Smooth Invariant Interpolation of Rotations

F. C. PARK
Seoul National University
and
BAHRAM RAVANI
University of California, Davis

We present an algorithm for generating a twice-differentiable curve on the rotation group SO(3) that interpolates a given ordered set of rotation matrices at their specified knot times. In our approach we regard SO(3) as a Lie group with a bi-invariant Riemannian metric, and apply the coordinate-invariant methods of Riemannian geometry. The resulting rotation curve is easy to compute, invariant with respect to fixed and moving reference frames, and also approximately minimizes angular acceleration.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*animation, geometric algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Cubic spline, interpolation, Lie algebra, Lie group, mathematics, rotation

1. INTRODUCTION

A common problem that arises not only in computer graphics and animation, but also in applications ranging from robot motion planning to machine vision, is the interpolation of smooth curves on the rotation group, 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 $\{\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_n\}$, and a set of $n + 1$ scalars $t_0 < t_1 < \cdots < t_n$,

This work was supported in part by US Army Research Office grant number DAAL03-90-G-0005 and in part by the California Department of Transportation through the AHMCT program. F. C. Park was also supported in part by the Engineering Research Center for Advanced Control and Instrumentation at Seoul National University.
Authors' addresses: F. C. Park, Department of Mechanical Design and Production Engineering, Seoul National University, Seoul, Korea; B. Ravani, Department of Mechanical and Aeronautical Engineering, University of California, Davis, Davis, CA 95616; email: ⟨bravani@ucdavis.edu⟩.

find a twice-differentiable curve $\mathbf{R}(t)$ on the rotation group such that $\mathbf{R}(t_i) = \mathbf{R}_i$, $i = 0, 1, \ldots, n$. Our goal is to find a fast, reference frame-invariant method of interpolating smooth curves on the rotation group.

Although well-established methods exist for interpolation in finite-dimensional vector spaces, the majority of these techniques do not easily generalize in a uniform and coordinate-invariant way to curved spaces such as the rotation group. Some recent work toward this end includes that of Brunnett and Crouch [1994], Noakes et al. [1989], and Zefran et al. [1995]. In Brunnett and Crouch [1994], a detailed variational analysis of minimum acceleration curves on the sphere is given, whereas in Noakes et al. [1989] and Zefran et al. [1995] the variational equations are derived for minimum acceleration curves on Riemannian manifolds and the Lie group of rigid body motions, respectively. The focus of these variational analyses is on obtaining qualitative results rather than computational algorithms; in general their solutions require the integration of nonlinear differential equations with split boundary conditions, which severely limits their practicality for CAD applications.

Because of its practical importance the problem of interpolation on the rotation group has received much attention in the literature. Most of the existing work involves choosing some local coordinate parametrization for rotations (e.g., Euler angles or unit quaternions) and applying existing vector space spline algorithms to these local coordinates. Although computationally efficient, one obvious drawback to such approaches is that, unlike the variational methods previously mentioned, the resulting curve typically will depend on the choice of local coordinates. When the underlying space is curved, often the price for coordinate-invariance is a substantial increase in the amount of computation. Interactive curve design, on the other hand, demands speed as well as accuracy, and in such situations the practical solution is usually a compromise between the efficiency offered by coordinate-based approaches, and the physical and mathematical consistency offered by purely intrinsic approaches. It is therefore important to understand how the choice of coordinates affects the behavior of the curve vis-á-vis the intrinsic solution.

In the case of rotations, another equally important issue is the question of *bi-invariance*. Given two ordered sets of rotation matrices $\{\mathbf{R}_0, \ldots, \mathbf{R}_n\}$ and $\{\bar{\mathbf{R}}_0, \ldots, \bar{\mathbf{R}}_n\}$, related by $\bar{\mathbf{R}}_i = \mathbf{P}\mathbf{R}_i\mathbf{Q}$, $i = 0, 1, \ldots, n$ for a pair of constant rotation matrices $\mathbf{P}$ and $\mathbf{Q}$, it is reasonable to demand that the interpolating curves through these two sets, denoted $\mathbf{R}(t)$ and $\bar{\mathbf{R}}(t)$, respectively, be related by $\bar{\mathbf{R}}(t) = \mathbf{P}\mathbf{R}(t)\mathbf{Q}$. Physically this relation reflects the fact that the orientation component of the interpolated motion for a rigid body moving in space should not be influenced by the choice of fixed (i.e., global) or moving (i.e., attached to the moving body) reference frames.

In this article we present an algorithm for interpolating twice-differentiable bi-invariant trajectories on the rotation group. The resulting trajectory approximately minimizes angular acceleration in the same sense that cubic splines can be viewed as approximations to minimum curvature

curves. The key ingredients of the algorithm are obtained by specializing standard concepts from the theory of Lie groups to SO(3), which can also be regarded as a Lie group with a bi-invariant Riemannian metric. In particular, the *canonical coordinates of the first kind* act as a natural set of local coordinates on SO(3), and the *exponential map* leads to a simple geometric visualization of the rotation group.

The article is organized as follows. In Section 2 we present in the context of SO(3) the basic concepts of matrix Lie groups. We also discuss the Lie group structure of the Special Unitary Group SU(2), which can in turn be identified with the unit quaternions. In Section 3 we review the variational equations for a minimum angular acceleration curve that interpolates two given rotations, and show that if the rotations are "sufficiently close," then the solution is given by the exponential of a certain cubic matrix polynomial. This curve moreover is shown to be bi-invariant, independent of the closeness assumption, and in several special but important cases coincides with the minimum angular acceleration curve. In Section 4 the two-point interpolation results of the previous section are extended to a multiple-point interpolation algorithm. We conclude in Section 5 with some remarks on the interpolation of general rigid body motions.

Before proceeding we mention some of the relevant previous work in SO(3) interpolation. One of the more widely cited approaches is the work of Shoemake [1985], who presents a class of interpolation schemes based on the unit quaternion representation for rotations. Part of the appeal behind the unit quaternions is that they provide a globally nonsingular four-parameter representation for rotations. As we explain later in more detail, it is important to keep in mind that the unit quaternions, or equivalently SU(2), are not the same as SO(3), but rather a double covering. This latter fact is often overlooked in many quaternion-based approaches, which sometimes leads to wildly spinning trajectories. Even with a proper accounting of this fact the resulting algorithms can often become cumbersome.

Shoemake's algorithm can essentially be viewed as a generalization of De Casteljau's construction for Bézier curves to SU(2), in which straight lines are now replaced by minimal geodesics. This algorithm, although producing intrinsic curves, has the expected disadvantage of high computational cost, as well as the additional bookkeeping required because of the double-covering property. Also, some of the standard properties possessed by Bézier curves in Euclidean space are not preserved in Shoemake's formulation. In Park and Ravani [1995] we argue that the present circle of geometric ideas can be used to formulate a more elegant and compact construction of Bézier curves in SO(3) (and other Lie groups) much more succinctly.

A more careful geometric analysis of unit quaternion-based methods is given by Ge and Ravani [1994a,b], in which the underlying curved geometry is considered in performing the interpolation, and a proper analysis and evaluation of the characteristics of the resulting Bézier representations are given. Kim and Nam [1996] also propose a unit quaternion-based method of

generating Hermite interpolants on SO(3) that uses circular blending arcs. Juttler and Wagner [1994; 1996] consider the generation of rigid-body motions using dual quaternion curves, and discuss the issues associated with the dependence of the existing methods on the coordinate system used. Hart et al. [1994] have presented an interesting method for visualization of quaternion curves that represent three-dimensional rotations, and Barr et al. [1992] also present an interpolation algorithm based on quaternions that handles angular velocity constraints.

