A C^2-continous B-spline Quaternion Curve Interpolating a Given Sequence of Solid Orientations

Myoung-Jun Kim
Computer Science Department
KAIST
Taejeon 305-701, Korea
mjkim@jupiter.kaist.ac.kr

Myung-Soo Kim
Department of Computer Science
POSTECH
Pohang 790-784, Korea
mskim@vision.postech.ac.kr

Sung Yong Shin
Computer Science Department
KAIST
Taejeon 305-701, Korea
syshin@jupiter.kaist.ac.kr

Abstract

An algorithm is presented that constructs a C^2-continuous B-spline quaternion curve which interpolates a given sequence of unit quaternions on the rotation group $SO(3)$. The de Casteljau type construction method of B-spline curves can be extended to generate B-spline quaternion curves [Sch92]; however, the B-spline quaternion curves do not have C^2-continuity in $SO(3)$.

The authors [KKS94a] recently suggested a new construction method that can extend a B-spline curve to a similar one in $SO(3)$ while preserving the C^k-continuity of the B-spline curve. We adapt this method for the construction of a B-spline quaternion interpolation curve. Thus, the problem essentially reduces to the problem of finding the control points for the B-spline interpolation curve. However, due to the non-linearity of the associated constraint equations, it is non-trivial to compute the B-spline control points. We provide an efficient iterative refinement solution which can approximate the control points very preciesly.

Keywords: Quaternion, rotation, $SO(3)$, S^3, B-spline, Bèzier, interpolation, control vertices

1 Introduction

Unit quaternion curves play an important role in computer animation as a computationally reliable tool for controlling rotations for both object models and virtual cameras [GW91, Sho85]. An advantage of unit quaternions is that they are free from singularities such as gimbal lock. Furthermore, unit quaternions are computationally more efficient than the 3×3 matrix representation of a 3D rotation. Thus, the design of various quaternion curves has recently become an active research topic in computer animation [KKS94a, KN, KN94, NH92, Ple89, Sch92, Sho85, Sho91, WJ93].

The rigid motion of a 3D solid object can be represented as a continuous curve $(p(t), q(t)) \in R^3 \times SO(3)$, $0 \leq t \leq 1$, where R^3 is the 3-dimensional Euclidean space and $SO(3)$ is the rotation group of R^3 [Cur72]. The curves $p(t)$ and $q(t)$ represent the translational and rotational motions of the solid, respectively. The space $SO(3)$ is obtained as a projective space of the unit quaternion space S^3 under the identification of each pair of two antipodal points q and $-q \in S^3$ as a single element [Ham69, JT86, Sho85]. Thus, the local geometry of $SO(3)$ is identical to that of S^3. The curve constructions can be done in a more intuitive space S^3, and the constructed curves can be projected into $SO(3)$ by using the antipodal identification. Thus, the rotation control problem essentially reduces to that of constructing a unit quaternion curve in S^3.

In computer animation, it is a fundamental problem to generate a smooth motion for a rigid body so that the generated motion interpolates a given sequence of keyframe positions and orientations. For the interpolation of keyframe positions in R^3, there are many well-known techniques available such as B-splines, Hermite, and Bézier curves. However, it is relatively difficult to extend them to the construction of interpolation curves in S^3. The CAGD techniques can be used for the construction of rational spherical curves in S^3; however, the speed of rational curves are somewhat difficult to manipulate since they are constructed by a stereographic projection from some interpolation curves generated in R^4. Since the relative speed of a curve is very important for applications in computer animation, it is desirable to have curve construction schemes which are based on some intrinsic geometric properties of the space S^3 itself.

Many of the previous results are based on the recursive constructions of geodesic great circular arcs in S^3 [Ple89, Sch92, Sho85, Sho91]. Some of recent results are based on the construction of circular arcs in S^3 [KN94, WJ93]. Most of the previous methods construct C^1-continuous quaternion curves [KN94, Sho85, Sho91, WJ93]. Because of the discontinuity in second derivatives, there may occur large angular accelerations at the curve joints, which have undesirable effects on generating naturally-looking rotations [BCGH92]. Thus, high degree continuity is also an important factor for the quaternion curves in computer animation.

There are only a few C^2-continuous quaternion curves. Pletincks [Ple89] constructed quaternion curves by generating curve mid points recursively. The generated curves are extremely smooth since they converge to infinite degree curves. However, they have no closed form equations. Furthermore, the method always generates $(2^i - 1)$ in-betweens (for some integer $i > 0$), which is a serious drawback for keyframe animation systems. Kim and Nam [KN] constructed a C^k-continuous quaternion curve by blending two circular arcs in S^3 with a high degree blending function of degree $2k - 1$. Though a high degree blending can eliminate the C^2-discontinuity at the curve joints, the global smoothness of the whole curve is somewhat difficult to be achieved in this method. It is required to have the generalization of B-spline curves which have extreme smoothness in the overall curve shapes.

