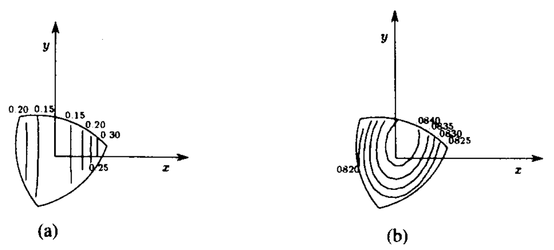

Fig. 8. Stiffness maps for the spatial six-degree-of-freedom parallel manipulator with $\psi = 10$; $\theta = 0°$, $\phi = 30°$: (a) k_x for $\psi = 10°$, $\theta = 0°$, $\phi = 30°$, $z = 52.5$; (b) k_{θ_x} for $\psi = 10°$, $\theta = 0°$, $\phi = 30°$, $z = 52.5$.

the programs developed here can be of significant help in the process of designing such manipulators. The method can also be extended to general complex kinematic chains—which can be used, for instance, as closed-loop hard automation modules—provided that a description of the workspace of the device to be studied and an expression for its stiffness matrix are available.

References

[1] K. H. Hunt, "Structural kinematics of in-parallel-actuated robot arms," *ASME J. Mechanisms, Transmissions, Automat. Design*, vol. 105, no. 4, pp. 705-712, 1983.

[2] E. F. Fichter, "A Stewart platform-based manipulator: General theory and practical construction," *Int. J. Robotics Res.*, vol. 5, no. 2, pp. 157-182, 1986.

[3] J.-P. Merlet, "Parallel manipulators, Part II: Theory, singular configurations and Grassmann geometry," Tech. Rep. 791, INRIA, France, 1988.

[4] C. Gosselin, "Kinematic analysis, optimization and programming of parallel robotic manipulators," Ph.D. thesis, McGill University, Montréal, Québec, Canada, 1988.

[5] C. Gosselin and J. Angeles, "Singularity analysis of closed-loop kinematic chains," *IEEE Trans. Robotics Automat.*, this issue, pp. 281-290.

[6] C. Gosselin, "Determination of the workspace of 6-dof parallel manipulators," *ASME J. Mechanisms, Transmissions, Automat. Design*, to be published.

[7] K. J. Waldron and K. H. Hunt, "Series-parallel dualities in actively coordinated mechanisms," in *Robotics Research 4* (R. Bolles and B. Roth, Eds.), 1988, pp. 175-181.

[8] J.-P. Merlet, "Parallel manipulators, Part I: Theory, design, kinematics, dynamics and control," Tech. Rep. 646, INRIA, France, 1987.

[9] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in *Proc. First Int. Symp. Robotics Res.*, 1984, pp. 735-747.

[10] H. Asada and J. A. Cro Granito, "Kinematic and static characterization of wrist joints and their optimal design," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1985, pp. 244-250.

[11] K.-M. Lee and R. Johnson, "Static characteristics of an in-parallel actuated manipulator for clamping and bracing applications," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1989, pp. 1408-1413.

[12] C. A. Klein and B. E. Blaho, "Dexterity measures for the design and control of kinematically redundant manipulators," *Int. J. Robotics Res.*, vol. 6, no. 2, pp. 72-83, 1987.

[13] J. K. Salisbury and J. J. Craig, "Articulated hands: Force control and kinematic issues," *Int. J. Robotics Res.*, vol. 1, no. 1, pp. 4-17, 1982.

[14] C. Gosselin and J. Angeles, "The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator," *ASME J. Mechanisms, Transmissions, Automat. Design*, vol. 110, no. 1, pp. 35-41, 1988.

[15] R. L. Williams, II and C. F. Reinholtz, "Closed-form workspace determination and optimization for parallel robotic mechanisms," in *Proc. ASME 20th Biennial Mech. Conf.* (Kissimmee, FL), Sept. 25-28, 1988.

[16] A. H. Shirkhodaie and A. H. Soni, "Forward and inverse synthesis for a robot with three degrees of freedom," *Proc. 1987 Summer Comput. Simulation Conf.* (Montréal), 1987, pp. 851-856.

On Homogeneous Transforms, Quaternions, and Computational Efficiency

JANEZ FUNDA, RUSSELL H. TAYLOR, SENIOR MEMBER, IEEE, AND RICHARD P. PAUL FELLOW, IEEE

***Abstract*—Three-dimensional modeling of rotations and translations in robot kinematics is most commonly performed using homogeneous transforms. In this paper, an alternate approach, employing quaternion/vector pairs as spatial operators, is compared with homogeneous transforms in terms of computational efficiency and storage economy. The conclusion drawn is that quaternion/vector pairs are as efficient, more compact, and more elegant than their matrix counterparts. A robust algorithm for converting rotational matrices into equivalent unit quaternions is described, and an efficient quaternion-based inverse kinematics solution for the Puma 560 robot arm is presented.**

I. Introduction

Robotics, computer vision, graphics, and other engineering disciplines require concise and efficient means of representing and applying generalized coordinate transformations in three dimensions, and a number of different representations have been developed. The most popular representation of spatial transformations of point vectors is the 4 × 4 real matrix (also termed homogeneous transform), based on the idea of homogeneous coordinates, introduced by Maxwell [9]. The appeal of homogeneous transforms is mostly due to their matrix formulation, which is familiar and lends itself to easy and efficient manipulation by a computer. On the other hand, such matrices are highly redundant, using 16 numbers (of which four are trivial) to represent six independent degrees of freedom. This redundancy can introduce numerical problems in calculations, wastes storage, and often increases the computational cost of algorithms. In parallel implementations, the extra data paths required to fetch the operands can also be a significant factor. Despite these drawbacks, matrix-based representations remain the dominant choice for most robotic systems applications.

This paper is concerned with an alternative representation, termed quaternion/vector pairs, in which a unit quaternion is used to represent rotations. Although they have been used extensively in kinematic analysis [1], [2], [16] and offer a potentially significant advantage in terms of numerical robustness and storage efficiency, they have been relatively neglected in practical robotic systems. This has been in part due to the fact that neither relative computational costs nor important details of key algorithms are well understood by the robotics community.

Quaternions were introduced by Hamilton [6], [7] as a way of defining the ratio between two vectors. They have since found applications in many areas of geometric analysis and modeling. Pervin and Webb [12] discussed general properties of quaternions as rotational operators, analyzed certain special types of rotations, and gave quaternion formulations of reflections and projections of stationary as well as moving geometric objects. Taylor [15] used quaternion/vector pairs for kinematic path planning and compared the computational cost with homogeneous transform methods using a Stanford arm as an example manipulator. Salamin [13] gave a brief analysis of com-

Manuscript received January 23, 1989; revised March 9, 1990. This material is based upon work supported by the National Science Foundation under Grant ECS-11879. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

J. Funda and R. P. Paul are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 19104-6389.

R. H. Taylor is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

IEEE Log Number 9036673.

putational behavior of quaternions and matrices as rotational operators and outlined an approach to the conversion problem. He also proposed an application of complex quaternions as a representation of Lorentz transformations in the theory of relativity. Shoemake [14] discussed advantages of using quaternions to achieve smooth interpolation of rotations in graphical animation and proposed a set of conversion routines between matrix, Euler-angle, and quaternion representations of rotations.

