CVGIP: IMAGE UNDERSTANDING
Vol. 54, No. 3, November, pp. 358–367, 1991

Estimating 3-D Location Parameters Using Dual Number Quaternions

MICHAEL W. WALKER AND LEJUN SHAO

Artificial Intelligence Laboratory, EECS Department, University of Michigan, Ann Arbor, Michigan 48109

AND

RICHARD A. VOLZ

Department of Computer Science, Texas A&M University, College Station, Texas 77843

Received May 24, 1989; accepted August 22, 1990

This paper describes a new algorithm for estimating the position and orientation of objects. The problem is formulated as an optimization problem using dual number quaternions. The advantage of using this representation is that the method solves for the location estimate by minimizing a single cost function associated with the sum of the orientation and position errors and thus is expected to have a better performance on the estimation, both in accuracy and in speed. Several forms of sensory information can be used by the algorithm. That is, the measured data can be a combination of measured points on an object's surfaces and measured unit direction vectors located on the object. Simulations have been carried out on a Compaq 386/20 computer and the simulation results are analyzed. © 1991 Academic Press, Inc.

1. INTRODUCTION

Object location parameter estimation is an important part of computer vision tasks, which usually leads to the computation of a 4×4 homogeneous transformation matrix $\boldsymbol{T}$ between the object coordinate frame and a reference coordinate frame,

$$\boldsymbol{T} = \begin{bmatrix} & \boldsymbol{R} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where the matrix $\boldsymbol{R}$ is a 3×3 rotation matrix which specifies the orientation of the object and the $\mathbf{t}$ is a 3×1 translation vector which specifies the position.

Techniques which combine redundant sensed features, whether or not they are of the same type, to determine the object location can improve the accuracy of localization. Least squares optimization techniques are frequently used to find the best estimate of the transformation matrix from those redundant features. Two approaches for using optimization techniques to find the best estimate of the transformation matrix have been introduced [1, 3, 6]. In one method, an optimal orientation of the object is determined first, which is then used as a basis to find the position of the object. That is, the translation vector is a function of an optimal rotation matrix $\boldsymbol{R}$ and other measured quantities. Many object localization algorithms use this method [1, 4–6, 9, 11]. The problem with this approach is the possibility of the existence of accumulated errors in calculating the translation vector due to errors from previous calculations and measurements. For example, in [1]'s SVD algorithm the translation vector $\mathbf{t}$ is computed from $\boldsymbol{R}$, $\tilde{\mathbf{p}}_i$, $\mathbf{p}_i$, e.g., $\mathbf{t} = f(\boldsymbol{R}, \tilde{\mathbf{p}}_i, \mathbf{p}_i)$, where $\tilde{\mathbf{p}}_i$ and $\mathbf{p}_i$ are sets of measured points and corresponding modeled points in 3-D space respectively. Because both $\tilde{\mathbf{p}}_i$ and $\boldsymbol{R}$ have errors, the error of the resulting translation vector will be compounded due to error propagation. The second approach is to compute two optimal solutions separately, one for the orientation and another for the position [3], which is not very efficient. The common characteristics of these two methods are that both approach the problems of determination of orientation and position separately. That is not surprising, because the transformation matrix itself can be easily decomposed into two parts: a rotation submatrix and a position vector. The difference between these two approaches lies in the way the optimization is done: the first optimizes only the rotational part of the homogeneous transformation matrix and the translational part is then derived from it, while the second method optimizes both rotational and translational parts separately.

In this paper, we present an efficient algorithm which is based on the use of dual number quaternions [2]. The method solves for the orientation and the position of an object by minimizing a single cost function associated with the sum of the orientation and position errors. The performance, both in accuracy and in speed, compared with that of the previous methods will be discussed. The required input data for the algorithm is a combination of

1049-9660/91 $3.00

measured points on the surfaces of an object, measured unit direction vectors from that object, and their corresponding modeled features. Examples of point features might be any combinations of the corner points (vertices) of an object, masked points, or the center of a sphere. Examples of unit vector features include the surface normals, edge direction vectors, the axis direction vectors, or surface normals.

A brief description of the concept and properties of dual numbers is given in Appendix I. More detailed discussion can be found in [12, 13]. In the following sections, we begin with the introduction of the definition of dual number quaternions. We show how they are used to represent object location, why they are a valid representation of location, and their correspondence with the more familiar homogeneous transforms. Then we give a brief description of the important properties of the dual number quaternions. Next, we formulate the object localization problem as a dual number quaternion optimization problem and an algorithm is derived to solve the problem. Simulation results are shown in section 4.

2. DUAL NUMBER QUATERNIONS

This section begins with the definition of dual number quaternions, their properties, and their physical interpretation. It concludes by showing how to convert back and forth from the dual number quaternion representation of location to the homogeneous transformation representation.

Table 1 lists and defines the symbols used in this paper.

2.1. *Properties of Dual Number Quaternions*

Quaternions are four-element vectors, which are thought of as consisting of a 3×1 vector component and a scalar component. For example, the quaternion $\boldsymbol{q}$ is

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \mathbf{q} \\ q_4 \end{bmatrix}. \tag{2}$$

In our notation, each quaternion is represented by a boldface italic character, such as $\boldsymbol{q}$; and each 3×1 vector is represented by a boldface roman character, such as $\mathbf{q}$.

Quaternions have been used extensively as a method of parameterizing orientation [7, 8, 10].

