To appear in CVGIP: Graphical Models and Image Processing, Vol. 55, 1993.

In *Computer Graphics: Development in Virtual Environments*,
R. Earnshaw and J. Vince (eds.), Academic Press, pp. 171-183, 1995.

Hermite Interpolation of Solid Orientations Based on a Smooth Blending of Two Great Circular Arcs on SO(3)

Kee-Won Nam* and Myung-Soo Kim*,+

* Dept. of Computer Science, POSTECH, Pohang 790-784, South Korea

+ Dept. of Computer Science, Purdue University, W. Lafayatte, IN47907, USA

Abstract

An efficient algorithm is presented that generates a hermite interpolation quaternion curve. Given two unit quaternions $q_1, q_2 \in SO(3)$, and two tangent vectors v_1 and v_2 at q_1 and q_2, respectively; two great circular arcs $C_1(t)$ and $C_2(t)$, $0 \le t \le 1$, are constructed so that $C_1(0) = q_1$, $C_1'(0) = v_1$, $C_2(1) = q_1$, and $C_2'(1) = v_2$. The two great circular arcs C_1 and C_2 are then smoothly blended together on $SO(3)$ to generate a hermite quaternion curve $Q(t)$, $0 \le t \le 1$, with the boundary conditions: $Q(0) = q_1, Q(1) = q_2, Q'(0) = v_1$, and $Q'(1) = v_2$. An efficient forward difference method is developed to generate the great circular arcs. With the speed up thus obtained, the method of this paper and the squad method of Shoemake [16] now become the most efficient methods for the construction of quaternion curves which interpolate a given sequence of 3D solid orientations.

Keywords: Quaternion, orientation, rotation, angular velocity, hermite interpolation, animation

1 Introduction

In computer graphics and animation, the rotational motion control has important applications in 3D user interface and virtual camera control [3, 11]. The 3D rotation of a solid object in R^3 can be uniquely specified by a continuous path $Q(t) \in SO(3)$, $0 \le t \le 1$, where $SO(3)$ is the rotation group of R^3. Since the non-Euclidean space $SO(3)$ has many interesting geometric properties which are much different from those of Euclidean space R^3, it is non-trivial to design curves and surfaces in $SO(3)$ while preserving their geometric properties in R^3 [7, 8]. Each element of $SO(3)$ can be identified with a pair of two antipodal points $q, -q \in S^3$; thus, the local geometry of $SO(3)$ is identical to that of S^3 [2]. Since the 3-sphere S^3 is more intuitive to understand than the projective 3-space $SO(3)$, the previous curve construction schemes in $SO(3)$ have been described in S^3. The constructed curves in S^3 are projected into $SO(3)$ by using the antipodal identification.

The design of various quaternion curves in $SO(3)$ and S^3 has recently become an active research topic in computer animation [1, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17].

The CAGD techniques provide useful tools for the construction of spherical curves in S^3; however, the spherical curves are generated as rational curves [5, 12, 17]. Since the CAGD research goal is focused on the final geometric shape rather than the speed and acceleration of the curve, it is sometimes quite difficult to deal with the spherical rational curves so that they satisfy certain velocity and acceleration conditions. For example, the n-th derivative of a rational curve of degree d is a rational curve of degree $2^n d$. In computer animation, high order derivative terms and their integrals are often optimized for the generation of natural looking motions [1]. Since the space S^3 has an intrinsic spherical structure, it seems more natural to construct the spherical curves on S^3 by composing various spheres and circles embedded in S^3. Kim and Nam [9, 10] demonstrated the feasibility of such an approach by constructing various quaternion curves in S^3 simply by blending circular arcs in S^3. In this paper, we show that similar quaternion curves can be constructed by blending great circular arcs in S^3. The main motivation of this work is to improve the computational efficiency of the quaternion curve generation. The main improvement is based on an efficient forward difference method (developed in this paper) for the generation of great circular arcs. The hermite quaternion curve scheme [10] is first extended by using this technique. The quaternion interpolation curve scheme [9] is also easily extended by constructing a sequence of hermite quaternion curve segments, which interpolate a given sequence of unit quaternions and default angular velocities derived from them.

The hermite interpolation problem in S^3 can be specified as follows: Given two unit quaternions $q_1, q_2 \in S^3$, and two tangent velocities v_1 and v_2 at q_1 and q_2, respectively; how to construct a hermite quaternion curve $Q(t) \in S^3$, $0 \le t \le 1$, so that it satisfies the boundary conditions:

$$Q(0) = q_1, \quad Q(1) = q_2, \quad Q'(0) = v_1, \quad \text{and} \quad Q'(1) = v_2.$$

The unit quaternions q_1 and q_2 represent the initial and final orientations of a given solid, and the velocities v_1 and v_2 represent the angular velocities at q_1 and q_2, respectively. Let $\omega_i = 2v_i \cdot q_i^{-1}$ $(i = 1, 2)$, then $\omega_i \in R^3$ denotes the instantaneous rotational velocity about the axis ω_i by an angle $\|\omega_i\|$, where $\cdot$ is the quaternion multiplication (see Appendix A and [9, 10] for more details). Note that there is a metric difference between S^3 and $SO(3)$; thus, there is a number 2 in the equation: $\omega_i = 2v_i \cdot q_i^{-1}$. Since $SO(3)$ is obtained by identifying two antipodal points $q, -q \in S^3$ as a single element, a great half circle in S^3 becomes a closed great circle in $SO(3)$, which implies a complete rotation of degree 2π. This way of understanding an angular velocity as a 3D vector ω_i is more familiar to us due to the general education in physics; however, it is not quite useful for designing and reasoning about hermite quaternion curves in S^3. Thus, we interpret an angular velocity $\omega_i \in R^3$ (at the solid orientation q_i) as a 4D tangent vector $v_i \in T_{q_i}S^3$ which is obtained by the translation of the vector $\frac{1}{2}\omega_i$ along the curved 3-dimensional space S^3, i.e., $v_i = \frac{1}{2}\omega_i \cdot q_i$, where $\cdot$ is the quaternion multiplication. The tangent space $T_{q_i}S^3$ is the 3-dimensional hyperplane which is orthogonal to the unit 4D vector $q_i \in S^3$ in R^4 (see Figure 1). The vector space $T_{q_i}S^3$ consists

