Fast Construction of Accurate Quaternion Splines

Ravi Ramamoorthi, Alan H. Barr
California Institute of Technology
ravir,barr@gg.caltech.edu

Abstract

In 1992, *Barr et al.* proposed a method for interpolating orientations with unit quaternion curves by minimizing covariant acceleration. This paper presents a simple improved method which uses cubic basis functions to achieve a speedup of up to three orders of magnitude. A new criterion for automatic refinement based on the Euler-Lagrange error functional is also introduced.

1 Introduction

In this paper, we discuss the interpolation of keyframe rotations with quaternion curves. Shoemake [1] introduced the idea of interpolating rotations with quaternions, but the curves (slerps) constructed did not satisfy any obvious variational principle [2] as splines do in flat space. Gabriel and Kajiya [3] then proposed a method that solved the (intrinsic) Euler-Lagrange equations for minimization of covariant acceleration on a manifold with a metric and applied these ideas to interpolation of rotations. In Barr et al. [4], a simpler method to minimize covariant acceleration using an extrinsic formulation based on quaternions was given. However, their approach can take several minutes to hours to compute the optimal curve. Analytic construction schemes such as those of Kim et al [5] are significantly faster and often yield satisfactory curves. If the computational cost can be reduced, minimizing covariant acceleration may be worthwhile.

This paper speeds up the method of Barr et al. [4] significantly. We use simple cubic basis functions and unconstrained minimization instead of the finite difference constrained optimization approach in [4]. Near-optimal curves can be produced in a few seconds—two to three orders of magnitude faster than previous methods. Cubic basis functions can be replaced by other construction schemes so our technique can be used in conjunction with methods such as in [5].

Our work is closely related to other areas of research that use optimization, such as spacetime constraints [6] and variational surface modeling [7]. An important difference is the introduction of a new technique for automatic adaptive refinement based on the "error" as measured by the Euler-Lagrange error functional. Other differences include the use of "variable frames" (the system inserts these) and unconstrained minimization.

The rest of this paper is organized as follows. In section 2, we formulate the quaternion interpolation problem. We present our solution in section 3 and discuss our results in section 4. Section 5 discusses future work and conclusions.

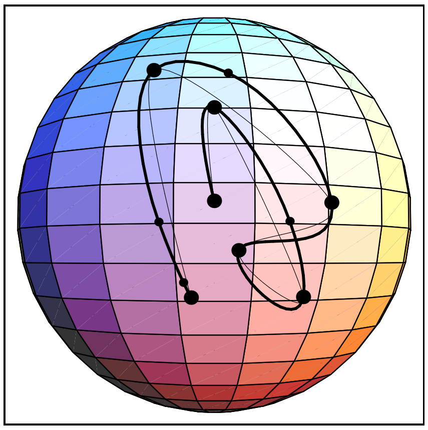

Figure 1: The interpolation problem in quaternion space. The large circles represent keyframes to be interpolated. The small circles show "variable frames" inserted by the system while computing the optimized path (shown with a solid line). A thin line shows the unoptimized path which is seen to have undue accelerations at the keyframes.

2 The Quaternion Interpolation Problem

As in [4] , we want to minimize the covariant acceleration in quaternion space, while interpolating keyframe orientations and (if specified) angular velocities at the end frames.

Let $q(t)$ denote the quaternion path as a function of time such that at keyframe times $t_1, t_2, ..., t_K$, $q(t_i) = Q^i$ where Q^i denotes the quaternion at keyframe i corresponding to time t_i. Further, if specified let ω_1 and ω_K denote the angular velocities at the end frames. Then, the problem we solve can be formulated thus:

$$minimize \int_{t_1}^{t_K} \| q'' \setminus q \|^2 \ dt \qquad (1)$$

subject to the constraints:

$$\text{INTERPOLATION: } \forall i : 1 \le i \le K : q(t_i) = Q^i$$
$$\text{UNITARINESS: } \forall t : t_1 \le t \le t_K : q(t) \cdot q(t) = 1$$
$$\text{VEL: } 2q'(t_1)q^{-1}(t_1) = \omega_1 \quad 2q'(t_K)q^{-1}(t_K) = \omega_K \qquad (2)$$