2. THE GEOMETRY OF SO(3)

We begin with a review of the necessary background on SO(3) as a matrix Lie group; the development closely parallels that of Park and Ravani [1995], and additional background can be found, for example, in Belinfante and Kolman [1972]. We close this section with a brief discussion of the Lie group structure of the Special Unitary Group SU(2), which can be identified with the unit quaternions.

2.1 SO(3) as a Lie Group

Before reviewing the Lie group structure of SO(3), it is instructive to consider the simplest nontrivial Lie group, the unit circle S^1, as a motivating example. Clearly S^1 forms a smooth one-dimensional space, or a *manifold* of dimension one. Points in S^1 can be parametrized in the complex plane as $\mathbf{z} = \cos\theta + i \sin\theta$ for $0 \leq \theta \leq 2\pi$. Multiplying two points in S^1 results in another point in S^1; in fact, it is not difficult to see that S^1 forms an algebraic group under complex multiplication. Another way to display this group structure is to arrange an element of S^1 as the 2×2 rotation matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{1}$$

The set of such matrices forms the algebraic group SO(2) under matrix multiplication. The final observation is that an element of S^1 can be written as the exponential $e^{i\theta} = \cos\theta + i \sin\theta = \mathbf{z}$, or in matrix form as

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \exp\begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}. \tag{2}$$

where $\exp(\cdot)$ here denotes the matrix exponential (defined in the following). In both the scalar and matrix representations the exponential map provides a natural means of parametrizing the circle S^1 by a single coordinate θ.

As previously shown, the circle S^1 is equivalent to the 2×2 rotation matrices SO(2). This set has the structure of both an algebraic group and differentiable manifold, and is an example of a *Lie group*. The rotation

group SO(3) is the set of all 3×3 real orthogonal matrices with unit determinant. It is not difficult to see that SO(3) will also have the structure of a group (under matrix multiplication) and a differentiable manifold, or a Lie group. Some other well-known examples of Lie groups include the rigid-body motions SE(3), whose elements admit the 4×4 matrix representation

$$\begin{bmatrix} \mathbf{R} & \mathbf{b} \\ 0 & 1 \end{bmatrix}, \tag{3}$$

where $\mathbf{R} \in \mathrm{SO}(3)$ and $\mathbf{b} \in \Re^3$; Gl(n), the general linear group of $n \times n$ real nonsingular matrices, and the special linear group Sl(n), which is a subgroup of Gl(n) whose elements have unit determinant.

Let $\mathbf{A}$ be a point on a matrix Lie group $\mathcal{G}$, and $\mathbf{X}(t)$ be any differentiable curve on $\mathcal{G}$ that passes through $\mathbf{A}$ at $t = 0$; that is, $\mathbf{X}(0) = \mathbf{A}$. The derivative $\dot{\mathbf{X}}(0)$ is said to be a *tangent vector* to $\mathcal{G}$ at $\mathbf{A}$; the set of all tangent vectors at $\mathbf{A}$, denoted $\mathrm{T}_{\mathbf{A}}\mathcal{G}$, forms a vector space, called the *tangent space* to $\mathcal{G}$ at $\mathbf{A}$. The tangent space at the identity $\mathbf{A} = \mathbf{I}$ is given a special name, the *Lie algebra* of $\mathcal{G}$, and is denoted by the lowercase $\mathbf{g}$. On SO(3) it is easily shown that the Lie algebra so(3) consists of the 3×3 skew-symmetric matrices: if $\mathbf{R}(t)$ is a curve in SO(3) such that $\mathbf{R}(0) = \mathbf{I}$, then differentiating both sides of $\mathbf{R}^{\mathrm{T}}(t)\mathbf{R}(t) = \mathbf{I}$, it follows that $\dot{\mathbf{R}}^{\mathrm{T}}(0) + \dot{\mathbf{R}}(0) = 0$, so that elements of so(3) are matrices of the form

$$[\mathbf{r}] \triangleq \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}, \tag{4}$$

where $\mathbf{r} \in \Re^3$. Note that an element $[\mathbf{r}] \in \mathrm{so}(3)$ can also be represented as a vector $\mathbf{r} \in \Re^3$.

More generally a Lie algebra is a vector space $\mathcal{V}$, together with a bilinear map $[\,\cdot\,,\,\cdot\,]: \mathcal{V} \times \mathcal{V} \to \mathcal{V}$ (called the *Lie bracket*) that satisfies, for every η, μ, $\xi \in \mathcal{V}$, (i) $[\eta, \eta] = 0$, and (ii) $[\eta, [\mu, \xi]] + [\xi, [\eta, \mu]] + [\mu, [\xi, \eta]] = 0$. From (i) and the bilinearity property it follows that $[\eta, \mu] = -[\mu, \eta]$. For matrix Lie groups and Lie algebras the corresponding Lie bracket reduces to the standard matrix commutator: if η and μ are square matrices, then $[\eta, \mu] = \eta\mu - \mu\eta$. In particular, on so(3) it is easily verified that the Lie bracket of two elements corresponds to their vector product: $[\mathbf{r}_1, \mathbf{r}_2] = [\mathbf{r}_1][\mathbf{r}_2] - [\mathbf{r}_2][\mathbf{r}_1] = [\mathbf{r}_1 \times \mathbf{r}_2]$.

One final notion that combines elements of both a Lie group and its Lie algebra is the *adjoint* map. If $\mathbf{X}$ and $\mathbf{x}$ are, respectively, arbitrary elements of a matrix Lie group and its Lie algebra, then $\mathbf{X}\mathbf{x}\mathbf{X}^{-1}$ is another element of the Lie algebra. On SO(3) the following identity can be established: if $\mathbf{R} \in \mathrm{SO}(3)$ and $\mathbf{r} \in \mathrm{so}(3)$, then $\mathbf{R}[\mathbf{r}]\mathbf{R}^{\mathrm{T}} = [\mathbf{R}\mathbf{r}]$. We make frequent use of this identity later.

2.2 The Exponential Map

One of the main connections between a Lie group and its Lie algebra is the *exponential mapping*; defined on each Lie algebra is the exponential mapping into the corresponding Lie group. On matrix groups the exponential mapping is given by the usual matrix exponential, that is, if $\mathbf{A}$ is an element of the Lie algebra, then $\mathbf{e}^{\mathbf{A}} = \mathbf{I} + \mathbf{A} + (\mathbf{A}^2/2!) + \ldots$ is an element of the Lie group. On so(3) the exponential mapping is *onto*; that is, for every $\mathbf{R} \in \mathrm{SO}(3)$ there exists at least one $[\mathbf{r}] \in \mathrm{so}(3)$ such that $\mathbf{e}^{[\mathbf{r}]} = \mathbf{R}$. On SO(3) and its Lie algebra well-known closed-form formulas exist for the exponential and its inverse. If $[\mathbf{r}] \in \mathrm{so}(3)$, then

$$\mathbf{e}^{[\mathbf{r}]} = \mathbf{I} + \frac{\sin \|\mathbf{r}\|}{\|\mathbf{r}\|} \cdot [\mathbf{r}] + \frac{1 - \cos \|\mathbf{r}\|}{\|\mathbf{r}\|^2} \cdot [\mathbf{r}]^2, \tag{5}$$