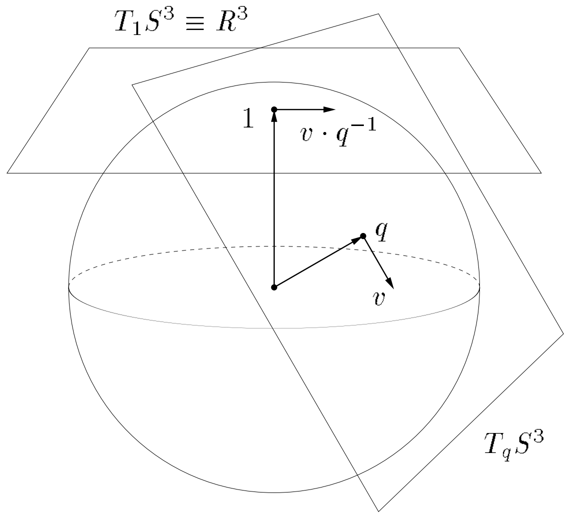

Figure 1: Tangent Spaces.

of all the tangent vectors (at q_i) of $S^3 \subset R^4$. In this paper, we use v_i and ω_i interchangeably as angular velocity.

Several methods have been suggested to construct hermite quaternion curves on S^3. Wang and Joe [17] constructed a hermite quaternion curve on S^3 by connecting two circular arcs with C^1-continuity. Hanotaux and Peroche [4] constructed an approximate hermite interpolation curve on S^3; however, the constructed curve does not interpolate boundary angular velocities. Kim, Kim, and Shin [6] extended the method so that the generated curve interpolates the two boundary velocities exactly; they also showed that the Bézier and squad quaternion curve methods of Shoemake [15, 16] can be extended to hermite quaternion curves by providing a compact differential formula for the quaternion curves. In a recent work, Kim, Kim, and Shin [8] proposed a general method that transforms spline curves (e.g., Bézier, Hermite, and B-spline curves) in R^3 into their quaternion curve analogues in $SO(3)$ and S^3 while preserving many important properties of the original curves. Kim and Nam [10] suggested a construction method for hermite quaternion curve by extending their circular blending method [9]. In the method of Kim and Nam [10], two circular arcs $C_1(t)$ and $C_2(t)$, $0 \le t \le 1$, are generated so that $C_1(0) = C_2(0) = q_1$, $C_1(1) = C_2(1) = q_2$, $C_1'(0) = v_1$, and $C_2'(1) = v_2$. The rather strong condition: $C_1(0) = C_2(0) = q_1$ and $C_1(1) = C_2(1) = q_2$, was used to simplify the differential formula of the quaternion curve at the two curve end points.

This paper assumes a weaker condition for the end points: $C_1(0) = q_1$ and $C_2(1) = q_2$. However, we use a blending function $f(t)$, $0 \le t \le 1$, which has a slightly higher degree than the blending function $f(t) = t$ of Kim and Nam [10]. That is, the blending function $f(t)$ of this paper is required to satisfy: $f(0) = 0$, $f(1) = 1$, $f'(0) = 0$, and $f'(1) = 0$. A cubic or piecewise quadratic polynomial function can produce such a function. To compensate the overhead of a higher blending degree, we utilize the extra degrees of freedom resulting from the weaker end point condition. Among all possible circular arcs $C_i(t) \in S^3$ (for $i = 1, 2$), $0 \le t \le 1$, with the condition: $C_i(t_i) = q_i$ and $C_i'(t_i) = v_i$ (where $t_1 = 0$ and $t_2 = 1$), we select the simplest curve; that is, the geodesic great circular arc.

There is a unique geodesic great circular arc $C_i(t)$ with the condition. If there were no computational shortcut for such a curve, there would be no advantage of using a higher degree blending function. Fortunately, there is a more efficient construction method for great circular arcs than for general circular arcs in S^3. It is based on a forward difference method (suggested in this paper) for great circular arcs. The same method is applicable to other circular curves as long as they are parametrized with uniform speeds. However, the circular curves of Kim and Nam [9, 10] do not have uniform speeds since they need to be reparametrized to meet the boundary conditions. In this paper, with a weaker version of the boundary conditions, we are free to use uniform speeds on the great circular curves and apply the efficient forward difference method for the curves (see Figure 4).

The rest of this paper is organized as follows. In §2, a hermite interpolation curve is constructed in S^3. In §3, an efficient forward difference method is introduced for the generation of great circular arcs. In §4, comparisons are made with other methods on the computational efficiencies. Finally, we conclude this paper in §5.

2 Hermite Interpolation in S^3

In this section, we consider the following hermite interpolation problem: Given two unit quaternions q_1 and $q_2 \in S^3$, and two velocities v_1 and v_2 at q_1 and q_2, respectively; how to construct a hermite interpolation curve $Q(t) \in S^3$, $0 \le t \le 1$, which satisfies the boundary conditions:

$$Q(0) = q_1, \quad Q(1) = q_2, \quad Q'(0) = v_1, \quad \text{and} \quad Q'(1) = v_2.$$

In §2.1, two geodesic great circular arcs $C_1(t)$ and $C_2(t) \in S^3$, $0 \le t \le 1$, are constructed so that they satisfy:

$$C_1(0) = q_1, \quad C_1'(0) = v_1, \quad C_2(1) = q_2, \quad \text{and} \quad C_2'(1) = v_2.$$

In §2.2, by blending the two great circular arcs C_1 and C_2, a hermite quaternion curve $Q(t) \in S^3$, $0 \le t \le 1$, is constructed.

