Fast Construction of Accurate Quaternion Splines

Ravi Ramamoorthi
ravir@gg.caltech.edu

Alan H. Barr
barr@gg.caltech.edu

California Institute of Technology *

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.

CR Categories: I.3.5 [*Computer Graphics*]: Computational Geometry and Object Modeling—Splines; G.1.6 [*Numerical Analysis*]: Optimization

Keywords: Euler-Lagrange error functional, Quaternions, Splines, Optimization

1 Introduction

In this paper, we discuss the interpolation of keyframe rotations with quaternion [6, 17] curves. Shoemake [16] introduced the idea of interpolating rotations with quaternions, but the constructed curves (slerps) did not satisfy an obvious variational principle [21] as splines [2] do in flat space. Gabriel and Kajiya [4] 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. [1], 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 [10] are significantly faster and often yield satisfactory curves.

This paper speeds up the method of Barr et al. [1] significantly, thus allowing minimization of covariant acceleration to be used as an interactive tool. We use simple cubic basis functions and unconstrained minimization instead of the finite difference constrained optimization approach in [1]. 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 to refine analytic splining methods such as those in [7, 9, 10, 11, 19].

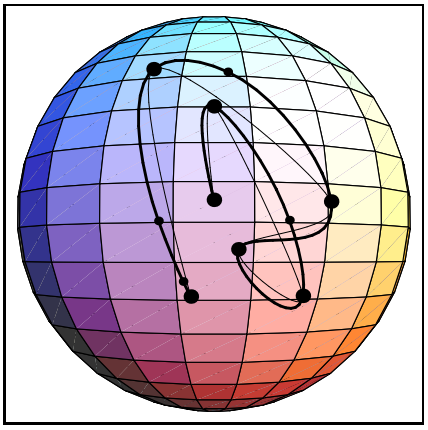

Figure 1: The interpolation problem in quaternion space. The large circles represent keyframes to be interpolated. The small circles show "variable frames" inserted by the system while computing the optimized path (shown with a solid line). A thin line shows the unoptimized path which is seen to have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

Our work is closely related to other areas of research that use optimization, such as spacetime constraints [3], dynamic nurbs [18] and variational surface modeling [20]. 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. In optimizing over a small number of variables for fast algorithms, our work is also similar to that of Liu and Cohen [12].

Although this paper describes the creation of quaternion splines primarily for animation, splining in curved spaces and constructing minimal energy motions is important to other communities such as computer aided geometric design, robotics and kinematics. A few related papers are found in [8, 15]. Our techniques can also be generalized easily to handle objective functions more complex than covariant acceleration for quaternions. Thus, our method can be used in conjunction with conventional animation techniques for rapid automatic improvement of "rough" animations or for sparser keyframing by interpolation[14].

The rest of this paper is organized as follows. In section 2, we

* Address: MS 350-74, Caltech, Pasadena, Ca 91125.

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.

2 The Quaternion Interpolation Problem

As in [1], we want to minimize the net squared magnitude of the covariant (tangential) acceleration in quaternion space, while interpolating keyframe orientations and (if specified) angular velocities at the end frames. The normal component of the acceleration is not penalized since it is necessary for maintaining unit magnitude of the quaternions. This formulation is analogous to minimizing angular acceleration.

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:

$$minimize \int_{t_1}^{t_K} \parallel q'' \setminus q \parallel^2 \, dt \tag{1}$$

subject to the constraints:

$$\begin{aligned} \text{INTERPOLATION:} \quad & \forall i : 1 \le i \le K : q(t_i) = Q^i \\ \text{UNITARINESS:} \quad & \forall t : t_1 \le t \le t_K : q(t) \cdot q(t) = 1 \\ \text{END-POINT ANGULAR VELOCITY:} \quad & \\ & 2q'(t_1)q^{-1}(t_1) = \omega_1 \quad 2q'(t_K)q^{-1}(t_K) = \omega_K \end{aligned} \tag{2}$$

Here, $q'' \setminus q$ is the tangential or covariant acceleration[1] , and $\parallel q'' \setminus q \parallel$ is its norm or magnitude.

