iSlerp: An Incremental Approach to Slerp

Xin Li
Computer Science Department
Digipen Institute of Technology
xli@digipen.edu

Abstract

In this paper, an incremental quaternion interpolation algorithm is introduced. With the assumption of a constant interval between a pair of quaternions, the cost of interpolation algorithm is significantly reduced. Expensive trigonometric calculations in Slerp are replaced with simple linear combination arithmetic. The round-off errors and drifting behavior accumulated through incremental steps are also analyzed in the paper.

Introduction

Quaternions are widely employed to represent rotations of objects in 3D graphics. During an animation, quaternions are often interpolated to generate intermediate orientations between a pair of key frames. A number of quaternion interpolation algorithms have been investigated (Lerp, Slerp [1], nLerp [2], log-Lerp [3], polySlerp [4], etc.). Among them, the spherical linear interpolation (Slerp) interpolates a pair of quaternions on the surface of a unit sphere. The resulting quaternions follow a geodesic path with constant angular velocity. Unfortunately, the application of Slerp in video games and real-time simulation software is limited due to its high computational cost.

This paper introduces an incremental approach of quaternion interpolation. With the assumption of a constant interval, for each interpolated quaternion q_k (k=1, …, n–1), an additional quaternion p_k is also computed. This "auxiliary" quaternion allows us to express the next quaternion q_{k+1} on the path as addition of two scaled quaternions from the previous step. Therefore the cost of interpolation is significantly reduced.

Slerp

We start with the original Slerp equation

$$q_t = \frac{1}{\sin(\alpha)}(\sin(\alpha - t\alpha)q_0 + \sin(t\alpha)q_n), \quad 0 \le t \le 1$$

where q_0 and q_n are unit quaternions and $q_0 \cdot q_n = \cos(\alpha)$. If we assume that α is interpolated incrementally through n steps with a fixed interval, then the angle between any intermediate quaternions q_k and q_{k+1} can be expressed by a constant $\beta = \alpha / n$. When t varies from 0 to 1, we always have $t\alpha = k\beta$ with k=0, 1, …, n and

$$q_k = \frac{1}{\sin(\alpha)}(\sin(\alpha - k\beta)q_0 + \sin(k\beta)q_n), \quad k = 0, 1, ..., n \qquad \text{(E-1)}$$

Note that (E-1) produces exactly the same sequence of interpolated quaternions as Slerp.

iSlerp

Now we introduce the incremental approach by presenting the following equations:

$$\begin{aligned} q_k &= \frac{1}{\sin(\alpha)}(\sin(k\beta)q_n + \sin(\alpha - k\beta)q_0) \\ p_k &= \frac{1}{\sin(\alpha)}(\cos(k\beta)q_n - \cos(\alpha - k\beta)q_0) \end{aligned} \qquad k=0, 1, \ldots, n \qquad \text{(E-2)}$$

The first equation recites the Slerp definition (E-1). It computes q_k between q_0 and q_n for any k. (The terms of q_0 and q_n are swapped for a cosmetic reason). The second equation specifies another quaternion p_k, which is the derivative of q_k (with respect to k) divided by β to normalize it. Therefore p_k is always a unit quaternion perpendicular to q_k. We hence call p_k as q_k's *tangent quaternion*

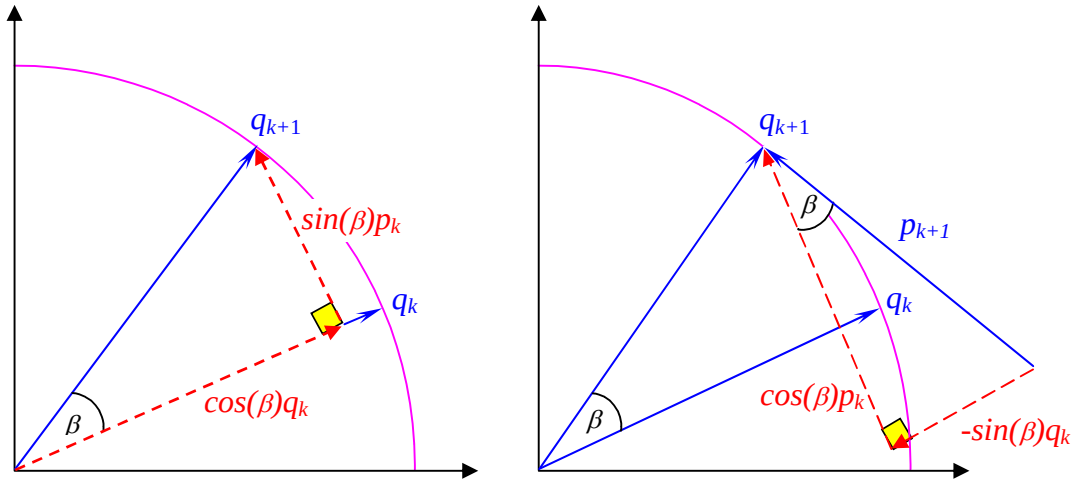
Plugging k=0 in (E-2), we have

$$\begin{aligned} q_0 &= \frac{1}{\sin(\alpha)}(\sin(0)q_n + \sin(\alpha)q_0) = q_0 \\ p_0 &= \frac{1}{\sin(\alpha)}(\cos(0)q_n - \cos(\alpha)q_0) = \frac{1}{\sin(\alpha)}(q_n - \cos(\alpha)q_0) \end{aligned} \qquad \text{(E-3)}$$