Schlag [Sch92] extended the de Casteljau construction scheme (of cubic Bézier quaternion curves [Sho85]) to the construction of B-spline quaternion curves. Nielson and Heiland [NH92] tried to construct a B-spline quaternion curve with C^2-continuity which interpolates a given sequence of unit quaternions. The B-spline quaternion curve is constructed as a sequence of cubic B-spline quaternion curve segments. The boundary positions and velocities of each cubic B-spline curve segment determine a cubic Bézier quaternion curve segment. Both curve segments are identical in R^3; however, they are different in S^3. The two curves do not even have the same derivative at the curve end points. Furthermore, two consecutive cubic B-spline quaternion curve segments do not meet with C^2-continuity, either. Nielson and Heiland [NH92] assumed the geometric properties which are not true in general for the quaternion curves in S^3. Thus, the constructed B-spline quaternion curve is not C^2-continuous.

The above discussion is the main observation of this paper; it also provides the main motivation for the development of another type of B-spline quaternion curves which the authors [KKS94a] recently proposed. The proposed B-spline quaternion curves are constructed so that they are intrinsically C^k-continuous when they are constructed with C^k-continuous basis functions. In this paper, we consider how to compute the B-spline control points so that they generate a B-spline quaternion curve which interpolates a given sequence of unit quaternions. Preserving many important geometric properties of B-spline curves in R^3, the B-spline quaternion curves have extreme smoothness with small gross angular accelerations.

The rest of this paper is organized as follows. In Section 2, we review the previous construction method of B-spline quaternion curves based on the de Casteljau algorithm, and discuss our observations on some important failures of the geometric properties of B-spline curves in S^3. Section 3 briefly introduces another type of B-spline quaternion curve which satisfies the C^k-continuity. Section 4 describes an iterative refinement method for approximating a cubic B-spline interpolation of unit quaternions. Section 5 demonstrates some experimental results. Finally, we conclude this paper in Section 6.

2 Some Remarks on B-spline Quaternion Curves

Nielson and Heiland [NH92] constructed a B-spline quaternion curve $Q(t)$ which interpolates a given sequence of solid orientations Q_i $(i = 0, 1, \cdots, n)$. The spline interpolation is done by constructing a sequence of cubic quaternion curve segments C_i $(i = 0, 1, \cdots, n)$. Each segment C_i interpolates two consecutive unit quaternions Q_i and Q_{i+1}, and two consecutive curve segments C_i and C_{i+1} meet with C^1-continuity at the common end point Q_i. Nielson and Heiland [NH92] assumed that the two curve segments C_i and C_{i+1} would meet with C^2-continuity at Q_i. However, this assumption is not true in $SO(3)$ and S^3, in which the quaternion multiplication is not commutative. We discuss more details below.

Given two unit quaternions q_1 and q_2, the geodesic great circular arc $G[q_1, q_2](t)$, which connects q_1 and q_2, is given by:

$$G[q_1, q_2](t) = q_1(q_1^{-1}q_2)^t = q_1 \exp(t \log(q_1^{-1}q_2)), \quad \text{for } 0 \leq t \leq 1. \tag{1}$$

Given four control points $q_{i-1}, q_i, q_{i+1}, q_{i+2}$, the cubic Bézier quaternion curve $\mathrm{GBz}[q_{i-1}, q_i, q_{i+1}, q_{i+2}](t)$ is defined by [Sho85]:

$$\begin{aligned}
\mathrm{GBz}[q_{i-1}, q_i, q_{i+1}](t) &= G[G[q_{i-1}, q_i](t), G[q_i, q_{i+1}](t)](t) \ , \\
\mathrm{GBz}[q_i, q_{i+1}, q_{i+2}](t) &= G[G[q_i, q_{i+1}](t), G[q_{i+1}, q_{i+2}](t)](t) \ , \\
\mathrm{GBz}[q_{i-1}, q_i, q_{i+1}, q_{i+2}](t) &= G[\mathrm{GBz}[q_{i-1}, q_i, q_{i+1}](t), \mathrm{GBz}[q_i, q_{i+1}, q_{i+2}](t)](t) \ .
\end{aligned} \tag{2}$$

Similarly, the cubic B-spline quaternion curve $\mathrm{GB}[q_{i-1}, q_i, q_{i+1}, q_{i+2}](t)$ is defined by [Sch92]:

$$\begin{aligned}
\mathrm{GB}[q_{i-1}, q_i, q_{i+1}](t) &= G[G[q_{i-1}, q_i](\tfrac{t+2}{3}), G[q_i, q_{i+1}](\tfrac{t+1}{3})](t) \ , \\
\mathrm{GB}[q_i, q_{i+1}, q_{i+2}](t) &= G[G[q_i, q_{i+1}](\tfrac{t+1}{3}), G[q_{i+1}, q_{i+2}](\tfrac{t}{3})](t) \ , \\
\mathrm{GB}[q_{i-1}, q_i, q_{i+1}, q_{i+2}](t) &= G[\mathrm{GB}[q_{i-1}, q_i, q_{i+1}](\tfrac{t+1}{2}), \mathrm{GB}[q_i, q_{i+1}, q_{i+2}](\tfrac{t}{2})](t) \ .
\end{aligned} \tag{3}$$

