Dynamic Splines with Constraints for Animation

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

Abstract

In this paper, we present a method for fast interpolation between animation keyframes that allows for automatic computer-generated "improvement" of the motion. Our technique is closely related to conventional animation techniques, and can be used easily in conjunction with them for fast improvements of "rough" animations or for interpolation to allow sparser keyframing.

We apply our technique to construction of splines in quaternion space where we show 100-fold speed-ups over previous methods. We also discuss our experiences with animation of an articulated human-like figure.

Features of the method include:

- Development of new subdivision techniques based on the Euler-Lagrange differential equations for splines in quaternion space.
- An intuitive and simple set of coefficients to optimize over which is different from the conventional B-spline coefficients.
- Widespread use of unconstrained minimization as opposed to constrained optimization needed by many previous methods. This speeds up the algorithm significantly, while still maintaining keyframe constraints accurately.

1 Introduction

Many investigators have examined the problem of creating animations from user-supplied keyframes. Spline-based techniques as in [BARTELS ET AL 87] have been used with much success.

One of the problems with spline-based animation, however, is that many motion quantities do not ideally fall along the natural spline paths. (In fact, when the undesirable aspects of splines are overly apparent, the motion is sometimes described as being too "spliney.")

There have been a number of approaches for improving motion and shape aesthetics while retaining user control. Examples of these include space-time constraint methods, as in [WITKIN & KASS 88], [COHEN 92], [LIU ET AL 94], inverse dynamics as in [BARZEL & BARR], dynamic nurbs, such as [TERZOPOULOS & QIN 94], and interpolation in non-Euclidean spaces, as in [GABRIEL & KAJIYA 85], [BARR ET AL 92].

These approaches are usually posed as constrained optimization problems, and utilize differential equations, lagrange multipliers and intensive numerical solution techniques.

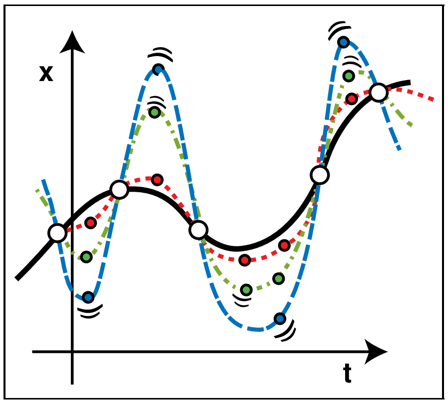

Figure 1. In our technique, the splines are constructed so that the animation paths pass through the keyframes (shown as the open circles along the solid curve) without using constrained optimization methods. "Variable" keyframes, shown as smaller circles, are created by the system to minimize an objective function affecting the behavior between the keyframes. Partially optimized solutions (shown as dashed lines) still pass through the keyframes.

In contrast, we use optimization but do not require constrained optimization. Our method first places a spline path through the user-defined keyframes, and measures the degree of constraint satisfaction within the spline segments. Unsatisfactory segments are automatically subdivided; "variable" points are inserted. An optimizer then improves the degree of constraint-satisfaction by moving the variable points to better positions.

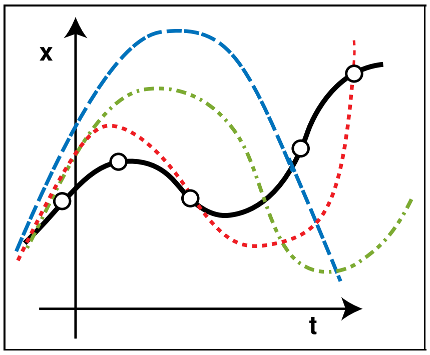

Figure 2. Methods that utilize constrained optimization must converge to pass through the keyframes. Keyframes are shown as open circles along the solid curve; note that the partially converged solutions (shown as dashed lines) do not pass through the keyframes; the optimizer must reach the full solution, shown in solid form, to pass through the keyframes, requiring much more computation.