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

$$\log \mathbf{R} = \frac{\phi}{2 \sin \phi} (\mathbf{R} - \mathbf{R}^T), \tag{6}$$

where ϕ satisfies $1 + 2 \cos \phi = \mathrm{Tr}(\mathbf{R})$ and $\|\log \mathbf{R}\|^2 = \phi^2$. If we restrict ϕ to be between 0 and π, then in the case when $\mathrm{Tr}(\mathbf{R}) = -1$, 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 coordinates $\mathbf{r}$ can in fact be identified with the *canonical coordinates of the first kind* for the Lie group SO(3) [Chevalley (1946)]. On general Lie groups these canonical coordinates provide a natural local parametrization about a neighborhood of the identity. In the case of SO(3) this neighborhood covers all of SO(3) except for a "thin" set. To visualize this parametrization, first recall that every arbitrary rotation can be represented by a rotation about some fixed axis in space passing through the origin. SO(3) can therefore be geometrically pictured as a three-dimensional solid ball of radius π, centered at the origin, with the antipodal points identified: a point $\mathbf{r}$ in the ball represents a rotation by $\|\mathbf{r}\|$ radians (in the right-hand sense) about the line directed from the origin through $\mathbf{r}$. The exponential and logarithm give explicit formulas for this solid ball representation of SO(3).

Some 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 2π periodicity of rotations, it follows that if $[\mathbf{r}]$ is one solution to $\log \mathbf{R}$, then so is $[\mathbf{r}](1 + (2\pi k/\|\mathbf{r}\|))$ for any integer k.

2.3 Angular Velocities and so(3)

The Lie algebra so(3) provides a set of local coordinates for SO(3) via the exponential map. Another important connection between so(3) and SO(3)

involves angular velocities. If $\mathbf{R}(t)$ is a curve in SO(3) describing the orientation of a rigid body relative to a fixed reference frame, then a simple calculation reveals that both $\dot{\mathbf{R}}\mathbf{R}^{\mathrm{T}}$ and $\mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}$ are skew-symmetric, and hence elements of so(3). $\mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}$ is in fact the angular velocity of the rigid body expressed in moving-frame coordinates, whereas $\dot{\mathbf{R}}\mathbf{R}^{\mathrm{T}}$ is the angular velocity expressed in fixed-frame coordinates. The angular acceleration vector with respect to either the fixed or moving frame is then obtained by differentiating the appropriate angular velocity representation.

The application to SO(3) of the following formula, which is valid for general matrix Lie groups, is one of the key steps in the derivation of our interpolation algorithm. Let $\mathbf{x}(t)$ be a curve on a matrix Lie algebra, and $\mathbf{X}(t) = \mathbf{e}^{\mathbf{x}(t)}$ be a curve on the corresponding matrix Lie group. Then it can be shown that

$$\mathbf{X}^{-1}\dot{\mathbf{X}} = \int_0^1 \mathbf{e}^{-\mathbf{x}(t)s}\dot{\mathbf{x}}(t)\mathbf{e}^{\mathbf{x}(t)s}ds \tag{7}$$

and

$$\dot{\mathbf{X}}\mathbf{X}^{-1} = \int_0^1 \mathbf{e}^{\mathbf{x}(t)s}\dot{\mathbf{x}}(t)\mathbf{e}^{-\mathbf{x}(t)s}ds. \tag{8}$$

If $[\mathbf{r}(t)] = \mathbf{x}(t)$ is a curve on so(3) and $\mathbf{R}(t) = \mathbf{e}^{[\mathbf{r}(t)]}$ is the corresponding curve on SO(3), then by explicitly evaluating the integral in the preceding formula a closed-form expression for the angular velocity can be obtained in terms of the canonical coordinates $\mathbf{x}$. First, denote by $\boldsymbol{\omega}_b(t)$ the angular velocity with respect to the moving (body) frame; that is, $[\boldsymbol{\omega}_b(t)] = \mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}$. Then

$$\boldsymbol{\omega}_b(t) = \mathbf{A}(\mathbf{r})\dot{\mathbf{r}}, \tag{9}$$

where

$$\mathbf{A}(\mathbf{r}) = \mathbf{I} - \frac{1 - \cos\|\mathbf{r}\|}{\|\mathbf{r}\|^2}[\mathbf{r}] + \frac{\|\mathbf{r}\| - \sin\|\mathbf{r}\|}{\|\mathbf{r}\|^3}[\mathbf{r}]^2. \tag{10}$$

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

$$\mathbf{B}(\mathbf{r}) = \mathbf{I} + \frac{1 - \cos\|\mathbf{r}\|}{\|\mathbf{r}\|^2}[\mathbf{r}] + \frac{\|\mathbf{r}\| - \sin\|\mathbf{r}\|}{\|\mathbf{r}\|^3}[\mathbf{r}]^2. \tag{11}$$

It can be verified that both $\mathbf{A}$ and $\mathbf{B}$ are nonsingular for all $\mathbf{r}$.

2.4 Lie Group Structure of SU(2)

In this section we discuss the Lie group structure of the Special Unitary Group SU(2), and demonstrate its equivalence with the unit quaternions. Although these results have been known in the mathematics and physics community, the formulas we establish here are either scattered throughout the literature or appear not to have been explicitly derived.

SU(2) is defined to be the set of 2×2 complex unitary matrices with unit determinant: any $\mathbf{M} \in \mathrm{SU}(2)$ satisfies $\mathbf{M}^*\mathbf{M} = \mathbf{M}\mathbf{M}^* = I$ and $\det\mathbf{M} = 1$, where $\mathbf{M}^*$ denotes the complex conjugate transpose of $\mathbf{M}$. From the definition it follows that $\mathbf{M}$ is of the form

$$\mathbf{M} = \begin{bmatrix} \alpha & \beta \\ -\bar{\beta} & \bar{\alpha} \end{bmatrix}, \tag{12}$$

where $\bar{\cdot}$ 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 be identified with the unit 3-sphere in $\Re^4$, which is clearly a differentiable manifold. That SU(2) forms an algebraic group (under matrix multiplication), and hence a Lie group, can also be easily verified. The identification with the unit quaternions can be made by writing $\mathbf{M} \in \mathrm{SU}(2)$ as $q_0 + q_1 i + q_2 j + q_3 k$. As seen from the foregoing, the rather unintuitive rule for quaternion multiplication turns out to be simply group multiplication in SU(2).

Following our derivation of the Lie algebra so(3), the Lie algebra of SU(2), denoted by the lower-case su(2), can be shown to consist of all complex 2×2 skew-Hermitian matrices with trace zero. More specifically, an element of su(2) has the form $i\mathbf{A}$, where

$$\mathbf{A} = \begin{bmatrix} x_1 & x_2 + ix_3 \\ x_2 - ix_3 & -x_1 \end{bmatrix}. \tag{13}$$

The Lie bracket on su(2) is also given by the matrix commutator: $[i\mathbf{A}, i\mathbf{B}] = \mathbf{BA} - \mathbf{AB}$. A simple calculation shows that the Lie bracket on su(2) can also be interpreted as the cross-product in $\Re^3$.

