbf{R} & \mathbf{0}\\ \mathbf{0}^T & 1\end{pmatrix}, \tag{35}$$

which can be seen both as 3D homogeneous transformation and a 4D rotation. Fig. 1 gives an intuitive graphical representation of this process.

Once we have clarified how to project 3D homogeneous transformations onto 4D rotations and vice versa, we can approximate (25), assuming that $\delta \to 0$, by

$$\mathcal{E} \cong \sum_{i=1}^{n}\left\|\mathbf{p}_i - \widetilde{\mathbf{R}}\mathbf{q}_i\right\|^2 = \sum_{i=1}^{n}\left(\mathbf{q}_i^T\mathbf{q}_i + \mathbf{p}_i^T\mathbf{p}_i - 2\mathbf{p}_i^T\widetilde{\mathbf{R}}\mathbf{q}_i\right), \tag{36}$$

whose minimization is equivalent to maximize

$$\sum_{i=1}^{n}\mathbf{p}_i^T\widetilde{\mathbf{R}}\mathbf{q}_i = \text{Trace}(\widetilde{\mathbf{R}}^T\mathbf{M}). \tag{37}$$

In other words, we have to find the nearest 4D rotation matrix to

$$\begin{aligned}\mathbf{M} &= \sum_{i=1}^{n}\mathbf{p}_i\mathbf{q}_i^T = \sum_{i=1}^{n}\begin{pmatrix}\delta\mathbf{b}_i\\ 1\end{pmatrix}\begin{pmatrix}\delta\mathbf{a}_i^T & 1\end{pmatrix}\\ &= \begin{pmatrix}\delta^2\sum_{i=1}^{n}\mathbf{b}_i\mathbf{a}_i^T & \delta\sum_{i=1}^{n}\mathbf{b}_i\\ \delta\sum_{i=1}^{n}\mathbf{a}_i^T & n\end{pmatrix}\\ &= \begin{pmatrix}\delta^2\mathbf{B}\mathbf{A}^T & \delta n\mathbf{b}\\ \delta n\mathbf{a}^T & n\end{pmatrix}.\end{aligned} \tag{38}$$

In the particular case in which $\{A_i\}$ and $\{B_i\}$ are centered with respect to their centroid, then

$$\mathbf{M}' = \begin{pmatrix}\delta^2\bar{\mathbf{B}}\bar{\mathbf{A}}^T & 0\\ 0 & n\end{pmatrix}. \tag{39}$$

Since the nearest rotation matrix to a given matrix is invariant with respect to any arbitrary non-zero scalar multiplying it, the problem boils down to compute the nearest rotation matrix to

$$\mathbf{M}'' = \begin{pmatrix}\frac{\delta^2}{n}\bar{\mathbf{B}}\bar{\mathbf{A}}^T & 0\\ 0 & 1\end{pmatrix}, \tag{40}$$

which simplifies to compute the nearest rotation matrix to

$$\mathbf{M}''' = \bar{\mathbf{B}}\bar{\mathbf{A}}^T \tag{41}$$

which, using equation (13), is given by

$$\begin{aligned}\mathbf{M}'''(\mathbf{M}'''^T\mathbf{M}''')^{-\frac{1}{2}} &= \bar{\mathbf{B}}\bar{\mathbf{A}}^T(\bar{\mathbf{A}}\bar{\mathbf{B}}^T\bar{\mathbf{B}}\bar{\mathbf{A}}^T)^{-\frac{1}{2}}\\ &= \bar{\mathbf{B}}\bar{\mathbf{A}}^T(\bar{\mathbf{A}}(\mathbf{R}\bar{\mathbf{A}})^T\mathbf{R}\bar{\mathbf{A}}\bar{\mathbf{A}}^T)^{-\frac{1}{2}}\\ &= \bar{\mathbf{B}}\bar{\mathbf{A}}^T(\bar{\mathbf{A}}\bar{\mathbf{A}}^T\bar{\mathbf{A}}\bar{\mathbf{A}}^T)^{-\frac{1}{2}}\\ &= \bar{\mathbf{B}}\bar{\mathbf{A}}^T(\bar{\mathbf{A}}\bar{\mathbf{A}}^T)^{-1}.\end{aligned} \tag{42}$$

As a consequence, the nearest 4×4 rotation matrix to (39) is

$$\widehat{\widetilde{\mathbf{R}}}' = \begin{pmatrix}\bar{\mathbf{B}}\bar{\mathbf{A}}^T(\bar{\mathbf{A}}\bar{\mathbf{A}}^T)^{-1} & 0\\ 0 & 1\end{pmatrix}. \tag{43}$$

Now, to obtain the nearest rotation matrix to (38), first observe that, from equation (31), it should be of the form

$$\widehat{\widetilde{\mathbf{R}}} = \begin{pmatrix}\widehat{\mathbf{R}} & \delta(\mathbf{b} - \widehat{\mathbf{R}}\mathbf{a})\\ \delta(\mathbf{a}^T - \mathbf{b}^T\widehat{\mathbf{R}}) & 1\end{pmatrix}, \tag{44}$$

which must be consistent with the limit case in which $\mathbf{a} = \mathbf{b} = 0$ given by (43). As a consequence, according to the derivation of (11), we can conclude that

$$\widehat{\mathbf{R}} = \left(\mathbf{B}\mathbf{A}^T - n\mathbf{b}\mathbf{a}^T\right)\left(\mathbf{A}\mathbf{A}^T - n\mathbf{a}\mathbf{a}^T\right)^{-1} \tag{45}$$

and

$$\hat{\mathbf{t}} = \mathbf{b} - \widehat{\mathbf{R}}\mathbf{a}, \tag{46}$$

as we already knew.

Before we came up with the above proof, we derived (45) and (46) by obtaining directly the nearest rotation matrix to (38) [41]. This latter approach is more insightful but, unfortunately, much more complicated as it involves operations such as the inverse and the series expansion of a block matrix.