Although most of the above references touch on the issue of computational efficiency of effecting three-dimensional rotations and their compositions using quaternions, none of them addresses parallel implementations of the corresponding algorithms. Section II compares storage and computational costs for both sequential and parallel implementations of common spatial operations using matrix and quaternion-based representations.

Many of the authors also discuss conversion algorithms [3], [10], [13], [14]. The suggested approaches to converting a rotational matrix to a corresponding unit quaternion, although generally correct, tend to lack robustness and stability in the neighborhood of rotational singularities. We address this problem in Section III and present numerically robust and computationally efficient algorithms for converting between matrix and quaternion representations.

Finally, quaternion-based kinematic solutions for robot manipulators have not been documented. This may be in part due to the fact that the derivations (in places) are not entirely straightforward. Section IV gives an efficient solution to the inverse kinematics problem for the Puma 560 robot arm, employing the quaternion/vector pair representation of spatial relationships.

II. Computational Comparison of Homogeneous Transforms and Quaternion/Vector Pairs

A. Internal Representation and Terms of Comparison

From the standpoint of evaluating relative advantages of different representational schemes, we need to be concerned with the compactness of a given representation as well as with its computational efficiency in performing basic spatial operations. Storage considerations are necessitated by the fact that on most existing hardware, the cost (measured in terms of the number of required CPU cycles) of fetching an operand from memory exceeds the cost of performing a basic arithmetic operation. Consequently, an advantage of a particular representation in terms of storage may well outweight a small disadvantage in terms of computational expense.

We will, in this paper, denote a quaternion $\boldsymbol{q}$ and its components by writing $\boldsymbol{q} = [s, \boldsymbol{v}]$ or $\boldsymbol{q} = [\cos(\theta/2), \sin(\theta/2)\boldsymbol{k}]$, where $s \in \mathbb{R}$, $\boldsymbol{v} \in \mathbb{R}^3$, and where θ and $\boldsymbol{k}$ correspond to the angle and the unit axis, respectively, of the fixed-point rotation encoded by $\boldsymbol{q}$. Similarly, a quaternion/vector pair operator denoting a rotation $\boldsymbol{q}$ and a translation $\boldsymbol{p}$ will be written as $\boldsymbol{Q} = (\boldsymbol{q}, \boldsymbol{p})$. For the purposes of the forthcoming discussion, assume that such a quaternion/vector pair is stored in a straightforward fashion as the triple $(s, \boldsymbol{v}, \boldsymbol{p})$, thus requiring storage for seven floating point values. Since $\boldsymbol{q} \cdot \boldsymbol{q} = 1$, one of these numbers is redundant and could, in principle, be recomputed from the other six, although this is not done in practice.

A transform $\boldsymbol{T} = [\boldsymbol{n}\,\boldsymbol{o}\,\boldsymbol{a}\,\boldsymbol{p}]$, on the other hand, contains a significant amount of redundant information, and its representation can thus be compacted. In particular, since the rotational part of $\boldsymbol{T}$ consists of three mutually orthogonal vectors $\boldsymbol{n}$, $\boldsymbol{o}$, and $\boldsymbol{a}$, we can eliminate one of them from the representation and recover it, when needed, as the cross product of the remaining two. Usually, $\boldsymbol{n}$ is omitted (note that $\boldsymbol{n} = \boldsymbol{o} \times \boldsymbol{a}$) and only the triple $(\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{p})$ is explicitly stored, requiring storage for nine floating-point values. Note that this convention necessitates restoring the normal vector $\boldsymbol{n}$ when the full matrix is needed for a particular operation. This represents pure overhead (one cross product) in the case of a spatial transformation of a point vector but translates into a significant saving (three dot products) during composition of spatial transformations, as the normal vector $\boldsymbol{n}$ of the resulting matrix need not be computed [5]. This, coupled with a 25% reduction in storage space, should provide ample justification to adopt the above convention.

TABLE I
Homogeneous Transforms Versus Quaternion/Vector Pairs

Operation	Norm	Sequential Execution						Parallel Execution			
		T			Q			T		Q	
		*	+	$\surd$	*	+	$\surd$	PEs	cyc	PEs	cyc
Spatial Trans	–	15	12	0	15	15	0	9	4	9	6
Composition	–	33	24	0	31	27	0	24	4	22	6
Inverse	–	15	9	0	15	12	0	6	5	9	5
Normalization		30	16	2	8	3	1	6	12	4	5
Spatial Trans	$\surd$	45	28	2	23	18	1	9	16	9	11
Composition	$\surd$	63	40	2	39	30	1	24	16	22	11
Inverse	$\surd$	45	25	2	23	15	1	6	17	9	10

Table I gives the results of evaluating the computational behavior of homogeneous transforms and quaternion/vector pairs in terms of three basic operations: 1) spatial transformation of a point vector, 2) composition of two spatial operators, and 3) computing the inverse of a spatial operator. The costs for both sequential and parallel executions of the corresponding algorithms are given (see [5] for derivations). To ensure correct interpretation of the stated results, a few comments about the way in which the numbers were obtained may be appropriate. This is particularly critical for the case of parallel algorithm cost analyses.

The costs for parallel algorithms are given in terms of the number of processing elements (PE's) and the number of machine cycles needed to perform the computation. A cycle, in this context, is defined as the time required by the hardware to execute a basic floating-point add/subtract or multiply/divide operation. Parallel computations are structured to minimize the necessary number of cycles (i.e., maximize computational concurrency) without regard to the number of PE's needed to support this optimal degree of parallelism.

B. Tabular Summary of Computational Costs and Discussion

Table I gives the computational costs of the three operations mentioned above under both sequential and parallel execution.[1] The symbol '—' in the Norm column indicates that the computational cost of the corresponding operation does not include the cost of normalizing the spatial operator prior to performing the operation. Conversely, the symbol '$\surd$' implies that the cost of normalization has been included in the cost of the operation.

Table I suggests that the two formalisms are virtually equivalent for the case of nonnormalizing sequential applications, i.e., the case where computations are carried out on a single-processor machine and where the rotational operators are not normalized prior to being used in the computations. We also see that the nonnormalizing algorithms based on homogeneous transforms parallelize slightly better than their quaternion/vector counterparts. If the cost of normalizing the rotational operator is included in the total cost, however, the quaternion/vector pair approach yields much more efficient implementations on both single and multiprocessor systems. This is due to the fact (see Table I) that a quaternion can be normalized with a minimal amount of computational expense, whereas normalization of a rotational matrix requires substantially more effort [5], [13]. Clearly, the nonnormalizing operations (where the rotational operators are never normalized) and their normalized versions as defined above (where the rotational operators are normalized at every call) correspond to the two extremes. In practice, the need for normalization will be intermediate to the above two scenarios, depending mostly on the presence of chains of transformational products, which are likely to denormalize rotational operators. Thus, in a situation where continuous chains of transformational products are relatively rare and highly parallel hardware is available, homogeneous matrices may prove superior to quaternion/vector pairs. Conversely, if the nature of computation necessitates frequent renormalization of rota-

[1]Divisions and multiplications by 2 can be implemented by incrementing exponents of the floating-point values being scaled and will therefore be neglected in this and all subsequent cost analyses.

tional operators, then the quaternion/vector approach arises as the better alternative. It is interesting to note that the cost of testing for whether or not a homogeneous transform (i.e., its rotational matrix) needs renormalization accounts for approximately two thirds of the cost of the normalizing process itself [5]. Consequently, it is uncertain if, for rotational matrices, "normalizing by need" will result in a more efficient overall performance.