2.1 Great Circular Arcs in S^3

Given a unit quaternion $q_i \in S^3$ and a velocity v_i at q_i ($i = 1, 2$), the velocity vector v_i is a tangent 4D vector to S^3 at q_i. The two 4D vectors q_i and v_i are orthogonal each other (see Figure 1). The great circular arc $C_i(t) \in S^3$, $0 \le t \le 1$, with $C_i(t_i) = q_i$ and $C_i'(t_i) = v_i$, is contained in the 2D plane which is determined by the two 4D vector q_i and $v_i \in R^4$, where $t_1 = 0$ and $t_2 = 1$. Let $\hat{v}_i = v_i/\|v_i\| \in S^3$, then the two unit 4D vectors q_i and $\hat{v}_i$ form an orthonormal basis of the 2D plane. Furthermore, the great circular arc is totally contained in the plane. The curve equations for $C_i(t)$'s, $0 \le t \le 1$, are given by:

$$C_1(t) = (\cos\theta_1 t)\ q_1 + (\sin\theta_1 t)\ \hat{v}_1, \tag{1}$$

$$C_2(t) = (\cos\theta_2(1-t))\ q_2 - (\sin\theta_2(1-t))\ \hat{v}_2, \tag{2}$$

where $\theta_i = \|v_i\|$. One can easily check that (with $t_1 = 0$ and $t_2 = 1$):

$$C_i(t_i) = q_i \quad \text{and} \quad C_i'(t_i) = \theta_i \hat{v}_i = v_i, \quad \text{for } i = 1, 2.$$

2.2 Hermite Curve in S^3

The hermite interpolation curve $Q(t)$ is constructed by blending the two great circular arcs $C_1(t)$ and $C_2(t)$ with a blending function $f(t)$, $0 \le t \le 1$. That is, let γ_t be the geodesic circular arc which connects the two points $C_1(t)$ and $C_2(t)$, then the blending curve point $Q(t)$ is defined to be the interior point of γ_t which subdivides the geodesic arc γ_t in the ratio of $f(t) : 1 - f(t)$. The curve equation of $Q(t)$, $0 \le t \le 1$, is given by:

$$Q(t) = \exp(f(t) \cdot \log(C_2(t) \cdot C_1(t)^{-1})) \cdot C_1(t) \tag{3}$$

$$= \exp((1 - f(t)) \cdot \log(C_1(t) \cdot C_2(t)^{-1})) \cdot C_2(t). \tag{4}$$

(More details on the log and exp functions, and the curve equations of $Q(t)$ are given in Appendix A.) The blending function $f(t)$ of this paper is required to satisfy the boundary conditions:

$$f(0) = 0, \quad f(1) = 1, \quad f'(0) = 0, \quad \text{and} \quad f'(1) = 0. \tag{5}$$

To show that the curve $Q(t)$ is a hermite quaternion curve, we need to prove the boundary conditions:

$$Q(0) = q_1, \quad Q(1) = q_2, \quad Q'(0) = v_1, \quad \text{and} \quad Q'(1) = v_2.$$

It is trivial to show that: $Q(0) = C_1(0) = q_1$ and $Q(1) = C_1(1) = q_2$; that is,

$$Q(0) = \exp(f(0) \cdot \log(C_2(0) \cdot C_1(0)^{-1})) \cdot C_1(0) = \exp(0) \cdot C_1(0) = C_1(0) = q_1,$$

$$Q(1) = \exp((1 - f(1)) \cdot \log(C_1(1) \cdot C_2(1)^{-1})) \cdot C_2(1) = \exp(0) \cdot C_2(1) = C_2(1) = q_2.$$

For the proof of $Q'(0) = v_1$ and $Q'(1) = v_2$, we need the following formulas:

Lemma 2.1 *For $t \in [0,1]$ with $f(t) = 0$, the differential $Q'(t)$ is given by:*

$$Q'(t) = f'(t) \cdot \log(C_2(t) \cdot C_1(t)^{-1}) \cdot C_1(t) + C_1'(t).$$

Proof: Kim, Kim, and Shin [6] ∎

Corollary 2.1 *For $t \in [0,1]$ with $1 - f(t) = 0$, the differential $Q'(t)$ is given by:*

$$Q'(t) = -f'(t) \cdot \log(C_1(t) \cdot C_2(t)^{-1}) \cdot C_2(t) + C_2'(t).$$

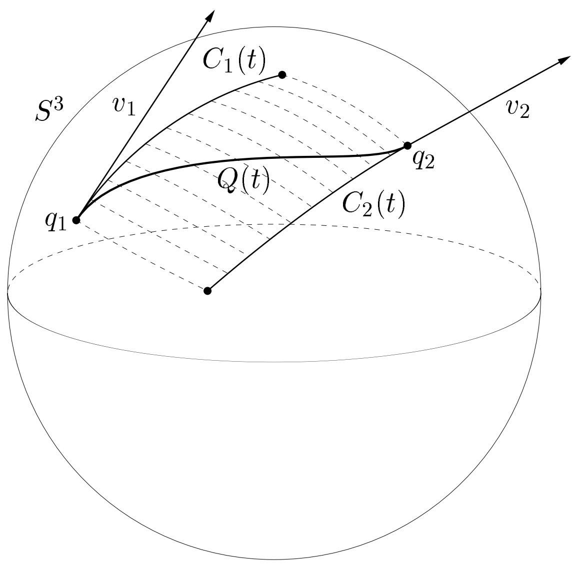

Figure 2: Blending Curve of Two Great Circular Arcs.