The corresponding exponential and logarithm maps for SU(2) are given by the following formulas:

$$\exp(i\mathbf{A}) = I \cos\|\mathbf{A}\| + i \frac{\mathbf{A}}{\|\mathbf{A}\|} \sin\|\mathbf{A}\| \tag{14}$$

$$\log(\mathbf{M}) = \frac{\phi/2}{2 \sin \phi/2} (\mathbf{M} - \mathbf{M}^*), \tag{15}$$

where $\|\mathbf{A}\| = (x_1^2 + x_2^2 + x_3^2)^{1/2}$. These two formulas define the canonical coordinates for SU(2).

As alluded to earlier, there exists a 2–1 group homomorphism (i.e., a mapping which preserves group operations) between SU(2) and SO(3), so

that SU(2) can be regarded as a double covering of SO(3). This mapping can be obtained as follows. Given $\mathbf{R} \in \mathrm{SO}(3)$, suppose $\log \mathbf{R} = [\hat{\boldsymbol{\omega}}]\phi$, where $\hat{\boldsymbol{\omega}} \in \Re^3$ is of unit length and $0 \leq \phi \leq \pi$; $\mathbf{R}$ is thus a rotation about the axis $\hat{\boldsymbol{\omega}}$ by an angle of ϕ radians. The two elements in SU(2) corresponding to $\mathbf{R}$ are then $(q_0, q_1, q_2, q_3) = (\cos(\phi/2), \omega_1 \sin(\phi/2), \omega_2 \sin(\phi/2), \omega_3 \sin(\phi/2))$ and $(q_0, q_1, q_2, q_3) = (-\cos(\phi/2), -\omega_1 \sin(\phi/2), -\omega_2 \sin(\phi/2), -\omega_3 \sin(\phi/2))$. Thus antipodal points in SU(2) correspond to the same rotation in SO(3).

3. TWO-POINT INTERPOLATION

3.1 Minimum Angular Acceleration Curves

In this section we formulate, in a purely matrix setting, the variational problem of determining minimum angular acceleration curves in SO(3). From physical reasons it should be apparent that the solution must be coordinate-invariant. It turns out that in a few limited but interesting cases an analytic solution to this variational problem can be found; we review both these solutions as well as the precise form of the general Euler-Lagrange equations.

In what follows we choose to represent the angular acceleration in terms of the moving frame coordinates, although this choice is completely arbitrary. The angular velocity in moving frame coordinates is given by $[\boldsymbol{\omega}] = \mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}$. Rewriting this equation as $\dot{\mathbf{R}} = \mathbf{R}[\boldsymbol{\omega}]$, the variational formulation of the minimum angular acceleration problem is as follows: find a curve $\mathbf{R}(t)$ in SO(3) that minimizes

$$\int_0^1 \dot{\boldsymbol{\omega}}^{\mathrm{T}}\dot{\boldsymbol{\omega}}dt \tag{16}$$

subject to $\dot{\mathbf{R}} = \mathbf{R}[\boldsymbol{\omega}]$, with $\mathbf{R}(0) = \mathbf{R}_0$, $\mathbf{R}(1) = \mathbf{R}_1$, $\boldsymbol{\omega}(0) = \boldsymbol{\omega}_0$, and $\boldsymbol{\omega}(1) = \boldsymbol{\omega}_1$ given as boundary conditions. The corresponding Euler-Lagrange equations are

$$\frac{d^3\boldsymbol{\omega}}{dt^3} + \boldsymbol{\omega} \times \ddot{\boldsymbol{\omega}} = 0. \tag{17}$$

This equation, although deceptively simple in appearance, does not admit a general closed-form solution. However, in the following three special cases a closed-form solution can be found. First, if $\omega(0) = \omega(1) \neq 0$, then the solution is given by

$$\mathbf{R}(t) = \mathbf{R}_0\mathbf{e}^{[\mathbf{r}]t}, \tag{18}$$

where $\mathbf{r} = \log(\mathbf{R}_0^{\mathrm{T}}\mathbf{R}_1)$. In this case $\mathbf{R}(t)$ is the minimum energy solution; that is, it minimizes $\int_0^1 \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}dt$. The second special case is when $\boldsymbol{\omega}(0) = \boldsymbol{\omega}(1)$

$= 0$; in this case the solution is given by

$$\mathbf{R}(t) = \mathbf{R}_0 \mathbf{e}^{[\mathbf{r}](3t^2 - 2t^3)}. \tag{19}$$

See Zefran et al. [1995] for a proof of the preceding two cases. The final special case is when $\boldsymbol{\omega}(0) = \dot{\boldsymbol{\omega}}(0) = 0$, in which case the solution to the Euler-Lagrange equations is

$$\mathbf{R}(t) = \mathbf{e}^{[\mathbf{r}]t^3} \tag{20}$$

as can be easily verified by using Equation (7). This is a particularly important case, as it corresponds to the popular *natural* spline in curve design (see the following).

In all these cases the solution is a rotation about the axis fixed in the direction of $\mathbf{r}$. One can verify that these expressions for $\mathbf{R}(t)$ are indeed solutions by substituting them into the Euler-Lagrange equations. In general, however, closed-form expressions for minimum angular acceleration curves are extremely difficult to find, so that one must resort to time-consuming and computationally expensive numerical methods to determine the solution.

3.2 Cubic Interpolation

Solving the preceding two-point boundary value problem numerically clearly is not a practical option for interactive CAD applications. However, if the two rotations are assumed "close" to one another (in a sense to be made precise in the following section), then by employing the canonical coordinates one can obtain a suboptimal "cubic" solution. The argument employed here is quite similar in spirit to the standard one given for approximating minimum energy splines, or elastica, by cubic polynomials: if the second derivative of the curve is not too large, then the squared norm of the second derivative provides a reasonable approximation to the elastic energy, and the corresponding solution will be a cubic polynomial. The solution on SO(3) that we obtain has the further desirable property of being bi-invariant. In this section we derive this particular cubic interpolant on SO(3).

By differentiating the angular velocity vector $\boldsymbol{\omega}_b(t)$ of Equations (9) and (10), one obtains the angular acceleration vector $\boldsymbol{\alpha}(t)$ relative to the moving frame, expressed in terms of the canonical coordinates. Following a somewhat lengthy calculation, the angular acceleration vector is given as follows:

$$\begin{aligned}\boldsymbol{\alpha}(t) = \ddot{\mathbf{r}} &- \frac{\mathbf{r}^{\mathrm{T}}\dot{\mathbf{r}}}{\|\mathbf{r}\|^4}(2\cos\|\mathbf{r}\| + \|\mathbf{r}\|\sin\|\mathbf{r}\| - 2)(\mathbf{r}\times\dot{\mathbf{r}}) - \frac{1-\cos\|\mathbf{r}\|}{\|\mathbf{r}\|^2}(\mathbf{r}\times\ddot{\mathbf{r}}) \\ &+ \frac{\mathbf{r}^{\mathrm{T}}\dot{\mathbf{r}}}{\|\mathbf{r}\|^5}(3\sin\|\mathbf{r}\| - \|\mathbf{r}\|\cos\|\mathbf{r}\| - 2\|\mathbf{r}\|)(\mathbf{r}\times(\mathbf{r}\times\dot{\mathbf{r}})) \\ &+ \frac{\|\mathbf{r}\| - \sin\|\mathbf{r}\|}{\|\mathbf{r}\|^3}(\dot{\mathbf{r}}\times(\mathbf{r}\times\dot{\mathbf{r}}) + \mathbf{r}\times(\mathbf{r}\times\ddot{\mathbf{r}})).\end{aligned} \tag{21}$$