In summary, the differences in computational efficiency between the two formalisms are perhaps not sufficiently significant to warrant a particular choice on that basis alone. However, quaternion/vector pairs require less storage and are therefore more conveniently retrievable from memory, saving valuable machine cycles in a real-time computation.

III. Conversions between Quaternions and Rotational Matrices

A. Quaternion → Rotational Matrix

Given a unit quaternion $q = [\cos(\theta/2), \sin(\theta/2)\boldsymbol{k}] = [s, \langle x, y, z\rangle]$, encoding a rotation through an angle of θ about unit axis $\boldsymbol{k}$, the corresponding equivalent rotational matrix $\boldsymbol{R}$ can be found to be [5], [12], [14]

$$\boldsymbol{R} = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2sz & 2xz+2sy \\ 2xy+2sz & 1-2x^2-2z^2 & 2yz-2sx \\ 2xz-2sy & 2yz+2sx & 1-2x^2-2y^2 \end{bmatrix}. \quad (1)$$

The matrix $\boldsymbol{R}$ can be constructed using nine multiples and 15 adds.

B. Rotational Matrix → Quaternion

Conversely, given a 3×3 rotational matrix $\boldsymbol{R} = [\boldsymbol{n}\,\boldsymbol{o}\,\boldsymbol{a}]$, its entries may be interpreted as in (1), i.e., as functions of s, x, y, z, and the corresponding equivalent unit quaternion $q = [s, \langle x, y, z\rangle]$ can be reconstructed as outlined below:

Since q is of unit magnitude, we have $s^2+x^2+y^2+z^2=1$, and

$$n_x + o_y + a_z + 1 = 4s^2. \quad (2)$$

Therefore

$$s = \frac{1}{2}\sqrt{n_x + o_y + a_z + 1}. \quad (3)$$

Note that the positive value of the square root is taken as the value of s since $-180° \le \theta \le 180°$, and thus $0 \le s = \cos(\theta/2) \le 1$, where θ is the angle of rotation implied by the matrix. The remaining task is to extract $\boldsymbol{v} = \langle x, y, z\rangle = \sin(\theta/2)\boldsymbol{k}$, where $\boldsymbol{k}$ is the unit axis of rotation.[2] Observe that pairing off-diagonal elements of $\boldsymbol{R}$ gives

$$\begin{aligned} x' &= 4sx = o_z - a_y \\ y' &= 4sy = a_x - n_z \\ z' &= 4sz = n_y - o_x \end{aligned} \quad (4)$$

where $\boldsymbol{v}' = \langle x', y', z'\rangle$ is colinear with the desired axis vector $\boldsymbol{v}$. We could therefore obtain $\boldsymbol{v}$ immediately as $(1/4s)\boldsymbol{v}'$. However, as θ approaches 0°, $\sin(\theta/2)$ approaches 0, and the vector $\boldsymbol{v}' = 4s\boldsymbol{v} = 4s\sin(\theta/2)\boldsymbol{k}$ becomes poorly defined. This corresponds to the physical reality of the axis of rotation being indeterminate for very small angular displacements. Similarly, as θ approaches 180°, $s = \cos(\theta/2)$ approaches 0, and again, the vector $\boldsymbol{v}'$ is poorly defined. However, physically, the rotational axis is well defined in this case, and we should be able to recover its components precisely. Towards this end, we will reexamine the rotational matrix $\boldsymbol{R}$ of (1) and extract an additional corrective vector $\boldsymbol{v}''$, which will be colinear with the axis of rotation and whose components will not vanish as θ approaches 180°. We will then add $\boldsymbol{v}'$ and $\boldsymbol{v}''$ and scale the sum appropriately to obtain the final axis vector $\boldsymbol{v}$.

[2]In the remainder of this discussion, we will refer to any vector colinear with $\boldsymbol{k}$ as the axis of rotation.

Using the fact that q is a unit quaternion, i.e., $s^2+x^2+y^2+z^2=1$, we can rewrite the diagonal elements of the matrix $\boldsymbol{R}$ as follows:

$$\begin{aligned} n_x &= s^2 + x^2 - (y^2+z^2) \\ o_y &= s^2 + y^2 - (x^2+z^2) \\ a_z &= s^2 + z^2 - (x^2+y^2). \end{aligned} \quad (5)$$

Therefore, the diagonal elements of a rotational matrix can be used to extract information about relative magnitudes of the components of the corresponding rotation axis $\boldsymbol{v} = \langle x, y, z\rangle$. In particular, the numerically largest (most positive) of the diagonal elements n_x, o_y, a_z indicates the primary direction (i.e., the largest component) of the rotation axis. This information can be used to obtain the largest possible corrective vector $\boldsymbol{v}''$. We will illustrate one of the three possible cases in detail. The other two cases are similar.

If $n_x = \max(n_x, o_y, a_z)$, i.e., the largest component of the $\boldsymbol{v}$ axis lies in the x direction, then

$$n_x - o_y - a_z + 1 = 4x^2 \quad (6)$$

and a vector $\boldsymbol{v}'' = \langle x'', y'', z''\rangle$, colinear with $\boldsymbol{v}'$, can be obtained as follows:

$$\begin{aligned} x'' &= 4xx = n_x - o_y - a_z + 1 \\ y'' &= 4xy = n_y + o_x \\ z'' &= 4xz = n_z + a_x. \end{aligned} \quad (7)$$

Note that the common factor in all components of $\boldsymbol{v}''$ is x (i.e., the largest component of $\boldsymbol{v}$), and therefore, $\boldsymbol{v}''$ represents the largest possible correction to $\boldsymbol{v}'$. In addition, observe that $|\boldsymbol{v}''| \gg |\boldsymbol{v}'|$ for θ close to 180°.

If $x' = 4sx \ge 0$, then as $s \ge 0$ (3), the x component of the $\boldsymbol{v}$ axis must be positive. Since $x'' = 4x^2$ is also positive, $\boldsymbol{v}'$ and $\boldsymbol{v}''$ are coparallel (i.e., point in the same direction). Conversely, if $x' = 4sx < 0$, then the x component of the $\boldsymbol{v}$ axis is negative, and $\boldsymbol{v}'$ and $\boldsymbol{v}''$ are antiparallel (i.e., point in opposite directions) with x' giving the correct direction. We can therefore combine the two vectors as follows:

$$\begin{aligned} x''' &= x' + \mathrm{sgn}(x')x'' = o_z - a_y + \mathrm{sgn}(x')(n_x - o_y - a_z + 1) \\ y''' &= y' + \mathrm{sgn}(x')y'' = a_x - n_z + \mathrm{sgn}(x')(n_y + o_x) \\ z''' &= z' + \mathrm{sgn}(x')z'' = n_y - o_x + \mathrm{sgn}(x')(n_z + a_x) \end{aligned} \quad (8)$$

where the sign function $\mathrm{sgn}(x)$ returns $+1$ if $x \ge 0$ and -1 otherwise. Thus obtained vector $\boldsymbol{v}''' = \langle x''', y''', z'''\rangle$ coincides with the axis of rotation of the given rotation matrix (and thus with the vector $\boldsymbol{v}$) and is well defined for θ approaching 180°.

