CUBIC SPLINE ALGORITHMS FOR ORIENTATION INTERPOLATION

I. G. KANG AND F. C. PARK *

School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, South Korea

SUMMARY

This article presents a class of spline algorithms for generating orientation trajectories that approximately minimize angular acceleration. Each algorithm constructs a twice-differentiable curve on the rotation group SO(3) that interpolates a given ordered set of rotation matrices at specified knot times. Rotation matrices are parametrized, respectively, by the unit quaternion, canonical co-ordinate, and Cayley–Rodrigues representations. All the algorithms share the common feature of (i) being invariant with respect to choice of fixed and moving frames (bi-invariant), and (ii) being cubic in the parametrized co-ordinates. We assess the performance of these algorithms by comparing the resulting trajectories with the minimum angular acceleration curve. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: orientation, rotation; SO(3); interpolation; cubic spline

1. INTRODUCTION

A problem that frequently arises in a wide range of engineering applications, e.g. computer graphics and animation, machine vision, satellite attitude control, multibody system dynamics and continuum mechanics, is the interpolation of smooth curves on the space of orientations, also known as the Special Orthogonal Group SO(3). In this article we address the following problem: given an ordered set of \( n + 1 \) rotation matrices \( \{ R_0, R_1, \ldots, R_n \} \), and a set of \( n + 1 \) scalars \( t_0 < t_1 < \cdots < t_n \), construct a twice-differentiable curve \( R(t) \) on SO(3) such that \( R(t_i) = R_i \), \( i = 0, 1, \ldots, n \).

Early approaches to orientation interpolation involved choosing some local co-ordinate representation for rotation matrices (e.g. Euler angles, roll-pitch-yaw angles) and applying existing vector space spline algorithms to these local co-ordinates. Although such approaches can offer algorithmic simplicity and computational efficiency, one of their obvious drawbacks is that, unlike methods based on minimizing physically meaningful quantities such as kinetic energy or angular acceleration, the resulting orientation trajectories will in general depend on the choice of local co-ordinates. When the underlying space is curved, as SO(3) is, often the price that one pays for insisting on co-ordinate invariance is a substantial increase in the amount of computation. In the case of
minimum angular acceleration curves, for example, one must solve a three-dimensional, fourth-order nonlinear two-point boundary value problem. For applications where speed, accuracy, and flexibility are all required together in some measure, the practical recourse is to relax the co-ordinate-invariance requirement, and search for a local co-ordinate-based solution that most closely approximates the desired physically inspired trajectory.

In the case of orientations, an additional issue that is arguably even more important than co-ordinate-invariance is the bi-invariance property. Mathematically, given two ordered sets of rotation matrices \( \{ R_0, \ldots, R_n \} \) and \( \{ \hat{R}_0, \ldots, \hat{R}_n \} \), related by \( \hat{R}_i = QR_iS \), \( i = 0, 1, \ldots, n \), where \( Q \) and \( S \) are two constant rotation matrices, the two interpolating curves through these two sets, denoted \( R(t) \) and \( \hat{R}(t) \), respectively, should then be related by \( \hat{R}(t) = QR(t)S \). In the context of rigid bodies, physically this relation reflects the fact that the orientation component of the rigid body’s motion should not be influenced by the choice of either the fixed (i.e. global) or moving (i.e. attached to the body) reference frames.

In this article we present a class of algorithms for efficiently generating \( C^2 \) bi-invariant trajectories on \( SO(3) \), and compare their performance vis-à-vis the minimum angular acceleration curve. The algorithms are based on three popular co-ordinate parametrizations for rotation matrices: the unit quaternion, Cayley–Rodrigues parameter, and canonical co-ordinate representations. What these three representations share in common is that they are all based on the idea of parametrizing a rotation by its axis of rotation, along with the angle of rotation about the axis. As such these representations are not only physically intuitive, but also overcome many of the well-documented disadvantages of Euler angles, e.g. the non-uniform distribution of singularities.

In the first part of the article we focus on the intuitive geometric interpretation and the relation between these various orientation representations. Detailed formulas—some well known, some obscure, some new—are also derived for the angular velocities and accelerations in fixed and body frame co-ordinates in terms of the three representations. In the second part we present three cubic spline interpolation algorithms based on these representations. Particular attention is paid to how well the resulting orientation trajectories approximate the minimum angular acceleration curve, in the same sense that Euclidean cubic splines can be viewed as an approximation to minimum curvature, or minimum bending energy, curves.

The algorithms presented in this paper are particularly useful for applications that require flexible and efficient generation of orientation trajectories. For example, in computer animation, generating physically realistic trajectories by integrating the dynamic equations of motion can be computationally very demanding; a more practical approach might be to apply the orientation splines proposed in this paper to minimize the total energy of the rigid bodies’ motions. Such an approach is also useful for, e.g. variational problems in continuum mechanics, in which the solution can be found by optimizing the energy function with respect to the orientation spline knot points. Another possible application is in tool path planning for machine tools and robot manipulators: by interactively specifying the orientation knot points and applying the interpolation algorithms considered here, smooth orientation trajectories can be efficiently and rapidly generated.

Most of the recent literature on \( SO(3) \) curve design focuses on methods for designing freeform \( SO(3) \) curves using extensions of De Casteljau’s subdivision algorithm (which in turn can be regarded as a generalization of Bézier curves to \( SO(3) \) [1–5]. These curves have the advantage of being time reversible—that is, the shape of the curve is unchanged if the order of the control points is reversed—often considered a desirable feature for interactive animation. However, these algorithms are quite cumbersome for interpolation applications, particularly when new intermediate orientations are added; in this case the control points must be suitably adjusted to ensure that the
resulting trajectory passes through all the required orientations. De Casteljau algorithms on SO(3) are also inherently computationally very demanding, and adding more control points limits even further their effectiveness for interactive applications. Another feature of many of the algorithms in the literature [6–8] are that they are formulated almost exclusively in terms of the unit quaternions, due in large part to the influential paper of Shoemake [1], which appears to have established the unit quaternions as the de rigeur choice for SO(3) curve design.

The paper is organized as follows. In Section 2 we review the geometric interpretation and relation between the unit quaternion, canonical co-ordinate, and Cayley–Rodrigues parameter representations. In Section 3 we present bi-invariant interpolation algorithms based on these representations, and in Section 4 we assess their performance using the minimum angular acceleration curve as a benchmark.