In this application, the components of the quaternion have the following interpretation:

$$\boldsymbol{q} = \begin{bmatrix} \sin(\theta/2)\mathbf{n} \\ \cos(\theta/2) \end{bmatrix}. \tag{3}$$

TABLE 1
A List of Symbols Appearing in This Paper

Symbol(s)	Description
Quaternion: $\boldsymbol{q}, \boldsymbol{e}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{n}, \boldsymbol{p}, \boldsymbol{n}_i, \boldsymbol{p}_i, \boldsymbol{n}_i^0, \boldsymbol{p}_i^0, \tilde{\boldsymbol{n}}_i, \tilde{\boldsymbol{p}}_i$	
$\boldsymbol{e}$	unit quaternion
$\boldsymbol{t}$	translation quaternion
$\boldsymbol{r}$	real part of a dual quaternion
$\boldsymbol{s}$	dual part of a dual quaternion
$\boldsymbol{n}$	direction quaternion
$\boldsymbol{p}$	position quaternion
$\boldsymbol{n}_i^0$	modeled direction quaternion
$\boldsymbol{p}_i^0$	modeled position quaternion
$\boldsymbol{n}_i$	transformed model direction quaternion
$\boldsymbol{p}_i$	transformed model position quaternion
$\tilde{\boldsymbol{n}}_i$	measured direction quaternion
$\tilde{\boldsymbol{p}}_i$	measured position quaternion
Vector: $\mathbf{t}, \mathbf{q}, \mathbf{a}, \mathbf{n}, \mathbf{p}, \mathbf{r}, \mathbf{p}_i, \mathbf{p}_i^0, \tilde{\mathbf{p}}_i, \mathbf{n}_i^0$	
$\mathbf{t}$	translation vector
$\mathbf{n}$	rotation axis unit direction vector
$\mathbf{p}$	position vector
$\mathbf{p}_i^0$	modeled position vector
$\tilde{\mathbf{p}}_i$	measured position vector
$\mathbf{p}_i$	transformed model position vector
$\mathbf{n}_i^0$	modeled direction vector
4×4 matrix: $\boldsymbol{T}, \boldsymbol{I}, \boldsymbol{A}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{Q}, \boldsymbol{W}$	
$\boldsymbol{T}$	homogeneous transformation matrix
$\boldsymbol{QW}$	quaternion matrices
3×3 matrix: $\boldsymbol{R}, \boldsymbol{K}$	
$\boldsymbol{R}$	rotation matrix
$\boldsymbol{K}$	skew-symmetric matrix
Scalar: $\theta, d, e, \lambda_1, \lambda_2, \alpha_i, \beta_i$	
θ	rotation angle
d	distance between two vectors
e	errors from least squares optimization
λ_1, λ_2	Lagrange multipliers
α_i, β_i	weighting factors
Dual quantities: $\hat{q}, \hat{\mathbf{q}}, \hat{\mathbf{n}}, \hat{\theta}$	
$\hat{q}$	dual quaternion
$\hat{\mathbf{n}}$	dual vector of rotation
$\hat{\theta}$	dual angle of rotation

The components of this quaternion are called the Euler Symmetric Parameters. As illustrated in Fig. 1, the vector $\mathbf{n}$ is the unit vector about which the coordinate system has rotated and θ is the amount of rotation about $\mathbf{n}$. The corresponding rotation matrix $\boldsymbol{R}$ can be expressed as

$$\boldsymbol{R} = (q_4^2 - \mathbf{q}^T\mathbf{q})\boldsymbol{I} + 2\mathbf{q}\mathbf{q}^T + 2q_4\boldsymbol{K}(\mathbf{q}), \tag{4}$$

where $\boldsymbol{K}$ is the skew-symmetric matrix

$$\boldsymbol{K}(\mathbf{q}) = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}. \tag{5}$$

The extension of this equation to include the representation of position and orientation is made by simply

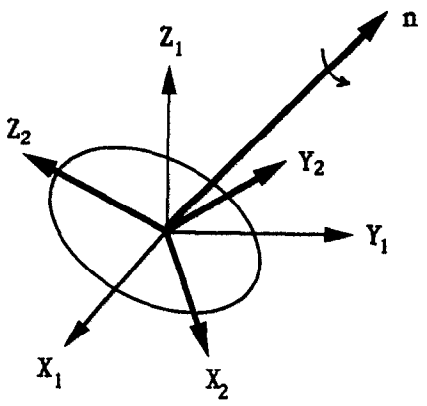

FIG. 1. Illustration of rotation for a real quaternion.

changing all of the quantities in the equation to dual quantities [2, 14]:

$$\hat{q} = \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \hat{q}_3 \\ \hat{q}_4 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{q}} \\ \hat{q}_4 \end{bmatrix}. \tag{6}$$

Note that whether an item is a 3×1 vector or a quaternion, if its components are dual numbers, this is signified by placing a hat over it as in the above example. A brief description of the concept of dual numbers and their important properties can be found in Section 1 of the Appendix.

There are two parts of a dual quaternion,

$$\hat{q} = r + \varepsilon s, \tag{7}$$

where r and s are both real quaternions and are called the real part and the dual part, respectively.

The dual quaternions have a similar interpretation as the real quaternion,

$$\hat{q} = \begin{bmatrix} \sin(\hat{\theta}/2)\hat{\mathbf{n}} \\ \cos(\hat{\theta}/2) \end{bmatrix}, \tag{8}$$

where the dual vector $\hat{\mathbf{n}}$ represents a line in 3-D space about which the coordinate system has rotated and translated and $\hat{\theta}$ is the dual angle of rotation and translation. The dual vector $\hat{\mathbf{n}}$ and dual angle $\hat{\theta}$ are

$$\hat{\mathbf{n}} = \mathbf{n} + \varepsilon \mathbf{p} \times \mathbf{n} \tag{9}$$

and

$$\hat{\theta} = \theta + \varepsilon d, \tag{10}$$

where $\mathbf{n}$ is a unit vector which specifies the direction of the rotation axis and also the direction of translation; the rotation is about the line having direction $\mathbf{n}$ passing through the point $\mathbf{p}$ with a rotation angle of θ; and d is the distance of translation along the direction specified by $\mathbf{n}$ (see the Appendix for the discussion of $\mathbf{n}$ and $\mathbf{p}$). The geometrical intepretation of the representation can be explained as follows:

> Traditionally, the transformation of a coordinate frame is specified by a translation vector $\mathbf{t}$, a rotation axis $\mathbf{n}$, and a rotation angle θ. A new coordinate frame is formed by first translating the original coordinate frame along $\mathbf{t}$ and then rotating it with respect to $\mathbf{n}$ by an angle θ. Of course, the sequence of translation and rotation can be reversed.
>
> With dual quaternion representation, the same transformation can be formed by first translating the original coordinate frame along the direction of $\mathbf{n}$ by a distance of d and then rotating it by an angle of θ with respect to a line having a unit vector $\mathbf{n}$ as its direction and passing through a point $\mathbf{p}$.