Let D_i $(i = 0, 1, \cdots, n)$ be the control points for the B-spline quaternion curve. Furthermore, let L_i and R_i be defined by:

$$L_i = G[D_{i-1}, D_i](\tfrac{2}{3}) \quad \text{and} \quad R_i = G[D_i, D_{i+1}](\tfrac{1}{3}) \ . \tag{4}$$

Then, we have the following relation:

$$G[L_i, R_i](\frac{1}{2}) = Q_i \ . \tag{5}$$

Nielson and Heiland [NH92] assumed that:

$$\mathrm{GB}[D_{i-1}, D_i, D_{i+1}, D_{i+2}](t) = \mathrm{GBz}[Q_i, R_i, L_{i+1}, Q_{i+1}(t)], \quad \text{for } 0 \leq t \leq 1. \tag{6}$$

Unfortunately, this is not true in the non-Euclidean space $SO(3)$ and S^3. Furthermore, even though the two curve sements have the same curve end points, their end derivaties are different:

$$\left.\frac{d}{dt}\right|_{t=0} \mathrm{GB}[D_{i-1}, D_i, D_{i+1}, D_{i+2}](t) \neq \left.\frac{d}{dt}\right|_{t=0} \mathrm{GBz}[Q_i, R_i, L_{i+1}, Q_{i+1}(t)] = Q_i(Q_i^{-1}R_i)^{\frac{1}{3}}. \quad (7)$$

The cubic Bézier quaternion curve segments satisfying Equations (4) and (5) do not meet with C^2-continuity. Even more, the B-spline quaternion curve is not C^2-continuous too. (See Kim, Kim, and Shin [KKS94b] for more details). Thus, we need a different scheme to develop a C^2-continuous quaternion spline interpolation.

3 B-spline Quaternion Curve with a Cumulative Form

Kim, Kim, and Shin [KKS94a] presented a general curve construction scheme that transforms a spline curve in R^3 (represented by a basis form) into a similar one in S^3. The effectiveness of this general scheme is demonstrated in [KKS94a] for the construction of Bézier, Hermite, B-spline quaternion curves. We briefly summarize the construction method of B-spline quaternion curves. For other quaternion curves and more details, see [KKS94a].

Given $n+1$ control points $\{p_i\}$, the B-spline curve $P(t)$ of order k is defined by:

$$P(t) = \sum_{i=0}^{n} p_i B_i^k(t) \ ,$$

where the base functions $B_i^k(t)$'s are defined by the following recurrence relation [Boo78]:

$$B_i^1(x) = \begin{cases} 1 & \text{if } t_i < t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

and

$$B_i^k(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_i^{k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1}^{k-1}(t).$$

The basis functions are C^{k-2} continuous piecewise polynomials of degree $(k-1)$. They are C^{k-2}-continuous everywhere, but may not be C^{k-1}-continuous at the knot sequence $\{t_i\}$. Each $B_i^k(t)$ has a non-zero support on the interval $[t_i, t_{i+k}]$, i.e., $B_i^k(t) = 0$ for $t < t_i$ or $t > t_{i+k}$.

The B-spline curve can be reformulated as follows:

$$P(t) = p_0 \widetilde{B}_0^k(t) + \sum_{i=1}^{n} (p_i - p_{i-1}) \widetilde{B}_i^k(t) \ ,$$

where

$$\widetilde{B}_i^k(t) \quad = \sum_{j=i}^{n} B_j^k(t) = \begin{cases} \sum_{j=i}^{i+k} B_i^k(t) & \text{if } t_i < t < t_{i+k-1} \\ 1 & \text{if } t \geq t_{i+k-1} \\ 0 & \text{if } t \leq t_i \end{cases} .$$

By replacing $P(t)$ to $Q(t)$, p_i to q_i, and the summations to the quaternion multiplacations, the corresponding B-spline quaternion curve with a cumulative basis form is formulated as follows:

$$Q(t) = q_0^{\widetilde{B}_0^k(t)} \prod_{i=1}^{n} (q_{i-1}^{-1} q_i)^{\widetilde{B}_i^k(t)} .$$

The B-spline quaternion curve is C^{k-2} continuous and locally controllable by moving the control points $\{q_i\}$ [KKS94a].

4 Quaternion Spline Interpolation

Given a sequence of data points P_i $(i = 0, 1, \cdots, n)$, the data interpolation can be done by constructing a uniform cubic B-spline curve $P(t)$ which interpolates each data point point P_i at $t = i$. The cubic B-spline curve $P(t)$ with $n + 2$ control point p_i's $(i = -1, 0, \cdots, n + 1)$ is defined by:

$$p(t) = \sum_{i=-1}^{n+1} p_i B_i^4(t) .$$

For notational simplicity, we denote $B_i(t) \equiv B_i^4(t)$. We use the uniform B-spline with the knot sequence: $t_i = i - 2$. Thus, the local support for each control point p_i is $[i - 2, i + 2]$, which is centralized to p_i. When the control points p_i's are given so that $P(i) = P_i$, the resulting B-spline curve $P(t)$ interpolates the given sequence of data points P_i's. The relation $P(t_i) = P(i) = P_i$ forms a system of linear equations:

$$p_{i-1} + 4\ p_i + p_{i+1} = 6P_i, \quad \text{for } i = 0, 1, \cdots, n . \tag{8}$$