Proof: In the second definition of $Q(t)$ in Equation (4), we may interpret $1 - f(t)$ as a blending function and switch the roles of C_1 and C_2, and then apply Lemma 2.1. ∎

Since $f(0) = 0$ and $f'(0) = 0$, by Lemma 2.1 we have:

$$Q'(0) = f'(0) \cdot \log(C_2(0) \cdot C_1(0)^{-1}) \cdot C_1(0) + C_1'(0) = C_1'(0) = v_1.$$

Similarly, since $1 - f(1) = 0$ and $-f'(1) = 0$, by Corollary 2.1 we have:

$$Q'(1) = -f'(1) \cdot \log(C_1(1) \cdot C_2(1)^{-1}) \cdot C_2(1) + C_2'(1) = C_2'(1) = v_2.$$

The blending function $f(t)$, $0 \leq t \leq 1$, satisfies the following boundary conditions of Equation (5):

$$f(0) = 0, \quad f(1) = 1, \quad f'(0) = 0, \quad \text{and} \quad f'(1) = 0.$$

There are many polynomial functions that satisfy the boundary conditions; however, at least degree three is required for a polynomial to satisfy these conditions. The cubic polynomial

$$f(t) = -2t^3 + 3t^2 = (-2t + 3)t^2$$

satisfies the boundary conditions (Figure 3(a)). The following piecewise quadratic polynomial also satisfies the boundary conditions (Figure 3(b)):

$$f(t) = \begin{cases} 2t^2 & \text{if } 0 \leq t \leq 0.5 \\ -2(t-1)^2 + 1 & \text{if } 0.5 < t \leq 1 \end{cases}$$

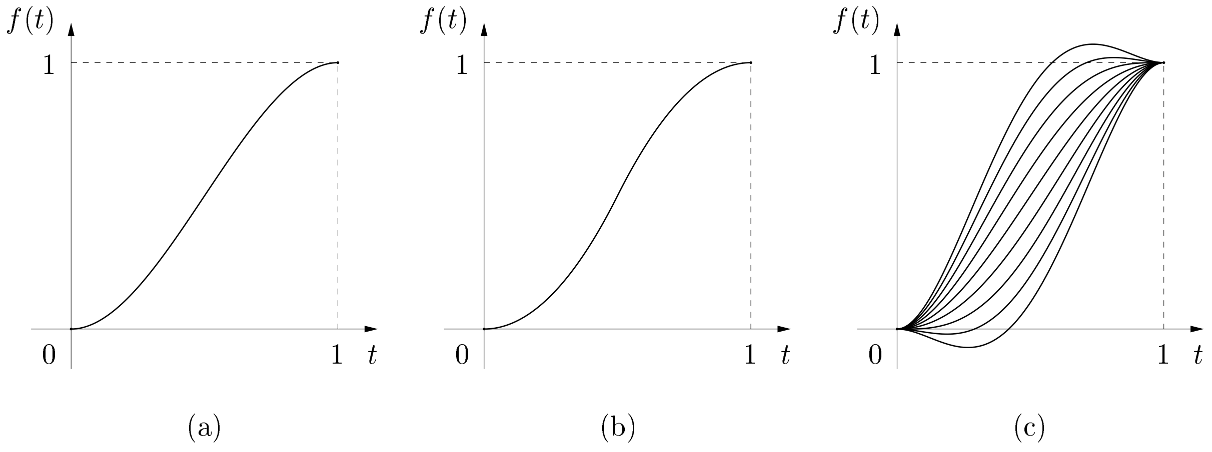

Figure 3: Blending Functions.

Low degree polynomials are preferred for computational efficiency, whereas higher degree polynomials are also preferred for flexibility. High degree polynomials have more degrees of freedom, and they can generate a variety of hermite interpolation curves while satisfying the given boundary conditions (Figure 3(c)). The user may choose the most suitable one from many possible solutions.

3 Forward Difference Method for Great Circular Arcs

In keyframe animation, between two adjacent keyframes at $t = t_1$ and $t = t_2$, a total of $(\frac{t_2 - t_1}{h} - 1)$ in-between frames are generated at $t = t_1 + h, t_1 + 2h, \ldots, t_2 - h$, where $h = \frac{1}{24}$ second for films and $\frac{1}{30}$ second for videos. Thus, the time parameter t is incremented by a fixed time interval $h > 0$. Let $n = \frac{t_2 - t_1}{h}$, then there are $(n-1)$ in-betweens. Though the quaternion curve $Q(t)$ is defined for any $t \in [0, 1]$, only $(n-1)$ in-between unit quaternions, $Q(t_1 + h), Q(t_1 + 2h), \ldots, Q(t_2 - h)$, are needed for the keyframe animation.

For simplicity, we may assume $t_1 = 0$ and $t_2 = 1$, and take $h = 1/n$. Then, the great circular arc $C_1(t)$, $0 \le t \le 1$, generates $(n-1)$ unit quaternions: $q_{1,j} = C_1(jh)$, for $j = 1, 2, \ldots, n-1$. Similarly, $C_2(t)$, $0 \le t \le 1$, generates $(n-1)$ unit quaternions: $q_{2,j} = C_2(jh)$, for $j = 1, 2, \ldots, n-1$. The $(n-1)$ in-between unit quaternions on the curve $Q(t)$, $0 \le t \le 1$, are generated by computing the interior point of $\gamma_{q_{1,j}, q_{2,j}}$ which subdivides the geodesic arc $\gamma_{q_{1,j}, q_{2,j}}$ in the ratio of $jh : 1 - jh$, where $\gamma_{q_{1,j}, q_{2,j}}$ is the geodesic great circular arc which connects the two unit quaternions $q_{1,j}$ and $q_{2,j} \in S^3$. This subdivision is essentially the same as the *slerp* construction of Shoemake [15, 16].