See Fig. 2 for an illustration of the interpretation. It can be proven that for each ($\mathbf{n}$, $\mathbf{p}$, d, θ) transformation representation, we can always find a unique corresponding ($\mathbf{t}$, $\mathbf{n}$, θ). On the other hand, for each ($\mathbf{t}$, $\mathbf{n}$, θ) transformation representation, there exists a set of corresponding ($\mathbf{n}$, $\mathbf{p}$, d, θ)'s. (See Section 2.2 and the Appendix for a detailed description).

If we place Eqs. (9) and (10), e.g., expressions for $\hat{\mathbf{n}}$ and $\hat{\theta}$, into Eq. (8), expand and simplify that equation by using the properties of dual numbers, and compare the results with Eq. (7), we have the following equations:

$$r = \begin{bmatrix} \sin(\theta/2)\mathbf{n} \\ \cos(\theta/2) \end{bmatrix} \tag{11}$$

and

$$s = \begin{bmatrix} (d/2)\cos(\theta/2)\mathbf{n} + \sin(\theta/2)(\mathbf{p} \times \mathbf{n}) \\ -(d/2)\sin(\theta/2) \end{bmatrix}. \tag{12}$$

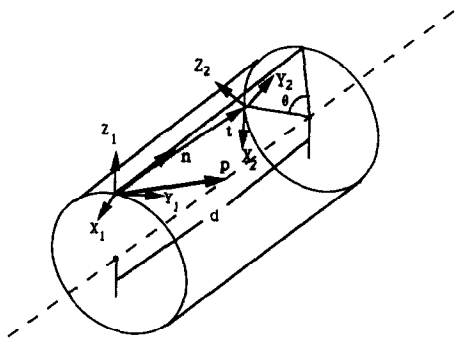

FIG. 2. Illustration of rotation and translation for a dual quaternion.

TABLE 2
Properties of Quaternion Matrices

$$\boldsymbol{Q}(a)^T\boldsymbol{Q}(\mathbf{a}) = \boldsymbol{Q}(a)\boldsymbol{Q}(a)^T = a^Ta\boldsymbol{I}$$

$$\boldsymbol{W}(a)^T\boldsymbol{W}(\mathbf{a}) = \boldsymbol{W}(a)\boldsymbol{W}(a)^T = a^Ta\boldsymbol{I}$$

$$\boldsymbol{Q}(a)b = \boldsymbol{W}(b)a$$

$$\boldsymbol{Q}(a)^Ta = \boldsymbol{W}(a)^Ta = (a^Ta)e$$

$$\boldsymbol{Q}(\boldsymbol{Q}(a)b) = \boldsymbol{Q}(a)\boldsymbol{Q}(b)$$

$$\boldsymbol{W}(\boldsymbol{W}(a)b) = \boldsymbol{W}(a)\boldsymbol{W}(b)$$

$$\boldsymbol{Q}(a)\boldsymbol{W}(b)^T = \boldsymbol{W}(b)^T\boldsymbol{Q}(a)$$

a and b are arbitrary quaternions

e is the unit quaternion $= [0001]^T$

A dual quaternion has eight elements, whereas the minimum number of independent variables to represent a 3-D object transformation is six, which means that two of the eight elements in dual quaternion representation are not independent. In fact, it can be shown from Eqs. (11) and (12) that the components of any dual quaternion, if they are defined by Eqs. (8)–(10), satisfy the following two constraints:

$$r^Tr = 1 \tag{13}$$

$$r^Ts = 0. \tag{14}$$

Two important matrix functions of quaternions are the matrices $\boldsymbol{Q}(r)$ and $\boldsymbol{W}(r)$, which are defined as

$$\boldsymbol{Q}(r) = \begin{bmatrix} r_4\boldsymbol{I} + \boldsymbol{K}(\mathbf{r}) & \mathbf{r} \\ -\mathbf{r}^T & r_4 \end{bmatrix} \tag{15}$$

$$\boldsymbol{W}(r) = \begin{bmatrix} r_4\boldsymbol{I} - \boldsymbol{K}(\mathbf{r}) & \mathbf{r} \\ -\mathbf{r}^T & r_4 \end{bmatrix}, \tag{16}$$

where $\boldsymbol{K}(\mathbf{r})$ is the skew-symmetric matrix as defined in Eq. (5).

Useful properties of the $\boldsymbol{Q}$ and $\boldsymbol{W}$ matrices which are utilized in the derivation of the localization algorithm are given in Table 2. All these properties can easily be verified by direct substitutions.

2.2. *Relation to Homogeneous Transforms*

A common method of representing the position and orientation of a coordinate system is with homogeneous transforms. Since homogeneous transforms are more common in use than dual number quaternions, the following is provided as a reference to show how to convert from one to the other.

2.2.1. *Computing the Homogeneous Transform Given the Dual Quaternion*

Equation (11) shows that the real part r of the dual quaternion has exactly the same form as that defined in Eq. (3). As a result, the rotation matrix $\boldsymbol{R}$ can be written in terms of the components of the dual quaternion in the familiar way

$$\boldsymbol{R} = (r_4^2 - \mathbf{r}^T\mathbf{r})\boldsymbol{I} + 2\mathbf{r}\mathbf{r}^T + 2r_4\boldsymbol{K}(\mathbf{r}) \tag{17}$$

or

$$\begin{bmatrix} \boldsymbol{R} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} = \boldsymbol{W}(r)^T\boldsymbol{Q}(r). \tag{18}$$

The position vector can be written in terms of the components of the dual quaternion (see Section 2 of the Appendix for the detailed derivation) as

$$t = \boldsymbol{W}(r)^Ts, \tag{19}$$

where t is the translation quaternion for the translation vector $\mathbf{t}$ and is defined as

$$t = \frac{1}{2}\begin{bmatrix} \mathbf{t} \\ 0 \end{bmatrix}. \tag{20}$$

2.2.2. *Computing the Dual Quaternion Given the Homogeneous Transform*

Given a homogeneous transform $\boldsymbol{T}$ specified by a rotation matrix $\boldsymbol{R}$ and a translation vector $\mathbf{t}$, one can compute the corresponding r and s.

$$r_4 = (1/2)\sqrt{R_{11} + R_{22} + R_{33} + 1}, \tag{21}$$

where R_{ij} denotes the ijth element of the matrix $\boldsymbol{R}$. A value of r_4 equal to zero represents a rotation of 180 degrees. If r_4 is not zero, then

$$\mathbf{r} = \frac{1}{4r_4}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}. \tag{22}$$

If r_4 is zero, then

$$\mathbf{r}\mathbf{r}^T = (1/2)(\boldsymbol{R} + \boldsymbol{I}). \tag{23}$$

So r can be determined from any nonzero column of $(1/2)(\boldsymbol{R} + \boldsymbol{I})$, call it $\mathbf{a}$. Thus,

$$\mathbf{r} = \pm \frac{\mathbf{a}}{\|\mathbf{a}\|}. \tag{24}$$

Either sign will work since the rotation angle is 180 degrees.

r having been determined, the value of s can be computed from Eq. (19) as

$$s = W(r)t. \tag{25}$$

Thus a homogeneous transformation matrix and a corresponding dual quaternion can be converted from one to another.