IV. Discussion

To get some insight into the obtained result, let us first consider, as above, the case in which both centroids are centered with respect to their centroids. In this case,

$$\widehat{\mathbf{R}}' = \bar{\mathbf{B}}\bar{\mathbf{A}}^T\left(\bar{\mathbf{A}}\bar{\mathbf{A}}^T\right)^{-1} = \bar{\mathbf{B}}\bar{\mathbf{A}}^+, \tag{47}$$

where $\bar{\mathbf{A}}^+$ denotes the right Moore-Penrose pseudoinverse of $\bar{\mathbf{A}}$. It is important to observe that equation (45) can be recovered from equation (47) by using equation (11) to obtain the effect of translating the pointclouds on the matrix products $\bar{\mathbf{B}}\bar{\mathbf{A}}^T$ and $\bar{\mathbf{A}}\bar{\mathbf{A}}^T$. Since (45) and (47) imply each other, they

are equivalent. Therefore, without loss of generality, we can proceed with our analysis using (47).

Now, if $\{\bar{B}_i\}$ is the result of applying a general displacement to $\{\bar{A}_i\}$, that is, $\bar{\mathbf{B}} = \mathbf{R}\bar{\mathbf{A}} + \mathbf{T}$, where $\mathbf{T} = (\mathbf{t}\,\mathbf{t}\,\ldots\,\mathbf{t})$. The substitution of this expression for $\bar{\mathbf{B}}$ in (47) yields

$$\widehat{\mathbf{R}}' = \mathbf{R} + \mathbf{T}\bar{\mathbf{A}}^{+}. \tag{48}$$

Now, it is easy to prove that $\mathbf{T}\bar{\mathbf{A}}^{+}$ is a null matrix. Indeed, since the sum of the entries of each row of $\bar{\mathbf{A}}$ is zero (because it corresponds to a pointcloud whose centroid is at the origin), and the entries of each row of $\mathbf{T}$ are equal, $\mathbf{T}\bar{\mathbf{A}}^{T}$ is a 3×3 null matrix. As a consequence, $\mathbf{T}\bar{\mathbf{A}}^{+}$ is also a null matrix. Then, we can conclude that (45) returns the exact rotation matrix provided that $\bar{\mathbf{A}}\bar{\mathbf{A}}^{T}$ is invertible, *i.e.*, $\{A_i\}$ contains at least four points defining a non-degenerate tetrahedron.

Under the presence of additive noise, according to (2), we have that $\mathbf{B} = \mathbf{R}\mathbf{A} + \mathbf{T} + \mathbf{N}$, where $\mathbf{N} = (\mathbf{n}_1\mathbf{n}_2\ldots\mathbf{n}_n)$, $\mathbf{n}_i$ being the error vector added to $\mathbf{b}_i$. The substitution of this expression for $\mathbf{B}$ in (47) yields

$$\widehat{\mathbf{R}} = \mathbf{R} + \boldsymbol{\Delta}, \tag{49}$$

where $\boldsymbol{\Delta} = \mathbf{N}\mathbf{A}^{+}$. If the entries of $\mathbf{N}$ are assumed to be independent zero mean random variables with equal variance, say σ, the mean value of the entries of $\boldsymbol{\Delta}$ is zero (*i.e.*, $\mathrm{E}(\widehat{\mathbf{R}}) = \mathbf{R}$), and their variances,

$$\begin{aligned}\mathrm{Var}(\delta_{ij}) &= \sum_{k=1}^{n} \mathrm{Var}(n_{ik})(a^{+}_{kj})^2 \\ &= \sigma \sum_{k=1}^{n} (a^{+}_{kj})^2 = \sigma \left\|\mathbf{a}^{+}_{j}\right\|^2, \end{aligned} \tag{50}$$

where δ_{ij} denotes the (i,j) entry of $\boldsymbol{\Delta} = \widehat{\mathbf{R}} - \mathbf{R}$, and $\mathbf{a}^{+}_{j}$, the j column of $\mathbf{A}^{+}$. Thus, the variance of the entries of $\widehat{\mathbf{R}}$ depend linearly on the modules of the three columns of $\mathbf{A}^{+}$. Actually, as the points in the pointcloud are close to being coplanar or collinear, these modules tend to infinity. Thus, the shape of the pointcloud has a direct influence on the error of the estimations. We can conclude that, under the presence of zero mean additive uncorrelated noise, the expected value of $\widehat{\mathbf{R}}$ is $\mathbf{R}$. Nevertheless, a certain orthogonality error should be expected, according to (50), in the estimated rotation matrices that directly depends on the noise perturbing the measured point locations and the shape of the pointcloud itself. This is discussed in Section V-C.

Finally, it is interesting to observe that, if the registration operation has to be repeated iteratively, $\mathbf{A}^{+}$ can be precomputed so that each registration is simply reduced to compute a matrix product. The reduction in computational burden with respect to all other methods is thus important. Moreover, observe that the rows of $\mathbf{A}^{+}$ can be interpreted as point coordinates. Therefore, every pointcloud has an associated *reciprocal* pointcloud. To better understand the concept of reciprocal pointcloud, let us take the Stanford Bunny data model [42]. After centering it with respect to its centroid and normalizing its point coordinates according to its intrinsic scale (see the Appendix), we obtain the pointcloud represented in Fig. 2(top). Then, if we compute its reciprocal pointcloud, and we also normalize it with respect to its intrinsic scale, we obtain the pointcloud represented in Fig. 2(bottom). The ellipsoids associated with the covariances of both pointclouds (see also the Appendix) have aligned principal axes. If the length of a semiaxis is ζ, the length of the corresponding semiaxis in the reciprocal pointcloud is $1/\zeta$. The product of the three semi-axes lengths of both ellipsoids is one owing to the normalization of the pointclouds with respect to their intrinsic scales.