The $(n-1)$ in-between unit quaternions on $q_{1,j} \in C_1(t)$, for $j = 1, \ldots, n-1$, may be generated by evaluating the curve equation of Equation (3) at $t = j/n$. However, there is a more efficient way of evaluating the curve points; that is, a forward difference method can be used to efficiently

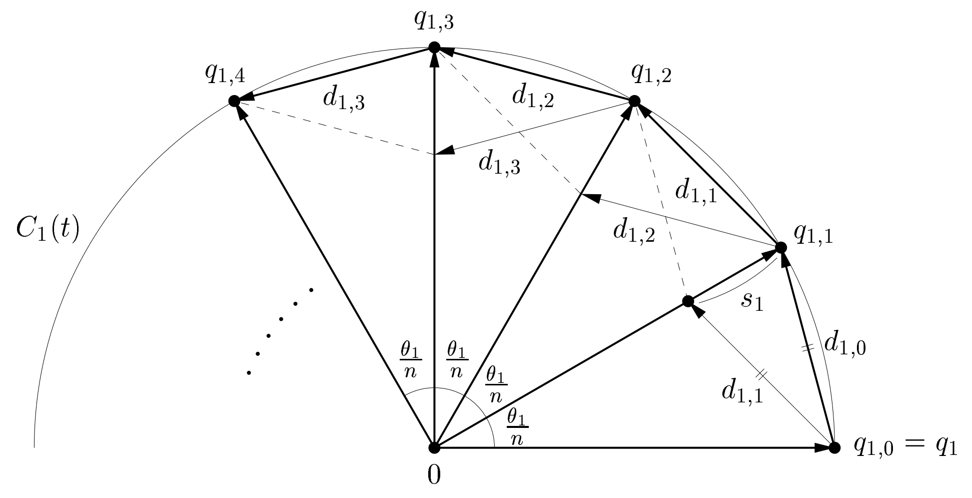

Figure 4: Forward Difference Method.

generate the $(n-1)$ in-between unit quaternions on $C_i(t)$. We present the algorithm only for the case of $C_1(t)$; however, the same construction can be applied to $C_2(t)$, too. Figure 4 shows how to generate a sequence of unit quaternions in a forward difference method.

First of all, we compute $q_{1,1}$ from Equation (3) as follows:

$$q_{1,1} = C_1(h) = q_1 \cos\theta_1 h + \hat{v}_1 \sin\theta_1 h,$$

where $h = 1/n$. The other $(n-2)$ in-between quaternions, $q_{i,j}$'s $(j = 2, \ldots, n-1)$, are generated in an increasing order of j by adding a forward difference vector $d_{i,j}$:

$$q_{1,j+1} = q_{1,j} + d_{1,j}, \quad \text{for } 1 \leq j \leq n-2.$$

The forward difference vector $d_{1,j}$'s are obtained by:

$$\begin{aligned} d_{1,0} &= q_{1,1} - q_1, \\ d_{1,j} &= d_{1,j-1} - s_1 \cdot q_{1,j} \quad \text{for } 1 \leq j \leq n-2. \end{aligned}$$

A parallelogram is formed by the four points: $q_{1,j-1}, q_{1,j}, q_{1,j+1}$, and $q_{1,j} - s_1 \cdot q_{1,j}$. The scalar constant s_1 is the magnitude of the vector that makes $d_{1,j-1}$ turn to a new vector $d_{1,j}$ without changing its length. It is computed by:

$$s_1 = 2\left(1 - \cos\frac{\theta_1}{n}\right).$$

(See Appendix B for the pseudo code for the construction of a hermite quaternion curve and two great circular arcs by the forward difference method.)

4 Computational Efficiency

The cardinal spline quaternion curve of Pletinckx [13] has been the most efficient method for the construction of a quaternion curve which interpolates a given sequence of solid orientations. One *slerp* and two *smid* constructions are required to generate each in-between unit quaternion. Each smid operation requires 9 multiplications and 1 square root. (We ignore the plus and minus operations.) Thus, each in-between quaternion is generated with 18 multiplications, 2 square roots, and 1 slerp. In the forward difference method introduced in this paper, except the extra setup cost for θ_i, $\hat{v}_i$, s_i, $q_{1,1}$, and $q_{2,n-1}$, each unit quaternion $q_{i,j}$ on $C_i(t)$ $(i = 1, 2)$ is generated with only 4 multiplications that are required for the computation of the difference vector $d_{i,j}$. Thus, our method can save 10 multiplications and 2 square roots for each in-between quaternion. The total extra setup cost for $Q(t)$ is 28 multiplications, 2 square roots, 2 sine operations, and 2 cosine operations (see the pseudo code in Appendix B). Thus, the forward difference method of this paper is faster than the cardinal spline curve of Pletinckx [13] when there are more than three in-betweens.

The forward difference method can also improve the computational efficiency of the squad method of Shoemake [16]. Given four unit quaternions $q_i, a_i, b_{i+1}, q_{i+1} \in S^3$, the squad quaternion curve $Q(t)$, $0 \le t \le 1$, is defined by:

$$Q(t) = \exp(2t(1-t) \cdot \log(\gamma_{a_i,b_{i+1}}(t) \cdot \gamma_{q_i,q_{i+1}}(t)^{-1})) \cdot \gamma_{q_i,q_{i+1}}(t),$$

where $\gamma_{p,q}(t)$, $0 \le t \le 1$, is the geodesic great circular arc which connects two unit quaternions $p, q \in S^3$. The two great circular arcs $\gamma_{a_i,b_{i+1}}(t)$ and $\gamma_{q_i,q_{i+1}}(t)$, $0 \le t \le 1$, can be generated by using the forward difference method. Thus, the computational efficiency of the squad method is similar to that of the method presented in this paper.

Given a sequence of unit quaternions $q_i \in S^3$ $(i = 1, \ldots, n)$, both our method and the squad method can be applied to generate a quaternion curve which interpolates the sequence of unit quaternions q_i's. Our method estimates each tangent vector v_i at q_i as:

$$v_i = \frac{1}{2} \log(q_{i+1} \cdot q_{i-1}^{-1}) \cdot q_i, \quad \text{for } i = 2, \ldots, n-1.$$

The squad method [16] estimates a_i and b_i as:

$$a_i = b_i = \exp\left(-\frac{\log(q_{i+1} \cdot q_i^{-1}) + \log(q_{i-1} \cdot q_i^{-1})}{4}\right) \cdot q_i, \quad \text{for } i = 2, \ldots, n-1.$$

Thus, these two methods now become the most efficient methods for the construction of a quaternion curve which interpolates a given sequence of 3D solid orientations.