Here $\|\cdot\|$ denotes the Euclidean norm in $\Re^3$. Having expressed everything in terms of canonical coordinates, the variational problem now reduces to finding the curve $\mathbf{r}(t) \in \Re^3$ that minimizes $\int_0^1 \|\boldsymbol{\alpha}(t)\|^2 dt$ subject to $\mathbf{r}(0) = 0$, $\dot{\mathbf{r}}(0) = \mathbf{A}^{-1}(\mathbf{r}_0)\boldsymbol{\omega}_0$, $\mathbf{r}(1) = \mathbf{r}_1$ with $[\mathbf{r}_1] = \log(\mathbf{R}_0^{\mathrm{T}}\mathbf{R}_1)$, and $\dot{\mathbf{r}}(1) = \mathbf{A}^{-1}(\mathbf{r}_1)\boldsymbol{\omega}_1$, with $\mathbf{A}$ given by Equation (10). The solution $\mathbf{R}(t)$ is then given by

$$\mathbf{R}(t) = \mathbf{R}_0 \mathbf{e}^{[\mathbf{r}(t)]}. \tag{22}$$

If $\mathbf{R}_0$ is close to $\mathbf{R}_1$ in the sense that $\mathbf{R}_0^{\mathrm{T}}\mathbf{R}_1$ is close to $\mathbf{I}$, then the boundary condition on $\mathbf{r}(1)$ is approximately $\mathbf{r}(1) = 0$. If in addition the initial and final angular velocities are not too large, then the solution $\mathbf{r}(t)$ can also be expected to remain small. Under this assumption the angular acceleration vector $\boldsymbol{\alpha}(t)$ is approximately $\ddot{\mathbf{r}}(t)$; the solution curve in this case is given by the cubic polynomial

$$\mathbf{r}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t, \tag{23}$$

where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \Re^3$ are constant vectors satisfying (i) $\mathbf{a} + \mathbf{b} + \mathbf{c} = \mathbf{r}_1$; (ii) $\mathbf{c} = \boldsymbol{\omega}_0$; (iii) $3\mathbf{a} + 2\mathbf{b} + \mathbf{c} = \mathbf{A}^{-1}(\mathbf{r}_1)\boldsymbol{\omega}_1$. The resulting solution curve in SO(3) is then given by

$$\mathbf{R}(t) = \mathbf{R}_0 \mathbf{e}^{[\mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t]}. \tag{24}$$

In the Appendix we show that the cubic solution (24) is bi-invariant. It should be apparent from physical considerations alone that the minimum angular acceleration curve—that is, the solution to the original variational problem—must necessarily be bi-invariant.

3.3 Accuracy of the Cubic Approximation

Because SO(3) is a curved space, establishing sharp bounds on the accuracy of the cubic approximation to the minimum angular acceleration curve turns out to be a difficult problem. Standard results from function approximation theory, which are usually formulated in a Hilbert space setting, cannot be directly applied to this situation. In fact, even the basic question of how to measure the distance between two elements of SO(3), which is fundamental to formulating any notion of error, needs to be addressed. In this section we try to gain some insight into how well-behaved the cubic approximation is as the endpoint rotations become more "distant" from each other.

To begin, observe that for the three special cases in which a closed-form analytic solution can be found (i.e., when the initial and final angular velocities are both zero, when they are equal and nonzero, and when the initial angular velocity and acceleration are zero), the cubic interpolant is the exact solution to the variational problem. As mentioned earlier, the last case is also quite meaningful from a CAD perspective, as these particular

initial conditions correspond to the popular natural spline. Clearly the preceding is a necessary condition for any reasonable approximation scheme. To discuss the approximation accuracy for the remaining cases, a notion of how to measure distance between two rotations is required. Here we give a brief review of some of the results established in Park [1995], where a natural distance metric on SO(3) is derived using our earlier geometric framework.

The natural way to define distance between two points on a surface is as the length of the shortest curve connecting them; the length of this *minimal geodesic* curve is computed by an integral involving the first fundamental form of the surface and a local coordinate parametrization of the curve. Generalizing these concepts to multidimensional surfaces, the surface now becomes a *Riemannian manifold*, and the first fundamental form the *Riemannian metric*. Usually one cannot hope to find a "natural" Riemannian metric for a given manifold, in the sense that the metric is determined by the geometry of the underlying space. However, on topologically compact Lie groups such as SO(3) there does exist a natural metric determined by the requirement of bi-invariance. We refer the reader to Park [1995] and any standard text on Riemannian geometry and Lie groups for the mathematical details. The key result for our purposes is that the minimal geodesic connecting two rotations $\mathbf{R}_0$ and $\mathbf{R}_1$, where length is measured relative to the bi-invariant Riemannian metric, is given by $\mathbf{R}(t) = \mathbf{R}_0 e^{[\mathbf{r}]t}$, where $[\mathbf{r}] = \log(\mathbf{R}_0^{\mathrm{T}}\mathbf{R}_1)$. The length of the minimal geodesic is simply given by the Euclidean length $\|\mathbf{r}\|$ or, equivalently, the amount of rotation (in radians) about $\mathbf{r}$.[1]

Figure 1 shows the deviation over $0 \leq t \leq 1$ of the cubic approximation from the optimal curve as a function of time. The optimal curve was determined numerically by the shooting method. Here the deviation is measured using the natural distance metric on SO(3), whose maximum value cannot exceed π. The boundary conditions used for this set of data are

$$\log(\mathbf{R}_0) = (0.2 \quad 0.1 \quad 0.1) \tag{25}$$

$$\log(\mathbf{R}_1) = (0.6 \quad 0.4 \quad 0.4). \tag{26}$$

The smaller of the two graphs depicts the error function for initial angular velocity $\boldsymbol{\omega}_0 = (0.5, 0.1, 0.1)$ and initial angular acceleration $\boldsymbol{\alpha}_0 = (0.5, 0.1, 0.1)$. The larger graph corresponds to the initial values $\boldsymbol{\omega}_0 = (5, 1, 1)$ and $\boldsymbol{\alpha}_0 = (0.5, 0.1, 0.1)$.

Figure 2 shows how the interpolated trajectory deviates from the minimum angular acceleration trajectory as a function of the initial and final

[1]It is now straightforward to formulate Bézier curves in SO(3), by simply replacing the straight lines in De Casteljau's algorithm with minimal geodesics; see Park and Ravani [1995] for details.