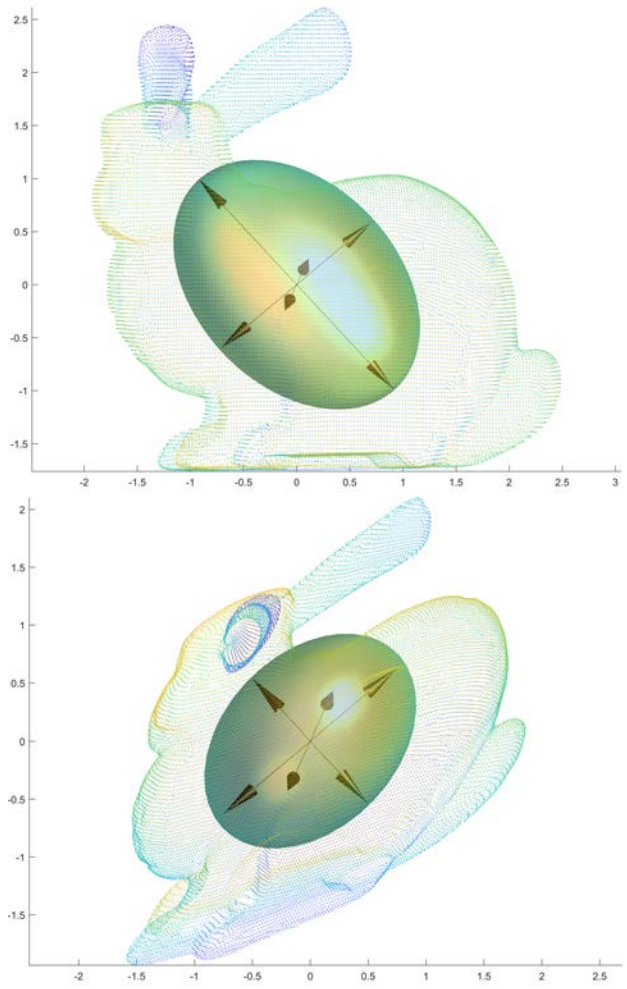


Fig. 2. A pointcloud (top) and its reciprocal pointcloud (bottom) together with the ellipsoids associated with their covariance matrices. Both pointclouds have been centered with respect to their centroids and normalized with respect to their intrinsic scales.

V. Performance Analysis

This paper has supplementary downloadable multimedia material consisting of a set of MATLAB functions that implement the methods described in the Sections II and III, and four scripts needed to reproduce the results of the four examples included in this section. All reported results have been obtained on a PC with a 4.2 GHz Intel Core i7 processor using double-precision arithmetics.

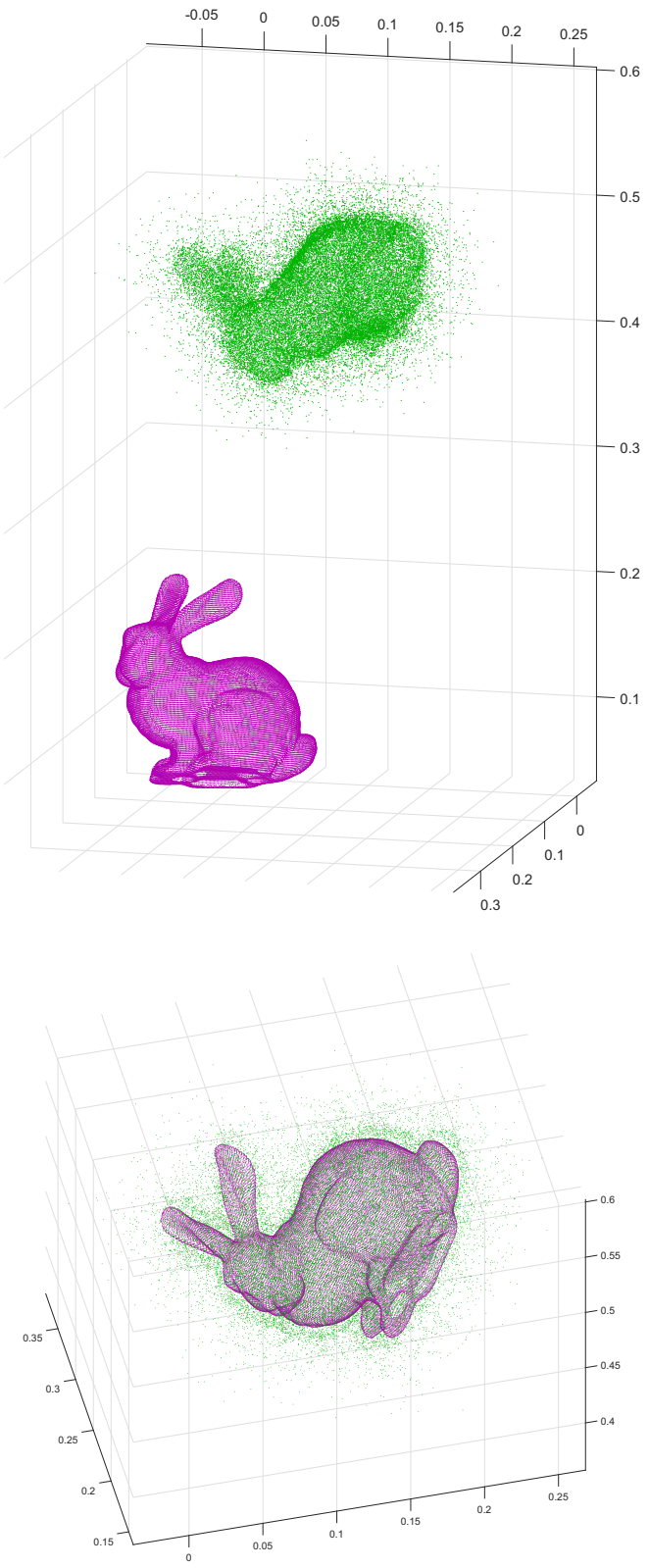


Fig. 3. Top: the reference and target pointclouds in magenta and green, respectively. Bottom: Registration result (there are no noticeable differences to the naked eye between the registrations obtained using all the analyzed methods).

A. Example I

As a first example, we have taken the Stanford Bunny data model as the reference pointcloud. It contains 35947 points, its enclosing box is $\mathcal{B} = [-0.095, 0.061]\times[0.033, 0.187]\times[-0.062, 0.059]$, and its centroid is $\mathbf{a} = (-0.0268, 0.0952, 0.0089)^T$.