In this paper, we use cubic polynomials which do not in general lie on the unit sphere. We enforce the constraint of unit quaternions with a "soft constraint" and minimize the objective given below (with the objective put explicitly in terms of quaternion components q_j)

$$OBJ = \int_{t_1}^{t_K} (\sum_{j=0}^{3} q_j'' q_j'' - \frac{(\sum_{k=0}^{3} q_k q_k'')^2}{\sum_{l=0}^{3} q_l q_l} + \alpha[1 - \sum_{n=0}^{3} q_n q_n]^2)\, dt \quad (3)$$

subject to the interpolation constraints. α is a positive constant that forces the quaternions to be nearly unitary. We also require the path $q(t)$ to be C^1 continuous.

Use of Soft Constraints In this paper, we have used cubic basis functions for simplicity, and "soft constraints" to maintain the unitary constraint. While soft constraints are just a weighting term added to the objective, and do not ensure the constraint is exactly met, they are simple to use and often result in the constraint being met well enough for our purposes. Any simple unconstrained minimization package can be used, and the technique is faster than constrained optimization because the constraint does not need to be evaluated and differentiated at each iteration.

3 Our Algorithm

Figure 2 presents a summary of our method. A more detailed discussion of each step is given below.

Algorithm

```
1. User provides keyframes, and number of
   ``variable frames'' (and optionally
   angular velocities at the end frames).
2. The system assigns quaternion velocities
   to the interior keyframes and interpolates
   keyframes and velocities with C^1
   continuous cubics
3. The velocities at the interior keyframes
   are adjusted by the optimizer to minimize
   the objective function
4. In each ``segment'', the average
   Euler-Lagrange error functional is
   computed.  ``Variable frames'' are added
   in a small number of regions having the
   largest deviation
5. In addition to keyframe velocities,
   velocities and positions at the variable
   frames are readjusted so as to minimize
   the objective again.
```

Figure 2: An overview of the algorithm.

Discussion of algorithm

Step 1: User provides keyframes In part 1 of the algorithm, The user must provide the keyframes to be interpolated. In addition, angular velocities at the end frames may be provided if wanted. Angular velocities are converted internally to quaternion velocities: $q'(t) = (1/2)\omega q(t)$.

The value of the soft constraint weighting factor α need also be given. Since we want quaternions of nearly unit magnitude, α should be large. By making α approximately 1000 we tell the optimizer that maintaining unit magnitude is significantly more important than minimizing covariant acceleration. This works very well in that deviation from unitariness is less than 1% in most cases, and the contribution to the objective from the soft constraint is about 1−2% thus ensuring that the soft constraint does not dominate the objective. The implementer may make the value of α a constant which the user need override only if he wants to.

Step 2: System assigns quaternion velocities At all keyframes for which the user has not supplied angular velocities (interior keyframes and end-frames if the user has not specified angular velocities), the system guesses velocities with a very simple algorithm such as: $q_j'(t_i) = (q_j(t_{i+1}) - q_j(t_{i-1}))/(t_{i+1} - t_{i-1})$. With these velocities, a suitable basis function is used to interpolate positions and velocities at the keyframes. In this paper we have used cubic polynomials, but there is no obstacle to using more advanced curves such as the hermite quaternion curves of [5]. Each component of the quaternion path is described by a piecewise cubic such that the position and velocity agree with those user-specified or system-inserted at the keyframes (or in the next stage, "variable frames"). Since both position and velocity are well-defined at the keyframes, the resulting paths must be C^1 continuous as required.

The integral in equation 3 is then evaluated by any simple numerical integration procedure.

Step 3: Minimization of the objective In part 3 of the algorithm, an unconstrained minimization package is used to alter our initial guess of the velocity at interior keyframes so as to minimize the objective OBJ given in equation 3. Note that since we have only C^1 continuity, the objective may be discontinuous, but this will not affect the integral in equation 3. We use the Sequential Quadratic Programming routine E04UCF in the NAG libraries [8]. We note that although the routine can do constrained optimization, it is significantly faster when using soft constraints as we have formulated the problem. To take advantage of sparseness in the problem, one may supply partial derivatives (for which one need evaluate only a small part of the region of integration since the cubic basis is local) instead of having the optimizer evaluate them. Since the method is fast enough even without this optimization, we have not used it in our tests.