Advantages of our method

- Splines are used to quickly create the motion path so as to ensure that the animation always goes through the keyframes (figure 1). Thus, constrained optimization (as would be required under the scheme shown in figure 2) is not required to satisfy the hard constraints (that the keyframes be met exactly). There are many advantages of soft constraints — enforced through penalty terms that are added to the objective function used in the optimization (soft constraints need not be exactly satisfied). The primary advantages are speed and simplicity as illustrated in figures 3 and 4.
- The motion is "better" than raw splines, at least as seen by the penalty function f(), and improves on successive iterations. We do not need to wait until full convergence.
- The method calculates function representations of non-Euclidean interpolation paths as in [BARR ET AL 92], but almost 100 times faster.
- Soft auxiliary constraints (which in contrast to hard constraints need not be exactly satisfied but are more in the nature of strong hints to the optimizer) are used to substantially reduce unphysical motion like a foot going through the floor, knees and elbows bending backward etc.
- There is an automatic subdivision scheme which subdivides in regions of high "unphysicality" or high penalty terms. In addition, for splines in quaternion space, Euler-Lagrange differential equations are used to accurately determine where the path of the object is locally non-optimal. This constitutes a new approach to subdivision.

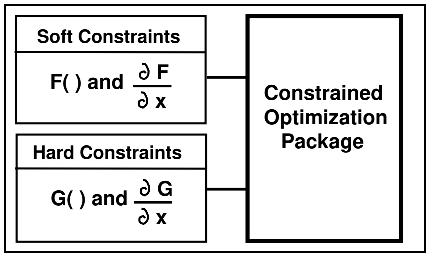

Figure 3. Solution methods for constrained optimization usually require intensive numerical methods. A prototypical constrained optimization problem is **minimize** $F(x)$ **subject to** $G(x) < \epsilon$, and requires many function evaluations and gradient calculations to maintain the hard constraints. Animation techniques that combine physics and constraints usually use constrained optimization to pass through the keyframes.

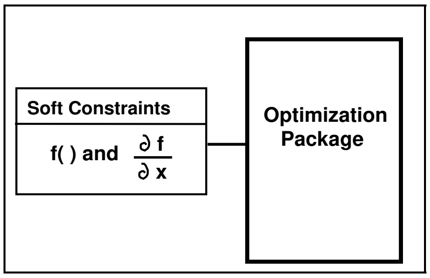

Figure 4. Methods that do not require hard constraints can use simpler numerical optimization methods, such as simple minimization.

While our approach is similar to the spacetime constraints paradigm ([WITKIN & KASS 88], [COHEN 92], [LIU ET AL 94]), in that we seek to minimize an objective function, our approach is more closely allied to traditional keyframing and our primary application is to interpolation of keyframes.

Unlike [LIU & COHEN 95], we fix the keyframes both spatially and temporally as is more common in conventional animation techniques (except that we allow partial keyframes that fix only some degrees of freedom). In addition, we improve on the method presented in [LIU & COHEN 95] in that we optimize the intermediate path between keyframes instead of using a simple spline or hermite approximation. In our approach, specification of generalized co-ordinate velocities is not necessary since our optimization procedure requires only the position of keyframes.

There has been much work in related areas of research. For example, physics-based or optimization techniques have been used for modeling of curves and surfaces ([WELCH & WITKIN 92], [TERZOPOULOS & QIN 94]). We present several new techniques such as error metrics and subdivision methods based on the Euler-Lagrange differential equations, use of optimization over variable intermediate frames instead of B-spline coefficients, and use of unconstrained minimization.

The rest of our paper is organized as follows: In section 2, we describe the algorithm; In section 3, we apply the method to covariant interpolation. In section 4, we compare our algorithm to previous work. In section 5, we illustrate the use of the algorithm applied to animation of a human-like figure. In section 6, we discuss future work.

2 Algorithm Description

The algorithm takes as input K keyframe (or "partial keyframe") vectors X_i, which specify the state of the animation at particular instants of time t_i, $i = 0, 1, \ldots, K-1$, and an objective function (or penalty function) $f(x)$ that affects the behavior of the animation between the keyframes. The task is to create an optimal set of "variable" frames between the keyframes; the variable frames are selected so that the spline curve that passes through all of the keyframes and variable frames minimizes the net integrated value of the penalty constraint $f()$, along the path. Either full keyframe vectors X_i, or "partial keyframes" may be specified. In partial keyframes, only some of the parameters of the animation are specified. The remaining parameters are selected to minimize the penalty or objective function $f()$.