With proper boundary conditions for p_{-1} and p_{n+1}, there are $n + 1$ equations for $n + 1$ unknowns. Since the system is strictly diagonally dominant, there exits a unique solution. An iterative method to compute the solution can be formulated as follows:

$$p_i^* = \frac{6}{4} \left(P_i - \frac{1}{6}\ p_{i-1} - \frac{1}{6}\ p_{i+1} \right) . \tag{9}$$

This equation is obtained by solving Equation (8) for p_i. Equation (9) is a contraction map, which is guaranteed to converge to a unique solution. Furthermore, in each iteration, the error is reduced by a factor of $\frac{6}{4}(\frac{1}{6} + \frac{1}{6}) = \frac{1}{2}$. By extending the same idea to the B-spline quaternion curve (as discussed in the previous section), we can construct a B-spline quaternion curve which interpolates a given sequence of unit quaternions Q_i $(i = 0, 1, \cdots n)$.

The B-spline quaternion curve $Q(t)$ is defined by:

$$q(t) = q_{-1}^{\widetilde{B}_0(t)} \prod_{i=0}^{n+1} (q_{i-1}^{-1} q_i)^{\widetilde{B}_i(t)} ,$$

where

$$\widetilde{B}_i(t) = \sum_{j=i}^{n+1} B_i(t) .$$

From the condition: $q(i) = Q_i$, we have $n + 1$ equations:

$$q_{i-1}(q_{i-1}^{-1} q_i)^{\frac{5}{6}} (q_i^{-1} q_{i+1})^{\frac{1}{6}} = Q_i , \quad \text{for } i = 0, 1, \cdots n . \tag{10}$$

Since there are $n + 1$ equations for $n + 3$ unknowns $q_{-1}, q_0, \cdots, q_{n+1}$, we need two boundary conditions. We use the end conditions for a natural spline:

$$Q''(0) = 0 \quad \text{and} \quad Q''(n) = 0.$$

When these two boundary conditions are applied to Equation (10), we obtain:

$$q_{-1}^{-1} q_0 = q_0^{-1} q_1 \quad \text{and} \quad q_n^{-1} q_{n+1} = q_{n-1}^{-1} q_n ,$$

or equivalently,

$$q_{-1} = q_0 (q_0^{-1} q_1)^{-1} \quad \text{and} \quad q_{n+1} = q_n (q_{n-1}^{-1} q_n) .$$

Thus, Equation (10) now reduces to:

$$\begin{aligned} & q_0 = Q_0 , \\ & q_{i-1}(q_{i-1}^{-1} q_i)^{\frac{5}{6}} (q_i^{-1} q_{i+1})^{\frac{1}{6}} = Q_i , \quad \text{for } i = 1, 2, \cdots n-1 , \\ & q_n = Q_n . \end{aligned} \tag{11}$$

Unfortunately, the system of Equation (11) is non-linear. There is no known method to compute the exact solution. We apply the iterative method of Equation (9) to solve the non-linear system of Equation (11). However, due to the non-linearity of the problem, there are some restrictions for the input values of Q_i's so that the convergence of the iterative method is guaranteed. For the convergence, the input data should satisfy one of the following two conditions: either (i) the key distances $|\log(Q_{i-1}^{-1} Q_i)|$ are within a certain bound or (ii) the rotational axis differences in three consecutive orientations Q_{i-1}, Q_i, and Q_{i+1} are within a certain bound. Each of these two conditions implies that, when the magnitudes are small, the quaternion multiplication is roughly commutative and thus Equation (11) is roughly linear. Each step of the iterative refinement introduces a smaller error than the previous step; thus, the iteration proceeds as a contraction map and the iterative solution converges to a unique solution.

5 Experimental Results

We formulate an iterative refinement procedure for the solution of Equation (11) as follows. The dominant term on the left hand side is $(q_{i-1}^{-1} q_i)^{\frac{5}{6}}$. By solving Equation (11) for the term, we have

$$(q_{i-1}^{-1} q_i)^{\frac{5}{6}} = q_{i-1}^{-1} Q_i (q_i^{-1} q_{i+1})^{-\frac{1}{6}} \tag{12}$$

Then, by solving for the variable q_i on the left hand side, we have:

$$q_i^* = q_{i-1} [q_{i-1}^{-1} Q_i (q_i^{-1} q_{i+1})^{-1/6}]^{6/5} \ . \tag{13}$$

For the initial guess, we set $q_i = Q_i$ on the right hand side.