A possible problem of the forward difference method might be the precision of the numerical computation. However, in our extensive experiments on various test data, the numerical errors are observed to be negligible for the applications in computer animation. Furthermore, the unit quaternions on the first curve $C_1(t)$ are generated in an ascending order, and those on the second curve $C_2(t)$ are generated in a descending order. This means that the beginning part of $C_1(t)$ and the end part of $C_2(t)$ are more accurate than the other parts; that is, the computations are accurate in the more important parts. Thus, the overall blending quaternion curve $Q(t)$ is more accurate at both ends of the curve than the middle of the curve.

5 Conclusions

We have described an efficient algorithm to construct a hermite interpolation quaternion curve on $SO(3)$. Two great arcs $C_1(t)$ and $C_2(t)$, for $0 \leq t \leq 1$, are constructed by an efficient forward difference method so that they satisfy: $C_1(0) = q_1, C_2(1) = q_1, C_1'(0) = v_1$, and $C_2'(1) = v_2$. By smoothly blending the two great circular arcs $C_1(t)$ and $C_2(t)$ on $SO(3)$, a hermite interpolation quaternion curve $Q(t)$, $0 \leq t \leq 1$, is constructed that satisfies all the four boundary conditions: $Q(0) = q_1, Q(1) = q_2, Q'(0) = v_1$, and $Q'(1) = v_2$. With the speed up obtained by the forward difference method, our method and the squad method [16] now become the most efficient methods for the construction of quaternion interpolation curves.

References

[1] Barr, A., Currin, B., Gabriel, S., and Hughes, J., "Smooth Interpolation of Orientations with Angular Velocity Constraints using Quaternions," *Computer Graphics (Proc. of SIGGRAPH '92)*, Vol. 26, No. 2, 1992, pp. 313–320.

[2] Curtis, M., *Matrix Groups*, Springer-Verlag, New York, 1979.

[3] Gleicher, M., and Witkin, A., "Through-the-Lens Camera Control," *Computer Graphics*, Vol. 26, No. 2, 1992, pp. 331–340,

[4] Hanotaux, G., and Peroche, B., "Interactive Control of Interpolations for Animation and Modeling," *Proc. of Graphics Interface '93*, pp. 201–208, 1993.

[5] Jüttler, B., "Visualization of Moving Objects Using Dual Quaternion Curves," *Computers & Graphics*, Vol. 18, No. 3, pp. 315–326, 1994.

[6] Kim, M.-J., Kim, M.-S., and Shin, S., "A Compact Differential Formula for the First Derivative of a Unit Quaternion Curve," to appear in *J. of Visualization and Computer Animation*.

[7] Kim, M.-J., Kim, M.-S., and Shin, S., "A C^2-continuous B-spline Quaternion Curve Interpolating a Given Sequence of Solid Orientations," *Proc. of Computer Animation '95*, pp. 72–81, Geneva, Switzerland, April 19–21, 1995.

[8] Kim, M.-J., Kim, M.-S., and Shin, S., "A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives," *Proc. of SIGGRAPH '95*, pp. 369–376, Los Angeles, August 6–11, 1995.

[9] Kim, M.-S., and Nam, K.-W., "Interpolating Solid Orientations with Circular Blending Quaternion Curves," *Computer-Aided Design*, Vol. 27, No. 5, pp. 385–398, 1995.

[10] Kim, M.-S., and Nam, K.-W., "Hermite Interpolation of Solid Orientations with Circular Blending Quaternion Curves," to appear in *J. of Visualization and Computer Animation*.

[11] Kyung, M.-H., Kim, M.-S., and Hong, S., "Through-the-Lens Camera Control with a Simple Jacobian Matrix," Technical Report CS-CG-94-006, Dept. of Computer Science, POSTECH, 1994.

[12] Nielson, G., "Smooth Interpolation of Orientations," *Models and Techniques in Computer Animation (Proc. of Computer Animation '93)*, N.M. Thalmann and D. Thalmann (Eds.), Springer-Verlag, pp. 75–93, 1993.

[13] Pletinckx, D., "Quaternion Calculus as a Basic Tool in Computer Graphics," *The Visual Computer*, Vol. 5, No. 1, pp. 2–13, 1989.

[14] Schlag, J., "Using Geometric Constructions to Interpolate Orientation with Quaternions," *Graphics Gems II*, J. Arvo (Ed.), Academic Press, Boston, pp. 377–380, 1991.

[15] Shoemake, K., "Animating Rotation with Quaternion Curves," *Computer Graphics (Proc. of SIGGRAPH '85)*, Vol. 19, No. 3, pp. 245–254, 1985.

[16] Shoemake, K., "Quaternion Calculus for Animation," *Math for SIGGRAPH (ACM SIGGRAPH '91 Course Notes #2)*, 1991.

[17] Wang, W., and Joe, B., "Orientation Interpolation in Quaternion Space Using Spherical Biarcs," *Proc. of Graphics Interface '93*, pp. 24–32, 1993.

A Logarithmic and Exponential Maps

Given a unit quaternion $q = (w, x, y, z) \in S^3$ (i.e., $w^2 + x^2 + y^2 + z^2 = 1$), let

$$\theta = \arccos w \in [0, \pi] \quad \text{and} \quad (a, b, c) = \frac{(x, y, z)}{\sqrt{x^2 + y^2 + z^2}} \in S^2.$$

Then we have

$$\begin{aligned} w &= \cos\theta, \\ \sin\theta &= \sqrt{1 - \cos^2\theta} \;=\; \sqrt{1 - w^2} \;=\; \sqrt{x^2 + y^2 + z^2}, \quad \text{and} \\ (x, y, z) &= \sqrt{x^2 + y^2 + z^2} \cdot (a, b, c) \;=\; \sin\theta \cdot (a, b, c). \end{aligned} \tag{6}$$

Thus, the unit quaternion $q \in S^3$ can be represented by:

$$q = (\cos\theta, \sin\theta \cdot (a, b, c)),$$

where $0 \le \theta \le \pi$ and $(a, b, c) \in S^2$.