3. PROBLEM FORMULATION AND SOLUTION

This section begins with the formulation of the problem as an optimization problem. The following section presents the solution to the problem.

3.1. *Problem Formulation*

As we have mentioned, two types of sensor measurements are considered: the position of points on an object and the unit vector on the object such as the unit normal, edge direction vector, etc. To facilitate the analysis we define quaternion representations of these quantities. Let $\mathbf{p}$ be the position vector of a point on the object surface. We define the position quaternion as

$$p = \frac{1}{2}\begin{bmatrix}\mathbf{p}\\ 0\end{bmatrix}. \tag{26}$$

Let $\mathbf{n}$ be a unit vector extracted from the object. We define the direction quaternion as

$$n = \begin{bmatrix}\mathbf{n}\\ 0\end{bmatrix}. \tag{27}$$

To determine the position and orientation of an object, we make measurements of k unit vectors for which we have a correspondence to the models and store them in the direction quaternions, $\tilde{n}_i$. The tilde denotes measured values. Similarly, we make measurements of l points on the object and store them in the position quaternions, $\tilde{p}_i$.

Corresponding to each measured point $\tilde{p}_i$, there is a database description of that point $\mathbf{p}_i^0$ which is described with respect to the object coordinate system. If $\mathbf{t}$ and $\boldsymbol{R}$ are the translational and rotational parts of the transformation matrix which is to be determined, the modeled point will be transformed into position $\mathbf{p}_i$,

$$\mathbf{p}_i = \mathbf{t} + \boldsymbol{R}\mathbf{p}_i^0. \tag{28}$$

If these modeled points are represented by position quaternions p_i^0 and p_i and a dual quaternion is used to represent the transformation parameters, from Eqs. (18) and (19), Eq. (28) will become

$$p_i = W(r)^T s + W(r)^T Q(r) p_i^0. \tag{29}$$

For the same reason, for the modeled direction quaternions n_i and n_i^0, we have the relation

$$n_i = W(r)^T Q(r) n_i^0. \tag{30}$$

The approach for computing the position and orientation of the object is to determine r and s which minimize the error between the $\tilde{p}_i$ and p_i and the $\tilde{n}_i$ and n_i. That is, we select the r and s to minimize the error function

$$E = \sum_{i=1}^{k} \alpha_i (n_i - \tilde{n}_i)^2 + \sum_{i=1}^{l} \beta_i (p_i - \tilde{p}_i)^2, \tag{31}$$

where the α_i and β_i are constant positive weighting factors.

We consider each of these terms individually:

$$(n_i - \tilde{n}_i)^2 = 2(1 - r^T Q(\tilde{n}_i)^T W(n_i^0) r) \tag{32}$$

$$(p_i - \tilde{p}_i)^2 = s^T s + 2s^T (W(p_i^0) - Q(\tilde{p}_i)) r - 2r^T Q(\tilde{p}_i)^T W(p_i^0) r + ((p_i^0)^T p_i^0 + \tilde{p}_i^T \tilde{p}_i). \tag{33}$$

Thus, the error function can be written as a quadratic function of r and s,

$$E = r^T C_1 r + s^T C_2 s + s^T C_3 r + \text{constant}, \tag{34}$$

where

$$C_1 = -2 \sum_{i=1}^{k} \alpha_i Q(\tilde{n}_i)^T W(n_i^0) - 2 \sum_{i=1}^{l} \beta_i Q(\tilde{p}_i)^T W(p_i^0) \tag{35}$$

$$C_2 = \left(\sum_{i=1}^{l} \beta_i\right) I \tag{36}$$

$$C_3 = 2 \sum_{i=1}^{l} \beta_i (W(p_i^0) - Q(\tilde{p}_i)) \tag{37}$$

$$\text{constant} = 2 \sum_{i=1}^{k} \alpha_i + \sum_{i=1}^{l} \beta_i ((p_i^0)^T p_i^0 + \tilde{p}_i^T \tilde{p}_i). \tag{38}$$

We compute $\boldsymbol{r}$ and $\boldsymbol{s}$ to minimize this error function subject to the constraints

$$\boldsymbol{r}^T\boldsymbol{r} = 1 \tag{39}$$

$$\boldsymbol{s}^T\boldsymbol{r} = 0 \tag{40}$$

3.2. *Problem Solution*

The optimal dual number location quaternion is obtained by adjoining the constraint equations to the error equation and then minimizing the resulting function without constraints,

$$\tilde{E} = \boldsymbol{r}^T\boldsymbol{C}_1\boldsymbol{r} + \boldsymbol{s}^T\boldsymbol{C}_2\boldsymbol{s} + \boldsymbol{s}^T\boldsymbol{C}_3\boldsymbol{r} + \text{constant} + \lambda_1(\boldsymbol{r}^T\boldsymbol{r} - 1) + \lambda_2(\boldsymbol{s}^T\boldsymbol{r}),$$

where λ_1 and λ_2 are Lagrange multipliers. Taking the partial derivatives gives

$$\frac{\partial\tilde{E}}{\partial r} = (\boldsymbol{C}_1 + \boldsymbol{C}_1^T)\boldsymbol{r} + \boldsymbol{C}_3^T\boldsymbol{s} + 2\lambda_1\boldsymbol{r} + \lambda_2\boldsymbol{s} = 0 \tag{41}$$

$$\frac{\partial\tilde{E}}{\partial s} = (\boldsymbol{C}_2 + \boldsymbol{C}_2^T)\boldsymbol{s} + \boldsymbol{C}_3\boldsymbol{r} + \lambda_2\boldsymbol{r} = 0. \tag{42}$$

Thus, the solution of Eqs. (39), (40), (41), and (42) for $\boldsymbol{r}$ and $\boldsymbol{s}$ gives the optimal solution for the position and orientation of the object.