2. ROTATION REPRESENTATIONS

Recall that the group of rotation matrices, or the Special Orthogonal Group SO(3), is defined as

\[ \text{SO}(3) = \{ \mathbf{R} \in \mathbb{R}^{3 \times 3} \mid \mathbf{R} \mathbf{R}^T = \mathbf{R}^T \mathbf{R} = \mathbf{I} \text{ and } \det \mathbf{R} = 1 \} \]  

Since SO(3) is a three-dimensional manifold, it can be locally parametrized by three parameters. Euler angles and roll-pitch-yaw angles are perhaps the most well known, but because of problems associated with singularities these co-ordinates are not widely used in engineering applications. Rather, the co-ordinates popularly used today all derive from the following physical interpretation of rotations. As is well known, any arbitrary rotation can be specified by an axis of rotation (which we represent by the unit vector \( \mathbf{r} \)), together with the rotation angle (\( \phi \), measured in the right-hand sense) about that axis. The parametrizations that we consider are all based on various normalizations and combinations of \( \mathbf{r} \) and \( \phi \).

2.1. Canonical co-ordinates

The canonical co-ordinates are a three-parameter representation for rotations obtained by multiplying \( \mathbf{r} \) and \( \phi \): \( \mathbf{r} = \hat{\mathbf{r}} \phi \). If we express \( \mathbf{r} = (r_1, r_2, r_3) \) as the skew-symmetric matrix

\[ [\mathbf{r}] = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \]  

then the corresponding rotation matrix \( \mathbf{R} \) can be obtained from \([\mathbf{r}]\) via the following formula:

\[ e^{[\mathbf{r}]} = \mathbf{I} + \frac{\sin \| \mathbf{r} \|}{\| \mathbf{r} \|} \cdot [\mathbf{r}] + \frac{1 - \cos \| \mathbf{r} \|}{\| \mathbf{r} \|^2} \cdot [\mathbf{r}]^2 \]  

where \( \| \mathbf{r} \| \) is the standard Euclidean norm. Conversely, if \( \mathbf{R} \in \text{SO}(3) \) such that \( \text{Tr}(\mathbf{R}) \neq -1 \), then

\[ \log \mathbf{R} = \frac{\phi}{2 \sin \phi} (\mathbf{R} - \mathbf{R}^T) \]  

where \( \phi \) satisfies \( 1 + 2 \cos \phi = \text{Tr}(\mathbf{R}) \) and \( \| \log \mathbf{R} \|^2 = \phi^2 \). If we restrict \( \phi \) to be between 0 and \( \pi \), then in the case when \( \text{Tr}(\mathbf{R}) = -1 \) two solutions for \( \log \mathbf{R} \) exist: if \( \hat{\mathbf{v}} \) is a unit length eigenvector of \( \mathbf{R} \) associated with the eigenvalue 1, then \( \log \mathbf{R} = \pm \pi [\hat{\mathbf{v}}] \).
The canonical co-ordinates suggest a geometric picture of SO(3) as a three-dimensional solid ball of radius $\pi$, centred at the origin, with antipodal points identified: a point $r$ in the ball represents a rotation by $\|r\|$ radians (in the right-hand sense) about the line directed from the origin through $r$. The exponential and logarithm give explicit formulas for this solid ball representation of SO(3). Two remarks are in order. First, this representation is unique only when restricted to the interior of the solid ball, as antipodal points on the boundary of the solid ball clearly correspond to the same physical rotation. Second, because of the $2\pi$ periodicity of rotations, it follows that if $[r]$ is one solution to $\log R$, then so is $[r](1+2\pi k/\|r\|)$ for any integer $k$.

It is also known that if $R(t)$ represents the orientation trajectory of a rotating rigid body, then $R^T \dot{R}$ is the angular velocity (expressed as a skew-symmetric matrix) of the rigid body expressed in moving frame co-ordinates, whereas $\dot{R}R^T$ is the angular velocity expressed in fixed frame co-ordinates. Explicit formulas for these quantities can be derived in terms of the canonical co-ordinates, e.g.

$$\omega_0(t) = A(r)\dot{r}$$

where

$$A(r) = I - \frac{1 - \cos \|r\|}{\|r\|^2} [r] + \frac{\|r\| - \sin \|r\|}{\|r\|^3} \|r\|^2 [r]$$

Similarly, the angular velocity with respect to the fixed frame, denoted $\omega_1(t)$, is given by $\omega_1(t) = B(r)\dot{r}$, with

$$B(r) = I + \frac{1 - \cos \|r\|}{\|r\|^2} [r] + \frac{\|r\| - \sin \|r\|}{\|r\|^3} \|r\|^2 [r]$$

It can be verified that both $A$ and $B$ are non-singular for all $r$.

By differentiating the angular velocity vector $\omega_0(t)$ of equations (5) and (6), one obtains the angular acceleration vector $\alpha_0(t)$ relative to the moving frame, expressed in terms of the canonical co-ordinates:

$$\alpha_0(t) = \ddot{r} - \frac{\dot{r} \dot{r}}{\|r\|^2} \left(2 \cos \|r\| + \|r\| \sin \|r\| - 2\|r\| \times \dot{r} \right) - \frac{1 - \cos \|r\|}{\|r\|^2} (r \times \dot{r})$$

$$+ \frac{\dot{r} \dot{r}}{\|r\|^3} \left(3 \sin \|r\| - \|r\| \cos \|r\| - 2\|r\| \times \dot{r} \right) (r \times \dot{r})$$

$$+ \frac{\|r\| - \sin \|r\|}{\|r\|^3} (\dot{r} \times (r \times \dot{r}) + r \times (r \times \dot{r}))$$

We refer the reader to Park and Ravani [9] for a derivation and further discussion of the above formulas.