A unit quaternion $q = (\cos\theta, \sin\theta \cdot (a,b,c)) \in S^3$ maps each point $p = (x,y,z) \in R^3$ into a point $\hat{p} = (\hat{x}, \hat{y}, \hat{z}) \in R^3$ which is given by the relation (see [2, 16]):

$$(0, \hat{x}, \hat{y}, \hat{z}) = q \cdot (0, x, y, z) \cdot \overline{q},$$

where the $\cdot$ operation is the quaternion multiplication and $\overline{q}$ is the conjugate quaternion of q, i.e., $\overline{q} = (\cos\theta, -\sin\theta \cdot (a,b,c)) \in S^3$. One can easily check that the point $\hat{p}$ is the same as the one which is obtained by rotating the point p about an axis (a,b,c) by an angle 2θ. Thus each unit quaternion $q \in S^3$ represents such a rotation; at the same time, it represents the orientation (of a 3D solid) which is obtained by rotating the solid from the standard orientation (which is represented by the identity quaternion $1 \equiv (1,0,0,0)$) about an axis (a,b,c) by an angle 2θ. Note that the two antipodal points q and $-q$ of S^3 give the same rotated point $(\hat{x}, \hat{y}, \hat{z}) \in R^3$; thus the two unit quaternions q and $-q \in S^3$ represent the same orientation. By identifying each pair of two antipodal points q and $-q \in S^3$ as a single orientation, the rotation group $SO(3)$ is obtained as a projective space of S^3.

Under the rotation about an axis $(a,b,c) \in S^2$ by an angle 2θ, each intermediate orientation obtained is the same as the one which is obtained by rotating the 3D solid about the same axis $(a,b,c) \in S^2$ by an angle $2\theta t$ for some $t \in [0,1]$. When we rotate a given 3D solid from the standard orientation, the trace of all the intermediate orientations can be represented by the geodesic circular arc $\gamma_{1,q}(t) \in S^3$, $0 \le t \le 1$, which connects the two unit quaternions: 1 and q; that is,

$$\gamma_{1,q}(t) = (\cos\theta t, \sin\theta t \cdot (a,b,c)), \quad \text{for } 0 \le t \le 1.$$

The geodesic circular arc in S^3 or $SO(3)$ is also called as *slerp* in computer graphics (following its first usage in Shoemake [15]). The velocity for the curve $\gamma_{1,q}(t)$ is given by:

$$\begin{aligned} \gamma_{1,q}'(t) &= (-\theta \sin\theta t, \theta \cos\theta t \cdot (a,b,c)) \\ &= (0, \theta \cdot (a,b,c)) \cdot (\cos\theta t, \sin\theta t \cdot (a,b,c)) \\ &= (0, \theta \cdot (a,b,c)) \cdot \gamma_{1,q}(t), \end{aligned}$$

for $0 \le t \le 1$, where the $\cdot$ operation between two 4D vectors means the quaternion multiplication. In this paper, the $\cdot$ operation is interpreted as scalar multiplication, inner product, or quaternion multiplication depending on the context. The initial velocity is thus given by:

$$\gamma_{1,q}'(0) = (0, \theta \cdot (a,b,c)).$$

The mapping from each unit quaternion $q = (\cos\theta, \sin\theta \cdot (a,b,c))$ to the initial velocity $(0, \theta \cdot (a,b,c))$ gives a well-defined map from the 3-sphere S^3 (of unit quaternions) into the tangent space $T_1 S^3 \equiv R^3$. The tangent space $T_1 S^3$ is the space of all angular velocities from the standard orientation. The Talyor series expansion of the logarithmic function for unit quaternions realizes such a map (see [2]):

$$\log(\cos\theta, \sin\theta \cdot (a,b,c)) = (0, \theta \cdot (a,b,c)). \tag{7}$$

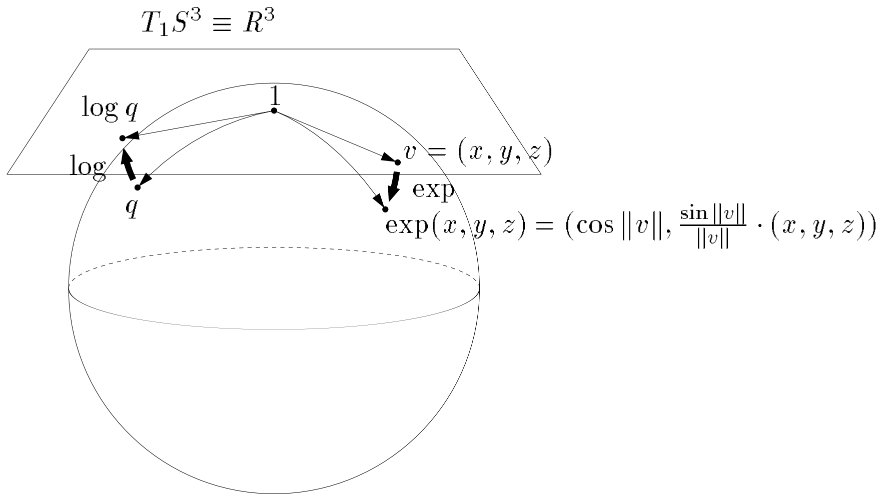

Figure 5: Log and Exp Maps.