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 integrand put explicitly in terms of quaternion components q_j)

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

subject to the interpolation constraints. In the integrand in equation 3, the first two terms are the squared magnitude of the covariant acceleration (an expanded form of the integrand in equation 1). The third term maintains quaternion magnitude, and 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.

[1] As explained in [1], an explicit formula for the i^{th} component of the covariant acceleration is:

$$(q'' \setminus q)_i = q_i'' - q_i \frac{\sum_{j=0}^{3} q_j'' q_j}{\sum_{k=0}^{3} q_k q_k}$$

The squared magnitude of the covariant acceleration, $\parallel q'' \setminus q \parallel^2$, is given by the first two terms of the objective in equation 3.

3 Our Algorithm

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

Algorithm

1. *Input user-specified keyframes*, and number of ''variable frames'' (with optional angular velocities at the end frames).
2. *Guess quaternion velocities* at the interior keyframes and interpolate keyframes and velocities with C^1 continuous cubics.
3. *An optimizer adjusts velocities* at the interior keyframes to minimize the objective functional (equation 3).
4. *"Variable frames" are added* in a small number of ''segments'' having the largest average value for the Euler-Lagrange error functional (after the average Euler-Lagrange error functional in each ''segment'' has been computed).
5. *Readjust variable frame quaternions and velocities* in addition to keyframe velocities to minimize the objective functional 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 (with the time difference between key frames normalized to be of order 1), we tell the optimizer that maintaining unit magnitude is significantly more important than minimizing covariant acceleration. This works well in that maximum deviation from unitariness is frequently less than 1% (with the average deviation about one tenth of that), 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.

Step 2: Assign initial guess for 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:

$$q_j'(t_i) = \frac{q_j(t_{i+1}) - q_j(t_{i-1})}{t_{i+1} - t_{i-1}}$$

With these velocities[2] , suitable basis functions are 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 [10]. 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 functional over keyframe velocities In part 3 of the algorithm, an unconstrained minimization package is used to alter our initial guess of the velocity at interior keyframes to minimize the objective OBJ given in equation 3. The variables over which the optimization is done are the velocities at the keyframe times; the user has not constrained the angular velocity at the interior keyframes. Note that since we have only C^1 continuity, the integrand in equation 3 may be discontinuous, but this will not affect the objective functional OBJ. We use the Sequential Quadratic Programming routine E04UCF in the NAG libraries [13]. 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 $\triangle t$ (segment n ranges from t_n to $t_n + \triangle t$). For each segment n, we compute the deviation (using the optimized path $Q(t)$ from step 3):

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

where $\mid EL \mid$ stands for the length of the Euler-Lagrange vector in equation 7. We then pick the segments having the highest values for $DEV(t_n)$ (and such that t_n does not coincide with a keyframe), and add in a "variable frame" at time t_n. 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 in step 5.

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). For our applications, we have found 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 typically divide each time-interval between keyframes into 4 to 5 segments.

[2] At the end-points, we can only take one sided differences:

$$q_j'(t_1) = \frac{q_j(t_2) - q_j(t_1)}{t_2 - t_1} \qquad q_j'(t_K) = \frac{q_j(t_K) - q_j(t_{K-1})}{t_K - t_{K-1}}$$

Note that we use an effectively continuous representation (with only fine-grain discretization for numerical integration). This is equivalent to saying numerical quadrature is done at several points within a segment. Further, since we generally want to add a small number of variable frames at the right places, it is appropriate to use a few segments in each interval between keyframes. Hence our recommendation above.

Since our method is relatively fast, the user may interactively modify the segment lengths if he wishes, but this should not be necessary in most cases.