To solve these equations, we begin by solving for λ_2. Multiplying Eq. (42) by $\boldsymbol{r}$ and solving for λ_2 gives

$$\lambda_2 = -\boldsymbol{r}^T\boldsymbol{C}_3\boldsymbol{r}. \tag{43}$$

Since $\boldsymbol{C}_3$ is skew symmetric,

$$\lambda_2 = 0. \tag{44}$$

We can now solve for $\boldsymbol{s}$ as a function of $\boldsymbol{r}$ from Eq. (42),

$$\boldsymbol{s} = -(\boldsymbol{C}_2 + \boldsymbol{C}_2^T)^{-1}\boldsymbol{C}_3\boldsymbol{r}. \tag{45}$$

Substituting Eqs. (44) and (45) into Eq. (41) gives

$$\boldsymbol{A}\boldsymbol{r} = \lambda_1\boldsymbol{r}, \tag{46}$$

where

$$\boldsymbol{A} = \frac{1}{2}(\boldsymbol{C}_3^T(\boldsymbol{C}_2 + \boldsymbol{C}_2^T)^{-1}\boldsymbol{C}_3 - \boldsymbol{C}_1 - \boldsymbol{C}_1^T). \tag{47}$$

Thus, the quaternion $\boldsymbol{r}$ is an eigenvector of the matrix $\boldsymbol{A}$ and λ_1 is the corresponding eigenvalue. In general there will be four solutions to this equation. Since $\boldsymbol{A}$ is real and symmetric, all of the eigenvalues and eigenvectors are real and the eigenvectors will be orthogonal. The desired solution is identified by referring back to the original error equation (34).

Multiplying Eq. (41) by $\boldsymbol{r}^T$ gives

$$\boldsymbol{r}^T\boldsymbol{C}_1\boldsymbol{r} = (1/2)\boldsymbol{r}^T(\boldsymbol{C}_1 + \boldsymbol{C}_1^T)\boldsymbol{r} = -(1/2)\boldsymbol{s}^T\boldsymbol{C}_3\boldsymbol{r} - \lambda_1. \tag{48}$$

Multiplying Eq. (42) by $\boldsymbol{s}^T$ gives

$$\boldsymbol{s}^T\boldsymbol{C}_2\boldsymbol{s} = (1/2)\boldsymbol{s}^T(\boldsymbol{C}_2 + \boldsymbol{C}_2^T)\boldsymbol{s} = -(1/2)\boldsymbol{s}^T\boldsymbol{C}_3\boldsymbol{r}. \tag{49}$$

Substituting these into Eq. (34) gives

$$E = \text{constant} - \lambda_1. \tag{50}$$

Thus, the error is minimized if we select the eigenvector corresponding to the largest positive eigenvalue.

Having computed $\boldsymbol{r}$, we can now substitute back into Eq. (45) to obtain $\boldsymbol{s}$ to complete the solution for the position and orientation of the object.

To give a clearer picture of the above derivation process, in the following the optimal dual number quaternion localization algorithm (DQ algorithm) will be summarized.

From the algorithm we can see that the execution times for steps 2–4 are basically constant and the execution time for step 1 has a linear relationship to the number of measured vectors. Therefore, the algorithm is an $O(n)$ algorithm in time complexity.

DQ Localization Algorithm.

Inputs: a set of k measured points $\tilde{\boldsymbol{p}}_i^0$, l measured unit vectors $\tilde{\boldsymbol{n}}_i^0$; the corresponding modeled points $\boldsymbol{p}_i$ and vectors $\boldsymbol{n}_i$, as well as weighting factors α_i and β_i chosen heuristically to reflect the reliability of the data points.

Output: an estimate of the transformation matrix **T**.

Step 1. Compute matrices $\boldsymbol{C}_1$, $\boldsymbol{C}_2$ and $\boldsymbol{C}_3$:

$$\boldsymbol{C}_1 = -2\sum_{i=1}^{k}\alpha_i\boldsymbol{Q}(\tilde{\boldsymbol{n}}_i)^T\boldsymbol{W}(\boldsymbol{n}_i^0) - 2\sum_{i=1}^{l}\beta_i\boldsymbol{Q}(\tilde{\boldsymbol{p}}_i)^T\boldsymbol{W}(\boldsymbol{p}_i^0)$$

$$\boldsymbol{C}_2 = \left(\sum_{i=1}^{l}\beta_i\right)\boldsymbol{I}$$

$$\boldsymbol{C}_3 = 2\sum_{i=1}^{l}\beta_i(\boldsymbol{W}(\boldsymbol{p}_i^0) - \boldsymbol{Q}(\tilde{\boldsymbol{p}}_i)).$$

Step 2. Compute the 4×4 symmetric matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = (1/2)(\boldsymbol{C}_3^T(\boldsymbol{C}_2 + \boldsymbol{C}_2^T)^{-1}\boldsymbol{C}_3 - \boldsymbol{C}_1 - \boldsymbol{C}_1^T).$$

Step 3. Compute the eigenvector $\boldsymbol{r}$ corresponding to the largest positive eigenvalue of matrix $\boldsymbol{A}$ and derive $\boldsymbol{s}$ from $\boldsymbol{r}$.

Step 4. Compute the matrix **T** from $\boldsymbol{s}$ and $\boldsymbol{r}$.

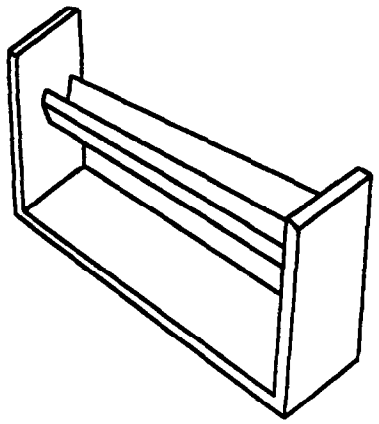

FIG. 3. The object "Feeder."

