Incremental Spherical Interpolation with Quadratically Varying Angle

Anders Hast *
Creative Media Lab,
University of Gävle

Tony Barrera †
Barrera Kristiansen AB

Ewert Bengtsson ‡
Centre for Image Analysis
Uppsala University

Abstract

Spherical linear interpolation has got a number of important applications in computer graphics. We show how spherical interpolation can be performed efficiently even for the case when the angle vary quadratically over the interval. The computation will be fast since the implementation does not need to evaluate any trigonometric functions in the inner loop. Furthermore, no renormalization is necessary and therefore it is a true spherical interpolation. This type of interpolation, with non equal angle steps, should be useful for animation with accelerating or decelerating movements, or perhaps even in other types of applications.

1 Introduction

Spherical linear interpolation (SLERP) [Glassner 1999] has got a number of important applications in computer graphics, for instance in animation [Shoemake 1985]. SLERP is different from linear interpolation (LERP) in the way that the angle between each vector or quaternion [Shankel 2000] will be constant, i.e. the movement will have constant speed. In the following text we will refer to quaternions even though everything presented can be used for vectors of any dimension. Nevertheless, the most probable application is for animation and here SLERP is used to interpolate quaternions.

LERP requires normalization and will also yield larger angles in the middle of the interpolation sequence. This will cause animated movements to accelerate in the middle, which is sometimes undesirable [Parent 2002]. However, if we can control the acceleration it should be useful in some cases since movements does not always have equal speed. There is already something called Spherical Quadratic Interpolation [Watt 1992], which involves quadratic interpolation between two linear interpolations and is therefore something completely different from what we propose herein. In our case we will have a quadratically varying angle, ie. the step size will will increase linearly.

The formula used for SLERP is

$$q(t) = q_1 \frac{\sin((1-t)\theta)}{\sin(\theta)} + q_2 \frac{\sin(t\theta)}{\sin(\theta)} \tag{1}$$

where $t \in [0,1]$, and θ is the angle between q_1 and q_2 computed as

$$\theta = cos^{-1}(q_1 \cdot q_2) \tag{2}$$

Note that q can be a quaternion or a vector of any dimension as explained above.

*e-mail: aht@hig.se
†e-mail: tony.barrera@spray.se
‡e-mail: ewert@cb.uu.se

It has been shown that SLERP can be efficiently performed without any computation of trigonometric functions in the inner loop [Barrera 2004]. In this paper we will show that it is possible to compute SLERP with a quadratically varying angle in a similar way. This type of interpolation should be useful for cases when the speed of the movement is close to quadratic, i.e accelerating or decelerating. The movement will be quadratic and the angle between two intermediate quaternions will increase linearly.

In [Barrera 2004] a number of alternative approaches are discussed that give efficient computation in the inner loop. The fastest one is the approach which uses Chebyshev's recurrence [Barrera 2004] but it requires equal angle interpolation. Another approach uses the De Moivre's formula [Hast 2003] and we shall concentrate on this approach in this paper since it can be changed to a quadratically varying angle interpolation quite easily.

1.1 Fast Incremental SLERP

The equation (1) can be rewritten as

$$q(n) = q_1 \cos(nK_\theta) + q_o \sin(nK_\theta) \tag{3}$$

where q_o is the quaternion obtained by applying one step of Gram-Schmidt's orthogonalization algorithm [Nicholson 1995] and then it is normalized. The following code in matlab performs this operation and is assumed to come before the following code examples.

```
qo=q2-(dot(q2,q1))*q1;
qo=qo/norm(qo);
```

The quaternion q_0 is orthogonal to q_1 and lies in the same hyper plane spanned by q_1 and q_2. Furthermore, if there are k steps then the angle between each quaternion is $K_\theta = \frac{\cos^{-1}(q_1 \cdot q_2)}{k}$.

```
theta=acos(dot(q1,q2));
kt=theta/k;
q(1,:)=q1;
```

The code that follows on these three lines can be used for computing incremental SLERP. Usually SLERP is computed using trigonometric functions in the inner loop in the following way

```
b=kt;
for n=2:k+1
  q(n,:)=q1*cos(b)+qo*sin(b);
  b=b+kt;
end
```

However, by applying the De Moivre's formula [Nicholson 1995] it can also be computed efficiently using complex multiplication in this way

```
Z1=cos(kt)+i*sin(kt);
Z=Z1;
for n=2:k+1
  q(n,:)=q1*real(Z)+qo*imag(Z);
  Z=Z1*Z;
end
```

We can split the complex multiplication for each step into two different steps. But first we shall discuss quadratic interpolation in general in next section.

1.2 Quadratic Interpolation

We will start by showing how the quadratic interpolation for SLERP can be setup. Then in next section we will show how it can be computed efficiently. In [Hast 2003] it is shown how a quadratic interpolation can be setup so that it is computed by 2 additions only as it was proposed for shading by Duff [Duff 1979].

It is shown that a quadratic recurrence

$$t(n) = An^2 + Bn + C \tag{4}$$

where $n = [1..k]$ can be evaluated as

$$t_{i+1} = t_i + dt_i \tag{5}$$

$$dt_{i+1} = dt_i + d^2t \tag{6}$$

where $dt_0 = A + B$, $d^2t = 2A$ and $C = 0$ since we always start from the beginning of the interval between the vectors or quaternions.

2 Spherical Interpolation with Varying Angle

We can now put together an algorithm that uses the De Moivre's formula to interpolate spherically with a varying angle. Let us first split the angle

$$\alpha = \sigma\theta \tag{7}$$

$$\beta = \theta - \alpha \tag{8}$$

where $\sigma \in [-1, 1]$ is a scaling factor that will affect the change in speed.