The iterative refinement can be represented by a non-linear map F on a quaternion vector $\mathbf{q} = \{q_1, q_2, \cdots, q_n\}$:

$$\mathbf{q}^{(n+1)} = F(\mathbf{q}^{(n)}) \ ,$$

where $\mathbf{q}^{(1)} = \{Q_1, Q_2, \cdots, Q_n\}$ is the initial guess. When the iteration F is strictly convergent, the map F is a contraction map with the contraction ratio $K < 1$:

$$\mathrm{dist}(F(\mathbf{q}), F \circ F(\mathbf{q})) < K \cdot \mathrm{dist}(\mathbf{q}, F(\mathbf{q})) \ , \tag{14}$$

where $\mathrm{dist}(\mathbf{q}_1, \mathbf{q}_2)$ is the distance between two quaternion vectors $\mathbf{q}_1$ and $\mathbf{q}_2$ under a certain metric. Equation (14) implies that:

$$\mathrm{dist}(\mathbf{q}^{(n+1)}, \mathbf{q}^{(n+2)}) < K \cdot \mathrm{dist}(\mathbf{q}^{(n)}, \mathbf{q}^{(n+1)}) < K^{n-1} \cdot \mathrm{dist}(\mathbf{q}^{(1)}, \mathbf{q}^{(2)}) \ .$$

Thus, the sequence $\mathrm{dist}(\mathbf{q}^{(n)}, \mathbf{q}^{(n+1)})$ converges to 0 as $n \to \infty$, and the sequence $\mathbf{q}^{(n)}$ converges to a unique solution $\mathbf{q}$. Unfortunately, the non-linear map F is not guaranteed to be contractive.

For a given input sequence of keyframe orientations Q_i's, let the key distance θ_i and the axis difference ϕ_i be defined by:

$$\begin{aligned} \theta_i &= |\log(Q_{i-1}^{-1} Q_i)|, \\ \phi_i &= |\mathrm{axis}(Q_{i-1}, Q_i) \times \mathrm{axis}(Q_i, Q_{i+1})|, \end{aligned}$$

where $\mathrm{axis}(q_1, q_2)$ is the rotational axis from q_1 to q_2. When all the keyframe orientations lie on the same rotation axis, i.e. $\phi_i = 0$, the iteration F becomes linear and a contraction map with the contraction ratio $K = 3/5 < 1$. Thus, it converges very quickly. Similarly, we can expect that the

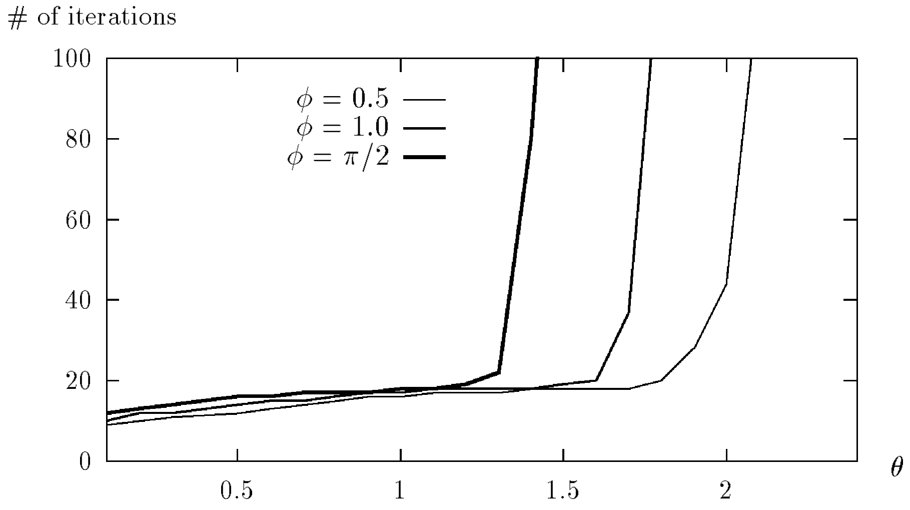

Figure 1: The number of iterations as a function of the key distance θ

iteration F would remain to be contractive for small values of θ_i or ϕ_i. This is because the iteration F behaves almost as a linear map in these cases. As the values of θ_i and ϕ_i increase, the iteration F shows non-linear behavior and the contraction property becomes weaker. Eventually, the map F will lose the contraction property and the corresponding convergency for large values of θ_i and ϕ_i. From this observation, if the iteration F converges for some values of θ_i and ϕ_i, we can expect that F will converge for smaller values of θ_i and ϕ_i, too. From experimental results, we have observed that the map F converges to a unique solution even for the input Q_i's which have somewhat large values of key distances and axis differences. In particular, the iteration converges for all $\{Q_i\}$ with $\theta_i \leq 1.4$. Note that the key distance 1.4 radian is sufficient for most applications in practice; this is because 1.4 radian in S^3 is 2.8 radian in the real world, which allows quite large key distances. When some key distances are larger than 2.8 radian, the animator may need to introduce some more additional keyframes between the two keyframes.

In the case of the axis difference, we have observed the worst convergency at $\phi_i = \pi/2$. Even for this case, the condition $\theta_i < 1.4$ makes the iteration converge. Of course, when the axis difference is smaller, we may allow larger key distances, too. The relationship between the key distance θ and the axis difference ϕ is shown in Figures 1 and 2. For a given sequence of keyframe orientations Q_i's with $\phi_i = \phi$ and $\theta_i = \theta$, the number of iterations is counted until the map F converges within an error bound 10^{-4}. Figures 1 and 2 show the required number of iterations for given values of θ and ϕ, respectively. For both cases, 10 keyframe orientations are used. In Figure 3, the required number of iterations is also shown as the number of keyframes increases. We can easily notice that the convergency does not depend on the number of keyframes. In most of the experiments, the average convergence ratio K is observed to be about 1/2, and the solutions are usually obtained within 10–20 iterations. Thus, it is possible to design rotational motions interactively.