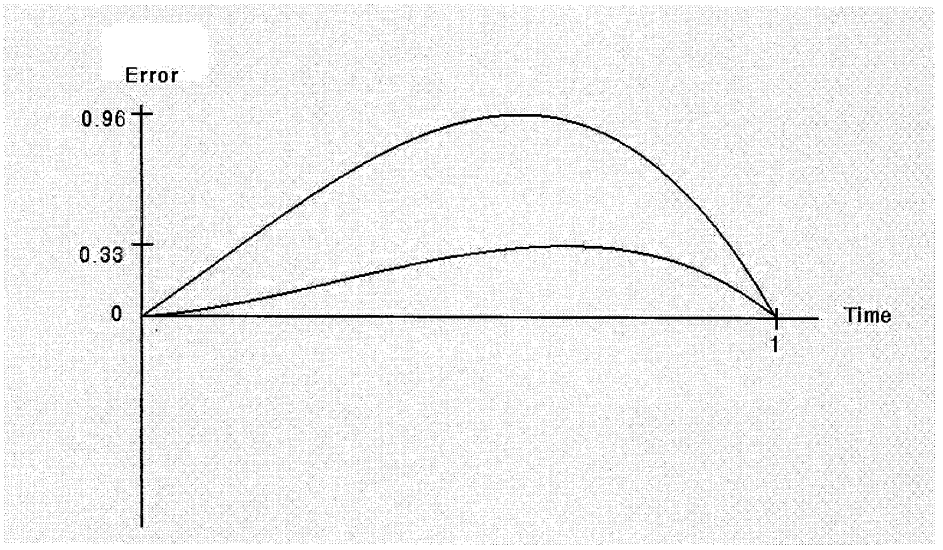

Fig. 1. A plot of deviation versus time for two interpolated trajectories. The smaller graph represents the natural distance between the cubic interpolant and the optimal (i.e., minimum angular acceleration) trajectory for small initial angular velocity and acceleration, and the larger graph depicts the deviation as the initial values are increased five-fold.

orientations. The initial conditions are

$$\mathbf{R}_0 = I$$

$$\boldsymbol{\omega}_0 = (\pi \quad 0 \quad 0)$$

$$\boldsymbol{\alpha}_0 = (0 \quad 0 \quad 0),$$

and the final orientation is expressed as a function of the parameter p, $0 \leq p \leq \pi$:

$$\mathbf{R}_1 = \exp\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\pi \\ 0 & \pi & 0 \end{bmatrix} p\right). \tag{27}$$

Note that when $p = 0$, both the interpolated trajectory and the minimum angular acceleration curve reduce to the minimal geodesic. As p increases, both the interpolated and minimum acceleration trajectories are forced to deviate from the minimal geodesic. If we denote the minimum angular acceleration trajectory by $\mathbf{R}(t)$ and the interpolated trajectory by $\hat{\mathbf{R}}(t)$, then for a given boundary value $\mathbf{R}_1$ the error is measured according to

$$\text{Error} = \int_0^1 \|\log(\mathbf{R}^T\hat{\mathbf{R}}\|^2 dt. \tag{28}$$

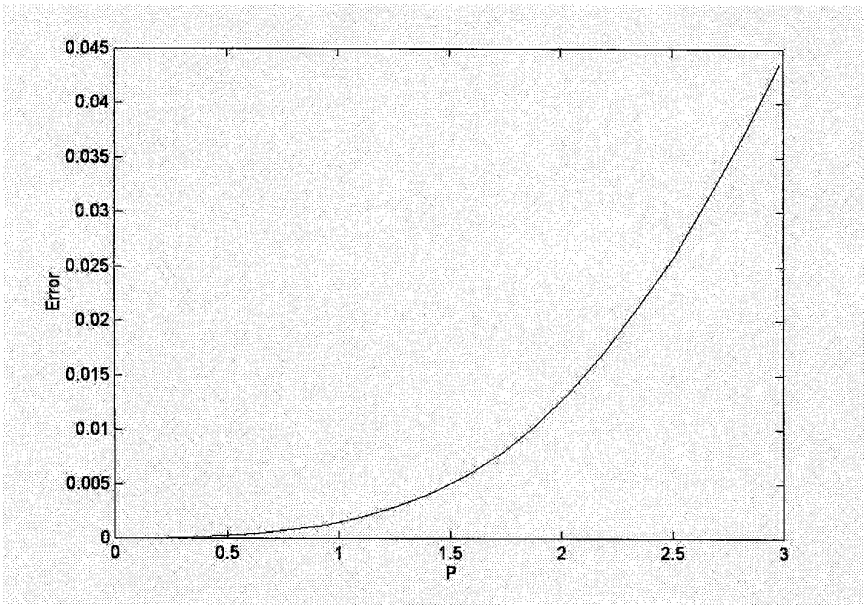

Fig. 2. A plot of average error as a function of the final orientation $\mathbf{R}_1$, which is parametrized by a scalar parameter p.

The resulting graph is symmetric about $p = \pi$, and therefore we plot only the error values for the range $p \in [0, \pi]$. These empirical results give an indication of how the approximation degrades as a function of the boundary conditions.

4. INTERPOLATION OF MULTIPLE POINTS

We now present a complete algorithm for interpolating through multiple points in SO(3) using cubic splines. Analogous to the Euclidean case, the interpolated curve in SO(3) maintains continuity of both angular velocities and accelerations at the knot points. The particular version of the algorithm that we present requires the following as inputs: an ordered set of $n + 1$ rotation matrices $\{\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_n\}$ (the knot points), a set of $n + 1$ scalars $t_0 < t_1 < \cdots < t_n$ (the knot times), an initial angular velocity $\boldsymbol{\omega}_0 \in \Re^3$, and an initial angular acceleration $\boldsymbol{\alpha}_0 \in \Re^3$. Both $\boldsymbol{\omega}_0$ and $\boldsymbol{\alpha}_0$ are expressed in moving frame coordinates. $\|\cdot\|$ here refers to the Euclidean norm.

Given:

$\{\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_n\}$ = knot points

$\{t_0, t_1, \ldots, t_n\}$ = knot times

$\boldsymbol{\omega}_0$ = angular velocity at t_0 in moving frame coordinates

$\boldsymbol{\alpha}_0$ = angular acceleration at t_0 in moving frame coordinates

(Condition: $\mathrm{Tr}(\mathbf{R}_{i-1}^{\mathrm{T}}\mathbf{R}_i) \neq -1$ for $i = 1, 2, \ldots, n$.)

Preprocessing: for $i = 1$ to n do

$$[\mathbf{r}_i] = \log(\mathbf{R}_{i-1}^{\mathrm{T}}\mathbf{R}_i)$$

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

Initialization:

$$\mathbf{c}_1 = \boldsymbol{\omega}_0$$

$$\mathbf{b}_1 = \boldsymbol{\alpha}_0/2$$

$$\mathbf{a}_1 = \mathbf{r}_1 - \mathbf{b}_1 - \mathbf{c}_1$$

Iteration: for $i = 2$ to n do

$\mathbf{s} = \mathbf{r}_i$ (temporary variable).

$\mathbf{t} = 3\mathbf{a}_{i-1} + 2\mathbf{b}_{i-1} + \mathbf{c}_{i-1}$ (temporary variable)

$\mathbf{u} = 6\mathbf{a}_{i-1} + 2\mathbf{b}_{i-1}$ (temporary variable)

$\mathbf{c}_i = \mathbf{A}_{i-1}\mathbf{c}_{i-1}$.

$$\mathbf{b}_i = \frac{1}{2}\left(\mathbf{u} - \frac{\mathbf{s}^{\mathrm{T}}\mathbf{t}}{\|\mathbf{s}\|^4}(2\cos\|\mathbf{s}\| + \|\mathbf{s}\|\sin\|\mathbf{s}\| - 2)(\mathbf{s}\times\mathbf{t}) - \frac{1 - \cos\|\mathbf{s}\|}{\|\mathbf{s}\|^2}(\mathbf{s}\times\mathbf{u})\right.$$

$$+ \frac{\mathbf{s}^{\mathrm{T}}\mathbf{t}}{\|\mathbf{s}\|^5}(3\sin\|\mathbf{s}\| - \|\mathbf{s}\|\cos\|\mathbf{s}\| - 2\|\mathbf{s}\|)(\mathbf{s}\times(\mathbf{s}\times\mathbf{t}))$$

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

$\mathbf{a}_i = \mathbf{s} - \mathbf{b}_i - \mathbf{c}_i$

Result: for $t_{i-1} \le t \le t_i$,

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

$$\mathbf{R}(t) = \mathbf{R}_{i-1}\mathbf{e}^{[\mathbf{a}_i\tau^3 + \mathbf{b}_i\tau^2 + \mathbf{c}_i\tau]}.$$

Formulas for the matrix exponential and logarithm are given by Equations (5) and (6), respectively.