2.2. Cayley–Rodrigues parameters

The Cayley–Rodrigues parameters (more commonly referred to in the kinematics literature as simply the Rodrigues parameters, see, e.g. [10]), are another three-parameter representation obtained from $\dot{r}$ and $\phi$ by setting $r = \dot{r} \tan \phi/2$. Geometrically, this parametrization has the effect of ‘stretching’ out to infinity the solid ball of radius $\pi$ as described in the previous section. These
parameters can be derived from a general formula attributed to Cayley that is also valid for rotation matrices of arbitrary dimension (see, e.g. [11]): if \( R \in \text{SO}(3) \) such that \( \text{tr}(R) \neq -1 \), then

\[
(I - R)(I + R)^{-1} = [r]
\]

(9)
is skew-symmetric. Conversely, \( R \) can be obtained from \([r]\) as follows:

\[
R = (I - [r])(I + [r])^{-1}
\]

(10)
The above two formulas establish a one-to-one correspondence between \( 3 \times 3 \) skew-symmetric matrices (or equivalently \( \mathbb{R}^3 \)) and those elements of \( \text{SO}(3) \) with trace not equal to \(-1\). In the event that \( \text{tr}(R) = -1 \), the following alternative formulas can be used to relate orthogonal (this time excluding those with unit trace) and skew-symmetric matrices in a one-to-one fashion:

\[
R = -(I - [r])(I + [r])^{-1}
\]

(11)

\[
[r] = (I + R)(I - R)^{-1}
\]

(12)

Equation (10) can be explicitly computed as

\[
R = \frac{(1 - r^T r)I + 2rr^T + 2[r]}{1 + r^T r}
\]

(13)
and its inverse mapping is given by

\[
[r] = \frac{R - R^T}{1 + \text{tr}(R)}
\]

(14)
for all \( R \in \text{SO}(3) \) such that \( \text{tr}(R) \neq -1 \). The vector \( r = 0 \) therefore corresponds to the identity matrix, and \(-r\) represents the inverse of the rotation corresponding to \( r \). The following two identities also follow from the above formulas:

\[
1 + \text{tr}(R) = \frac{4}{1 + r^T r}
\]

(15)

\[
R - R^T = \frac{4[r]}{1 + r^T r}
\]

(16)

One of the attractive features of the Cayley–Rodrigues parameters is the particularly simple form for the composition of two rotation matrices. If \( r_1 \) and \( r_2 \) denote the Cayley–Rodrigues parameter representations of the rotation matrices \( R_1 \) and \( R_2 \), respectively, then the Cayley–Rodrigues parameter representation for \( R_3 = R_1 R_2 \), denoted by \( r_3 \), is given by

\[
r_3 = \frac{r_1 + r_2 + (r_1 \times r_2)}{1 - r_1^T r_2}
\]

(17)

In the event that \( r_1^T r_2 = 1 \), or equivalently \( \text{tr}(R_1 R_2) = -1 \), the following alternative composition formula can be used. Define

\[
s = \frac{r}{\sqrt{1 + r^T r}}
\]

(18)
so that the rotation corresponding to \( r \) can be written

\[
R = I + 2\sqrt{1 - s^T s} s \left[ s \right] + 2[s]^2
\]

(19)
The direction of \( s \) coincides with that of \( r \), and \( \|s\| = \sin \phi/2 \). The composition law now becomes
\[
s_3 = s_1 \sqrt{1 - s_1^2} + s_2 \sqrt{1 - s_2^2} + (s_1 \times s_2)
\] (20)

We now discuss representations for the angular velocity and angular acceleration in terms of the Cayley–Rodrigues parameters. If \( r(t) \) denotes the Cayley–Rodrigues parameter representation of the orientation trajectory \( R(t) \), then in vector form
\[
\omega_s = 2 \frac{1 + \|r\|^2}{} (r \times \dot{r} + \dot{r})
\] (21)
\[
\omega_b = 2 \frac{1 + \|r\|^2}{} (-r \times \dot{r} + \dot{r})
\] (22)

The angular acceleration with respect to the inertial and body-fixed frames can now be obtained by time differentiating the above expressions:
\[
\dot{\omega}_s = 2 \frac{1 + \|r\|^2}{} (r \times \ddot{r} + \dot{r})
\] (23)
\[
\dot{\omega}_b = 2 \frac{1 + \|r\|^2}{} (-r \times \ddot{r} + \dot{r})
\] (24)

We refer the reader to Mladenova [12] for a derivation and further discussion of these formulas.

2.2.1. Higher-order Cayley transforms. The Cayley transform of equation (10) can be generalized to higher-order as follows [13]:
\[
R = (I - [r])^k (I + [r])^{-k}
\] (25)

For the case \( k = 2 \), the rotation \( R \) corresponding to \( r \) can be computed from the formula [14]
\[
R = I + 4 \frac{1 - r^T r}{(1 + r^T r)^2} [r] + \frac{8}{(1 + r^T r)^2} [r]^2
\] (26)

Conversely, given a rotation matrix \( R \), a vector \( r \) that satisfies equation (26) can be obtained as
\[
r = \tan(\phi/4) \hat{r}
\] (27)

where, as before, \( \hat{r} \) is the unit vector corresponding to the axis of rotation for \( R \), and \( \phi \) is the corresponding rotation angle. We note that this solution is not unique: another solution is given by
\[
r = -\frac{1}{\tan(\phi/4)} \hat{r}
\] (28)

This three-parameter representation for rotations is also commonly referred to in the literature as the modified Cayley–Rodrigues parameters [15]. The angular velocity in the body-fixed frame obeys the following relation:
\[
\dot{r} = \frac{1}{4} \{(1 - r^T r) [I + 2[r] + 2rr^T] \omega\}
\] (29)

One of the advantages of the modified Cayley–Rodrigues parameters is that the singularity at 180° is now relocated to 360°; rotations up to 360° are now possible. Viewed from the canonical
co-ordinate perspective, the modified Cayley–Rodrigues parameters can be obtained by ‘stretching’ the solid ball of radius $2\pi$ (as opposed to $\pi$ for the standard Cayley–Rodrigues parameters corresponding to the $k = 1$ case) to infinity. However, one now loses the one-to-one correspondence between $\mathbb{R}^3$ and $SO(3)$ that exists for the standard Cayley–Rodrigues parameters. Moreover, (i) the formulas for the angular velocity and acceleration become more complicated, (ii) one cannot obtain $r$ from $R$ by a simple rational expression as in the case of the standard Cayley–Rodrigues parameters, and (iii) multiplication of two rotation matrices in the modified parameters does not admit a simply rational expression like the standard parameters.