Again, the cases $o_y = \max(n_x, o_y, a_z)$ and $a_z = \max(n_x, o_y, a_z)$ yield symmetrically analogous derivations and results. Having obtained a robust description of the direction of the rotation axis, we must now scale the vector $\boldsymbol{v}'''$ into $\boldsymbol{v} = \langle x, y, z\rangle$ so that $q = [s, \boldsymbol{v}]$ is a unit quaternion. In order for q to be of unit magnitude, we must have

$$|\boldsymbol{v}|^2 = x^2 + y^2 + z^2 = 1 - s^2 \quad (9)$$

and therefore

$$\boldsymbol{v} = \left(\sqrt{\frac{1-s^2}{x'''^2 + y'''^2 + z'''^2}}\right)\boldsymbol{v}''' \quad (10)$$

where $q = [s, \boldsymbol{v}]$ is the desired unit quaternion, corresponding to the original matrix $\boldsymbol{R}$.

The computational cost of the above procedure is as follows: $3+$, $1\sqrt{}$ are needed to obtain s, $11+$ is needed to obtain $\boldsymbol{v}'''$ (note that it does not matter which of the three cases turns out to be relevant), and finally, $8*$, $3+$, and $1\sqrt{}$ are needed to compute $\boldsymbol{v}$ (10). Therefore, the overall cost of converting a rotational matrix $\boldsymbol{R} = [\boldsymbol{n}\,\boldsymbol{o}\,\boldsymbol{a}]$ to a unit

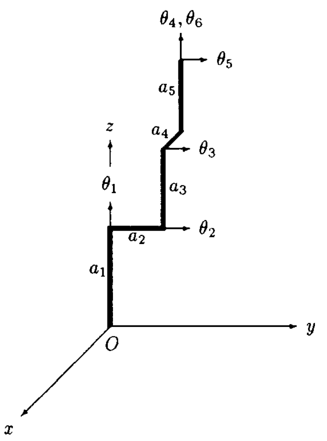

Fig. 1. Coordinate frame skeleton for Puma 560.

quaternion $q = [s, v]$ is eight multiplies, 17 adds, and two square roots.

IV. Quaternion-Based Inverse Kinematics for Puma 560

In this section, we use the quaternion/vector pair formalism to solve the inverse kinematics problem for a six-jointed revolute manipulator with a spherical wrist. The solution is based on a nonstandard (non Denavit/Hartenberg) choice of coordinate frame assignments, where for the fully vertically extended arm, the six-joint coordinate frames are aligned with the kinematic base frame (see Fig. 1). We present the procedure in some detail and in particular concentrate on the solution for the orientation angles (θ_4, θ_5, and θ_6) to illustrate the elegance and power of quaternion algebra.

The inverse kinematic problem for the Puma can be stated as the problem of retrieving the joint angles θ_i, $1 \le i \le 6$, from the equation

$$\begin{aligned}(R_w, T_w) = {} & T(z, a_1)R(z, \theta_1)T(y, a_2)R(y, \theta_2)T(z, a_3)R(y, \theta_3)\\ & \cdot T(x, a_4)T(z, a_5)R(z, \theta_4)R(y, \theta_5)R(z, \theta_6) \quad (11)\end{aligned}$$

where T and R denote translations along and rotations about the specified coordinate axes, a_i are the known link length parameters, and (R_w, T_w) represents the known orientation and displacement of the robot's wrist with respect to the (kinematic) base coordinate frame.

Since the rotations $R(y, \theta_2)$ and $R(y, \theta_3)$ do not change the orientation of the axis y, we can rearrange the terms in (11) as follows:

$$\begin{aligned}(R_w, T_w) = {} & T(z, a_1)R(z, \theta_1)R(y, \theta_2)T(z, a_3)R(y, \theta_3)T(x, a_4)\\ & \cdot T(y, a_2)T(z, a_5)R(z, \theta_4)R(y, \theta_5)R(z, \theta_6). \quad (12)\end{aligned}$$

By moving the translation $T(z, a_3)$ one step closer to the base of the kinematic chain and combining the rotations $R(y, \theta_2)$ and $R(y, \theta_3)$, we have

$$\begin{aligned}(R_w, T_w) = {} & T(z, a_1)R(z, \theta_1)T(x, S_2a_3)T(z, C_2a_3)R(y, \theta_{(2+3)})\\ & \cdot T(x, a_4)T(y, a_2)T(z, a_5)R(z, \theta_4)R(y, \theta_5)R(z, \theta_6) \quad (13)\end{aligned}$$

where S_i and C_i denote $\sin\theta_i$ and $\cos\theta_i$, respectively. Fixing the arm's base coordinate frame B at the intersection of the axes of joint angles θ_1 and θ_2 in Fig. 1, we can write the kinematic transformations A_i describing the spatial relationships between successive coordinate frames along the manipulator linkage as follows:

$$\begin{aligned}A_1 &= R(z, \theta_1) = ([\bar{C}_1, \bar{S}_1\boldsymbol{k}], \mathbf{0})\\ A_2 &= T(x, S_2a_3)T(z, C_2a_3) = (\mathbb{1}, \langle S_2a_3, 0, C_2a_3\rangle)\\ A_3 &= R(y, \theta_{(2+3)}) = ([\bar{C}_{(2+3)}, \bar{S}_{(2+3)}\boldsymbol{j}], \mathbf{0})\\ A_4 &= T(x, a_4)T(y, a_2)T(z, a_5)R(z, \theta_4) = ([\bar{C}_4, \bar{S}_4\boldsymbol{k}], \langle a_4, a_2, a_5\rangle)\\ A_5 &= R(y, \theta_5) = ([\bar{C}_5, \bar{S}_5\boldsymbol{j}], \mathbf{0})\\ A_6 &= R(z, \theta_6) = ([\bar{C}_6, \bar{S}_6\boldsymbol{k}], \mathbf{0}) \qquad (14)\end{aligned}$$

where the transformations A_i have been expressed as quaternion/vector pairs. For reasons of compactness, we will adopt the notational conventions of denoting the unit identity quaternion $[1, \mathbf{0}]$ by $\mathbb{1}$, the half angle $(\theta_i/2)$ by $\bar{\theta}_i$, and $\sin(\theta_i/2)$ and $\cos(\theta_i/2)$ by $\bar{S}_i$ and $\bar{C}_i$, respectively. Note that each A_i depends only on θ_j with $j \le i$. The corresponding inverse transformations are

$$\begin{aligned}A_1^{-1} &= ([\bar{C}_1, -\bar{S}_1\boldsymbol{k}], \mathbf{0})\\ A_2^{-1} &= (\mathbb{1}, \langle -S_2a_3, 0, -C_2a_3\rangle)\\ A_3^{-1} &= ([\bar{C}_{(2+3)}, -\bar{S}_{(2+3)}\boldsymbol{j}], \mathbf{0})\\ A_4^{-1} &= ([\bar{C}_4, -\bar{S}_4\boldsymbol{k}], \langle -C_4a_4 - S_4a_2, S_4a_4 - C_4a_2, -a_5\rangle)\\ A_5^{-1} &= ([\bar{C}_5, -\bar{S}_5\boldsymbol{j}], \mathbf{0})\\ A_6^{-1} &= ([\bar{C}_6, -\bar{S}_6\boldsymbol{k}], \mathbf{0}). \qquad (15)\end{aligned}$$