The equations can also be extended to the following forms for k+1:

$$\begin{aligned} q_{k+1} &= \frac{1}{\sin(\alpha)}(\sin((k+1)\beta)q_n + \sin(\alpha - (k+1)\beta)q_0) \\ &= \cos(\beta)q_k + \sin(\beta)p_k \\ p_{k+1} &= \frac{1}{\sin(\alpha)}(\cos((k+1)\beta)q_n - \cos(\alpha - (k+1)\beta)q_0) \\ &= \cos(\beta)p_k - \sin(\beta)q_k \end{aligned} \qquad \text{(E-4)}$$

(E-3) and (E-4) are recursive forms of (E-2). They indicate that, along the geodesic path of interpolated orientations, the next q_{k+1} (and its counterpart p_{k+1}) can be calculated based on two constants ($\cos(\beta)$ and $\sin(\beta)$), the current quaternion q_k and its tangent p_k. Figure 1 illustrates the geometric intuitions.



$q_{k+1}=\cos(\beta)q_k + \sin(\beta)p_k$ $\quad$ $p_{k+1}=\cos(\beta)p_k - \sin(\beta)q_k$

Figure 1: Computing q_{k+1} and p_{k+1} by q_k and p_k

Discussion

From (E-4), q_{k+1} is a linear combination of q_k and p_k with two scalars. Hence q_{k+1} must be on the same plane defined by q_k and p_k. Moreover, since q_k and p_k are defined by linear combinations of q_0 and q_n (E-2) and hence must be on the same plane with q_0 and q_n. Consequently, all quaternions with different k define a geodesic path between q_0 and q_n.

Obviously iSlerp is faster than most of quaternion interpolation algorithms. Yet it preserves all geometric characteristics of Slerp. The algorithm is simple to understand and very easy to implement. No logarithmic, polynomial or spline functions involved, no parameter-tuning needed. Since the incremental interpolation only use additions and multiplications, the calculation is immune to numerical instability.

However, since the method is based on incremental steps over a constant angle, the implementation on any hardware and software platform would inevitably introduce some cumulative floating-point round-off errors. To measure accumulated errors and examine the "drifting behavior", a test is conducted on a PC in Windows environment. The

program was written in C++ to test 1,000 pairs of quaternions with angles of 15^{o} to 90^{o} between each pair. Those quaternion pairs are interpolated through 200 steps by both iSlerp and original Slerp functions. Then the results are compared and the differences are computed. Two types of errors are recorded: (1) the maximum discrepancies of angles between the start and interpolated quaternions, and (2) the maximum differences of *x*, *y*, *z* and *w* components of two interpolated quaternions. Figures in next page illustrate the angle and components errors of each step of 1,000 pairs of quaternions.

As expected, both angles and components of iSlerp-interpolated quaternions tend to drift away from results of conventional Slerp function. The cumulative round-off errors increase near-linearly with the number of interpolation steps between the start and end quaternions.

However, the test also indicates that this drifting behavior is insignificantly trivial. Even in the worst cases, all maximum round-off errors are well below 10^{-5}. Note that only the single-precision 32-bits floating point was used in the above test. The round-off errors should be further reduced under double-precision 64-bits operations. In conclusion, since the deviation is relatively small, the accuracy of iSlerp algorithm should be sufficient to most applications in real-time simulation and video game software.