4. SIMULATION RESULTS

To examine the performance of the DQ algorithm, the accuracy and the speed, computer simulations have been carried out on a Compaq 386/20 with an 80387-20 math processor. The simulation data are from a modeled "Feeder" (see Fig. 3) which has 32 vertices, 50 surfaces and 48 edges. The size of the Feeder is 322 × 84 × 151 units. The SVD algorithm is selected as a sample algorithm to compare its accuracy and performance with our DQ algorithm.

Because the SVD algorithm accepts only 3-D points as its inputs, the sole inputs for the two algorithms are the sampled points in order to have a fair comparison of their accuracy. The algorithms were tested using 5, 10, 20, and 30 points as input data. Each of these tests was repeated for 25 different choices of points, e.g., 25 different sets of 5, 10, 20, and 30 points were run, and each of these was run 20 times with random errors (see discussion below) added to the sample values. In our simulation, the required number of 3-D points $\boldsymbol{p}_i^0$ are randomly selected from a Feeder's vertices at the beginning of each trial. The corresponding measured points $\tilde{\boldsymbol{p}}_i$ are then generated by first rotating an angle of 36° around an axis through the origin with direction vector (3.0, 4.0, 6.0) followed by a translation of (7, 8, 13), and finally by adding to each coordinates of the resulting points Gaussian random noise with mean zero and standard deviation of 0.5. These measured data and modeled data are used to compute the estimated orientation and translation parameters. To simplify the simulation, all the weighting factors α_i and β_i are set to 1. The standard deviations for the resulting orientation and translation parameters are calculated from these twenty trials. Table 3 lists the simulation results. All the algorithms are written in Turbo Pascal. The Mathpak 87 subroutine package from Precision Plus Software was used to carry out all the matrix computation, as well as SVD and eigenvalue calculations.

TABLE 3
Comparison of Standard Deviation of Transformation Parameters

Number of point correspondences	Method used: SVD x	SVD y	SVD z	SVD θ	Dual number x	Dual number y	Dual number z	Dual number θ
5	1.434	3.013	1.190	0.147	0.461	0.277	0.509	0.147
10	1.133	2.373	0.843	0.046	0.133	0.215	0.169	0.046
20	0.296	0.607	0.254	0.040	0.102	0.187	0.108	0.040
30	0.171	0.246	0.125	0.037	0.115	0.115	0.087	0.037

From Table 3 we see that the two algorithms produce the same rotation errors no matter how many points are used during the simulations, which is expected. For the translation errors, the DQ algorithm exhibits better performance than the SVD algorithm in all the cases. Even in the case of 30 points, which is supposed to provide a good estimate, the DQ algorithm provides average accuracy improvement of 20% for the translation parameter calculation compared with the SVD algorithm.

Figure 4 shows the solution times. Except for the case of three samples, which uses three points, the minimum number of required points, all the samples consist of an equal number of points and vectors. From this figure we see that the computation time is approximately linear in the number of samples taken, which confirmed our analysis about the algorithm's time complexity. The computation time increases at about a rate of 1.0 msec/sample when a math processor is used.

5. CONCLUSION

A new algorithm has been presented for solving the object location problem using dual number quaternions, $\hat{\boldsymbol{q}} = \boldsymbol{r} + \varepsilon \boldsymbol{s}$. The primary result is the recognition that the localization problem can be cast into an optimization problem involving $\boldsymbol{r}$ and $\boldsymbol{s}$. This can be compared to previous results which directly obtain the position vector $\mathbf{p}$

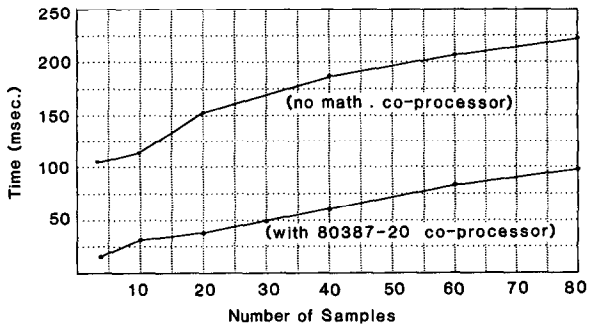

FIG. 4. Solution times for the DQ algorithm on a Compaq 386/20 computer.

rather than the position component of the dual quaternion, s. By solving for s instead of $\mathbf{p}$ we obtain a quadratic cost function which can be easily solved using standard eigenvalue–eigenvector math packages.

The method is computationally efficient and very accurate in the result. The required measurements are quite general. The inputs to the algorithm could be feature points and feature unit vectors.

This paper has not discussed the selection of the weighting factors. Intuitively, the weighting factors are closely related to the reliability of corresponding matching features. Let us take the extraction of two surface normals as an example. If two surface normals extracted from two planar surfaces are used as part of the algorithm's inputs, one surface is toward the sensor, and another surface is near parallel to the sensor's optical axis, it is obvious that the extracted surface normal obtained from the first one is more reliable than that of the second one. A quantitative analysis is needed and will certainly further improve the accuracy of the algorithm.

APPENDIX: DUAL QUATERNIONS AND THEIR APPLICATIONS IN REPRESENTING POSITION AND ORIENTATION

1. *The Concept and Properties of Dual Numbers*

A dual number $\hat{a} = a + \varepsilon b$ can be defined as a combination of two ordered real numbers a and b with a special multiplication rule for ε given by $\varepsilon^2 = 0$. The two real numbers a and b can be said to belong to the real part and the dual part of the dual number, respectively.

Addition, subtraction, and multiplication of dual numbers are defined by the formulae

$$\begin{aligned}(a + \varepsilon b) + (c + \varepsilon d) &= (a + c) + \varepsilon(b + d)\\ (a + \varepsilon b) - (c + \varepsilon d) &= (a - c) + \varepsilon(b - d)\\ (a + \varepsilon b)(c + \varepsilon d) &= ac + \varepsilon(ad + bc).\end{aligned} \tag{51}$$

Dual numbers were first considered by the famous German geometer E. Study (1862–1930) in the beginning of this century [12]. In his study he used the dual number to represent the dual angle which measures the relative position of two skew lines in space. That is, a dual angle was defined as

$$\hat{\theta} = \theta + \varepsilon d, \tag{52}$$

where the d is a distance between two lines in three-dimensional space and the θ is the angle between their directions.