Step 5: Optimizing again using variable frames We now repeat the optimization process of step 3 introducing some number M of additional optimization variables in the form of the positions and velocities at the variable frames. Thus, the minimizer now varies interior keyframe velocities as well as variable frame positions and velocities in order to minimize the objective functional. 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 [21] tells us that the optimal curve satisfies the corresponding Euler-Lagrange equations:

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

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:

$$\begin{aligned} a &= \sum_{j=0}^{3} q_j q_j'' \\ b &= \sum_{j=0}^{3} q_j q_j \\ T &= \frac{a}{b} \\ U(i) &= q_i T^2 - q_i'' T \\ V(i) &= q_i'' - q_i T \\ W(i) &= 2\alpha(b-1)q_i \\ \frac{d^2}{dt^2}V(i) &\equiv q_i'''' - (q_i'' T + 2q_i' T' + q_i T'') \end{aligned} \tag{6}$$

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 i^{th} component of 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 may be simpler to differentiate T numerically rather than analytically. For reference, the analytic formulae (assuming cubic basis functions so the fourth derivative vanishes) are given below:

First, we must define:

$$\begin{aligned} d_{ij} &= d_{ji} = \sum_{k=0}^{3} q_k^{(i)} q_k^{(j)} \\ t1 &= \frac{2d_{13} + d_{22}}{d_{00}} - 4\frac{d_{01}(d_{12} + d_{03})}{(d_{00})^2} \\ t2 &= -2\frac{d_{02}(d_{02} + d_{11})}{(d_{00})^2} \\ t3 &= 8\frac{(d_{01})^2 d_{02}}{(d_{00})^3} \end{aligned} \tag{8}$$

where the superscripts on the quaternions stand for the appropriate derivatives. Note that $d_{00} = b$, and $d_{02} = a$. We can now write:

$$\begin{aligned} T' &= \frac{d_{12} + d_{03}}{d_{00}} - 2\frac{d_{01}d_{02}}{(d_{00})^2} \\ T'' &= t1 + t2 + t3 \end{aligned} \tag{9}$$

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

It should be noted that the Euler-Lagrange error functional vanishes as expected in case the path followed is a unit great circle (and thus has no covariant acceleration). For instance, consider the path:

$$q_0(t) = \cos(t) \quad q_1(t) = \sin(t) \quad q_2(t) = q_3(t) = 0$$

It is clear that for this path,

$$q_i'' = -q_i \qquad b = 1 \qquad a = T = -1$$

Using these relations, it can readily be verified that $U(i) = V(i) = W(i) = 0$, and the Euler-Lagrange error functional correctly recognizes that the path is optimal. Since all great circles can be constructed by rotating the example given, the Euler-Lagrange error functional vanishes for great circles as expected.

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 [20] 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.

Here, we have used the Euler-Lagrange error functional only for adaptive refinement. An alternative approach that could be tried in the future is to solve the Euler-Lagrange equations directly or to minimize the Euler-Lagrange error functional instead of the objective function of equation 3. Note that while we have derived analytic formulae for the error functional, numerical approaches based on numerically approximating the left hand side of equation 5 can also be used, and may be necessary if more complicated objective functions than the one in this paper are used.

Advantages of Cubics There are a number of advantages that continuous (in our case cubic) basis functions possess over discrete methods [1, 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.

Importance of Variable Frames Variable frames (where the optimizer can vary the position and velocity) have some advantages over approaches based on spline coefficients [3].

- Since variable frames correspond directly to frames on the actual animation path, they are easier to understand and deal with, especially for a user. Changes in variable frame positions or velocities usually correspond in a simple manner to changes in the actual animation. Large coefficient changes may balance each other, and may not correspond as directly and intuitively to the final animation.
- The use of variable frames ensures that keyframe interpolation is automatic. With coefficient based methods, constraints need to be added to ensure keyframe interpolation. This may require a more complex optimizer and/or a more time-consuming algorithm.

4 Results

We present a representative 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.