Step 4: Checking of Euler-Lagrange error functional We divide the entire path into "segments" of time $\triangle t$. For each segment, we compute the deviation (using the optimized path $Q(t)$ from step 3):

$$DEV(t) = \frac{1}{\triangle t} \int_{t}^{t+\triangle t} \mid EL(t) \mid \, dt \quad (4)$$

where $\mid EL \mid$ stands for the length of the Euler-Lagrange vector discussed below. We then pick the regions having the highest values for $DEV(t)$ (and such that t does not coincide with a keyframe), and add in a "variable frame" at time t. A "variable frame" is like a keyframe except that not only the velocities but also the positions q_i can be varied by the optimizer.

The number of regions in which we add variable frames is a tradeoff between accuracy (the more variable frames the better) and speed (the more variable frames the slower). We have found (as is clearly shown in the results section) that nearly optimal results can be produced with about 5 variable frames.

Some care must be taken in the way the path is divided into segments. If the segments are too short, there may be too many (and unnecessary) variable points inserted in one region at the expense of other regions. On the other hand, the segments should be near enough for multiple variable points to be concentrated in a region. For these reasons, we recommend dividing each time-interval between keyframes into 4 to 5 segments. Since our method is fast, the user may interactively modify the segment lengths.

Step 5: Optimizing again We now repeat the optimization process of step 3 introducing additional variables in the form of the positions and velocities at the variable frames. Figure 1 shows an example of an optimized quaternion spline along with keyframes, variable frames and the unoptimized "keyframish" path.

Euler-Lagrange Error functional

Let $F(q_i, q_i', q_i'')$ denote the integrand in equation 3 (the integral of which is the objective function OBJ). Then, the variational calculus [2] tell us that the corresponding Euler-Lagrange equations must be satisfied on minimization.

$$\frac{\partial F}{\partial q_i} - \frac{d}{dt}\frac{\partial F}{\partial q_i'} + \frac{d^2}{dt^2}\frac{\partial F}{\partial q_i''} = 0 \tag{5}$$

There are four equations corresponding to each component q_i. By measuring the magnitude (Euler-Lagrange error) of the left-hand side of the equation above, we can get an idea of where to refine our coarse representation. The equations for the objective of equation 3 are given below.

Define:

$$a = \sum_{j=0}^{3} q_j q''_j$$

$$b = \sum_{j=0}^{3} q_j q_j$$

$$T = \frac{a}{b}$$

$$U(i) = q_i T^2 - q_i'' T$$

$$V(i) = q_i'' - q_i T$$

$$W(i) = 2\alpha(b-1)q_i$$

$$\frac{d^2}{dt^2}V(i) \equiv q_i'''' - (q_i''T + 2q_i'T' + q_iT'') \tag{6}$$

In the notation of the canonical equation 5, $U(i)$ corresponds to the term $\partial F/\partial q_i$ for covariant acceleration. $W(i)$ is the corresponding term for maintenance of unitary quaternions. $V(i)$ corresponds to the term $\partial F/\partial q_i''$ in equation 5. The term $\partial F/\partial q_i' = 0$ since the objective function does not depend directly on q'.

The Euler-Lagrange deviation or error is then:

$$EL_i = 2(U(i) + \frac{d^2}{dt^2}V(i) + W(i)) \tag{7}$$

Since the first and second derivatives for T are complicated, and our calculations need only be accurate enough to order the segments by deviation, it is simpler to differentiate T numerically rather than analytically.

As our "error" function for adaptive refinement (Step 4 in our algorithm), we use the length of the Euler-Lagrange vector $EL = \sqrt{\sum_{i=0}^{3} EL_i^2}$.

Euler-Lagrange Error Functional as a Metric for Adaptive Refinement We believe our use of Euler-Lagrange equations for adaptive refinement is an improvement over other approaches such as objective or constraint based subdivision [7] because the Euler-Lagrange equations should be 0 on complete minimization while the objective need not go to 0. Thus, a high objective does not necessarily indicate a large error while a "large" value for the Euler-Lagrange deviation is generally a sure indication of a "bad" region. This paper thus also shows how to combine Euler-Lagrange and gradient based methods effectively.