It is easier to prove the inverse relation:

$$\exp(0, \theta \cdot (a, b, c)) = (\cos \theta, \sin \theta \cdot (a, b, c))$$

by expanding the Taylor series of the exponential function for $(0, \theta \cdot (a, b, c))$ (see [9, 10]). Thus the unit quaternion curve $\gamma_{1,q}(t)$ $(0 \leq t \leq 1)$ can be represented by the log and exp maps:

$$\begin{aligned}
\gamma_{1,q}(t) &= (\cos \theta t, \sin \theta t \cdot (a, b, c)) \\
&= \exp(0, \theta t \cdot (a, b, c)) \\
&= \exp(t \cdot (0, \theta \cdot (a, b, c))) \\
&= \exp(t \cdot \log(\cos \theta, \sin \theta \cdot (a, b, c))) \\
&= \exp(t \cdot \log(q)).
\end{aligned}$$

Furthermore, the geodesic circular arc $\gamma_{q_1,q_2}(t) \in S^3$ $(0 \leq t \leq 1)$ which connects two unit quaternions q_1 and $q_2 \in S^3$ is given by (also see [9, 10]):

$$\gamma_{q_1,q_2}(t) = \exp(t \cdot \log(q_2 \cdot q_1^{-1})) \cdot q_1, \quad \text{for } 0 \leq t \leq 1. \tag{8}$$

The curve differential is also given by:

$$\gamma'_{q_1,q_2}(t) = \gamma'_{1,q_2 \cdot q_1^{-1}}(t) \cdot q_1 = \log(q_2 \cdot q_1^{-1}) \cdot \gamma_{q_1,q_2}(t), \quad \text{for } 0 \leq t \leq 1.$$

Using the identities in Equation (6) and ignoring the first component of the right-hand side of Equation (7) which is always 0, we can define the logarithmic map $\log : S^3 \rightarrow R^3$ as follows (see Figure 5).

Definition A.1 *The logarithmic map* $\log : S^3 \to R^3$ *is defined by:*

$$\log(w,x,y,z) = \arccos w \cdot \frac{(x,y,z)}{\sqrt{x^2+y^2+z^2}} = \frac{\arccos w}{\sqrt{1-w^2}} \cdot (x,y,z), \quad \text{for } (w,x,y,z) \in S^3.$$

Furthermore, given an initial velocity $v = (x,y,z) = \theta \cdot (a,b,c)$, by using the relation:

$$\begin{aligned} \theta &= \|v\| = \sqrt{x^2+y^2+z^2} \\ (a,b,c) &= \frac{1}{\sqrt{x^2+y^2+z^2}} \cdot (x,y,z) = \frac{1}{\|v\|} \cdot (x,y,z), \end{aligned}$$

the exponential map exp (which is the inverse map of log) can be defined as follows.

Definition A.2 *The exponential map* $\exp : R^3 \to S^3 \subset R^4$ *is defined by:*

$$\exp(x,y,z) = \begin{cases} (\cos\|v\|, \frac{\sin\|v\|}{\|v\|} \cdot (x,y,z)) & \text{if } v = (x,y,z) \neq (0,0,0) \\ (1,0,0,0) & \text{if } v = (x,y,z) = (0,0,0) \end{cases}$$

B Pseudo Code for a Hermite Quaternion Curve

Algorithm HermiteCurveS3 ($q1$, $q2$, $v1$, $v2$, n)
  begin
  { Step 1: Generate points on the two great arcs. }
  $Q1[1,\ldots,n-1] \leftarrow$ GreatCircularArcIncr ($q1$, $v1$, n);
  $Q2[1,\ldots,n-1] \leftarrow$ GreatCircularArcDecr ($q2$, $v2$, n);

  { Step 2: Generate points on the hermite quaternion curve. }
  $Q[0] \leftarrow q_1$;
  $Q[n] \leftarrow q_2$;
  for $i := 1$ **to** $n-1$ **do**
    $Q[i] \leftarrow$ Slerp ($Q1[i]$,$Q2[i]$,$f(i/n)$);
  return ($Q[0,\ldots,n]$);
  end

Algorithm GreatCircularArcIncr (q, v, n)
  begin
  { Step 1: Setup the forward difference method. }
  $\theta \leftarrow \| v \| /n$;

$\hat{v} \leftarrow$ normalize v;
$Q[1] \leftarrow q \cdot \cos\theta + \hat{v} \cdot \sin\theta$;
$d \leftarrow Q[1] - q$;
$s \leftarrow 2(1 - \cos\theta)$;

{ Step 2: Apply the forward difference method in the increasing order of i. }
for $i := 2$ **to** $n - 1$ **do**
  begin
  $d \leftarrow d - s \cdot Q[i-1]$;
  $Q[i] \leftarrow Q[i-1] + d$;
  end
return $(Q[1, \ldots, n-1])$;
end

Algorithm GreatCircularArcDecr (q, v, m)
begin
{ Step 1: Setup the forward difference method. }
$\theta \leftarrow \parallel v \parallel /n$;
$\hat{v} \leftarrow$ normalize v;
$Q[n-1] \leftarrow q \cdot \cos\theta - \hat{v} \cdot \sin\theta$;
$d \leftarrow Q[n-1] - q$;
$s \leftarrow 2(1 - \cos\theta)$;

{ Step 2: Apply the forward difference method in the decreasing order of i. }
for $i := n - 2$ **downto** 1 **do**
  begin
  $d \leftarrow d - s \cdot Q[i+1]$;
  $Q[i] \leftarrow Q[i+1] + d$;
  end
return $(Q[1, \ldots, n-1])$;
end

Kee-Won Nam received the B.S. and M.S. degrees in Computer Science from POSTECH, Korea, in 1992 and 1994, respectively. He is currently a Ph.D. student in the Department of Computer Science, POSTECH. His research interests include Computer Graphics and Computer Animation.
Address: Department of Computer Science, POSTECH, San 31, Hyoja Dong, Pohang 790-784, South Korea.
Electronic mail: namkw@vision.postech.ac.kr

Myung-Soo Kim is an associate professor in the Department of Computer Science, POSTECH, Korea. He is currently sabbatical leave from POSTECH, and a visiting associate professor in the Department of Computer Science, Purdue University, U.S.A. His research interests are in Computer Graphics, Computer Animation, and Geometric Modeling.
Address: Department of Computer Science, Purdue University, West Lafayette, IN 47907-1398, U.S.A.
Electronic mail: mskim@cs.purdue.edu