Now we need to split up θ so that we will have a recurrence going from 0 to θ using α and β in k steps. We can do this by

$$A = \alpha/k^2 \tag{9}$$

$$B = \beta/k \tag{10}$$

To prove that this is the right way to do it, we put this into equation (4) for $n = k$ and we get

$$t(k) = \left(\frac{\alpha}{k^2}\right)k^2 + \left(\frac{\theta - \alpha}{k}\right)k = \theta \tag{11}$$

We know how to interpolate the angle quadratically and we can perform the spherical interpolation as exemplified in the following matlab code

```
theta=acos(dot(q1,q2));
A=sigma*theta; B=theta-A;
A=A/(k*k); B=B/k;
d2t=2*A;
dt=A+B;
t=0;

for n=1:k
  t=t+dt;
  dt=dt+d2t;
  qn(n,:)=q1*cos(t)+qo*sin(t);
end
```

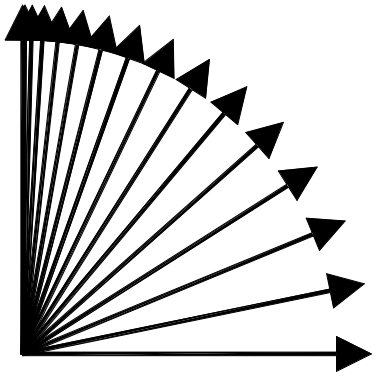

Figure 1: Interpolation with $\sigma = 1.0$.

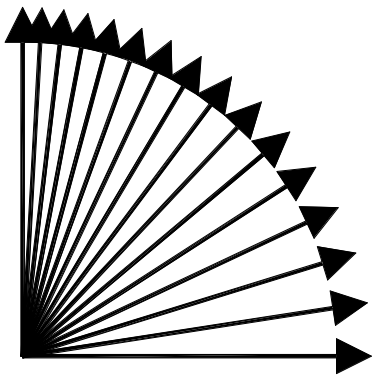

Figure 2: Interpolation with $\sigma = 0.5$.

We can now set up the SLERP with Quadratically varying angle by using the approach recently explained and the code becomes

```
A=sigma*theta; B=theta-A;
A=A/(k*k); B=B/k;

Z1=cos(A+B)+i*sin(A+B);
Z2=cos(2*A)+i*sin(2*A);
Z=1;

for n=1:k
  Z=Z*Z1;
  Z1=Z1*Z2;
  q1*real(Z)+qo*imag(Z)
  qd(n,:)=q1*real(Z)+qo*imag(Z);
end
```

3 Results and Conclusions

We have shown how fast incremental spherical interpolation can be performed for a quadratically varying angle, which can be used for animation of accelerating and decelerating movements.

Figure 1 shows how the angle increases in the interpolation. Here $\sigma = 1.0$. Figure 2 shows that the rate of change can be modified by σ. Here $\sigma = 0.5$. If σ is very small then the approach becomes regular SLERP as shown in figure 3.

It is also possible to create a decelarating movement in a similar way by letting $\sigma < 0$. We have not investigated how the other

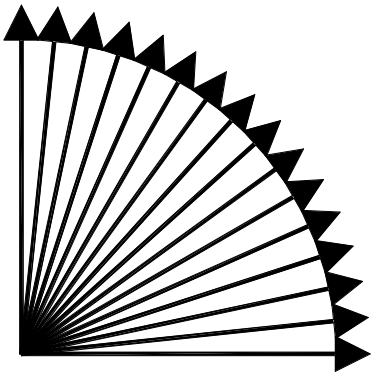

Figure 3: Interpolation with $\sigma = 0.000001$.

approaches for incremental SLERP can be modified for the varying angle approach but is something that should be done in the future. Hopefully this approach can be useful for animation and other applications. It is fast since no trigonometric functions need to be evaluated in the inner loop and the resulting quaternions or vectors will have unit length (if the starting and ending quaternions/vectors have unit length)

References

T. BARRERA, A. HAST, E. BENGTSSON 2004. *Faster shading by equal angle interpolation of vectors* IEEE Transactions on Visualization and Computer Graphics, pp. 217-223.

T. BARRERA, A. HAST, E. BENGTSSON 2005. *Incremental Spherical Linear Interpolation* SIGRAD 2004, pp. 7-10.

T. DUFF 1979. *Smoothly Shaded Renderings of Polyhedral Objects on Raster Displays* ACM, Computer Graphics, Vol. 13, 1979, pp. 270-275.

A. GLASSNER 1999. *Situation Normal* Andrew Glassner's Notebook- Recreational Computer Graphics, Morgan Kaufmann Publishers, pp. 87-97.

A. HAST, T. BARRERA, E. BENGTSSON 2003. *Shading by Spherical Linear Interpolation using De Moivre's Formula* WSCG'03, Short Paper, pp. 57-60.

A. HAST, T. BARRERA, E. BENGTSSON 2003. *Improved Shading Performance by avoiding Vector Normalization,* WSCG'01, Short Paper,2001, pp. 1-8.

W. K. NICHOLSON 1995. *Linear Algebra with Applications* PWS Publishing Company, pp. 275,276.

R. PARENT 2002. *Computer Animation - Algorithms and Techniques* Academic Press, pp. 97,98.

J. SHANKEL 2000. *Interpolating Quaternions* Game Programming Gems. Edited by M. DeLoura. Charles River Media, pp. 205-213

K. SHOEMAKE 1985. *Animating rotation with quaternion curves* ACM SIGGRAPH, pp. 245-254.

A. WATT, M. WATT 1992. *Advanced Animation and Rendering Techniques - Theory and Practice* Addison Wesley, pp. 363, 366.