Advantages of Cubics There are a number of advantages that continuous (in our case cubic) basis functions possess over discrete methods [3, 4] of which some of the most important are given below.

- Accurate formulae for quaternion derivatives with respect to time are provided.
- An accurate representation of a curve can be made from a very small number of basis functions, leading to extremely fast algorithms.

4 Results

We present an example with 7 keyframes. Figure 1 shows the path on the sphere (one quaternion component is zero always as are end-point velocities). Large dots represent keyframes; small dots are variable frames. Figure 3 shows the object rotating while moving in a parabola. Keyframes are black; the tops of the variable frames are black. A few of the computed animation frames are shown in grey. In figure 3, one variable frame is not drawn. Refer to figure 1 for its location.

In figure 4 we show the unit magnitude being maintained. The initial path (dotted) has many places where quaternions are far from unit magnitude. Our optimized path—4 variable frames and $\alpha = 1000$—(dashdot) is within 1% of unit magnitude, while the solid line shows the effect of making $\alpha = 10000$ and having 24 variable frames. We see that the quaternions are practically indistinguishable from having unit magnitude (The maximum deviation is .07% in the last example).

Performance

We show the decrease in objective (as a function of program execution time) as more variable frames are added in figure 5. Each point in the graph shows a further level of optimization. The first point shows no optimization, the second adjustment only of velocities at keyframes, the third addition of one variable frame, the fourth two variable frames and so on. We see that with 4 variable frames, we have a result that is only .1% away from optimal.

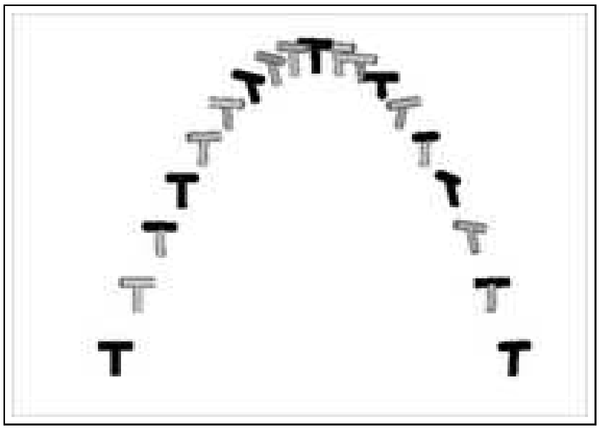

Figure 3: Showing the object path. Keyframes are black, variable frames have black tops and computed animation frames are grey. One variable frame is not drawn in the last region.

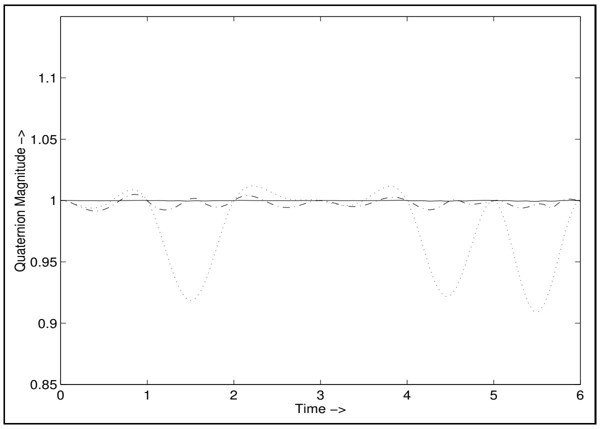

Figure 4: Maintaining unit magnitude. In the original path (dotted), there are significant deviations, while increasing levels of optimization bring the quaternions very close to unit magnitude.

The Euler-Lagrange deviation before and after optimization are shown in figure 6. We see that our method has equidistributed the Euler-Lagrange error well.