Some important properties of dual numbers are

1. The product of a dual number $\hat{a}$ and its conjugate $\bar{\hat{a}} = a - \varepsilon b$ is

$$\hat{a}\bar{\hat{a}} = a^2. \tag{53}$$

2. The modulus of a dual number

$$|\hat{a}| = a \tag{54}$$

which can be negative.

3. Due to the fact that $\varepsilon^2 = 0$, the dual number function has a very simple form of Taylor series expansion:

$$f(a + \varepsilon b) = f(a) + \varepsilon b f'(a). \tag{55}$$

4. For the dual angle $\hat{\theta}$, we have

$$\sin(\hat{\theta}) = \sin(\theta + \varepsilon d) = \sin(\theta) + \varepsilon d \cos(\theta) \tag{56}$$

and

$$\cos(\hat{\theta}) = \cos(\theta + \varepsilon d) = \cos(\theta) - \varepsilon d \sin(\theta). \tag{57}$$

In the above, the properties (53) and (54) can be derived directly from the definition of multiplication of two dual numbers. To show property (55), we only need to know a function of a dual number $f(a + \varepsilon b)$, like the usual functions over the field of complex numbers, can be expanded into a formal Taylor series. That is, according to the theorem of Taylor series expansion, if a function $f(\hat{a})$ is analytic within a circle $|\hat{a} - c| < R$ $(R > 0)$, where c is the center of the circle, then $f(\hat{a})$ can be expanded into a Taylor series within that circle,

$$f(\hat{a}) = \sum_{n=0}^{\infty} u_n(\hat{a} - c)^n, \tag{58}$$

where $u_n = f^{(n)}(c)/n!$ and the series is unique. If we expand $f(\hat{a})$ at point a, the Taylor series will have the form

$$f(a + \varepsilon b) = f(a) + \varepsilon b f'(a) + \varepsilon^2 \frac{b^2}{2} f''(a) + \cdots. \tag{59}$$

Because $\varepsilon^2 = 0$, all the terms with the power of ε greater than one in the Taylor series will be zero. To get the last property, we simply expand $\sin(\hat{\theta})$ and $\cos(\hat{\theta})$ into their corresponding Taylor series at θ.

The idea of dual quantities can be extended to define dual vectors, dual quaternions, and dual matrices. These dual quantities enable two different quantities to be combined into one in many ways. For example, a dual vector can be defined to form any line in 3-D space. The direction and position of the line can be specified as

$$\hat{\mathbf{n}} = \mathbf{n} + \varepsilon \mathbf{p} \times \mathbf{n}, \tag{60}$$

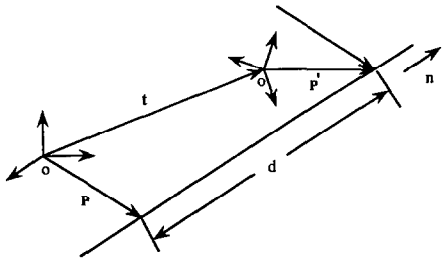

FIG. 5. Illustration of the transformation of an object coordinate system.

where $\mathbf{n}$ is a unit direction vector of the line and $\mathbf{p}$ is a position vector of any point on the line. As another example, a dual quaternion can be defined to represent any transformation between two coordinate frames which has been shown in Eq. (8).

2. *Position Vector Expressed in Terms of the Components of the Dual Quaternion*

The derivation of Eq. (19),

$$t = \boldsymbol{W}(r)^T s.$$

Consider an object coordinate system which is transformed into a new location, where the transformation is done by first translating the coordinate system in the direction of the unit vector $\mathbf{n}$ by a distance d and then rotating it by an angle θ with respect to a line having a unit vector $\mathbf{n}$ as its direction and passing through a point $\mathbf{p}$ (Fig. 5). The position vector $\mathbf{p}$ as shown in Fig. 5 will be transformed into vector $\mathbf{p}'$. The translation vector $\mathbf{t}$ can thus be expressed as

$$\begin{aligned} \mathbf{t} &= \mathbf{p} + d\mathbf{n} - \mathbf{p}' \\ &= \mathbf{p} + d\mathbf{n} - \boldsymbol{R}\mathbf{p} \\ &= (\boldsymbol{I} - \boldsymbol{R})\mathbf{p} + d\mathbf{n}. \end{aligned} \tag{61}$$

The rotation matrix $\boldsymbol{R}$, when defined by a rotation angle θ and a rotation axis $\mathbf{n}$, has the form

$$\boldsymbol{R} = \cos\theta \boldsymbol{I} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta \boldsymbol{K}(\mathbf{n}), \tag{62}$$

where the $\boldsymbol{K}(\mathbf{n})$ is the often mentioned skew-symmetric matrix.

Because $\mathbf{n}\mathbf{n}^T = \boldsymbol{I} + \boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n})$, Eq. (62) can be changed into

$$\begin{aligned} \boldsymbol{R} &= \cos\theta \boldsymbol{I} + (1 - \cos\theta)(\boldsymbol{I} + \boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n})) + \sin\theta \boldsymbol{K}(\mathbf{n}) \\ &= \boldsymbol{I} + (1 - \cos\theta)\boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n}) + \sin\theta \boldsymbol{K}(\mathbf{n}) \\ &= \boldsymbol{I} + 2\sin^2(\theta/2)\boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n}) + \sin\theta \boldsymbol{K}(\mathbf{n}). \end{aligned} \tag{63}$$

Replacing Eq. (63) into Eq. (61), we have

$$\begin{aligned} \mathbf{t} &= (-2\sin^2(\theta/2)\boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n})\mathbf{p} - \sin\theta \boldsymbol{K}(\mathbf{n}))\mathbf{p} + d\mathbf{n} \\ &= 2\sin^2(\theta/2)\mathbf{n} \times (\mathbf{p} \times \mathbf{n}) + \sin\theta(\mathbf{p} \times \mathbf{n}) + d\mathbf{n}. \end{aligned} \tag{64}$$