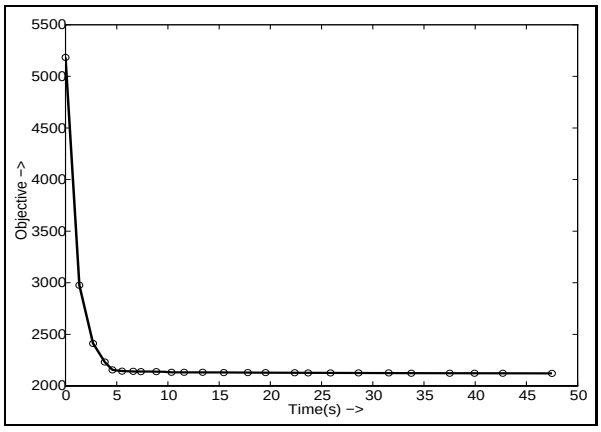

Figure 3: A very small number of variable frames can yield near-optimal results. As described in the text below, each circle represents an increasing number of variable frames or a higher level of optimization.

Performance

We show the decrease in objective (as a function of program execution time) as more variable frames are added in figure 3. Each circle on the graph shows a further level of optimization. The first circle 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.

Unit Magnitude

In figure 4 we show the unit magnitude being maintained. The initial path (blue dotted) has many places where quaternions are far from unit magnitude. Our optimized path—4 variable frames and $\alpha = 1000$—(magenta dashdot) is within 1% of unit magnitude, while the solid red 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).

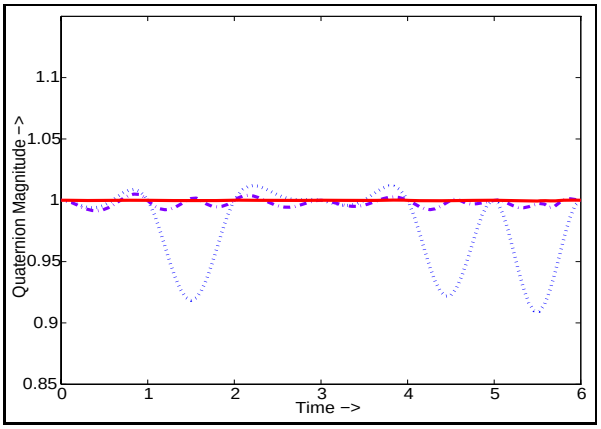

Figure 4: Maintaining unit magnitude. In the original path (blue dotted), there are significant deviations, while increasing levels of optimization (magenta dashdot and red solid line) bring the quaternions very close to unit magnitude.

Euler-Lagrange error functional

The Euler-Lagrange deviation before and after optimization are shown in figure 5. 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 [1] 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 [1] was 8800 seconds, a factor of more than 2000 slower! (of course, the exact timings may vary depending on the specific machine and minimizer used). 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.

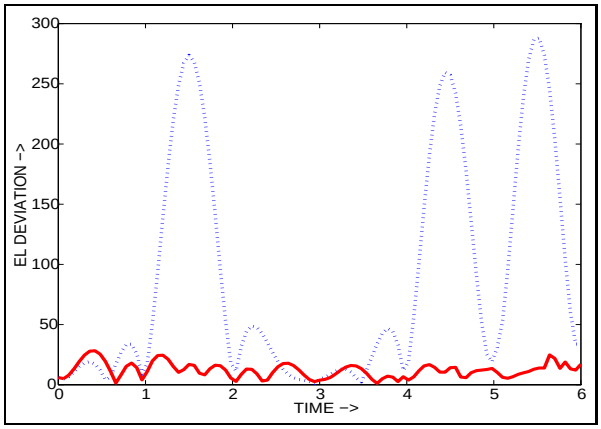

Figure 5: The blue dotted graph shows the initial Euler-Lagrange error. The solid red graph shows how this error is significantly decreased upon optimization.