Going to higher order, for the case $k = 4$ it can be shown that the corresponding $r$ is given by

$$r = \tan(\phi/8)\tilde{r}$$

(30)

As $k$ increases, one obtains successively closer approximations (up to constant scaling factor) to the canonical co-ordinates—the non-linear warping effect caused by the tangent function becomes less severe. As expected, however, the formulas are no longer simple rational expressions, but become increasingly complicated expressions involving transcendental functions. Because of the disadvantages mentioned above, and in light of our emphasis on numerical efficiency, we believe that for our specific application higher-order Cayley transforms do not offer any distinct advantages vis-à-vis the standard Cayley–Rodrigues parameters or the canonical co-ordinates. However, the interpolation algorithm based on the standard Cayley–Rodrigues parameters can be straightforwardly generalized to these higher-order parameters once explicit formulas for the angular velocities and accelerations are derived.

### 2.3. Unit quaternions

The unit quaternion representation for rotations is obtained by encoding the rotation axis and angle information as a four-dimensional vector:

$$(q_0, q_1, q_2, q_3) = \left( \frac{\cos \frac{\phi}{2}}{\sqrt{2}}, \frac{r_1 \sin \frac{\phi}{2}}{\sqrt{2}}, \frac{r_2 \sin \frac{\phi}{2}}{\sqrt{2}}, \frac{r_3 \sin \frac{\phi}{2}}{\sqrt{2}} \right)$$

(31)

However, another possible representation for the same rotation is given by the negative of the original vector, i.e. $(-q_0, -q_1, -q_2, -q_3)$. This two-to-one nature of the unit quaternion representation must always be kept in mind when using these co-ordinates to represent rotations; we comment on this further below.

The unit quaternions can, in fact, be identified with the Special Unitary Group SU(2), which is the set of $2 \times 2$ complex unitary matrices with unit determinant: any $M \in SU(2)$ satisfies $M^*M = MM^* = I$ and $\text{det} M = 1$, where $M^*$ denotes the complex conjugate transpose of $M$. From the definition it follows that $M$ is of the form

$$M = \begin{bmatrix} \alpha & \beta \\ -\bar{\beta} & \bar{\alpha} \end{bmatrix}$$

(32)

where $^*$ denotes complex conjugation. If $\alpha = q_0 + iq_1$ and $\beta = q_2 + iq_3$, then from the definition we must have $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. Hence, $SU(2)$ can also be identified with the unit 3-sphere in $\mathbb{R}^4$. The identification with the unit quaternions can be made by writing $M \in SU(2)$ as $q_0 + q_1i + q_2j + q_3k$. As seen from above, the rather unintuitive rule for quaternion multiplication simply turns out to be group multiplication in $SU(2)$.
Also, antipodal points on the unit 3-sphere correspond to the same rotation matrix. Explicit formulas for the angular velocity and acceleration can be derived by calculating $M^{-1}M$ and $MM^{-1}$ and its derivatives. For example, the angular velocity in body-fixed co-ordinates is given by

$$\omega_0 = 2\begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$ (33)

and the angular acceleration can be obtained by differentiating $\omega_0$.

### 3. INTERPOLATION ALGORITHMS

In this section we provide pseudo-code descriptions of the orientation interpolation algorithms based on the three co-ordinate representations discussed. To fix notation we restate the mathematical statement of the problem: given an ordered set of $n+1$ rotation matrices $\{R_0, R_1, \ldots, R_n\}$, and a set of $n+1$ scalars $t_0 < t_1 < \cdots < t_n$, find a twice-differentiable curve $R(t)$ on $SO(3)$ such that $R(t_i) = R_i, i = 0, 1, \ldots, n$. For convenience we assume that the initial angular velocity $\omega_0 \in \mathbb{R}^3$ and the initial angular acceleration $\alpha_0 \in \mathbb{R}^3$ are specified as boundary conditions, both expressed in terms of body-fixed frame co-ordinates. In particular, specifying both these quantities to zero is a popular choice in many applications; in the spline literature such interpolating curves are called natural cubic splines [16].

#### 3.1. Canonical co-ordinates

The interpolation algorithm for canonical co-ordinates is as follows (see [9] for a derivation and complete discussion):

1. **Given.**
   - $\{R_0, R_1, \ldots, R_n\} = n+1$ rotation matrices satisfying $\text{Tr}(R_i^T R_0) \neq -1, i = 1, \ldots, n$
   - $\{t_0, t_1, \ldots, t_n\} = \text{knot times}$
   - $\omega_0 = \text{angular velocity at } t_0$ in body-fixed co-ordinates
   - $\alpha_0 = \text{angular acceleration at } t_0$ in body-fixed co-ordinates

2. **Preprocessing.** For $i = 1$ to $n$ do
   $$[r_i] = \log(R_i^T R_i)$$

3. **Initialization.**
   - $c_1 = \omega_0$
   - $b_1 = \alpha_0/2$
   - $a_1 = r_1 - b_1 - c_1$
(4) **Iteration.** For $i = 2$ to $n$ do

\[ s = r_i \quad \text{(temporary variable)} \]

\[ t = 3a_{i-1} + 2b_{i-1} + c_{i-1} \quad \text{(temporary variable)} \]

\[ u = 6a_{i-1} + 2b_{i-1} \quad \text{(temporary variable)} \]

\[ c_i = \left( 1 - \frac{1 - \cos \|s\|}{\|s\|^2} [s] + \frac{\|s\| - \sin \|s\|}{\|s\|^3} [s]^2 \right) t \]

\[ b_i = \frac{1}{2} \left( u - \frac{s^T t}{\|s\|^2} (2 \cos \|s\| + \|s\| \sin \|s\| - 2)(s \times t) - \frac{1 - \cos \|s\|}{\|s\|^2} (s \times u) \right. \]

\[ + \frac{s^T t}{\|s\|^2} (3 \sin \|s\| - \|s\| \cos \|s\| - 2\|s\|)(s \times (s \times t)) \]

\[ \left. + \frac{\|s\| - \sin \|s\|}{\|s\|^3} (t \times (s \times t) + s \times (s \times u)) \right) \]

\[ a_i = s - b_i - c_i \]

(5) **Result.** For $t_{i-1} \leq t \leq t_i$

\[ \tau = \frac{t - t_{i-1}}{t_i - t_{i-1}} \]

\[ R(t) = R_{i-1} e^{a_i t^3 + b_i t^2 + c_i t} \]

Formulas for the matrix exponential and logarithm are given by equations (3) and (4), respectively.