The objective function $f()$ can be thought of as a weighted collection of non-negative penalty terms, $f = \sum_i a_i f_i(x)$, that measures deviations from desired states and behaviors, such as nonunit quaternions, object interpenetration and overflexing of joint angles.

We are looking for the the optimal animation path– that is, we wish to find the C^2 continuous vector function $Y(t)$ such that the integral:

$$E(Y) = \int_{t_0}^{t_{K-1}} \alpha_1 f_1(Y(t)) + \alpha_2 f_2(Y(t)) + \ldots \, dt \quad (1)$$

is minimized subject to the constraints that $Y(t_i) = X_i$ for $i = 0, 1, \ldots K-1$. In the above integral, f_j is an objective function, i.e., a nonnegative function that measures by how much a soft constraint— a condition that the user desires but one that need not be exactly met— is violated. The α_j are positive weighting constants that balance the relative strengths of the soft constraints.

Overview of Algorithm

1. User provides keyframes (or partial key frames), as well as any soft constraints to be satisfied in the animation.
2. The system inserts variable frames between the keyframes. These variable frames (or variable points) are like keyframes except that they are not fixed but are varied during the course of the optimization procedure.
3. From the variable frames and key frames (or fixed frames) compute an interpolating C^2 (cubic B-spline) function.
4. With the function computed above, calculate E(Y) as in integral 1. Then, move the variable points to minimize E.
5. Check to see where the true solution is badly approximated by the current set of splines and subdivide by adding in more variable points in those regions only.

Figure 5. Overview of the algorithm. A partial key frame is similar to a traditional key frame except that some or all of the state of the animation at a given time may be supplied. A suitable metric is used in step 5. In the case of splines in quaternion space, we use the Euler-Lagrange differential equations based on covariant quaternion acceleration.

In general, the N components of Y(t) can be given as

$$Y_i(t) = \sum_k C_{ik}\phi_k(t) \quad (2)$$

with i ranging from 0 to $N-1$ — with $N = 3$ if Y is a vector and $N = 4$ if Y is a quaternion — where the $\phi(t)$ are spline basis functions, such as the B-spline or nonuniform rational B-spline bases (see [FARIN 1992]). The C_{ik} are the basis function coefficients.

Variable points In general, we can use equation 2 to write (where Y is a vector representing the state of the animation)

$$Y_i(t_j) = \sum_k C_{ik}\phi_k(t_j) \quad (3)$$

since equation 2 holds for all t. Equation 3 holds for all keyframe times t_j. If we want to interpolate a set of M points with times $t_1, t_2, \ldots t_M$, we must have equation 3 satisfied at all t_j at which the interpolated points occur. But, if we know the values of the vectors Y at the interpolated points, we can write:

$$Y_{im} = \sum_k C_{ik}\phi_k(t_m) \tag{4}$$

for all m from 1 to M. This is a series of M simultaneous equations. We can write it in matrix form as:

$$[Y_i] = [\phi][C_i] \tag{5}$$

In the above equation, $[Y_i]$ is a column matrix with $[Y_i]_m = Y_{im}$. $[\phi]$ is an $M \times M$ matrix with $[\phi]_{mn} = \phi_n(t_m)$. $[C_i]$ is a column matrix with $[C_i]_n = C_{in}$. If the matrix $[\phi]$ is invertible, we can write

$$[C_i] = [\phi]^{-1}[Y_i] \tag{6}$$

With the help of this equation we can calculate the coefficients in matrix $[C_i]$ knowing the points to be interpolated, that is the matrix $[Y_i]$. The matrix $[\phi]$ is inverted only once at the start of the optimization procedure, and the time taken by the calculation in equation 6 is negligible in comparison to the time taken for computation of the integral to be minimized.

As stated in figure 5, the user supplies keyframes (or even incomplete, partial keyframes). The system puts in variable frames and sends these to the optimizer. Then, based on the system state at the variable frames and the already known state (hard constraint) at the fixed frames, the matrix $[Y_i]$ is created from which the matrix $[C_i]$ can be calculated.