Animations

The animations[3] accompanying this paper show some examples of rotational paths created using the methods in the paper. Our first example is the rotation of a rigid object similar to the example discussed above (except that there is more rotation which involves all four quaternion components). Figure 6 shows the (I-shaped) object rotating while moving in a parabola. The trunks are black for keyframes (which also have a dark circle behind them that makes their position clear), blue for variable frames and red for the few computed animation frames that are shown.

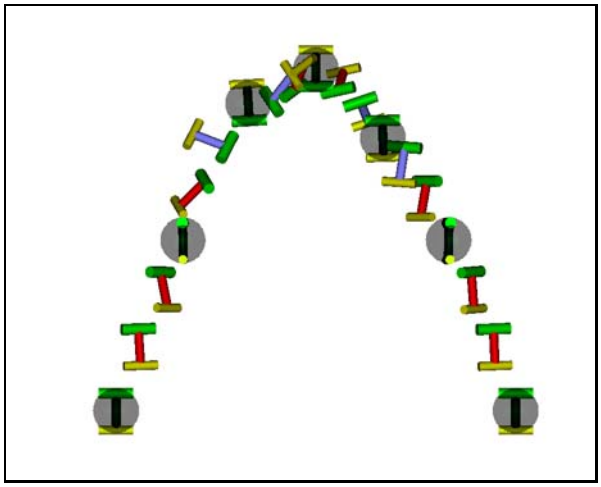

Figure 6: Showing the object path. Keyframes have a dark circle behind them. Variable frames have blue trunks and those for computed animation frames are red. The top and bottom of the object are shown in green and yellow consistently for all frames.

Our second example is derived from a molecular dynamics simulation (with exotic initial conditions). We resampled the rotations sparsely, interpolating to make animations. We show an example using 4 keyframes. A still is shown in figure 7.

[3]The animations are present on the CD-ROM accompanying the proceedings, and can be accessed via our website:
http://www.gg.caltech.edu/Animations/quaternions.html

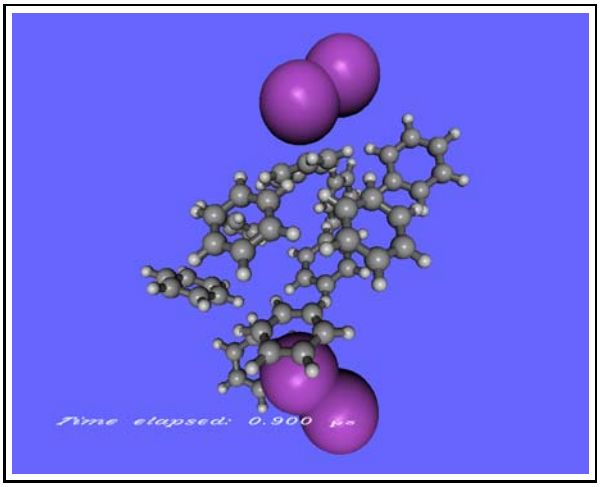

Figure 7: A still from an animation on the CD-ROM/website. The animation is derived from a molecular dynamics simulation of iodine and benzene molecules slowed down by a factor of approximately 10^{12}.

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 the following areas:

- *Deriving theoretical bounds* on the error in the objective using the Euler-Lagrange error functional and comparing Euler-Lagrange based adaptive refinement to other methods. We [14] have derived a physical interpretation for the Euler-Lagrange error functional and some simple bounds, but these are currently too loose for meaningful error estimates.
- *Using more advanced basis functions*. Our current estimate for quaternion velocities (step 2 of the algorithm) can be improved. In addition, we use simple cubics. Using quaternion curves such as the B-spline quaternion curves of [10] for the initial guess of quaternion velocities and the corresponding hermite quaternion curves for interpolation instead of cubics might yield faster results. This would also allow the optimization procedure to be carried out on the quaternion sphere directly [5] instead of using soft constraints.
- *Higher degrees of continuity and better objective functions* This paper has discussed curves with C^1 continuity. A simple way of achieving C^2 continuity is with interpolating B-splines as in [14]. Perceptually (rather than mathematically) based objective functions also need to be investigated.