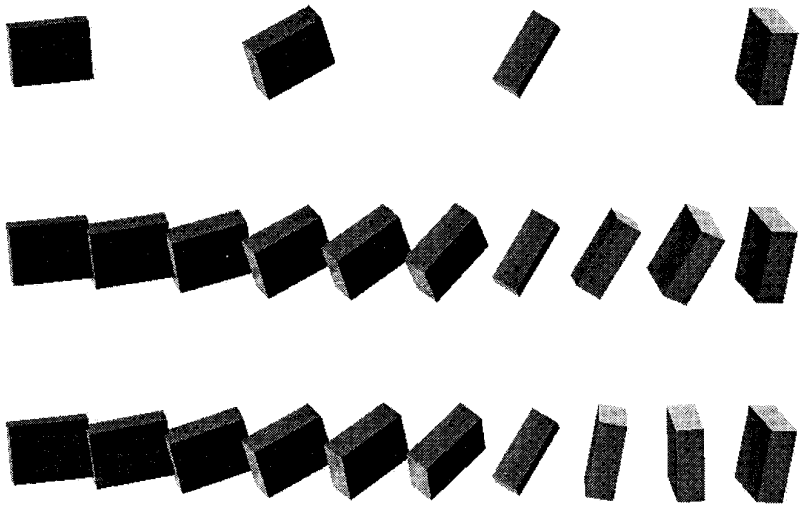

Fig. 3. Top row: 4 knot points; middle row: the cubic interpolant; bottom row: minimum angular acceleration interpolant.

Specifying the initial angular velocity and acceleration as inputs leads to the preceding version of the algorithm. 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. For other choices of input, for example, specifying the initial and final angular velocities (the so-called *clamped* spline), the formulas in the algorithm will in general become more complicated. To illustrate why, it is instructive to consider the case of clamped cubic splines in Euclidean space. Recall that determining the polynomial coefficients requires the solution of a tridiagonal linear system, which arises because the coefficients $(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i)$ are linear functions of $(\mathbf{a}_{i-1}, \mathbf{b}_{i-1}, \mathbf{c}_{i-1})$. In the case of SO(3), however, observe that the formula for $\mathbf{b}_i$ in the preceding algorithm is quadratic in $\mathbf{b}_{i-1}$. Although in principle a closed-form analytic solution for quadratics can be obtained, the resulting formulas are symbolically quite complex. For applications in which the final angular velocity is specified, we have found it practical to apply a numerical shooting type method to determine the corresponding initial angular acceleration.

Example. We illustrate the features of the interpolation algorithm with a simple example involving four knot points. The top row of Figure 3 shows the four orientations corresponding to knot points, with the knot times uniformly spaced at $t = 0, 1, 2, 3$. The initial angular velocity and acceleration are set to $\boldsymbol{\omega}_0 = (0.5, 0.1, 0.1)$ and $\boldsymbol{\alpha}_0 = (0.5, 0.1, 0.1)$, respectively. The middle row of Figure 3 shows the resulting interpolated motion. For comparison we include the corresponding minimum angular acceleration trajectory in the bottom row, where the intermediate velocities are chosen to coincide with those for the cubic interpolant. Here again the optimal solution is determined numerically by a shooting algorithm. For

this particular set of knot points and initial conditions the angular velocity and acceleration steadily increase with time. In general the size of the deviation will be proportional to the velocity and acceleration, and hence we observe the larger deviation occurring toward the tail end of the trajectory. As always, the best way to minimize deviations is to choose control points that are physically sensible and reasonably spaced.

5. CONCLUSIONS

By regarding SO(3) as a Lie group equipped with a natural Riemannian metric, we have presented an algorithm for interpolating multiple points in SO(3) that can also be viewed as a generalization of Euclidean cubic splines to rotation space. The main advantage of this approach is that bi-invariant curves can be generated quickly and efficiently. In the context of moving rigid bodies, bi-invariance ensures that the orientation trajectories are independent of choice of fixed and moving reference frames. Rotational cubic splines are also an effective compromise among the computational speed requirements for interactive CAD versus greater curve smoothness and geometric consistency.

Currently the most popular methods for designing rotation trajectories appear to be based on the unit quaternion representation. Although unit quaternions possess many attractive features, particularly their rational polynomial form, some of these advantages may disappear when one insists upon bi-invariance. Also, unit quaternions by definition must be of unit length, and the accumulation of errors due to finite precision may become significant in some cases. The need for renormalization procedures, combined with the additional bookkeeping incurred as a result of the 2–1 covering property, are two potential downsides of the unit quaternions that surprisingly are seldom mentioned in the literature.

Among SO(3) curve design algorithms, a popular choice is the generalized de Casteljau algorithm, in which straight lines are now replaced by minimal geodesics (arcs of great circles in the case of unit quaternions) [Shoemake 1985; Park and Ravani 1995]. These algorithms have the advantage of generating curves using purely geometric constructions, so that they are truly coordinate-independent. Moreover, reversing the order of the control points also results in the same curve, which may be a desirable feature from an animation perspective. The price for complete coordinate-independence, of course, is that the amount of computation increases significantly. Moreover, these algorithms are intended for free-form curve design, and in order to perform interpolation they require the user to specify additional control points, for which an infinite number of possibilities exist. The resulting curve also lacks physical meaning (in the sense of, e.g., approximately minimizing angular accelerations), and this can be important if the goal is to generate physically realistic trajectories. In short, all these factors must be considered in selecting an algorithm for a particular application.

There are a number of possible approaches to interpolating motions for a rigid body moving in physical space. Recalling that the configuration space of a rigid body is the Lie group of rigid body motions SE(3), one can exploit its Lie group structure to develop an algorithm analogous to the preceding one: canonical coordinates can be defined on the Lie algebra se(3), and closed-form formulas can be derived for the exponential and logarithm (see, e.g., Park [1995]). However, the resulting motion for two-point interpolation will be a screw motion, which from a physical viewpoint is unnatural: physics stipulates that a rigid body should move in such a way that, in the absence of external forces, its center of mass will move linearly and its orientation change according to Euler's equations. Hence the method of rigid-body motion generation that we advocate is to interpolate orientations and positions separately.

The other point to be mentioned is that for rigid body motions bi-invariance is impossible; this is strictly a consequence of the geometry of SE(3), and not of any features of a particular algorithm (see Park [1995]). From a graphics perspective the physical implications are that any interpolated motion for a rigid body will depend on where the moving frame is attached to the body (assuming a virtual body of infinite size). In most applications, however, there is a natural choice of location for this point. For example, if the objective is to generate physically realistic motions, then the center of mass is the obvious location for the moving frame. Because a preferred point can usually be determined from the application, the important requirement from the viewpoint of mechanics is that the motion be invariant with respect to the choice of fixed frame, or left-invariant. Regardless of where the moving frame is attached, generating an orientation trajectory for the body that is independent of this location is clearly desirable from both physical and mathematical considerations.