Given the orientation and displacement of the wrist coordinate frame with respect to the base frame B (i.e., (R_w, T_w)), we can recover the joint coordinates $\theta_1, \theta_2, \cdots, \theta_6$ via the following iterative procedure:

```
• compute $U_i = A_iA_{i+1} \cdots A_6$ for $1 \le i \le 6$
• let $V_1 = (R_w, T_w)$
• for $j \leftarrow 1$ to 3 do
      set $U_j = V_j$ and extract $\theta_j$
      $V_{j+1} \leftarrow A_j^{-1}V_j$
  endfor
• set $U_4 = V_4$ and extract $\theta_4, \theta_5, \theta_6$.
```

We therefore first define the quaternion/vector products U_i, $1 \le i \le 6$, where $U_i = A_iA_{i+1} \cdots A_6$. In order to optimize computational efficiency, we will define local variables wherever appropriate:

$$\begin{aligned}U_6 &= A_6\\ &= ([\bar{C}_6, \bar{S}_6\boldsymbol{k}], \mathbf{0}) \qquad (16)\\ U_5 &= A_5U_6\\ &= ([\bar{C}_5\bar{C}_6, \langle \bar{S}_5\bar{S}_6, \bar{S}_5\bar{C}_6, \bar{C}_5\bar{S}_6\rangle], \mathbf{0}) \qquad (17)\\ U_4 &= A_4U_5\\ &= ([U_{41}, \langle U_{42}, U_{43}, U_{44}\rangle], \langle a_4, a_2, a_5\rangle) \qquad (18)\end{aligned}$$

where

$$U_{41} = \bar{C}_5\bar{C}_{(4+6)} \qquad U_{42} = \bar{S}_5\bar{S}_{(6-4)}$$
$$U_{43} = \bar{S}_5\bar{C}_{(6-4)} \qquad U_{44} = \bar{C}_5\bar{S}_{(4+6)}.$$

$$\begin{aligned}U_3 &= A_3U_4\\ &= ([U_{31}, \langle U_{32}, U_{33}, U_{34}\rangle], \langle U_{35}, a_2, U_{37}\rangle) \qquad (19)\end{aligned}$$

where

$$\begin{aligned}U_{31} &= \bar{C}_{(2+3)}U_{41} - \bar{S}_{(2+3)}U_{43} & U_{32} &= \bar{C}_{(2+3)}U_{42} + \bar{S}_{(2+3)}U_{44}\\ U_{33} &= \bar{C}_{(2+3)}U_{43} + \bar{S}_{(2+3)}U_{41} & U_{34} &= \bar{C}_{(2+3)}U_{44} - \bar{S}_{(2+3)}U_{42}\\ U_{35} &= C_{(2+3)}a_4 + S_{(2+3)}a_5 & U_{37} &= C_{(2+3)}a_5 - S_{(2+3)}a_4.\end{aligned}$$

$$\begin{aligned}U_2 &= A_2U_3\\ &= ([U_{31}, \langle U_{32}, U_{33}, U_{34}\rangle], \langle U_{25}, a_2, U_{27}\rangle) \qquad (20)\end{aligned}$$

where

$$U_{25} = U_{35} + S_2a_3 \qquad U_{27} = U_{37} + C_2a_3.$$

$$\begin{aligned}U_1 &= A_1U_2\\ &= ([\bar{C}_1U_{31} - \bar{S}_1U_{34},\\ &\quad \langle \bar{C}_1U_{32} - \bar{S}_1U_{33}, \bar{C}_1U_{33} + \bar{S}_1U_{32}, \bar{C}_1U_{34} + \bar{S}_1U_{31}\rangle],\\ &\quad \langle C_1U_{25} - S_1a_2, C_1a_2 + S_1U_{25}, U_{27}\rangle). \qquad (21)\end{aligned}$$

1. Solving for $\theta_1, \theta_2, \theta_3$: Following the procedure outlined above, we can extract the first three angles by equating the corresponding U and V terms. The quaternion/vector pairs V_i, $1 \le i \le 3$, have the forms

$$\begin{aligned} V_1 &= (R_w, T_w) \\ &= ([w, \langle a, b, c\rangle], \langle p_x, p_y, p_z\rangle) \qquad (22) \\ V_2 &= A_1^{-1} V_1 \\ &= ([V_{21} \langle V_{22}, V_{23}, V_{24}\rangle], \langle V_{25}, V_{26}, p_z\rangle) \qquad (23) \\ V_3 &= A_2^{-1} V_2 \\ &= ([V_{21}, \langle V_{22}, V_{23}, V_{24}\rangle], \langle V_{35}, V_{26}, V_{37}\rangle) \qquad (24) \end{aligned}$$

where the local variables V_{ij} are defined as follows:

$$\begin{aligned} V_{21} &= \bar{C}_1 w + \bar{S}_1 c & V_{22} &= \bar{C}_1 a + \bar{S}_1 b \\ V_{23} &= \bar{C}_1 b - \bar{S}_1 a & V_{24} &= \bar{C}_1 c - \bar{S}_1 w \\ V_{25} &= C_1 p_x + S_1 p_y & V_{26} &= C_1 p_y - S_1 p_x \\ V_{35} &= V_{25} - S_2 a_3 & V_{37} &= p_z - C_2 a_3. \end{aligned}$$

Since the details of the resulting derivation, as well as the final expressions for the joint angles, are similar to the ones obtained by Paul and Zhang using matrix-based methods [11], we will restrict ourselves to stating the results. The reader is referred to [5] for derivational details:

$$\theta_1 = \begin{cases} \arctan\left[\dfrac{p_y}{p_x}\right] - \arcsin\left[\dfrac{a_2}{\sqrt{p_x^2 + p_y^2}}\right] \\ \arctan\left[\dfrac{p_y}{p_x}\right] + \arcsin\left[\dfrac{a_2}{\sqrt{p_x^2 + p_y^2}}\right] - \pi \end{cases}$$

$$\theta_2 = \arctan\left[\frac{V_{25}}{p_z}\right] \pm \arccos\left[\frac{V_{25}^2 + p_z^2 + a_3^2 - a_4^2 - a_5^2}{2a_3\sqrt{V_{25}^2 + p_z^2}}\right]$$

$$\theta_3 = \arctan\left[\frac{V_{35}}{V_{37}}\right] - \arctan\left[\frac{a_4}{a_5}\right] - \theta_2.$$

Note that θ_1 and θ_2 are both double valued functions corresponding to the two possible "shoulder" and "elbow" configurations of the arm, respectively.