6 Acknowledgements

We would like to thank Julius Su (from Ahmed Zewail's femtochemistry group at Caltech) for contributing the molecular dynamics simulation from which the video accompanying this paper was derived, Brian D'Urso for suggesting the particular initial conditions used, and the anonymous Siggraph reviewers for comments that were very helpful in preparing the final version of the paper. Thank also go to 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 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] A.H. BARR, B. CURRIN, S. GABRIEL, and J.F. HUGHES. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *SIGGRAPH 92 proceedings*, pages 313–320, 1992.

[2] R. BARTELS, J. BEATTY, and B. BARSKY. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufman, Palo Alto, 1987.

[3] M.F. COHEN. Interactive spacetime control for animation. In *SIGGRAPH 92 proceedings*, pages 293–302, 1992.

[4] S. GABRIEL and J. KAJIYA. Spline interpolation in curved space. In *SIGGRAPH 85 course notes for State of the Art in Image Synthesis(#11)*, 1985.

[5] H.-J. HA. A new camera control method preserving view-up vectors. Master's thesis, KAIST, Taejon 305-701, Korea, 1996.

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

[7] B. JUTTLER. Visualization of moving objects using dual quaternion curves. *Computer & Graphics*, 18(3):315–326, 1994.

[8] B. JUTTLER and M.G. WAGNER. Computer-aided design with spatial rational B-spline motions. *Journal of Mechanical Design*, 118(2):193–201, June 1996.

[9] M.-J. KIM, M.-S. KIM, and S. SHIN. A C^2-continuous B-spline quaternion curve interpolating a given sequence of solid orientations. In *Computer Animation 95 Proceedings*, pages 72–81, 1995.

[10] M.-J. KIM, M.-S. KIM, and S. SHIN. A general construction scheme for unit quaternion curves with simple high order derivatives. In *SIGGRAPH 95 proceedings*, pages 369–376, 1995.

[11] M.-S. KIM and K.-W. NAM. Interpolating solid orientations with circular blending quaternion curves. *Computer Aided Design*, 27(5):385–398, 1995.

[12] Z. LIU and M.F. COHEN. Keyframe motion optimization by relaxing speed and timing. In 6^{th} *Eurographics Workshop on Animation and Simulation (Maastricht 1995)*, pages 144–153, 1995.

[13] Numerical Algorithms Group, Ltd. *NAG Fortran Library Document*, 1988.

[14] R. RAMAMOORTHI, C. BALL, and A.H. BARR. Dynamic splines with constraints for animation. Technical Report CS-TR-97-03, California Institute of Technology, Included on CD-ROM, January 1997. ftp://ftp.cs.caltech.edu/tr/cs-tr-97-03.ps.Z.

[15] B. RAVANI and F. PARK. Bezier curves on riemannian manifolds and lie groups with kinematic applications. *Journal of Mechanical Design*, 117(1):36–40, March 1995.

[16] K. SHOEMAKE. Animating rotation with quaternion curves. In *SIGGRAPH 85 proceedings*, pages 245–254, 1985.

[17] K. SHOEMAKE. Quaternion calculus for animation. In *SIGGRAPH 91 course notes for Math for Siggraph (#2)*, 1991.

[18] D. TERZOPOULOS and H. QIN. Dynamic Nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, April 1994.

[19] W. WANG and B. JOE. Orientation interpolation in quaternion space using spherical biarcs. In *Graphics Interface 93*, pages 24–32, 1993.

[20] W. WELCH and A. WITKIN. Variational surface modeling. In *SIGGRAPH 92 proceedings*, pages 157–166, 1992.

[21] D. ZWILLINGER. *Handbook of Differential Equations*. Academic Press, San Diego, 1989.