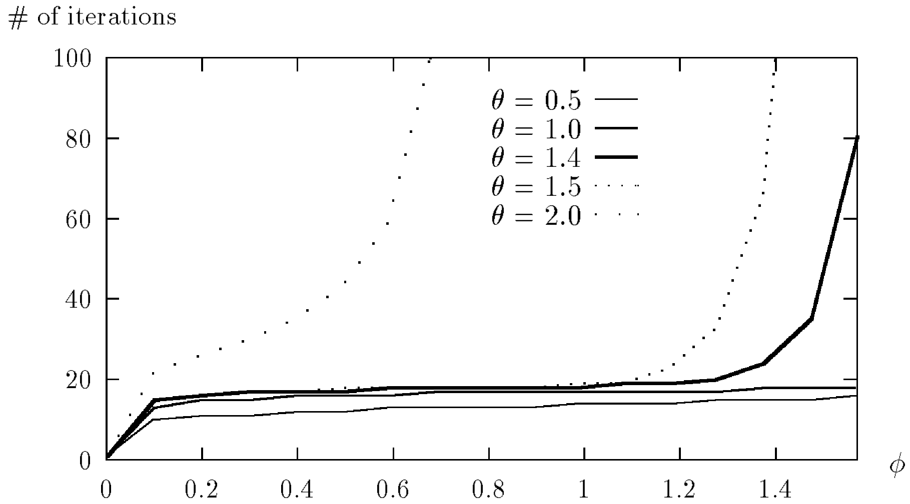

Figure 2: The number of iterations as a function of the axis difference ϕ

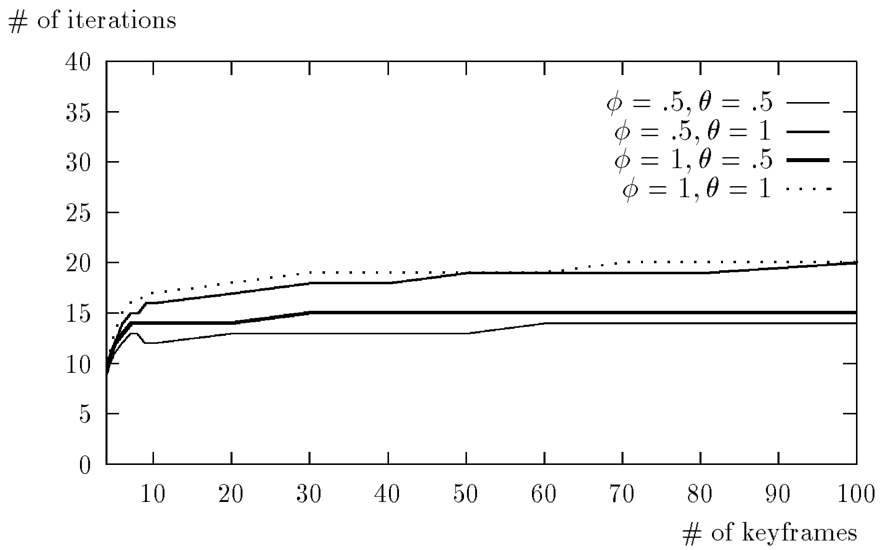

Figure 3: The number of iterations as a function of the number of keyframes

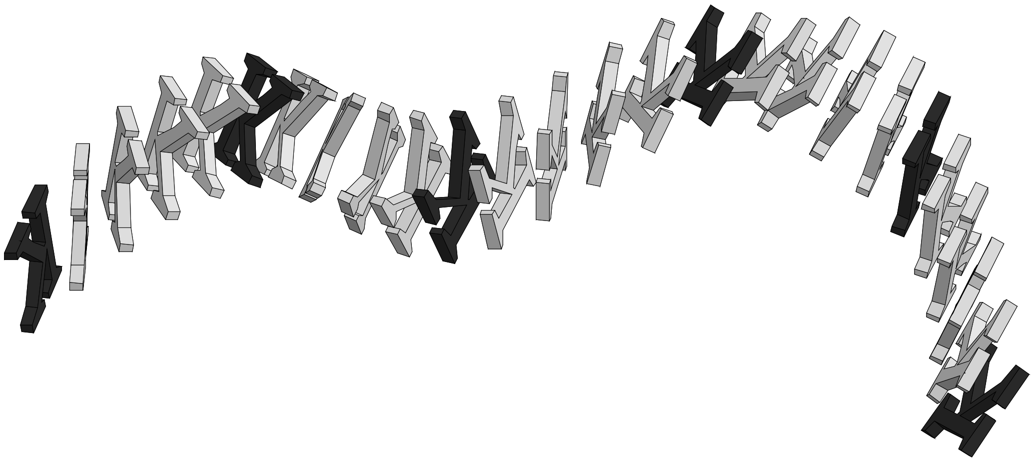

Figure 4: An example of quaternion B-spline interpolation

Figure 4 shows an example of motion interpolation for a rigid body, where the position and orientation interpolation curves are constructed by the B-spline interpolation curves in R^3 and S^3, respectively. Six keyframes are used in this example, and they are shown in dark color.