The target pointcloud has been obtained by applying to all points in the reference pointcloud the following rotation

$$\mathbf{R} = \mathbf{R}_x\left(\frac{\pi}{3}\right)\mathbf{R}_y\left(\frac{\pi}{6}\right)\mathbf{R}_z\left(\frac{\pi}{4}\right) = \begin{pmatrix} 0.6124 & -0.6124 & 0.5000 \\ 0.6597 & 0.0474 & -0.7500 \\ 0.4356 & 0.7891 & 0.4330 \end{pmatrix},$$

and then the translation given by $\mathbf{t}=(0.2,\, 0.5,\, 0.1)^T$. The vector of Euler parameters quaternion corresponding to $\mathbf{R}$ is

$$\mathbf{e} = (0.7233, 0.5320, 0.0223, 0.4397)^T. \tag{51}$$

Then, we have perturbed some point locations in the target pointcloud. A noise with Gaussian distribution $\mathcal{N}(0, 0.02)$ has been added to the three coordinates of 20% of the points, with distribution $\mathcal{N}(0.005, 0.018)$ to 10% of the points, and with distribution $\mathcal{N}(-0.005, 0.018)$ to another 10% of the points. These three sets are disjoint and randomly selected. The reference and the resulting target pointclouds are depicted in Fig. 3(top).

The implemented methods to register the reference pointcloud with respect to the target pointcloud have been:

1) The method based on the SVD as described in Section II-A.
2) The standard quaternion method based on the computation of a dominant eigenvector as described in Section II-B
3) Wu et al.'s improved quaternion method based on the closed-form formulas also included in Section II-B.
4) The dual quaternion method described in Section II-C.
5) The new method presented in Section III.

Their accuracy was evaluated in terms of:

1) The error in the recovered rotation, computed as $\arccos(|\hat{\mathbf{e}}_i\cdot\mathbf{e}|)$, where $\mathbf{e}_i$ is the unit quaternion (strictly speaking it is just a vector of Euler parameters) corresponding to the estimated rotation matrix. This metric was apparently first used in [43] for 3D object pose estimation. It is a pseudometric in the unit quaternions but it is a metric in $SO(3)$ [2].
2) The error of translations computed simply as $\left\|\hat{\mathbf{t}}_i - \mathbf{t}\right\|$, where $\hat{\mathbf{t}}_i$ is the estimated translation.

These errors vary each time the program is executed because the perturbed points in the target pointcloud are randomly selected at each execution.

It has been verified that the rotational and translational errors are always the same for the methods based on the SVD, quaternions, or dual quaternions. The only truly distinguishing factor is execution time. This concurs with the results presented in [27]. The conclusion of superior accuracy of the method based on dual quaternions, as assured in [18], is thus incorrect.

TABLE I
ERROR FIGURES AND EXECUTION TIMES FOR THE FIVE COMPARED METHODS IN A PARTICULAR CASE.

Method	Rotational error (rad·10^{-3})	Translational error (m·10^{-3})	Time (ms)
SVD	1.4146	0.33677	2.10
Quaternion	1.4146	0.33677	4.40
Wu et al.'s	1.4146	0.33677	5.47
Dual quaternion	1.4146	0.33677	95.61
4D rotation	1.2779	0.23559	1.30

The new method leads to completely different error figures when compared to the other methods. In one particular execution, we obtained the results compiled in Table I. We have chosen this instance because an important conclusion can be drawn from it: all analytical methods used so far are not optimal because it is possible to obtain solutions that are better both in terms of translations and rotations. In this particular case, the determinant of the estimated rotation matrix using the new method is 1.0042 (see Example III for a discussion on orthogonality errors). Nevertheless, it is important to remark that the rotational error has been computed from the corresponding unit quaternion which is not affected by this error.

In all cases, no differences between the registrations obtained using all the analyzed methods are noticeable to the naked eye [see Figure 3(bottom)].

While Wu et al's and the new method simply evaluate some formulas, the other three rely on iterative numerical methods that either compute the SVD or the dominant eigenvector. This has important consequences in the execution times using MATLAB. For example, the SVD implemented in MATLAB calls the xGESVD routine of LAPACK (Linear Algebra Package). Thus, it does not make much sense to compare execution times of functions fully written in interpretable code and others calling optimized compiled routines. Due to this fact, it is even more remarkable the low computational cost of the new method. The time performance of Wu et al.'s method is not as good as that reported by their authors probably because, for the sake of robustness, we had to introduce the computation of all cofactors of a 4×4 matrix as explained in Section II-B.

B. Example II

In the above example, the new method delivered better results both in terms of the rotation and the translation, but this is not the general case. To properly assess the quality of its results, we have executed the following procedure 10^4 times to compare them with those obtained using Wu et al.'s method:

1) Generate a random vector point in $\mathbb{S}^3$ using the algorithm detailed in [44], identify it as a quaternion and convert it to the rotation matrix $\mathbf{R}_i$.
2) Generate a random vector point in $\mathbb{S}^2$, say $\mathbf{p}_i$, using the algorithm also detailed in [44] and set the translation vector $\mathbf{t}_i = r\mathbf{p}_i$.
3) Rotate and translate the reference pointcloud according to $\mathbf{R}_i$ and $\mathbf{t}_i$, respectively.
4) Contaminate the obtained pointcloud with noise as in Example I to get the target pointcloud.
5) Registrate the reference pointcloud with respect to the target pointcloud using the new method and Wu et al.'s method.
6) Compute the rotational and translational error of the registrations.

We have repeated this procedure for different values of r and the results are essentially the same even for values of r as high as 100 (an amount three orders of magnitude higher than the mean side of the enclosing box of the reference pointcloud). The resulting statistics appear in Table II. This table has two separated parts for the rotational and translational errors with three columns each. The first column in each part refers to the attained maximum error; the second one, to the average error; and the third one, to the standard deviation. Observe that the standard deviation is given in microradiands. The maximum obtained orthogonally error for the new method has been 0.0138 (evaluated as $\left|1-\det(\hat{\mathbf{R}})\right|$); its average, 0.00286; and its variance, $4.5{\cdot}10^{-6}$.

We can conclude that the main advantage of the new method is its low computational cost and simplicity. Nevertheless, as already observed in Section IV, the estimated rotation matrix departs from orthogonality under the presence of noise. This latter point is further discussed in the following examples.