### 3.2. Cayley–Rodrigues parameters

Before presenting the pseudo-code description of the algorithm we first consider the problem of interpolating between two orientations using Cayley parameters [17]. The goal is to find a curve $R(t)$ in SO(3) that satisfies the boundary conditions $R(0) = R_0$, $R(1) = R_1$, and $R^T(0)R(0) = [\omega_0]$, and $R^T(1)R(1) = [\omega_1]$, where $R_0, R_1 \in SO(3)$ such that $\text{tr}(R_0^T R_1) \neq -1$, and $\omega_0, \omega_1 \in \mathbb{R}^3$ are given (i.e. angular velocity vectors expressed in body-fixed frame co-ordinates).\(^1\)

The class of admissible curves we consider are left-invariant, i.e. of the form

\[ R(t) = R_0(I - [r(t)])(I + [r(t)])^{-1} \]

(34)

Left-invariance assures that $R(t)$ is invariant with respect to left translations of the knot points by some constant rotation matrix. We furthermore require that

\[ r(t) = at^3 + bt^2 + ct \]

(35)

with the coefficients $a, b, c \in \mathbb{R}^3$ to be determined; $r(t)$ is therefore a three-dimensional cubic polynomial with zero constant term. (In the appendix we show that this curve is also right-invariant, and

\(^1\)In the event that $\text{tr}(R_0^T R_1) = -1$, the angle of rotation about the fixed axis that takes $R_0$ into $R_1$ is $\pm \pi$, so that there exists two shortest paths, or minimal geodesics, between $R_0$ and $R_1$. Rather than arbitrarily choose a particular minimal geodesic, we instead impose the modest requirement that $\text{tr}(R_0^T R_1) \neq -1$ in our interpolation algorithm.
hence bi-invariant.) From equations (13)–(14) and the above boundary conditions, the following equations uniquely determine the coefficients of \( r(t) \):

\[
\begin{align*}
 r(0) &= 0 \quad (36) \\
 [r(1)] &= \frac{1}{1 + \text{tr}(R^T_0 R_1)} (R^T_0 R_1 - R^T_1 R_0) \quad (37) \\
 \dot{r}(0) &= \frac{\omega_0}{2} \quad (38) \\
 \dot{r}(1) &= \frac{1 + \|r(1)\|^2}{2} (I - [r(1)])^{-1} \omega_1 \quad (39)
\end{align*}
\]

Observe that \((I - [r(1)])^{-1}\) always exists regardless of the value of \( r(1) \). Boundary condition (36) is automatically satisfied, and boundary condition (38) implies that \( c = \omega_0/2 \). \( a \) and \( b \) can now be determined by solving the two linear equations

\[
\begin{align*}
 a + b &= r(1) - \frac{\omega_0}{2} \quad (40) \\
 3a + 2b &= \frac{1 + \|r(1)\|^2}{2} (I - [r(1)])^{-1} \omega_1 - \frac{\omega_0}{2} \quad (41)
\end{align*}
\]

where \( r(1) \) is determined from (37). Once values for \( a, b, \) and \( c \) (and hence \( r(t) \)) are obtained, the interpolating curve \( R(t) \) can be determined from the closed-form formula of equation (13).

Using the above results on two-point interpolation, we now provide a pseudo-code description of the interpolation algorithm for multiple knot points:

(1) \textit{Given.}

\[
\{ R_0, \ldots, R_n \} = n + 1 \text{ rotation matrices satisfying} \\
\text{tr}(R^T_{i-1} R_i) \neq -1, \quad i = 1, \ldots, n
\]

\[
\{ t_0, \ldots, t_n \} = n + 1 \text{ knot times} \\
\omega_0 = \text{angular velocity at } t_0 \text{ in body-fixed co-ordinates} \\
\alpha_0 = \text{angular acceleration at } t_0 \text{ in body-fixed co-ordinates}
\]

(2) \textit{Preprocessing.} For \( i = 1 \) to \( n \) do

\[
[d_i] = \frac{R^T_{i-1} R_i - R^T_i R_{i-1}}{1 + \text{tr}(R^T_{i-1} R_i)}
\]

(3) \textit{Initialization.}

\[
\begin{align*}
 c_1 &= \omega_0 \\
 b_1 &= \frac{\alpha_0}{2} \\
 a_1 &= d_1 - c_1 - b_1
\end{align*}
\]

(4) \textit{Iteration.} For \( i = 2 \) to \( n \) do

\[
\begin{align*}
 w &= d_i \\
 v &= 3a_{i-1} + 2b_{i-1} + c_{i-1} \\
 u &= 6a_{i-1} + 2b_{i-1}
\end{align*}
\]
Once all the coefficients $a_i$, $b_i$, $c_i$, $i = 1, \ldots, n$ have been found, then for a given value of $t$, where $t_{i-1} \leq t < t_i$, the corresponding orientation $R(t)$ for the interpolating trajectory is found as follows:

\begin{align*}
\tau &= \frac{t - t_{i-1}}{t_i - t_{i-1}} \\
\mathbf{r} &= a_i \tau^3 + b_i \tau^2 + c_i \tau \\
R(t) &= R_{i-1} \left( \frac{(1 - \mathbf{r}^T \mathbf{r}) \mathbf{I} + 2\mathbf{r} \mathbf{r}^T + 2[\mathbf{r}]}{1 + \mathbf{r}^T \mathbf{r}} \right)
\end{align*}

### 3.3. Unit quaternions

The basic idea behind the unit quaternion-based method is quite straightforward: to perform cubic spline interpolation in $\mathbb{R}^4$, and then to project the resulting curve onto the unit three-sphere. The following is a pseudo-code description for the unit quaternion-based orientation interpolation algorithm.