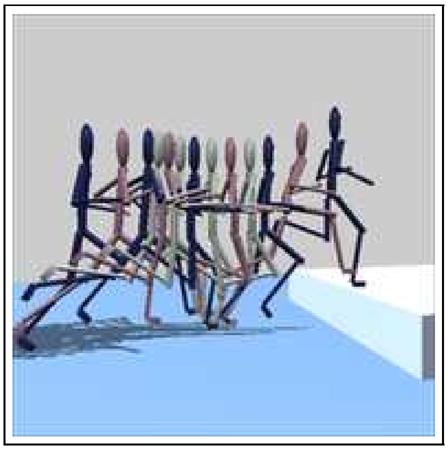

Figure 6. Illustrating the algorithm with reference to animation of a running human-like figure. The human figures are drawn thin so their path can be more apparent. The initial keyframes are drawn darkest. The variable frames are less dark; the "in-between" frames are drawn in the lightest color.

Addition of variable points We add variable points in the region between two frames (fixed or variable) where the motion is unphysical as described by our subdivision criteria. We add the variable points at the midpoint of the region, and also add a knot there in the knotvector for our B-spline basis.

Advantages

- The variables being optimized are now actual animation states, so feedback from the system can immediately tell a user about the state of the animation.
- The functions $Y_i(t)$ always interpolate the fixed frames. Thus, the hard constraints of the animation having a particular state at the key frames as per the animator's wishes is always satisfied.

The algorithm is illustrated in Figure 6, where the key frames, variable frames (or variable points) and intermediate frames (given by the spline that interpolates all the variable and key frames) are clearly shown.

Minimization of the objective As described in equation 1, our functional consists of the weighted sum of several objective functions and soft-constraint deviations. We use sequential quadratic programming (SQP) to position the variable frames so as to minimize the value of the integral in equation 1 (as described in [NAG]). One can either use equation 1 directly, or renormalize the various parts of the integrand based on the initial values of the soft constraints as follows:

$$I = \int_{t_0}^{t_{K-1}} \alpha_1 \frac{f_1(Y(t))}{f_1(Y(t))_{initial}} + \alpha_2 \frac{f_2(Y(t))}{f_2(Y(t))_{in}} + \ldots \, dt \tag{7}$$

The subscript stands for the initial state before any optimization has taken place (the original spline path). We integrated numerically by summing the values of the integrand at a discrete number of uniformly distributed points and multiplying by the time interval represented by each point (the total time range divided by the number of points). In keeping with the approximate nature of soft constraints, we did not feel that more sophisticated integration methods were necessary.

This renormalized representation depends less on the scale of the various objectives and soft constraints and is thus better suited for some applications.

For instance, we may measure deviation of quaternions from unit magnitude by the absolute magnitude of the deviation or by the square of the deviation.

While the scales of these two deviations will in general differ largely, our representation makes it easier to attach the same relative weight (in comparison to other parts of the objective) to ensure that the quaternions remain nearly unitary.

Note that in our formulation the integral I decreases at each major iteration of the SQP solver. Also, we ensure that the hard constraints where provided at the partial key frames are exactly met. Thus, at each iteration we have a better result than we did before. As soon as the optimization process is started, we get improved results, and we do not need to wait for convergence.

Subdivision

Even though convergence may have been reached, the superposition of basis functions might not come sufficiently close to the actual optimal path. In regions where this is the case, we add a knot to the knot vector (and a corresponding variable point) for B-splines to allow that region to be evaluated in further detail. Previous techniques for determining if the superposition of basis functions adequately represents the true solution include work by [Welch & Witkin 1992] who compare versions of a solution computed at two different resolutions. Another test commonly used is the magnitude of the objective in a particular region. However, these approaches are not always theoretically sound. For instance, in a particular region, the objective might be large but that may be the best that can be done. For the case of splines in quaternion space (discussed in more detail in the section on numerical results), we present a different approach based on the Euler-Lagrange differential equations, that gives a more direct indication of the difference between the current path and the optimal one.

Euler-Lagrange equations. For an optimization problem stated in terms of extremizing an integral as in equation 1, it is possible to derive an alternative formulation in terms of differential equations. These differential equations are known as the Euler-Lagrange equations. See [Zwillinger 89], for instance, for details. Appendix B gives the terms of the canonical Euler-Lagrange equation.