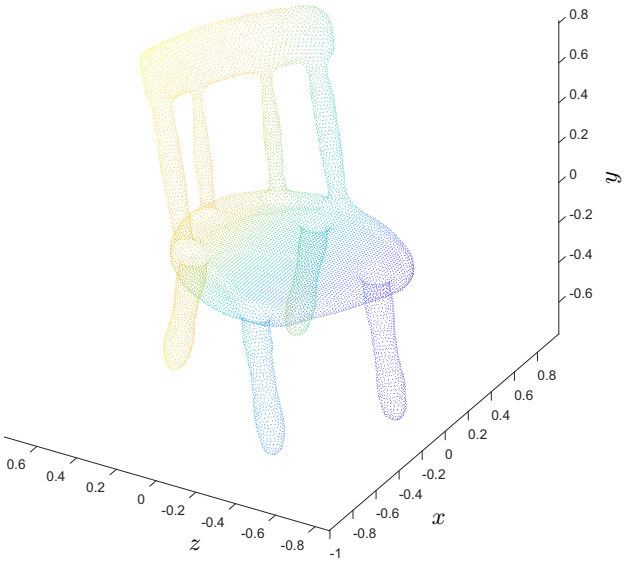


Fig. 4. A pointcloud representing a chair.

C. Example III

To examine the influence of the level of noise in the estimated rotations and translations using the new method, we have used the pointcloud appearing in Fig. 4, which has been taken from the Princeton Segmentation Benchmark [45][1]. The target pointcloud has been obtained by applying the same

[1]freely available at: https://segeval.cs.princeton.edu/.

TABLE II
ERROR STATISTIC FOR 10^4 RANDOM DISPLACEMENTS USING WU ET AL.'S AND THE NEW METHOD.

Method	Maximum rot. error (rad·10^{-3})	Average rot. error (rad·10^{-3})	Standard deviation (rad·10^{-6})	Maximum trans. error (m·10^{-3})	Average trans. error (m·10^{-3})	Standard deviation (m·10^{-6})
Wu et al.'s	3.51	1.07	0.21	0.65	0.19	0.0080
4D rotation	4.72	1.16	0.26	0.87	0.29	0.0015

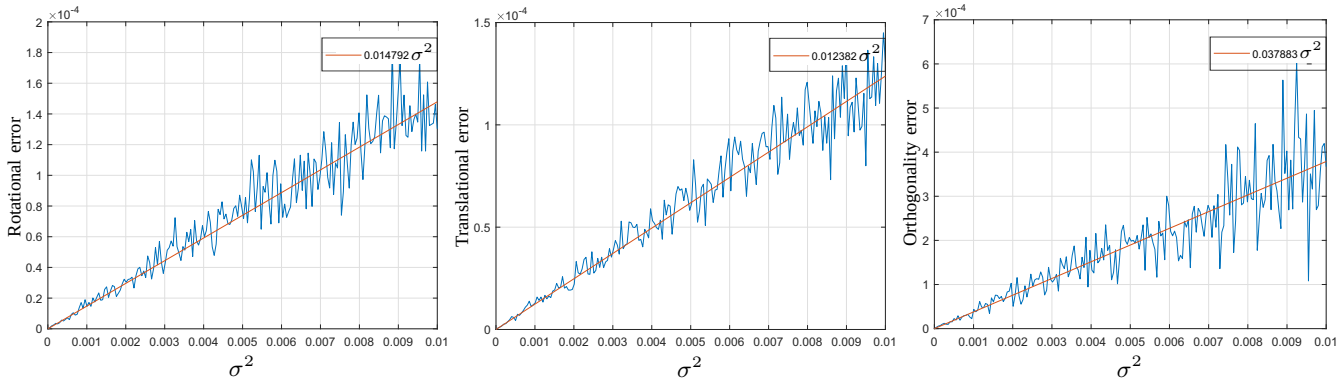
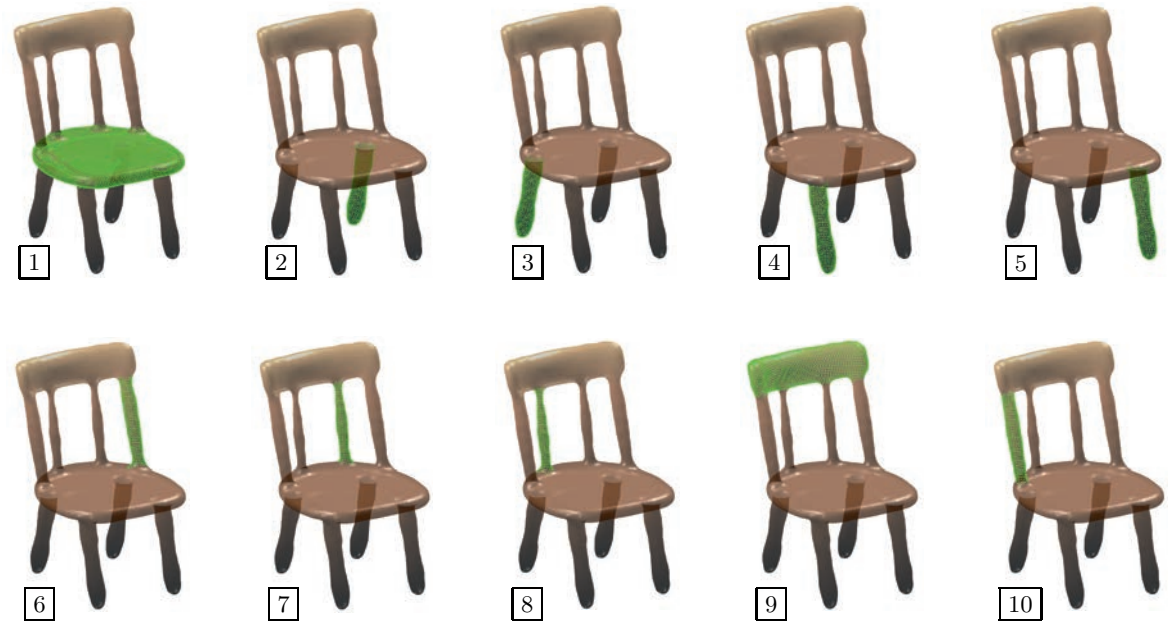


Fig. 5. Rotational, translational and orthogonality errors, as a function of the added noise standard deviation, for the registration of the pointcloud in Fig. 4 using the proposed method.