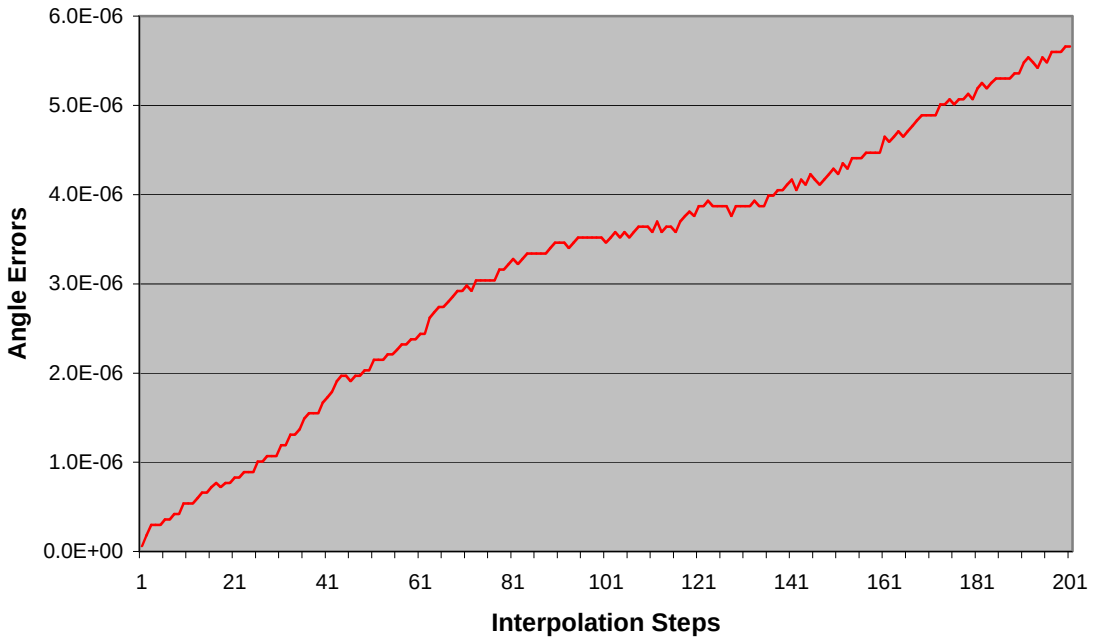


Figure 2: Cumulative round-off errors of angles

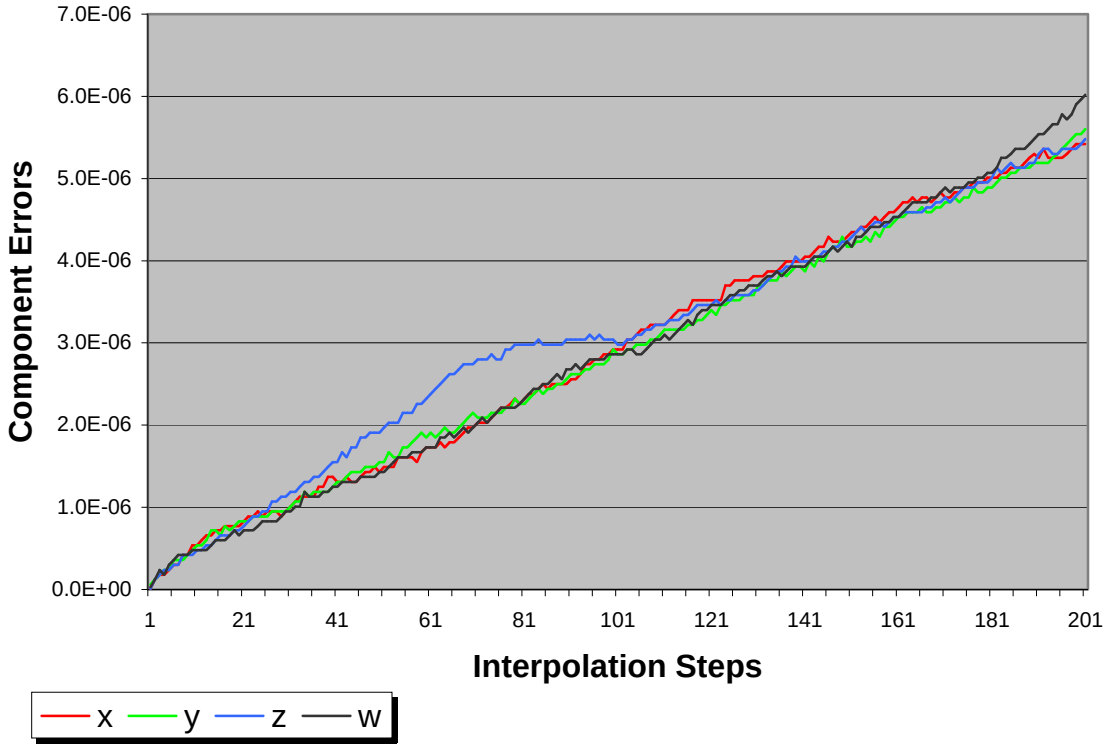


Figure 3: Cumulative round-off errors of quaternion components

References:

[1] Ken Shoemake, “Animating Rotation with Quaternion Curves”, Computer Graphics, Volume 19, Number 3, 1985.

[2] Jonathan Blow, “Hacking Quaternions”, Game Developer Magazine, March 2002.

[3] F. Sebastian Grassia, “Practical parameterization of rotations using the exponential map”, Journal of Graphics Tools, volume 3.3, 1998.

[4] Thomas Busser, “PolySlerp, A Fast and Accurate Polynomial Approximation of Slerp”, Game Developer Magazine, February, 2004.