Comparison to previous work

Our result with 4 variable frames took 4 seconds to compute. We did not make use of sparseness in the E04UCF routine. We also implemented a discrete method as in [4] where we used the formulae for derivatives given there [except that a factor of two must multiply their results for correctness]. Within this framework, we used the same optimization method and objectives as for the method described in this paper (but since we supplied derivatives, we did make use of sparseness in this case). The running time for the discrete method [4] was 8800 seconds, a factor of more than 2000 slower!! Note that while we used 4 variable frames in our approach, we had to use the entire 600 frames over which the integral was evaluated in the discrete method. Our method is significantly faster because a much smaller number of variables are optimized.

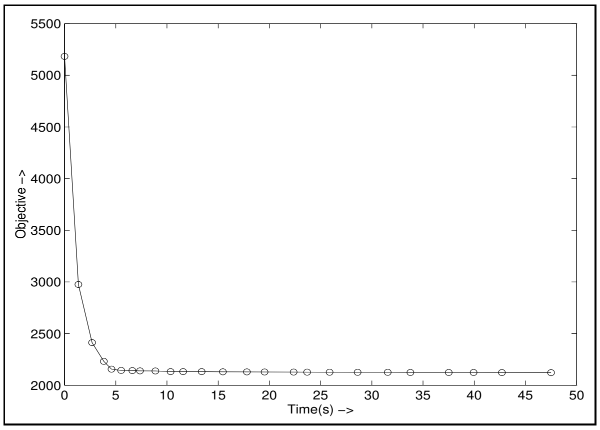

Figure 5: A very small number of variable frames can yield near-optimal results. Each point represents an increasing number of variable frames/higher level of optimization

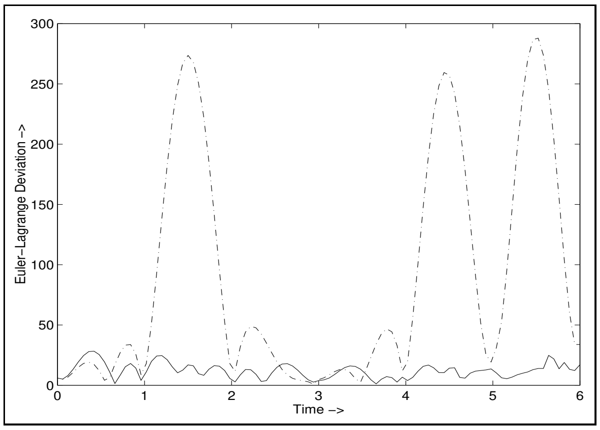

Figure 6: The dotted graph shows the initial Euler-Lagrange error. The solid graph shows how this error is significantly decreased, upon optimization.

5 Conclusion and Future Work

We have shown how to construct quaternion splines orders of magnitude faster than previous methods and introduced a new technique for adaptive refinement based on the Euler-Lagrange error functional. While the results are encouraging, improvements can be made in deriving theoretical bounds on the error in the objective using Euler-Lagrange errors and in combining our techniques with more advanced basis functions.

6 Acknowledgements

We want to thank everyone at the Caltech graphics group for their help and encouragement. This work was supported in part by grants from DEC, Hewlett Packard, and IBM. Additional support was provided by NSF (ASC-89-20219), as part of the NSF/DARPA STC for Computer Graphics and Scientific Visualization. All opinions, findings, conclusions or recommendations expressed here are those of the authors only and do not necessarily reflect the views of the sponsoring agencies.

References

[1] K. Shoemake. *Animating Rotation with Quaternion Curves. Siggraph 85*

[2] D. Zwillinger. *Handbook of Differential Equations. Academic Press, San Diego, 1989.*

[3] S. Gabriel and J. Kajiya, *Spline interpolation in curved space." Siggraph 85 Course notes*

[4] A. H. Barr, B. Currin, S. Gabriel and J. F. Hughes, *Smooth interpolation of orientations with angular velocity constraints using quaternions ,Siggraph 92*

[5] M.-J. Kim , M.-S. Kim, and S. Shin. *A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives. Siggraph 95*

[6] M.Cohen *Interactive Spacetime Control for Animation, Siggraph 92*

[7] W. Welch and A. Witkin *Variational Surface Modeling. Siggraph 92*

[8] Numerical Algorithms Group Ltd. *NAG fortran library document, 1988.*