6 Conclusion

The B-spline curves in R^3 have many useful geometric properties; however, extreme care need to be taken to extend the curve construction scheme to S^3 so that the useful properties are preserved. The de Casteljau type construction of B-spline curves does not generate a C^2-continuous B-spline quaternion curve. Furthermore, a cubic B-spline quaternion curve is totally different from the Bézier quaternion curve which is defined by the curve boundary conditions.

By adapting a new construction method for B-spline quaternion curves (recently suggested by the authors [KKS94a]), the C^2-continuity is guaranteed for the B-spline quaternion curves. In this paper, we concentrated on the B-spline interpolation problem: how to find the B-spline control points so that the generated B-spline quaternion curve interpolates a given sequence of unit quaternions. Since the interpolation conditions are non-linear in the case of S^3, we developed an efficient iterative refinement method which can approximate the solution very precisely. The iteration converges when the input orientations are given so that they satisfy certain difference bounds; furthermore, these bounds actually cover most of important cases in practice. The iteration itself converges very quickly with the convergence ratio K being almost equal to $\frac{1}{2}$. Thus, it is possible to modify the keyframe orientations interactively.

References

[BCGH92] A. Barr, B. Currin, S. Gabril, and J. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Computer Graphics (Proc. of*

SIGGRAPH '92), pages 313–320, 1992.

[Boo78] C. Boor. *A Practical Guide to Splines.* Springer-Verlag, 1978.

[Cur72] M. Curtis. *Matrix Groups.* Spinger-Verlag, 1972.

[GW91] M. Gleicher and A. Witkin. Through-the-lens camera control. In *Computer Graphics (Proc. of SIGGRAPH '92)*, pages 331–340, 1991.

[Ham69] W.R. Hamilton. *Elements of Quaternions (Volume I, II).* Chelsea Publishing Company, 1969.

[JT86] J.L. Junkins and J.D. Turner. *Optimal Spacecraft Rotational Manueuvers.* Elsevier, 1986.

[KKS94a] M.J. Kim, M.S. Kim, and S.Y. Shin. A general unit quaternion curve construction scheme based on cumulative basis transformation. Technical Report CS/TR-94-88, KAIST, Taejon 305-701, Korea, 1994.

[KKS94b] M.J. Kim, M.S. Kim, and S.Y. Shin. Some remarks on the de casteljau construction of unit quaternion curves. Technical Report CS/TR-94-89, KAIST, Taejon 305-701, Korea, 1994.

[KN] M.S. Kim and K.W. Nam. Interpolating solid orientations with circular blending quaternion curves. to appear in *Computer-Aided Design.*

[KN94] M.S. Kim and K.W. Nam. Hermite interpolation of solid orientations with circular blending quaternion curves. In N.M. Thalmann and D. Thalmann (Eds.), editors, *(Proc. of CG Internatinal '94).* Spinger-Verlag, 1994.

[NH92] G. Nielson and R. Heiland. Animated rotations using quaternions and splines on a 4d sphere. In *Programmirovanie(Russia)*, pages 17–27. Spinger-Verlag, July-August 1992. English edition, *Programming and Computer Software*, Plenum Pub., New York.

[Ple89] D. Pletincks. Quaternion calculus as a basic tool in computer graphics. *The Visual Computer*, 5(1):2–13, 1989.

[Sch92] J. Schlag. Using geometric constructions to interpolate orientation with quaternions. In *Graphics GEMS II*, pages 377–380. Academic Press, 1992.

[Sho85] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics (Proc. of SIGGRAPH '85)*, pages 245–254, 1985.

[Sho91] K. Shoemake. Quaternion calculus for animation. *Math for SIGGRAPH (ACM SIGGRAPH '91 Course Notes #2)*, 1991.

[WJ93] W. Wang and B. Joe. Orientation interpolation in quaternion space using spherical biarcs. In *Proc. of Graphics Interface '93*, pages 24–32, 1993.

A Preliminaries

The quaternion space $\mathcal{H}$ is a skew-field of elements $q = w + x\hat{\imath} + y\hat{\jmath} + z\hat{k}$, $w, x, y, z \in R$, with element-wise addition and the following multiplication rules [Cur72, Ham69]:

$$\begin{aligned}
&\hat{\imath}^2 = \hat{\jmath}^2 = \hat{k}^2 = -1 \ , \\
&\hat{\imath}\hat{\jmath} = \hat{k}, \ \hat{\jmath}\hat{\imath} = -\hat{k} \ , \\
&\hat{\jmath}\hat{k} = \hat{\imath}, \ \hat{k}\hat{\jmath} = -\hat{\imath} \text{ , and} \\
&\hat{k}\hat{\imath} = \hat{\jmath}, \ \hat{\imath}\hat{k} = -\hat{\jmath} \ .
\end{aligned}$$