Point set	1	2	3	4	5	6	7	8	9	10
Number of points	7697	922	934	911	869	700	491	511	2216	722
Eigenvalues	0.0021 0.0707 0.0806	0.0015 0.0016 0.0350	0.0015 0.0017 0.0364	0.0015 0.0016 0.0349	0.0015 0.0016 0.0315	0.0009 0.0010 0.0301	0.0005 0.0005 0.0255	0.0006 0.0006 0.0258	0.0040 0.0071 0.0807	0.0009 0.0011 0.0313
Eccentricity	6.16	4.80	4.89	4.78	4.57	5.67	7.07	6.80	4.48	5.79
Volume·10^3	3.50	0.29	0.30	0.29	0.28	0.17	0.08	0.09	1.50	0.17
c_{rot}	0.0790	0.3948	0.3742	0.3819	0.3966	0.5765	0.9684	0.8586	0.1379	0.5670
c_{trans}	0.0337	0.4467	0.4136	0.6475	0.6221	0.7537	1.0785	1.0733	0.3198	0.7708
c_{ortho}	0.2132	0.9053	1.0077	0.9965	0.9805	1.3817	2.2943	2.2613	0.3114	1.3770

Fig. 6. The chair model in Fig. 4 789++++++++++++++++++++segmented into ten disjoint point sets. The shape characteristics of each set appear in the bottom table. This table also includes the coefficients that indicate the sensitivity of each set to rotational, translational, and orthogonality errors. As expected, the smaller the volume of the point set, the higher the sensitivity to noise.

transformation as in Example I. Then, the coordinates of the resulting points have been perturbed with additive uncorrelated zero-mean Gaussian noise with standard deviation ranging from 0 to 0.1. After recovering the applied rotation and translation using the proposed method, the committed rotational, translational, and orthogonality errors have been evaluated. If this is repeated ten times for each value of the standard derivation of the added noise, and the results averaged, the plots in Fig. 5 are obtained. Using linear regression, these three plots can be approximated by the linear functions $\epsilon_{\text{rot}} = c_{\text{rot}}\,\sigma^2$, $\epsilon_{\text{trans}} = c_{\text{trans}}\,\sigma^2$, and $\epsilon_{\text{ortho}} = c_{\text{ortho}}\,\sigma^2$, where $c_{\text{rot}} = 0.014792$, $c_{\text{trans}} = 0.012382$, and $c_{\text{ortho}} = 0.037883$, respectively. To examine how these three coefficients vary with the shape of the pointcloud, we have segmented the analyzed pointcloud into the ten disjoint point sets appearing in Fig. 6, and the above experiment has been repeated for each of them. A table with the parameters that characterize the shape of each point set (see the Appendix), as well as the obtained values for c_{rot}, c_{trans}, and c_{ortho}, is also included in Fig. 6.

While it is true that, for large levels of noise, some orthogonality errors arise in the proposed method, it is no less true that orthonormalizing a noisy rotation matrix, with the observed *small* orthogonality errors, can be performed by applying the following updating rule:

$$\widehat{\mathbf{R}} \leftarrow \widehat{\mathbf{R}}(3\mathbf{I} + \widehat{\mathbf{R}}^T\widehat{\mathbf{R}})(\mathbf{I} + 3\widehat{\mathbf{R}}^T\widehat{\mathbf{R}})^{-1} \tag{52}$$

One of the nice features of this formula is its cubically convergence to the solution (For details on this fact see, for example, [46], [47]). As we will see in the next section, only one iteration of leads to rotation matrices with negligible orthogonality errors.

D. Example IV

In real-world problems, the correspondence between the points of two pointclouds to be registered is, in general, unknown. Moreover, the number of points in both sets is, in most cases, different. As an example of this kind of problems, consider the two pointclouds represented in Fig. 7 which were obtained with a Cyberware 3030MS optical triangulation scanner by the Stanford Graphics Laboratory. The problem here consists in obtaining the rigid transformation that registers the green pointcloud with respect to the magenta one. Since the correspondence between both point sets is unknown, we have integrated the compared methods in an ICP method. To this end, we have used the built-in Matlab function *pcregistericp* as a reference. It is based on the algorithms presented in [21], [22]. In this function, we have substituted the method that computes the registration for a hypothetical correspondence by each of the methods to be compared. The performance figures obtained by applying 20 iterations using each method appear in Table V-D.

Except for the execution time, the other two performance figures for the method derived in this paper are worse than for the other methods. The reason is that its rate of convergence is lower and hence its lower number of matched points and its higher root mean square error for the distance between the matched points. This is because the generated correspondences during the search process have a very large number