2. Solving for $\theta_4, \theta_5, \theta_6$: To compute the orientation angles of the Euler wrist, we need to evaluate the quaternion/vector pair V_4. We have

$$\begin{aligned} V_4 &= A_3^{-1} V_3 \\ &= ([V_{41}, \langle V_{42}, V_{43}, V_{44}\rangle], \langle V_{45}, V_{26}, V_{47}\rangle) \qquad (25) \end{aligned}$$

where

$$\begin{aligned} V_{41} &= \bar{C}_{(2+3)} V_{21} + \bar{S}_{(2+3)} V_{23} & V_{42} &= \bar{C}_{(2+3)} V_{22} - \bar{S}_{(2+3)} V_{24} \\ V_{43} &= \bar{C}_{(2+3)} V_{23} - \bar{S}_{(2+3)} V_{21} & V_{44} &= \bar{C}_{(2+3)} V_{24} + \bar{S}_{(2+3)} V_{22} \\ V_{45} &= C_{(2+3)} V_{35} - S_{(2+3)} V_{37} & V_{47} &= C_{(2+3)} V_{37} + S_{(2+3)} V_{37}. \end{aligned}$$

Equating the rotational parts of U_4 and V_4 yields the following set of equations:

$$\begin{aligned} \bar{C}_5 \bar{C}_{(4+6)} &= V_{41} \\ \bar{S}_5 \bar{S}_{(6-4)} &= V_{42} \\ \bar{S}_5 \bar{C}_{(6-4)} &= V_{43} \\ \bar{C}_5 \bar{S}_{(4+6)} &= V_{44}. \qquad (26) \end{aligned}$$

TABLE II
INVERSE KINEMATICS FOR PUMA 560

Component	Cost				
	+	*	$\sqrt{}$	trig^{-1}	sin + cos
θ_1	3	3	1	2	0
θ_2	4	6	1	2	1
θ_3	4	2	0	1	1
$[V_{41}, \langle V_{42}, V_{43}, V_{44}\rangle]$	8	16	0	0	2
θ_5	2	4	2	1	0
θ_4, θ_6	2	0	0	2	0
Total	23	31	4	8	4

From this we see that

$$V_{41}^2 + V_{44}^2 = \bar{C}_5^2$$
$$V_{42}^2 + V_{43}^2 = \bar{S}_5^2.$$

Since $\theta_5 \in [-\pi, \pi)$, $\bar{C}_5 = \cos(\theta_5/2)$ is nonnegative, whereas the sign of $\bar{S}_5 = \sin(\theta_5/2)$ depends on the sign of the angle θ_5, i.e., $\text{sgn}(\bar{S}_5) = \text{sgn}(\theta_5)$. Hence, we have

$$\theta_5 = 2\arctan\left[\frac{\pm\sqrt{V_{42}^2 + V_{43}^2}}{+\sqrt{V_{41}^2 + V_{44}^2}}\right]$$

where the positive sign of the numerator corresponds to the wrist configuration with $\theta_5 > 0°$, and conversely, the negative sign of the numerator corresponds to the configuration with $\theta_5 < 0°$.

Assuming for the moment that the wrist is nonsingular (i.e., $\bar{S}_5 \neq 0$), we can cancel both $\bar{S}_5$ and $\bar{C}_5$ from the ratios V_{42}/V_{43} and V_{44}/V_{41}, respectively. Preserving the signs of the V_{ij} terms to facilitate reconstruction of the correct quadrants of the angles being retrieved, we have

$$\tan(\bar{\theta}_4 + \bar{\theta}_6) = \frac{\text{sgn}(\bar{C}_5)V_{44}}{\text{sgn}(\bar{C}_5)V_{41}}; \qquad \tan(\bar{\theta}_6 - \bar{\theta}_4) = \frac{\text{sgn}(\bar{S}_5)V_{42}}{\text{sgn}(\bar{S}_5)V_{43}}. \qquad (27)$$

Again, $\bar{C}_5 \ge 0$, $\text{sgn}(\bar{S}_5) = \text{sgn}(\theta_5)$, and therefore

$$\theta_4 = \arctan\left[\frac{V_{44}}{V_{41}}\right] - \arctan\left[\frac{\text{sgn}(\theta_5)V_{42}}{\text{sgn}(\theta_5)V_{43}}\right]$$

$$\theta_6 = \arctan\left[\frac{V_{44}}{V_{41}}\right] + \arctan\left[\frac{\text{sgn}(\theta_5)V_{42}}{\text{sgn}(\theta_5)V_{43}}\right].$$

Observe that as θ_5 approaches 0°, $\bar{S}_5$ goes to 0, and the term $\arctan(V_{42}/V_{43})$ becomes undefined. However, the sum of θ_4 and θ_6 remains well defined, as expected, since the condition $\theta_5 = 0°$ corresponds to the axes of the joints 4 and 6 being coaxial (i.e., aligned in the same direction). This consideration is consistent with the physical reality of the Euler wrist, which loses a degree of freedom as it approaches a singular configuration.

Table II summarizes the cost of the procedure outlined above. The cost analysis assumes that table lookup is employed to compute trigonometric functions (approximately 10 arithmetic operations). The computation of a sin/cos pair for an angle has been counted as a single operation because both values can be obtained at a cost only negligibly greater than that of computing one. Finally, any subexpressions involving the constant offsets a_i are assumed to have been precomputed offline.

The computational costs listed in Table II are roughly comparable with those for matrix-based representations [8], [11], and consequently, one would not want to choose one representation over the other on the basis of computational speed alone. On the other hand, the greater numerical robustness and lower storage requirements of quaternion-based representations can often represent a decisive advantage, and it is reassuring that this bonus comes at no additional cost in terms of the kinematic solution time.

V. Conclusion

The main purpose of this paper is to propose to the robotics community that quaternions, coupled with a displacement vector, function as very elegant and efficient spatial transformation operators. With the additional advantage of being very compact, and therefore storage efficient, quaternion/vector pairs provide an attractive alternative to the homogeneous matrices. This is particularly true in situations where storage access is expensive (in terms of machine cycles) and where the nature of the computation necessitates frequent renormalization of rotational operators. In addition, quaternions seem to offer a more intuitive framework for using and understanding the effects of three-dimensional rotations because the axis and angle of rotation appear explicitly in their definition. Finally, the compactness and elegance of the quaternion algebra makes the quaternion/vector pair formalism a rewarding alternative in research and pedagogical environments.

We have also presented an efficient and robust procedure for converting a rotational matrix operator into the corresponding unit quaternion, which exhibits stable behavior in the neighborhood of rotational singularities. As an example of the elegance and efficiency of the quaternion algebra, we have given a quaternion-based solution to the inverse kinematics problem for the Puma 560 robot arm, which in terms of the computational expense is comparable to existing matrix-based solutions [8], [11].