After minimization is completed, the velocity at each fixed or variable point is known. Thus, we now have enough data to apply the Euler-Lagrange formulation to each segment. If the path were locally optimal, we would have $EL = 0$ everywhere where EL is the left-hand side of the Euler-Lagrange differential equation. By measuring the deviation of the left hand side from 0, we obtain a reliable estimate of the deviation of the computed path from the optimal path at each point (or region by integrating the deviation over the region).

We subdivided in regions where

$$EL > \alpha$$

where α is a constant. The magnitude of the left hand side of the Euler-Lagrange equation is a plausible estimate for the "unphysicality" if the change in the displacement from the optimal path with time is small compared with the change in the deviation of the Euler-Lagrange equations as discussed in Appendix C.

Since satisfactory convergence has been achieved (a near-minimal path has been computed given the limited number of basis functions), this gives us an accurate indication of whether the current set of basis functions suffices. In regions where there is a large violation of the Euler-Lagrange norm, we add variable points, assigning the values of variables there initially as what they would have been as per our original low-resolution solution.

A physical interpretation for the Euler-Lagrange equations is given as follows (refer to Appendix C for notation):

$$\frac{\triangle E/(t_1 - t_0)}{\epsilon_{opt}} \leq EL_{max} \tag{8}$$

This states that the net time averaged violation (beyond what is required) in the penalty or objective function per unit displacement (from the optimal path) is less than or equal to the maximal magnitude of the left-hand side of the Euler-Lagrange equation. We can also apply this equation locally to claim that the time averaged violation per unit displacement at a point is less than or equal to the magnitude of the left hand side of the Euler-Lagrange equation evaluated at that point. A derivation of this result can be found in Appendix C.

Advantages and Disadvantages of Euler-Lagrange equations A disadvantage of the Euler-Lagrange equations is that one usually needs to specify the velocities at the key frames. These velocities are required as boundary conditions for the differential equations. Also, a piecewise solution to the Euler-Lagrange equations with velocity specified would give us only C^1 continuity, not the C^2 continuity that cubic splines guarantee us. However, an important advantage of the Euler Lagrange equations is that they are stated as an explicit equality of the left hand side to 0 rather than a minimization.

3 Numerical Results: Algorithm applied to Covariant interpolation of quaternions

One application of the method is minimizing the covariant acceleration of the quaternion of a single body. This is the same problem solved in [Barr et al 1992] except that we optimize over quaternion values at variable points and do not need to create discretized samples. In this case, we specified that the condition that the quaternions be of unit magnitude be a soft constraint.

We present the results in Figures 7-10. There were seven keyframes in this run. Initially, we had one variable point between any two key frames. The algorithm converged to a minimum in a few seconds on a single HP 700. This compares favorably to the time of 4 minutes reported by [Barr et al 1992] on an identical architecture. The deviations of the quaternions from unit magnitude was less than 0.2% at all times.

Analogous to [barr et al 92], there are two soft constraints: the quaternion is kept close to unit magnitude, and the covariant acceleration magnitude is minimized.

$$f() = \beta f_1() + f_2() \qquad (9)$$

where $f_1() = (1 - q \cdot q)^2$ and $f_2() = (q'' \setminus q)^2$

Here q represents the quaternion which — as a function of time — describes the path of the rotating object. $f_2()$ refers to the covariant component of the quaternion acceleration. This is the quaternion acceleration with any radial component removed. β is a scaling factor which affects the relative strengths of the two soft constraints. Suitable results can also be obtained by using $f_1() = \mid 1 - q \cdot q \mid$.

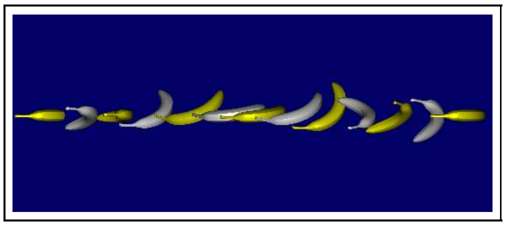

Figure 7: Showing the path generated by covariant interpolation of quaternions using our technique. For clarity, the object is translated from left to right. The yellow bananas represent key frames while the white ones represent the variable frames in between.