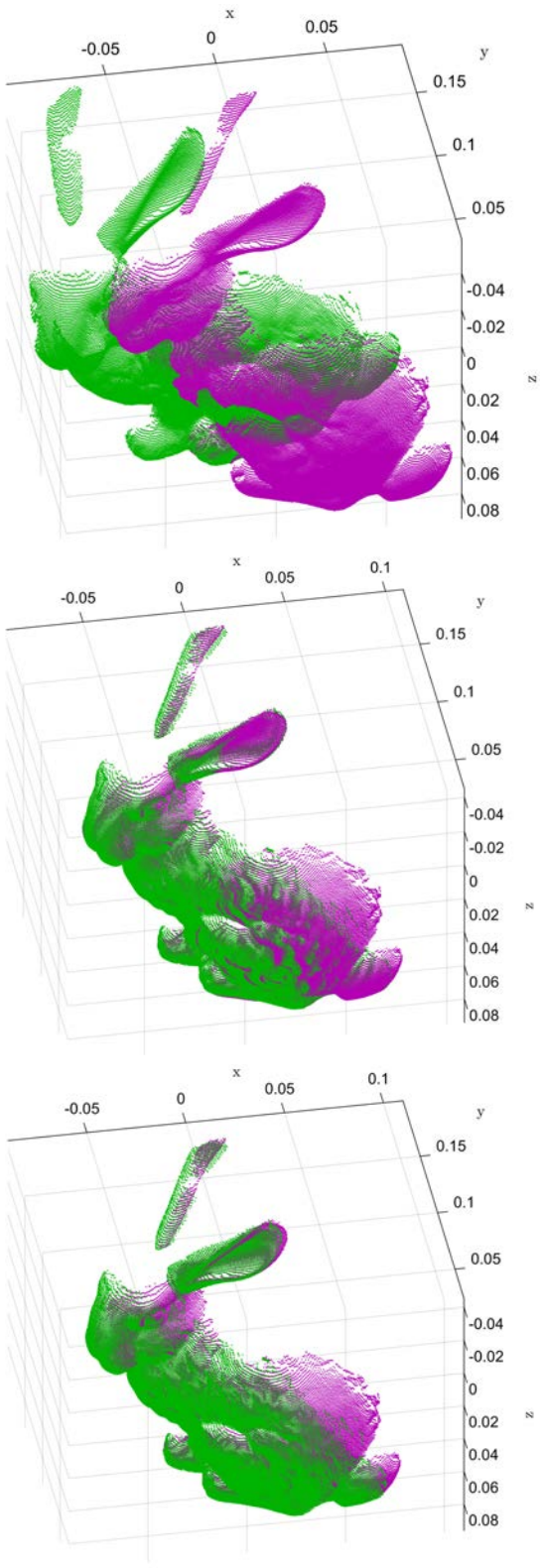


Fig. 7. Top: Two pointclouds corresponding to two partial 3D scans of a model shown in Fig. 2. They contain 40,097 points (the magenta pointcloud) and 40,256 points (the green pointcloud). These pointclouds correspond to files *bun000.ply* and *bun045.ply*, respectively, both freely available from [42]. Middle: Registration result between the two pointclouds using all analyzed methods, except the one derived in this paper. Bottom: Registration result obtained using the method derived in this paper.

TABLE III
PERFORMANCE OF THE DIFFERENT METHODS AT SOLVING THE REGISTRATION PROBLEM IN FIG. 7.

Method	Time (s)	RMS error ($\times 10^{-3}$)	Matched points
SVD	0.0218	3.106	27,508
Quaternion	0.0214	3.106	27,508
Wu et al.'s	0.0656	3.106	27,508
Dual quaternion	22.0840	3.106	27,508
4D rotation	0.0122	3.721	14,444
Refined 4D rotation	0.0127	3.028	26,872

of mismatches which leads to large orthogonality errors in the estimated rotation matrix. If we introduce one single application of the updating rule (V-C) after each estimation, the performance figures change dramatically (see the last row in Table V-D). It is also interesting to observe that a lower number of matched points is not necessarily a bad feature. Actually, all other methods reached the maximum number of matched points at the eighteenth iteration with 27,603 points, and then they start to discard outliers to improve the quality of the match. The results of the registrations appear in Fig. 7(middle) (for all previous approaches), and in Fig. 7(bottom) (for the one derived in this paper). The application of the updating rule V-C to reduce the orthogonality errors obviously increases the computational cost, but the resulting method is still almost twice as faster than the best of the other methods.

VI. CONCLUSION

Inspired by some developments in the area of kinematics, this paper explores the idea of using a characteristic length to solve the pointcloud registration problem. We have used the reciprocal of this length as a symbol that represents a very small number. Therefore, only the terms of lowest degree in this symbol have been kept in all algebraic expressions generated during the resolution of the problem. As a result, a closed-form formula is obtained which has been proved to be an attractive alternative to the previous analytic methods. It is indeed useful for implementation in embedded micro-controllers with limited computational resources because it requires neither square roots nor trigonometric computations. We have shown that this is true even when orthogonality errors have to be reduced due to high levels of noise.

APPENDIX

If a pointcloud, say $\{A_i\}$, is seen as a random set of points in $\mathbb{R}^3$, its covariance matrix is defined as

$$\mathbf{\Sigma}_A = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{a}_i - \mathbf{a})(\mathbf{a}_i - \mathbf{a})^T. \tag{53}$$

where $\mathbf{a}$ is the centroid of the pointcloud.

$\mathbf{\Sigma}_A$ is a positive semi-definite 3×3 matrix, *i.e.*, $\det(\mathbf{\Sigma}) \geq 0$. Then, the set

$$\Xi_A = \left\{\mathbf{x} | \mathbf{x}^T \mathbf{\Sigma}_A \mathbf{x} \leq 1\right\} \tag{54}$$

is an ellipsoid in $\mathbb{R}^3$, centered at the origin. The semi-axes of this ellipsoid are given by $\mathbf{s}_i = \pm\sqrt{\lambda_i}\mathbf{q}_i$, where λ_i is eigenvalue i, $i = 1, 2, 3$, and $\mathbf{q}_i$, its corresponding eigenvector. In other words, eigenvectors determine the directions of the semi-axes and eigenvalues determine their lengths. The axes of ellipsoid Ξ_A are aligned with the principal axes of the pointcloud $\{A\}$. Thus, we can say that Ξ_A captures the spatial distribution of $\{A_i\}$.

We define the following three coefficients associated with a pointcloud

1) *Eccentricity*. The eccentricity of a pointcloud is defined as
$$\varsigma = \sqrt{\lambda_{\max}/\lambda_{\min}}. \tag{55}$$
2) *Volume*. The volume of a pointcloud is defined as
$$\upsilon = \sqrt{\det(\mathbf{\Sigma})} = \sqrt{\lambda_1\,\lambda_2\,\lambda_3}. \tag{56}$$
The larger υ, the greater the pointcloud dispersion. If $\upsilon = 0$, the points lie on a line or a plane.
3) *Intrinsic scale*. The intrinsic scale of a pointcloud is defined as
$$\kappa = \upsilon^{1/3}. \tag{57}$$