Since neither representation exhibits a decisive advantage over the other in terms of computational speed, we suggest that other criteria such as mathematical elegance, numerical robustness, and storage efficiency should be considered in selecting a representation for spatial transformations. The advantages of quaternion/vector pairs in these respects may make them an attractive choice in many practical applications.

Appendix A

The Two Operators

The homogeneous transform operator effecting a rotation through an angle of θ about a spatial axis $\boldsymbol{k}$ and a translation through $\boldsymbol{p}$ is given by the following matrix [10]:

$$\boldsymbol{T} = \begin{bmatrix} k_x k_x V_\theta + C_\theta & k_y k_x V_\theta - k_z S_\theta & k_z k_x V_\theta + k_y S_\theta & p_x \\ k_x k_y V_\theta + k_z S_\theta & k_y k_y V_\theta + C_\theta & k_z k_y V_\theta - k_x S_\theta & p_y \\ k_x k_z V_\theta - k_y S_\theta & k_y k_z V_\theta + k_x S_\theta & k_z k_z V_\theta + C_\theta & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{28}$$

where $S_\theta = \sin\theta$, $C_\theta = \cos\theta$, and $V_\theta = 1 - \cos\theta$. The equivalent quaternion/vector pair operator, describing the same point vector transformation, has the form [4], [6], [7], [14]

$$\boldsymbol{Q} = \left(\left[\cos\frac{\theta}{2}, \sin\frac{\theta}{2} \langle k_x, k_y, k_z \rangle \right], \langle p_x, p_y, p_z \rangle \right) \tag{29}$$

where the quaternion $[\cos(\theta/2), \sin(\theta/2)\boldsymbol{k}]$ encodes the appropriate rotation, and the vector $\boldsymbol{p}$ denotes the corresponding translational displacement.

Appendix B

Mathematical Formulations of Basic Spatial Operations

Let $\boldsymbol{T} = [\boldsymbol{noap}]$ be a homogeneous transform, $\boldsymbol{Q} = (\boldsymbol{q}, \boldsymbol{p})$ a quaternion/vector pair with $\boldsymbol{q} = [s, \boldsymbol{v}]$, and $\boldsymbol{r}$ an arbitrary point vector. Table III gives the mathematical formulations of some basic spatial operations for both homogeneous transforms and quaternion/vector pairs.

The only homogeneous transform operation above that requires commentary is the normalization operation. Under the storage convention of Section II-A, only the orientation and approach vectors (i.e., $\boldsymbol{o}$ and $\boldsymbol{a}$) of the rotational matrix to be normalized are available. We assume that the spatial orientation of the $\boldsymbol{o}$-$\boldsymbol{a}$ plane is correct and reconstruct the normal vector as $\boldsymbol{n} = \boldsymbol{o} \times \boldsymbol{a}$. We then distribute the orthogonality error between the orientation and approach vectors $\boldsymbol{o}$ and $\boldsymbol{a}$ by setting $\boldsymbol{a}' = \boldsymbol{a} + (\boldsymbol{n} \times \boldsymbol{o})$ and $\boldsymbol{o}' = \boldsymbol{a}' \times \boldsymbol{n}$. Finally, the new orientation and approach vectors are normalized. The resulting normal vector is not explicitly stored and is denoted by ? in Table III (see [5] for more detail).

TABLE III
Basic Spatial Operations with T and Q

1. Spatial transformation of a point vector	
$\mathbf{r}' = [\mathbf{n\,o\,a\,p}][\mathbf{r}]$	$\mathbf{r}' = \mathbf{q} * \mathbf{r} * \mathbf{q}^{-1} + \mathbf{p}$
2. Composition of spatial operators	
$\mathbf{T}'' = [\mathbf{n\,o\,a\,p}][\mathbf{n}'\,\mathbf{o}'\,\mathbf{a}'\,\mathbf{p}']$	$\mathbf{Q}'' = (\mathbf{q}, \mathbf{p}) * (\mathbf{q}', \mathbf{p}') = (\mathbf{q} * \mathbf{q}', \mathbf{q} * \mathbf{p}' * \mathbf{q}^{-1} + \mathbf{p})$
3. Inverse of a spatial operator	
$\mathbf{T}^{-1} = [[\mathbf{n\,o\,a}]^T - ([\mathbf{n\,o\,a}]^T[\mathbf{p}])]$	$\mathbf{Q}^{-1} = (\mathbf{q}^{-1}, -\mathbf{q}^{-1} * \mathbf{p} * \mathbf{q})$
4. Normalization of a spatial operator	
$\mathbf{T}' = [?\ \frac{\mathbf{a}' \times \mathbf{n}}{\lvert\mathbf{a}' \times \mathbf{n}\rvert}\ \frac{\mathbf{a}'}{\lvert\mathbf{a}'\rvert}\ \mathbf{p}]$ where $\mathbf{n} = \mathbf{o} \times \mathbf{a}$, and $\mathbf{a}' = \mathbf{a} + (\mathbf{n} \times \mathbf{o})$	$\mathbf{Q}' = (\mathbf{q}', \mathbf{p})$ where $\mathbf{q}' = \frac{\mathbf{q}}{\sqrt{s^2 + \lvert\mathbf{v}\rvert^2}}$

On the quaternion/vector side, we have let the symbol '$*$' denote quaternion multiplication, which is defined as [6]

$$\boldsymbol{q} * \boldsymbol{q}' = [s, \boldsymbol{v}] * [s', \boldsymbol{v}'] = [ss' - \boldsymbol{v} \cdot \boldsymbol{v}', s\boldsymbol{v}' + s'\boldsymbol{v} + \boldsymbol{v} \times \boldsymbol{v}'].$$

The inverse of a quaternion is obtained by simply negating its vector part, i.e.

$$\boldsymbol{q}^{-1} = [s, \boldsymbol{v}]^{-1} = [s, -\boldsymbol{v}].$$

Finally, rotation of a vector $\boldsymbol{r}$ through a quaternion $\boldsymbol{q} = [s, \boldsymbol{v}]$ can be expressed as follows:

$$\boldsymbol{q} * \boldsymbol{r} * \boldsymbol{q}^{-1} = \boldsymbol{r} + 2s(\boldsymbol{v} \times \boldsymbol{r}) + 2\boldsymbol{v} \times (\boldsymbol{v} \times \boldsymbol{r}).$$

The reader is refered to [5] for a more detailed discussion of the above operations. Other references include [2], [4], [6], [7], [12], and [14].

References

[1] W. Blaschke, *Gesammelte Werke*. Essen: Thales-Verlag, 1982.

[2] O. Bottema and B. Roth, *Theoretical Kinematics*. New York: Elsevier-North Holland, 1979, pp. 518-525.

[3] J. Craig, *Introduction to Robotics, Mechanics and Control* (2nd ed.). Reading, MA: Addison-Wesley, 1989.

[4] O. F. Fischer, *Five Mathematical Structural Models in Natural Philosophy with Technical Physical Quaternion*. Stockholm: Axion Institute, 1957, ch. 1, 2.