The multiplication is associative but non-commutative. A quaternion $q = w + x\hat{\imath} + y\hat{\jmath} + z\hat{k}$ can also be represented by a 4-dimensional point (w, x, y, z) or an ordered pair (w, v), where v is a three-dimensional vector (x, y, z). Many interesting properties on the quaternion space can be found in [Ham69]. One important property is that the unit quaternion space:

$$S^3 = \{q \in \mathcal{H} \mid \|q\| = \sqrt{q^2} = 1\}$$

has a natural mapping to the rotation group $SO(3)$. The rotation by angle θ about an axis $\hat{u}$ is represented by a quaternion:

$$q = (\cos\frac{\theta}{2}, \hat{u}\sin\frac{\theta}{2}).$$

For notational convenience, we represent a three-dimensional vector $v = (x, y, z)$ as a quaternion $x\hat{\imath} + y\hat{\jmath} + z\hat{k}$. Using the unit quaternion q as a rotational operator, the vector $\overline{v}$ rotated from v by q is given by the quaternion multiplication:

$$\overline{v} = qvq^{-1} \ .$$

We denote the rotation by q as the rotational operator R_q. Two successive rotations R_{q_2} after R_{q_1} of a vector v is expressed as $R_{q_2}(R_{q_1}(v))$. That is, we have:

$$R_{q_2}(R_{q_1}(v)) = q_2 q_1 v q_1^{-1} q_2^{-1} = (q_2 q_1) v (q_2 q_1)^{-1} = R_{q_2 q_1}(v) \ .$$

Thus, the quaternion product $q_2 q_1$ represents the composite rotation of two successive rotations q_1 and q_2.

The unit quaternion space S^3 in R^4 has the same local topology and geometry as that of the rotation group $SO(3)$. Since a quaternion can be represented by only four elements (in constrast to the nine elements in the 3×3 rotation matrix), it is more compact and computationally more efficient to use quaternions to represent the rotations in many applications Thus, we will use the quaternion to represent a rotation. Since q and $-q$ represent the same rotation, the map from S^3

to $SO(3)$ is 2-to-1. This is a disadvantage of quaternion but does not cause any big problem. The problem can be resolved if we are careful in identifying two antipol points, q and $-q$.

The quaternion multiplication is not commutative; thus, we have to be very careful to preserve the order of multiplications in a correct sequence. Let $q_1, q_2, \cdots, q_n$ be a sequence of rotations, each of which is given in the global world coordinate frame. Then, the product $q_n q_{n-1} \cdots q_1$ represents the net rotation of the successive rotations in the world coordinate. However, when each quaternion q_i represents a rotation in the local object coordinate frame (which is obtained by a sequence of rotations $q_1, q_2, \cdots, q_{i-1}$), the product $q_1 q_2 \cdots q_n$ represents the net rotation. Note that the same rotated result would be obtained by rotating in the sequence of rotations: $q_n, q_{n-1}, \cdots, q_1$, in the global world coordinate frame. The order of quaternion multiplications is extremely confusing. The rule is that the next rotation is multiplied to the left of the previous rotation if the rotation is done in the global frame, and to the right of the previous rotation if it is done in the local frame.

A good representation for a rotation is to use a 3-dimensional vector, where a vector v represents a rotation by an angle $\theta = \|v\|$ about the axis $\hat{v} = v/\|v\|$. This is a well-known formulation for angular momentum in classical mechanics. The map from a rotation vector v to its corresponding quaternion gives a natural map from R^3 to S^3:

$$v \rightarrow (\cos \frac{1}{2}\|v\|, \hat{v} \sin \frac{1}{2}\|v\|) \ . \tag{15}$$

One revolution (rotation by 2π) in the real space corresponds to the half revolution in S^3, i.e., the rotation from q to $-q$. For a complete revolution in S^3, from q to q via $-q$, the real space counterpart has two complete revolutions. The constant factor $\frac{1}{2}$ of Equation (15) results from this relationship between the real space and the space S^3. We may eliminate the factor $\frac{1}{2}$ by specifying the amount of rotation directly in S^3; that is, the phrase "rotate by θ in S^3" means the rotation by angle 2θ in the real space. Therefore, a rotation vector v denotes the rotation by angle $\|v\|$ about the axis $\hat{v} \in S^2$; the map (15) now takes a simpler form:

$$v \rightarrow (\cos \|v\|, \hat{v} \sin \|v\|) \ ,$$

which is identical to the definition of quaternion exponential [Ham69]:

$$\exp(v) = (\cos \|v\|, \hat{v} \sin \|v\|) \ .$$

The quaternion exponential is a many-to-1 function. To find its inverse function, which is the quaternion logarithm, we limit the domain within $\|v\| < \pi/2$. The logarithm of a quaternion

$q = (w, u)$, where w is the real part and u is the vector part, is defined by:

$$\log q = \log(w, u) = \begin{cases} \tan^{-1}(\|u\|/w)\hat{u}, & \text{if } w \neq 0 \\ \frac{\pi}{2}\hat{u}, & \text{if } w = 0 \end{cases}$$

The power of a quaternion, q^{α}, for a real exponent α, is defined by: $q^{\alpha} = \exp(\alpha \log q)$.