APPENDIX

In this section we show that the cubic solution (24) is bi-invariant. We first consider the easier case of left-invariance. If the fixed frame is moved to another location, then the endpoint orientations $\mathbf{R}_0$ and $\mathbf{R}_1$ are left-multiplied by some constant rotation $\mathbf{P}$ to $\bar{\mathbf{R}}_0 = \mathbf{P}\mathbf{R}_0$ and $\bar{\mathbf{R}}_1 = \mathbf{P}\mathbf{R}_1$, and the angular velocity vectors $\boldsymbol{\omega}_0$ and $\boldsymbol{\omega}_1$ remain the same. The boundary conditions on the new curve $\bar{\mathbf{R}}(t) = \bar{\mathbf{R}}_0 \mathbf{e}^{[\bar{\mathbf{a}}t^3 + \bar{\mathbf{b}}t^2 + \bar{\mathbf{c}}t]}$ are (i) $\bar{\mathbf{a}} + \bar{\mathbf{b}} + \bar{\mathbf{c}} = \mathbf{r}_1$; (ii) $\bar{\mathbf{c}} = \boldsymbol{\omega}_0$; (iii) $3\bar{\mathbf{a}} + 2\bar{\mathbf{b}} + \bar{\mathbf{c}} = \mathbf{A}^{-1}(\mathbf{r}_1)\boldsymbol{\omega}_1$. These equations are identical to the original conditions. Therefore $\bar{\mathbf{R}}(t) = \mathbf{P}\mathbf{R}(t)$, and the new curve is just the original curve left-multiplied by $\mathbf{P}$, proving left-invariance.

To show right-invariance, suppose now that the moving frame is relocated to another point on the rigid body. The boundary conditions in this case become $\bar{\mathbf{R}}_0 = \mathbf{R}_0\mathbf{Q}$ and $\bar{\mathbf{R}}_1 = \mathbf{R}_1\mathbf{Q}$ for some constant rotation $\mathbf{Q}$, and the angular velocity vectors are $\bar{\boldsymbol{\omega}}_0 = \mathbf{Q}^{\mathrm{T}}\boldsymbol{\omega}_0$ and $\bar{\boldsymbol{\omega}}_1 = \mathbf{Q}^{\mathrm{T}}\boldsymbol{\omega}_1$; here we make use of the adjoint mapping identity $\mathbf{Q}^{\mathrm{T}}[\boldsymbol{\omega}]\mathbf{Q} = [\mathbf{Q}^{\mathrm{T}}\boldsymbol{\omega}]$ on SO(3). Denote $\mathbf{Q}^{\mathrm{T}}\mathbf{r}_1$ by $\bar{\mathbf{r}}_1$. The boundary conditions on the new curve $\bar{\mathbf{R}}(t) = \bar{\mathbf{R}}_0 \mathbf{e}^{[\bar{\mathbf{a}}t^3 + \bar{\mathbf{b}}t^2 + \bar{\mathbf{c}}t]}$ then are (i) $\bar{\mathbf{a}} + \bar{\mathbf{b}} + \bar{\mathbf{c}} = \mathbf{Q}^{\mathrm{T}}\mathbf{r}_1$; (ii) $\bar{\mathbf{c}} = \mathbf{Q}^{\mathrm{T}}\boldsymbol{\omega}_0$; (iii) $3\bar{\mathbf{a}} + 2\bar{\mathbf{b}} + \bar{\mathbf{c}} = \mathbf{A}^{-1}$

$(\mathbf{Q}^T\mathbf{r}_1)\mathbf{Q}^T\boldsymbol{\omega}_1$. It can be shown that $\mathbf{A}(\bar{\mathbf{r}}_1) = \mathbf{Q}^T\mathbf{A}(\mathbf{r}_1)\mathbf{Q}$, from which it follows that $\mathbf{A}^{-1}(\bar{\mathbf{r}}_1) = \mathbf{Q}^T\mathbf{A}^{-1}(\mathbf{r}_1)\mathbf{Q}$. From the preceding boundary conditions we see that $[\bar{\mathbf{a}}\ \bar{\mathbf{b}}\ \bar{\mathbf{c}}] = \mathbf{Q}^T[\bar{\mathbf{a}}\ \bar{\mathbf{b}}\ \bar{\mathbf{c}}]$, and $\bar{\mathbf{R}}(t) = \mathbf{R}_0\mathbf{Q}\mathbf{e}^{\mathbf{Q}^T}[\mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t]\mathbf{Q} = \mathbf{R}_0\mathbf{Q}\mathbf{Q}^T\mathbf{e}^T[\mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t]\mathbf{Q} = \mathbf{R}(t)\mathbf{Q}$, establishing right-invariance.

ACKNOWLEDGMENTS

We wish to acknowledge In-Gyu Kang for debugging the multiple point algorithm, and generating all the simulation results and figures.

REFERENCES

BARR, A. H., CURRIN, B., GABRIEL, S., AND HUGHES, J. F. 1992. Smooth interpolation of orientations with angular velocity constraints using quaternions. *Comput. Graph. 25,* 2, 313–320.

BELINFANTE, J. G. AND KOLMAN, B. 1972. *A Survey of Lie Groups and Lie Algebras with Applications and Computational Methods.* SIAM, Philadelphia, PA.

BRUNNETT, G. AND CROUCH, P. 1994. Elastic curves on the sphere. *Advances in Computational Mathematics,* vol. 2, Baltzer, Amsterdam, The Netherlands, 23–40.

CHEVALLEY, C. 1946. *Theory of Lie Groups.* Princeton University Press, Princeton, NJ.

GE, Q. J. AND RAVANI, B. 1994a. Geometric construction of Bézier motions. *ASME J. Mech. Des. 116,* 749–755.

GE, Q. J. AND RAVANI, B. 1994b. Computer-aided design of motion interpolants. *ASME J. Mech. Des. 116,* 756–762.

HART, J. C., FRANCIS, G. K., AND KAUFMAN, L. H. 1994. Visualizing quaternion rotation. *ACM Trans. Graph. 13,* 3, 256–276.

JUTTLER, B. 1994. Visualization of moving objects using dual quaternion curves. *Comput. Graph. 18,* 3, 315–326.

JUTTLER, B. AND WAGNER, M. 1996. Computer-aided design with rational B-spline motions. *ASME J. Mech. Des. 118,* 2, 193–201.

KIM, M. S. AND NAM, K. W. 1996. Hermite interpolation of solid orientations with circular blending quaternion curves. *J. Visual. Comput. Anim. 7,* 95–110.

NOAKES, L., HEINZINGER, G., AND PADEN, B. 1989. Cubic splines on curved spaces. *IMA J. Math. Control Appl. 6,* 465–473.

PARK, F. C. 1995. Distance metrics on the rigid body motions with applications to mechanism design. *ASME J. Mech. Des. 117,* 1, 48–54.

PARK, F. C. AND RAVANI, B. 1995. Bézier curves on Riemannian manifolds and Lie groups with kinematics applications. *ASME J. Mech. Des. 117,* 1, 36–40.

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. *ACM SIGGRAPH 19,* 3, 245–254.

ZEFRAN, M., AND KUMAR, V. 1996. Planning smooth motions on SE (3). In *Proceedings of the IEEE International Conference on Robotic Automation* (Minneapolis, MN, April), 121–126.

Received March 1995; revised May 1996; accepted November 1996