[5] J. Funda, "Quaternions and homogeneous transforms in robotics," Master's thesis, Univ. Pennsylvania, 1988, MS-CIS-88-06.

[6] W. R. Hamilton, *Elements of Quaternions, Volume I and II*. New York: Chelsea, 1869.

[7] W. R. Hamilton, *Lectures on Quaternions*. London: Whittaker, 1853.

[8] C. S. G. Lee, "Robot arm kinematics, dynamics, and control," *IEEE Trans. Comput.*, pp. 62-80, Dec. 1982.

[9] E. A. Maxwell, *General Homogeneous Coordinates in Space of Three Dimensions*. Cambridge, England: Cambridge Univ. Press, 1951.

[10] R. P. Paul, *Robot Manipulators: Mathematics, Programming and Control*. Cambridge, MA: MIT Press, 1981.

[11] R. P. Paul and H. Zhang, "Computationally efficient kinematics for manipulators with spherical wrists based on the homogeneous transform representation," *Int. J. Robotics Res.*, vol. 5.2, 1986.

[12] E. Pervin and J. A. Webb, "Quaternions in vision and robotics," Dept. Comput. Sci., Carnegie-Mellon Univ., Tech. Rep. CMU-CS-82-150, 1982.
[13] E. Salamin, "Application of quaternions to computation with rotations," Tech. Rep., AI Lab, Stanford Univ., 1979.
[14] K. Shoemake, *Animating Rotation with Quaternion Curves.* Binghamton, NY: Singer Co., Link Flight Simulation Div., 1985.
[15] R. H. Taylor, "Planning and execution of straight line manipulator trajectories," *IBM J. Res. Development*, vol. 23, no. 4, pp. 424-436, July 1979.
[16] A. T. Yang and F. Freudenstein, Application of dual-number quaternion algebra to the analysis of spatial mechanisms." *Trans. ASME, J. Appl. Mechanics*, vol. 31, pp. 300-308, 1964.

A Proof of the Structure of the Minimum-Time Control Law of Robotic Manipulators Using a Hamiltonian Formulation

YAOBIN CHEN, MEMBER, IEEE, AND
ALAN A. DESROCHERS, SENIOR MEMBER, IEEE

Abstract—**A Hamiltonian Canonical formulation, which yields a new and very straightforward proof of the structure of the minimum-time control (MTC) law for *m*-link robotic manipulators is used. It is shown that the structure of the MTC law requires that at least one of the actuators is always in saturation. In addition, a numerical algorithm is presented. The algorithm converts the original problem, possibly a partially singular one, into a totally nonsingular optimal control problem by introducing a perturbed energy term in the performance index. It is shown that the solution to the perturbed problem converges to that of the MTC problem in the sense of the performance index as the perturbation parameter approaches zero. The control algorithm is then used in a simulation to verify the MTC law structure.**

I. Introduction

The problem of minimum-time control (MTC) of robot manipulators is concerned with the determination of control signals that will drive the manipulator from a given initial configuration to a given final configuration in as short a time as possible. In general, it is extremely difficult, if not impossible, to obtain an exact closed-form solution to this problem due mainly to the highly nonlinear and coupled nature of the robotic manipulator dynamics.

The general time-optimal control problem has been treated by many authors [1], [2]. The theoretical results for the general system are in the form of necessary and sufficient conditions that have to be satisfied by the candidate optimal solution. It will be shown in this work that in general, for robotic manipulators, the singular optimal control does not exist, and the so-called "partially singular control," which will be defined later, may exist. Research in singular control problems [3] has been directed towards finding some additional necessary conditions that the controls have to satisfy, obtaining the singular optimal trajectories in the interval, and characterizing singular extremals.

Several researchers have addressed the general point-to-point MTC problem for robot manipulators [4], [12], [18]. In this work, a Hamiltonian canonical formulation is used that not only yields a new and very straightforward proof of the structure of the minimum-time control law but also makes the numerical solution easier. It will be shown that the structure of the MTC law requires that at least one of the actuators always be in saturation over every finite time interval, whereas the others adjust their torques so that some constraints on motion are not violated while enabling the arm to reach its final desired destination. Then, singular and partially singular extremal trajectories are characterized for the purpose of finding the nonbang-bang control law.

Solving this partially singular control problem has proven to be difficult [5]. In this work, we present a reliable method [15] for solving the two-point boundary value problem (TPBVP) arising from the MTC law structure. Computer simulations for a two-degree of freedom (DOF) arm are performed to verify our theoretical result of the MTC structure for robot systems.

Manuscript received January 28, 1988; revised March 12, 1990.

Y. Chen is with the Electrical Engineering and Computer Science Department, George Washington University, Washington, DC 20052.
A. A. Desrochers is with the Electrical, Computer, and Systems Engineering Department and the Center for Intelligent Robotic Systems for Space Exploration, Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

IEEE Log Number 9036836.

II. Structure of Minimum-Time Control Law

A. Dynamic Model of an m-DOF Robot Arm

We first assume that the torques applied to each joint are bounded and can take any values between the bounds. Using the Hamiltonian canonical equations, the dynamic equations of an m-degree-of-freedom robot arm can be derived.

The total kinetic energy of the robot arm in the joint space is expressed as

$$K(q, \dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{1}$$

where $q(t)$ is the $m \times 1$ generalized joint coordinate vector, and $M(q)$ is the $m \times m$ total inertial matrix of the system. In addition, let the total potential energy of the system be represented by $P(q)$. Then, the Lagrangian of the system is written as

$$L(q, \dot{q}) = K(q, \dot{q}) - P(q) = \frac{1}{2}\dot{q}^T M(q)\dot{q} - P(q). \tag{2}$$

Now define a Hamiltonian function H as

$$\bar{H} = [\dot{q}(t)]^T r(t) - L(q, \dot{q}) \tag{3}$$

where $r(t)$ is a generalized momentum vector defined as the partial derivative of the Lagrangian $L(q, \dot{q})$ with respect to the joint velocity vector $\dot{q}$. By (2) the generalized momentum can be expressed as

$$r(t) = M(q)\dot{q}(t). \tag{4}$$

From (2)-(4) and the symmetric positive definite property of the inertia matrix, we obtain

$$\bar{H} = \frac{1}{2}r^T(t)M^{-1}(q)r(t) + P(q). \tag{5}$$

Using the Hamiltonian canonical equations [13], a set of $2m$ differential equations can be written as

$$\frac{dq(t)}{dt} = M^{-1}(q)r(t)$$
$$\frac{dr(t)}{dt} = -\frac{1}{2}(I \otimes r^T(t))\frac{\partial M^{-1}(q)}{\partial q}r(t) - FM^{-1}(q)r(t) + G(q) + u(t) \tag{6}$$

where $\otimes$ is the Kronecker product, F is the $m \times m$ viscous frictional matrix, and $u(t)$ is a control torque vector applied to the joints. Now define the generalized coordinate vector $q(t)$ and the generalized momentum vector $r(t)$ as states, i.e., $x^T(t) = (x_1^T(t), x_2^T(t)) =$