1. **Given.**
   \[
   \{R_0, \ldots, R_n\} = n + 1 \text{ rotation matrices satisfying} \\
   \text{tr}(R_{i-1}^T R_i) \neq -1, \ i = 1, \ldots, n \\
   \{t_0, \ldots, t_n\} = n + 1 \text{ knot times} \\
   \omega_0 = \text{angular velocity at } t_0 \text{ in body-fixed co-ordinates} \\
   \alpha_0 = \text{angular acceleration at } t_0 \text{ in body-fixed co-ordinates}
   \]

2. **Preprocessing.** For $i = 0$ to $n$ do
   \[
   \phi = \frac{\text{tr}(R_{i-1}^T R_i) - 1}{2} \\
   [\psi] = R_{i-1}^T R_i - R_i^T R_{i-1} \\
   q_i = \begin{bmatrix}
   \sqrt{\frac{1 + \phi}{2}} \\
   \psi \\
   \sqrt{\frac{1 - \phi}{2}}
   \end{bmatrix}
   \]

3. **Initialization.**
   \[
   c_1 = \frac{1}{2} \begin{bmatrix}
   0 \\
   \omega_0
   \end{bmatrix}
   \]
\[ b_i = \frac{1}{4} \left( \begin{bmatrix} 0 \\ \varepsilon_0 \end{bmatrix} - 2 \Omega(c_i) c_i \right) \]

\[ a_i = q_i - b_i - c_i - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]

(4) Iteration. For \( i = 2 \) to \( n \) do

\[ s = q_i \quad \text{(temporary variable)} \]
\[ t = 3a_{i-1} + 2b_{i-1} + c_{i-1} \quad \text{(temporary variable)} \]
\[ u = 6a_{i-1} + 2b_{i-1} \quad \text{(temporary variable)} \]
\[ c_i = \Omega(s) t \]
\[ b_i = \frac{1}{2} (\Omega(t) t + \Omega(s) u - \Omega(c_i) c_i) \]
\[ a_i = s - b_i - c_i - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]

Once all the coefficients \( a_i, b_i, c_i, \) \( i = 1, \ldots, n \) have been found, then for a given value of \( t \), where \( t_{i-1} \leq t < t_i \), the corresponding orientation \( R(t) \) for the interpolating trajectory is found as follows:

(5) Result.

\[ \tau = \frac{t - t_{i-1}}{t_i - t_{i-1}} \]
\[ x = a_i \tau^3 + b_i \tau^2 + c_i \tau + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]
\[ R(t) = \frac{1}{x^T x} R_{i-1} \Theta(x) \]

\( \Omega(x) \) and \( \Theta(x) \), where \( x = (x_0, x_1, x_2, x_3) \), are defined as follows:

\[ \Omega(x) = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ -x_1 & x_0 & x_3 & -x_2 \\ -x_2 & -x_3 & x_0 & x_1 \\ -x_3 & x_2 & -x_1 & x_0 \end{bmatrix} \]

\[ \Theta(x) = \begin{bmatrix} x_0^2 + x_1^2 - x_2^2 - x_3^2 & 2(x_1 x_2 - x_3 x_0) & 2(x_1 x_3 + x_2 x_0) \\ 2(x_1 x_2 + x_3 x_0) & x_0^2 - x_1^2 + x_2^2 - x_3^2 & 2(x_2 x_3 - x_1 x_0) \\ 2(x_1 x_3 - x_2 x_0) & 2(x_2 x_3 + x_1 x_0) & x_0^2 - x_1^2 - x_2^2 + x_3^2 \end{bmatrix} \]

4. PERFORMANCE ANALYSIS

In this section we evaluate the performance of the three interpolation algorithms above, by examining computational issues as well as how closely each trajectory approximates the minimum angular acceleration curve for a fixed set of boundary conditions. This optimal curve, which is
obviously both co-ordinate and bi-invariant, is determined by solving the Euler–Lagrange equations for the functional

\[ J(R) = \int_0^1 \omega^T \dot{\omega} \, dt \] (42)

where \([\omega] = R^T \dot{R}\) is the angular velocity in body-fixed frame co-ordinates. The corresponding Euler–Lagrange equations are [18, 19]

\[ \frac{d^3 \omega}{dt^3} + \omega \times \ddot{\omega} = 0 \] (43)

The above equation, although deceptively simple in appearance, does not admit a general closed-form solution.

The trajectories generated by the three interpolation algorithms proposed in this paper can all be viewed in some sense as an approximation to the minimum angular acceleration solution. For the canonical co-ordinate-based algorithm, if \(R_0\) is close to \(R_1\) in the sense that \(R_0^T R_1\) is close to \(I\), then the boundary condition on the canonical co-ordinates \(r(1)\) is approximately \(r(1) = 0\). If in addition the initial and final angular velocities are not too large, then the solution \(r(t)\) can also be expected to remain small. Under this assumption the angular acceleration vector \(\dot{\omega}(t)\) is approximately \(\ddot{\omega}\); the solution curve in this case is given by the cubic polynomial

\[ r(t) = at^3 + bt^2 + ct \] (44)

and the resulting solution curve in \(SO(3)\) is given by

\[ R(t) = R_0 e^{[ar^3 + br^2 + cr]} \] (45)

For the Cayley–Rodrigues parameters, it is clear from equation (24) that if \(||r||\) (which now represents the Cayley–Rodrigues parameters) is much smaller than unity, then the angular acceleration \(\dot{\omega}_b\) can be approximated by \(\ddot{r}\). Here \(r\) corresponds to the Cayley–Rodrigues parameter representation for \(R_0^T R_1\), so that if \(r\) is small, then \(R_0\) and \(R_1\) are close to each other. Hence, if \(R_0\) and \(R_1\) are sufficiently close, then the minimum angular acceleration trajectory in Cayley–Rodrigues co-ordinates is approximated by a cubic polynomial.

Similarly for the unit quaternions, since the body-fixed angular velocity \(\omega_b\) is proportional to \(M^* M\), where \(M \in SU(2)\), it follows that \(\omega_b\) is proportional to \(M^* M + M + M^*\). Thus, if \(R_0\) and \(R_1\) are sufficiently close, then \(M\) (the \(SU(2)\) representation for \(R_0^T R_1\)) is approximately \(I\), and if the angular velocity is small, then \(M\) can be approximated to be zero. In this sense a cubic on the unit quaternions can be viewed as approximating the minimum angular acceleration curve, although the requirement that the unit quaternion curve always be of unit length makes a careful analysis more involved.