REFERENCES

[1] J. Angeles, "Is there a characteristic length of a rigid-body displacement?" *Mechanism and Machine Theory*, vol. 41, no. 8, pp. 884–896, 2006.
[2] D. Huynh, "Metrics for 3D rotations: comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.
[3] P. Larochelle and J. M. McCarthy, "Planar motion synthesis using an approximate bi-invariant metric," *ASME Journal of Mechanical Design*, vol. 117, no. 4, pp. 646–651, 1995.
[4] K. Etzel and J. M. McCarthy, "A metric for spatial displacements using biquaternions on SO(4)." Minneapolis, USA: Proceedings of the IEEE International Conference on Robotics and Automation, 1996, pp. 3158–3190.
[5] F. Thomas, "Approaching dual quaternions from matrix algebra," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1037–1048, 2014.
[6] S. Qiao, Q. Liao, S. Wei, and H.-J. Su, "Inverse kinematic analysis of the general 6R serial manipulators based on double quaternions," *Mechanism and Machine Theory*, vol. 45, pp. 193–199, 2010.
[7] P. M. Larochelle, A. P. Murray, and J. Angeles, "SVD and PD based projection metrics on SE(n)," in *On Advances in Robot Kinematics*, J. Lenarčič and C. Galletti, Eds. Dordrecht: Springer Netherlands, 2004, pp. 13–22.
[8] P. Larochelle and V. Venkataramanujam, "An improved principal coordinate frame for use with spatial rigid body displacement metrics," T. Uhl, Ed., vol. 73, Advances in Mechanism and Machine Science (IFToMM WC 2019). Cham: Springer, 2019, pp. 319–328.
[9] P. Larochelle, "On the geometry of approximate bi-invariant projective displacement metrics." Proceedings of the World Congress on the Theory of Machines and Mechanisms, 1999.
[10] R. Brégier, F. Devernay, L. Leyrit, and J. L. Crowley, "Defining the pose of any 3D rigid object and an associated distance," *International journal of Computer Vision*, vol. 126, pp. 571–596, 2018.
[11] K. Kazerounian and J. Rastegar, "Object norms: a class of coordinate end metric independent norms for displacements," in *Proceedings of the ASME Design Engineering Technical Conferences*, vol. 47. ASME Press, New York, 1992, pp. 271–275.
[12] D. S. Grant, *Cloud to Cloud Registration for 3D Point Data*. PhD dissertation, Purdue University, 2013.
[13] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn, "A survey of rigid 3D pointcloud registration algorithms," *The Fourth International Conference on Ambient Computing, Applications, Services and Technologies (AMBIENT 2014)*, pp. 8–13, 2014.
[14] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.
[15] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, 1987.

[16] B. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.
[17] ——, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, no. 7, pp. 1127–1135, 1988.
[18] M. Walker, L. Shao, and R. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 358–367, 1991.
[19] J. Wu, M. Liu, Z. Zhou, and R. Li, "Fast symbolic 3D registration solution," *IEEE Transactions on Automation Science and Engineering*, to appear, 2020.
[20] B. Bellekens, V. Spruyt, R. Berkvens, R. Penne, and M. Weyn, "A benchmark survey of rigid 3D point cloud registration algorithms," *International Journal on Advances in Intelligent Systems*, vol. 8, no. 1–2, 2015.
[21] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *Proceedings IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2724–2729, 1991.
[22] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.
[23] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.
[24] D. Zhengl, D. Yue, and J. Yue, "Geometric feature constraint based algorithm for building scanning point cloud registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 4, pp. 464–468, 2008.
[25] Z.-C. Lin, T. Huang, and S. Blostein, "Motion estimation from 3-D point sets with and without correspondences." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1986, pp. 94–201.
[26] K. Daniilidis, "Hand-eye calibration using dual quaternions," *International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.
[27] D. Eggert, A. Lorusso, and R. Fisher, "Estimating 3-D rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, no. 5, pp. 272–290, 1997.
[28] "Deep learning based point cloud registration: an overview," *Virtual Reality & Intelligent Hardware*, vol. 2, no. 3, pp. 222–246, 2020.
[29] I. Bar-Itzhack, "Iterative optimal orthogonalization of the strapdown matrix," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-11, no. 1, pp. 30–37, 1975.
[41] S. Sarabandi, "Solving the nearest rotation matrix problem in three and four dimensions with applications in robotics," Ph.D. dissertation, Technical University of Catalonia, Barcelona, Spain, 2021.
[30] S. Sarabandi, A. Shabani, J. Porta, and F. Thomas, "On closed-form formulas for the 3D nearest rotation matrix problem," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1333–1339, 2020.
[31] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica*, vol. A32, pp. 922–923, 1976.
[32] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.
[33] K. Spring, "Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: a review," *Mechanism and Machine Theory*, vol. 21, no. 5, pp. 365–373, 1986.
[34] Y. Bar-Itzhack, "New method for extracting the quaternion from a rotation matrix," *Journal of Guidance Control and Dynamics*, vol. 23, no. 6, pp. 1085–1087, 2000.
[35] N. Higham and V. Noferini, "An algorithm to compute the polar decomposition of a 3x3 matrix," *Numerical Algorithms*, vol. 73, no. 2, pp. 349–369, 2016.
[36] M. Chasles, "Note sur les propriétés générales du système de deux corps semblables entr'eux," *Bulletin des Sciences Mathématiques, Astronomiques, Physiques et Chemiques*, no. 14, pp. 321–326, 1830.
[37] C. Wang, Y. Cho, and J. Park, "Performance tests for automatic 3D geometric data registration technique for progressive as-built construction site modeling." Orlando, Fl, USA: 2014 International Conference on Computing in Civil and Building Engineering, 2014, pp. 1053–1061.
[38] F. Cole, "On rotations in space of four dimensions," *American Journal of Mathematics*, vol. 12, no. 2, pp. 191–210, 1890.
[39] H. Gains, "Attitude matrix orthonormality correction," *Honeywell Interoffice Correspondence*, 1965.
[40] A. Lewis and J. Malick, "Alternating projections on manifolds," *Mathematics of Operations Research*, vol. 33, no. 1, pp. 216–234, 2008.
[42] The Stanford 3D scanning repository. [Online]. Available: http://www.graphics.stanford.edu/data/3Dscanrep/
[43] P. Wunsch, S. Winkler, and G. Hirzinger, "Real-time pose estimation of 3D objects from camera images using neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, 1997, pp. 3232–3237.
[44] G. Marsaglia, "Choosing a point from the surface of a sphere," *Annals of Mathematical Statistics*, vol. 43, pp. 645–646, 1972.
[45] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3D mesh segmentation," *ACM Transactions on Graphics*, vol. 28, no. 3, article 73, 2009.
[46] N. J. Higham, "Newton's method for the matrix square root," *Mathematics of Computation*, vol. 46, no. 174, pp. 537–550, 1986.
[47] S. Sarabandi and F. Thomas, "Solution methods to the nearest rotation matrix problem in R^3: A comparative survey," *submitted for publication*.