On the other hand, from Eqs. (11) and (12) we have

$$\begin{aligned} r_4\mathbf{s} - s_4\mathbf{r} &= (d/2)\mathbf{n} + \sin(\theta/2)\cos(\theta/2)\mathbf{p} \times \mathbf{n} \\ &= (d\mathbf{n} + \sin\theta \mathbf{p} \times \mathbf{n})/2 \end{aligned} \tag{65}$$

and

$$\begin{aligned} \mathbf{r} \times \mathbf{s} &= [\sin(\theta/2)\mathbf{n}] \times [d/2\cos(\theta/2)\mathbf{n} + \sin(\theta/2)\mathbf{p} \times \mathbf{n}] \\ &= \sin^2(\theta/2)\mathbf{n} \times (\mathbf{p} \times \mathbf{n}) \end{aligned} \tag{66}$$

because $\mathbf{n} \times \mathbf{n} = 0$. Therefore

$$\mathbf{t} = 2(r_4\mathbf{s} - s_4\mathbf{r} + \mathbf{r} \times \mathbf{s}). \tag{67}$$

On the other hand,

$$\begin{aligned} \boldsymbol{W}(r)^T s &= \begin{bmatrix} r_4\boldsymbol{I} + \boldsymbol{K}(\mathbf{r}) & -\mathbf{r} \\ \mathbf{r}^T & r_4 \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ s_4 \end{bmatrix} \\ &= \begin{bmatrix} (r_4\boldsymbol{I} + \boldsymbol{K}(\mathbf{r}))\mathbf{s} - s_4\mathbf{r} \\ r^T s \end{bmatrix} \\ &= \begin{bmatrix} r_4\mathbf{s} - s_4\mathbf{r} + \boldsymbol{K}(\mathbf{r})\mathbf{s} \\ 0 \end{bmatrix} = \begin{bmatrix} 1/2\mathbf{t} \\ 0 \end{bmatrix}. \end{aligned}$$

If we define the translation quaternion t for the translation vector $\mathbf{t}$ as

$$t = (1/2)\begin{bmatrix} \mathbf{t} \\ 0 \end{bmatrix} \tag{68}$$

then we have the formula

$$t = \boldsymbol{W}(r)^T s. \tag{69}$$

3. *Finding a Transformation Tuple* ($\mathbf{n}$, θ, d, $\mathbf{p}$) *from a Given Dual Quaternion*

Given a dual quaternion, the rotation submatrix can be derived as previously described, and the rotation axis $\mathbf{n}$ and angle θ can be extracted from the matrix. The translation distance d can be derived from Eq. (12),

$$d = \frac{2s_4}{\sin(\theta/2)}. \tag{70}$$

Equation (12) can also be used to derive $\mathbf{p}$. From Eq. (12) we have

$$\mathbf{s} = (d/2)\cos(\theta/2)\mathbf{n} + \sin(\theta/2)(\mathbf{p} \times \mathbf{n}). \tag{71}$$

This equation can be written as

$$\mathbf{p} \times \mathbf{n} = \frac{\mathbf{s} - (d/2)\cos(\theta/2)\mathbf{n}}{\sin(\theta/2)}. \tag{72}$$

Because $\mathbf{p} \times \mathbf{n} = \boldsymbol{K}(-\mathbf{n})\mathbf{p}$, we have

$$\boldsymbol{K}(-\mathbf{n})\mathbf{p} = \frac{\mathbf{s} - (d/2)\cos(\theta/2)\mathbf{n}}{\sin(\theta/2)}, \tag{73}$$

where the only unknown is $\mathbf{p}$.

The rank of matrix $\boldsymbol{K}(-\mathbf{n})$ is 2 and thus the dimension of null space of the matrix is 1. Equation (73) has a general solution in the form

$$\mathbf{p} = \mathbf{p}_0 + \alpha\mathbf{n}, \tag{74}$$

where $\mathbf{p}_0$ is any vector in null space.

That is, the solution for $\mathbf{p}$ is not unique. Usually, we select one of the vectors which are perpendicular to $\mathbf{n}$ as the desired solution.

One possible solution is

$$\mathbf{p} = \frac{\alpha}{n_1}\begin{bmatrix} 0 \\ -n_3 \\ n_2 \end{bmatrix}, \tag{75}$$

where $n_1 \neq 0$.

REFERENCES

1. K. S. Arun, T. S. Huang, and S. D. Blostein, Least-squares fitting of two 3-D point sets, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-9,** 1987, 698–700.
2. W. K. Clifford, Preliminary sketch of bi-quaternions, *J. London Math. Soc.* **4,** 1873, 381–395.
3. O. D. Faugeras and M. Herbert, A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces, in *Proceedings, Eighth International Conference on Artificial Intelligence, August 1983,* pp. 996–1002.
4. W. Eric, L. Grimson and Tomas Lozano-Perez, Model-based recognition and localization from sparse range or tactile data, *Int. J. Robotics Res.,* Fall 1984, 3–35.
5. Steven J. Gordon and Warren P. Seering, Real-time part position sensing, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-10,** 1988.
6. Kristjan T. Gunnarsson, *Optimal Part Localization by Database Matching with Sparse and Dense Data,* Ph.D. dissertation, Dept. of Mechanical Engineering, Carnegie Mellon Univ., May 1987.
7. Wahba Grace, A least squares estimate of satellite attitude, problem 65.1, *SIAM Rev.* July 1966, 384–386.
8. J. Keat, *Analysis of Least-Squares Attitude Determination Routine DOAOP,* Technical Report, Comp. Sci. Corp., CSC/TM-77/6034, February 1977.
9. S. Linnainmaa, D. Harwood, and L. S. Davis, Pose determination of a three-dimensional object using triangle pairs, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-10,** 1988, 634–648.
10. Edward Pervin and Jon A. Webb, *Quaternions in Computer Vision and Robotics,* Technical Report, Dept. of Computer Science, Carnegie-Mellon University, CMU-CS-82-150, 1982.
11. W. S. Rutkowski, R. Benton, and E. W. Kent, Model-driven determination of object pose for a visually served robot, in *IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, 1987,* pp. 1419–1428.
12. E. Study, *Geometrie der Dynamen,* Leipzig, 1903.
13. I. M. Yaglom, *Complex Numbers in Geometry,* Academic Press, New York, 1968.
14. A. T. Yang and F. Freudenstein, Application of dual-number quaternion algebra to the analysis of spatial mechanisms, *Trans. ASME,* June 1964, 300–308.