4.1. Two-point interpolation

We first try to gain some insight into how well behaved the various approximations are as the endpoint rotations become more distant from each other. Given two rotations \(R_0\) and \(R_1\), we define the distance between them as \(||r||\), where \([r] = \log(R_0^T R_1)\). Physically, this corresponds to
the angle of rotation (in radians) about $r$ in rotating from $R_0$ to $R_1$ (see [20]). The test case we consider has the following boundary conditions:

$$R_0 = I $$
$$R_1 = \text{Rot}(\hat{y}, p) \cdot \text{Rot}(\hat{x}, \pi)$$

where $p$ varies over the range $[0, \pi]$. In addition, the initial body-fixed angular velocity and angular acceleration are set to $(\pi, 0, 0)$ and $(0, 0, 0)$, respectively. While varying $p$ over its range, for the resulting trajectory produced by each algorithm we plot the average squared deviation from the optimal trajectory. That is, if $R_m(t)$ denotes the optimal trajectory and $R_i(t)$ the interpolated trajectory, then the average squared deviation is defined to be

$$\int_0^1 \| \log(R_m^T R_i) \|^2 \, dt$$

Figure 1 shows the deviation from the minimum angular acceleration trajectory for the canonical co-ordinate and unit quaternion-based algorithms. The Cayley–Rodrigues parameter trajectory has an error on the order of a 100 times greater than the scale of the graph, and hence is not shown. The $x$-axis corresponds to $p$, while the $y$-axis corresponds to the average squared deviation. The canonical co-ordinate trajectory has smaller error than the unit quaternion trajectory for values of $p$ less than 1. In particular, observe that for the canonical co-ordinate trajectory the error is 0 when $p = 0$; this corresponds to one of the three special cases in which the cubic interpolant in canonical co-ordinates is the exact solution to the variational problem: when the initial and final conditions are tested.
CUBIC SPLINE ALGORITHMS FOR ORIENTATION INTERPOLATION 59

Table I

<table>
<thead>
<tr>
<th>Knot point</th>
<th>Time (s)</th>
<th>Canon. co-ord.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0.0, 0.0, 0.0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0.5, 0.1, −0.3)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(1.0, 0.2, −1.0)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>(0.0, −0.2, −2.2)</td>
</tr>
</tbody>
</table>

angular velocities are equal and non-zero. In this case the solution \( R(t) \) is given by

\[
R(t) = R_0 e^{[r]t}
\]  (49)

where \( r = \log(R_0^t R_1) \). In fact, \( R(t) \) corresponds to the minimal geodesic, i.e. it minimizes \( \int_0^1 \omega^T \omega \, dt \). The other two special cases occur when the initial and final angular velocities are both zero, and when the initial angular velocity and acceleration are zero. In the former the solution is given by

\[
R(t) = R_0 e^{[r](3t^2 − 2t^3)}
\]  (50)

while in the latter the solution is

\[
R(t) = e^{[r]t^3}
\]  (51)

See [9, 19], for proofs of these results.

For the Cayley–Rodrigues parameter and unit quaternion trajectories, when \( p = 0 \) the path followed by these trajectories is identical to the minimal geodesic, but these curves fail to be unit-speed: the Cayley–Rodrigues parameters warp the time axis according to \( \tan t/2 \), while for the unit quaternions time warping occurs as a result of projecting a line segment (connecting two points on the three-sphere in \( \mathbb{R}^4 \)) onto the three-sphere.

For \( p > 1 \) the unit quaternion trajectory has less distortion than the canonical co-ordinate trajectory, although for other arbitrary initial angular velocities the crossover point varied over the entire range of \( p \). In general, our experimental results suggest that for \( p \) between 0 and \( \pi/2 \) the canonical co-ordinate trajectory has the least deviation from the optimal trajectory, while for \( p \) greater than \( \pi/2 \) the unit quaternion trajectory has less deviation. In all cases the Cayley–Rodrigues parameter trajectory had the most deviation from the optimal trajectory, often by one or two orders of magnitude.

4.2. Multiple-point interpolation

We now examine the performance of the algorithms for interpolation through four orientations. The top row of Figure 2 depicts the four knot orientations, which are given in canonical co-ordinates in Table I.

The initial angular velocity and angular acceleration at \( t = 0 \) are set to \( \omega_0 = (0.6, 0.1, −0.3) \) and \( \alpha_0 = (0, 0, 0) \), respectively. The second row depicts the minimum angular acceleration curve, which we in fact compute \textit{a priori} before selecting the four knot points. The next three rows depict the trajectories produced by the canonical co-ordinate, Cayley–Rodrigues parameter, and unit-quaternion-based algorithms, respectively. All the intermediate orientations are shown at equally spaced time intervals.
To numerically evaluate the differences among the three trajectories, we plot the deviation of each trajectory from the minimum angular acceleration trajectory at each time $t$. As before, the deviation is measured using the isotropic distance metric in $\text{SO}(3)$: if $R_m(t)$ and $R_i(t)$, respectively, denote the minimum angular acceleration and interpolated orientation trajectories, then the deviation at time $t$ is given by $\|\log(R_m(t)R_i(t))\|$. Figure 3 plots the error graph for the three trajectories. Note that at the knot times the deviation is zero as expected.

5. CONCLUSIONS

This article has presented an analysis of three bi-invariant cubic spline algorithms for orientation interpolation. The algorithms are based on the three most basic angle-axis representations for rotation matrices: the canonical co-ordinates, Cayley–Rodrigues parameters, and unit quaternions. A geometric interpretation of the various representations, together with explicit formulas for calculating, among others, angular velocities and angular accelerations, are given. Using these formulas we have also provided pseudo-code descriptions of the interpolation algorithms based on the three representations.

In the latter half of the article we have compared the performance of the three algorithms with respect to computational efficiency, and how closely the resulting trajectories approximate the minimum angular acceleration curve. Our experimental results suggest that in general the canonical
CUBIC SPLINE ALGORITHMS FOR ORIENTATION INTERPOLATION

Figure 3. Deviation as a function of time for the interpolated trajectories. Solid line: canonical co-ordinate trajectory. Dashed line: unit quaternion trajectory. Dotted line: Cayley–Rodrigues parameter trajectory

co-ordinate-based trajectories offer the best approximation, followed by the unit quaternion, and finally the Cayley–Rodrigues parameter-based trajectories. The differences were particularly significant for two-point interpolation, but less severe for multiple-point interpolation, in which the knot points are usually spaced closer together. In the latter case the errors for the three algorithms are roughly of the same order.

The Cayley–Rodrigues parameter-based trajectories result in greater distortion than trajectories generated from the other two methods, but have the advantage of extremely compact and clean formulas for the angular velocity and acceleration. Both the Cayley–Rodrigues parameter and unit quaternion-based algorithms produce rational motions; this is quite significant from a CAD perspective, where NURBS are gaining increasing popularity. [21, 22]. One can expect that higher-order Cayley transforms will reduce the distortion, but the corresponding formulas become significantly more complicated, and the resulting motions may no longer be rational. It should also be pointed out that in our simulation experiments we have not explicitly considered the inertial properties of a rigid body.

One of the main differences in the Cayley–Rodrigues interpolation algorithm from the other two algorithms is that winding effects are no longer present. Depending on one’s perspective this may be either an advantage or disadvantage: if the interpolated motion generates sufficiently large angular velocities and angular accelerations, it may in some cases be more energy efficient to interpolate between two orientations by ‘winding’ around the same axis several times without reversing direction. On the other hand, the designer may not want such unpredictable behaviour to occur when interactively designing orientation trajectories.
APPENDIX

We show that the rotation trajectory generated by the Cayley–Rodrigues parameter-based algorithm is bi-invariant. Recall that bi-invariance implies that the actual orientation trajectory generated in physical space is independent of the choice of inertial and body-fixed reference frames. As before, let the boundary conditions be 
\[ R(0) = R_0, \quad R(1) = R_1, \quad R^T(0)R(0) = [\omega_0], \quad R^T(1)R(1) = [\omega_1], \]
and let \( R(t) \) be the interpolating trajectory obtained as outlined in the previous section. Then \( R(t) \) is in the left-invariant form
\[
R(t) = R_0(I - [r(t)])(I + [r(t)])^{-1}
\]
where \( r(t) \) is a cubic polynomial in \( \mathbb{R}^3 \):
\[
r(t) = at^3 + bt^2 + ct
\]
for \( 0 \leq t \leq 1 \), where \( c = \omega_0/2 \), and \( a \) and \( b \) are determined from the linear equations (40) and (41).

We first show the easier case of left-invariance, corresponding to a change in the location of the inertial reference frame. In this case the boundary conditions with respect to the new inertial frame become
\[
R(0) = QR_0, \quad R(1) = QR_1, \quad R^T(0)R(0) = [\omega_0], \quad \text{and} \quad R^T(1)R(1) = [\omega_1]
\]
for some constant \( Q \in SO(3) \). Denote by \( \hat{R}(t) \) the interpolating trajectory satisfying these new boundary conditions, i.e.
\[
\hat{R}(t) = Q R_0(I - [\hat{r}(t)])(I + [\hat{r}(t)])^{-1}
\]
where \( \hat{r}(t) \) is a cubic polynomial in \( \mathbb{R}^3 \). Then by inspecting equations (36)–(39) it is clear that the boundary conditions on \( \hat{r}(t) \) are identical to those for \( r(t) \), so that \( \hat{r}(t) = r(t) \). Hence, \( \hat{R}(t) = Q R(t) \), establishing left-invariance.

Right-invariance can also be shown by a straightforward but slightly more involved calculation. Here we assume the body-fixed frame is attached to another point on the rigid body, possibly in a different orientation. The boundary conditions in this case become 
\[
R(0) = R_0Q, \quad R(1) = R_1Q, \quad R^T(0)R(0) = Q^T[\omega_0]Q, \quad \text{and} \quad R^T(1)R(1) = Q^T[\omega_1]Q
\]
for some constant \( Q \in SO(3) \). Let
\[
\tilde{R}(t) = R_0Q(I - [\tilde{r}(t)])(I + [\tilde{r}(t)])^{-1}
\]
denote the interpolating trajectory satisfying these boundary conditions, where \( \tilde{r}(t) = \hat{a}t^3 + \hat{b}t^2 + \hat{c}t \) is some cubic polynomial in \( \mathbb{R}^3 \) to be determined. Then \( \tilde{r}(1) \) must satisfy
\[
[\tilde{r}(1)] = Q^T[\omega_0] - R_0R_1R_0^TQ
\]
\[
= Q^T[\omega_1]Q
\]
Applying the general matrix identity \( R^T[b]R = [R^Tb] \) for any rotation matrix \( R \) and three-vector \( b \), we have \( \tilde{r}(1) = Q^T[\omega_1]Q \). Then the coefficients of \( \tilde{r}(t) \) satisfy
\[
\hat{c} = \frac{Q^T[\omega_0]}{2}
\]
\[ \hat{a} + \hat{b} = \hat{c}(1) - \hat{c} \]  
\[ 3\hat{a} + 2\hat{b} = \frac{1 + \|\hat{r}(1)\|^2}{2}(I - [\hat{r}(1)])^{-1}Q^T\omega_1 \]

Observing that the initial and final angular velocities are \( Q^T\omega_0 \) and \( Q^T\omega_1 \), respectively, a routine calculation then verifies that \( \hat{a} + \hat{b} = Q^T(a + b) \) and \( 3\hat{a} + 2\hat{b} = Q^T(3\hat{a} + 2\hat{b}) \). Then \( \hat{r}(t) = Q^T\hat{r}(t) \), and

\[ \dot{R}(t) = R_0Q(I - Q^T[\hat{r}(t)]Q)(I + Q^T[\hat{r}(t)]Q)^{-1} \]
\[ = R_0(I - [\hat{r}(t)])(I + [\hat{r}(t)])^{-1}Q \]
\[ = R(t)Q \]

which proves right-invariance. Our interpolating trajectory is therefore bi-invariant as claimed.

REFERENCES

20. Park FC. Distance metrics on the rigid-body motions with applications to mechanism design. ASME Special 50th Anniversary Design Issue 1995; 117(B):87–92.