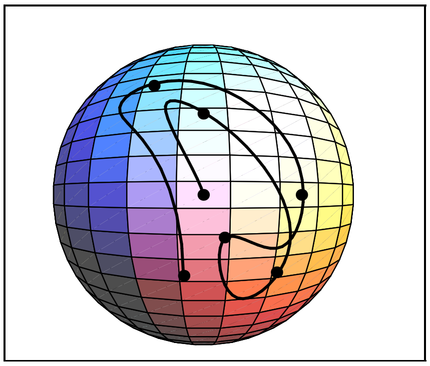

Figure 8. Covariant interpolation of quaternions using this method is approximately 100 times faster than previous methods. Here we covariantly interpolate through seven keyframe quaternions, at most taking a few seconds on an Hp700 workstation.

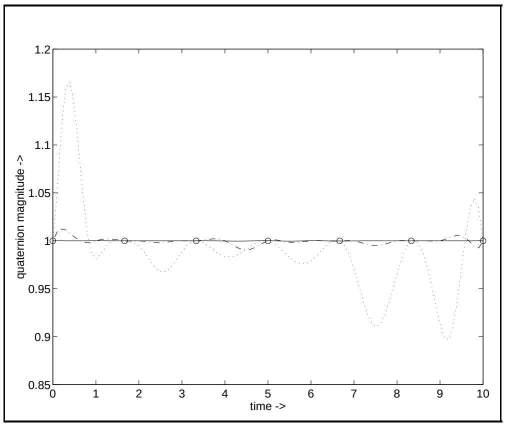

Figure 9: The unit magnitude of the quaternions is maintained by the method. Dashed lines are incompletely optimized results; the solid line is the optimized result, and is seen to be quite close to 1.0.

NOTES These results were obtained after complete convergence. Further, the small constraint violation at the end is present even in the algorithm of [barr et al 1992] but in their case, it comes from the post-minimization interpolation between discrete points. The sources of our speed up are:

- Use of splines instead of a discrete basis allows for fewer variable points while allowing a large number of quadrature points for numerical integration. This compares favorably with the discrete method employed by [barr et al 1992].
- The use of soft constraints has speed advantages referred to earlier. It is clear from the results that the soft constraint has been adequate in maintaining the unitary magnitude of the quaternions to a small tolerance. Note that in this example the ratio (using the notation of equation 7) of α_1(unit magnitude) to

α_2(covariant acceleration) was 2 : 5. While [BARR ET AL 1992] used hard constraints at a finite number of points (they required the quaternions to be unitary at these points), subsequent interpolation between these points to create a continuous representation would violate the unitariness of quaternions.

Note that our use of variable frames is not a source of speed-up in this case since [BARR ET AL 1992] essentially used variable frames with a "box" basis.

SUBDIVISION As discussed in the section on Euler-Lagrange based subdivison, we use criteria for adaptive refinement of the solution based on the Euler-Lagrange differential equations. For the case of covariant acceleration of quaternions as defined in equation 9, these are as given below (a sketch of the derivation is presented in Appendix B):

Define:

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

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

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

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

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

$$Z(i) = 2\beta(b-1)q_i$$

The Euler-Lagrange equation is then:

$$X(i) + \frac{d^2}{dt^2}Y(i) + Z(i) = 0 \qquad (10)$$

Here , we have:

$$\frac{d^2}{dt^2}Y(i) = q_i'''' - (q_i''T + 2q_i'T' + q_iT'') \qquad (11)$$

In the notation of the canonical equation 24, $X(i)$ corresponds to the term $\partial F/\partial y$ for covariant acceleration. $Z(i)$ is the corresponding term for maintenance of unitary quaternions. $Y(i)$ corresponds to the term $\partial F/\partial y''$ in equation 24. The term $\partial F/\partial y' = 0$ since the objective function does not depend directly on q'.

Analytical expressions for T' and T'' can be derived but these are complicated. It is simpler to differentiate T numerically. We have found that this yields satisfactory results.

Note that by varying the subscript i from 0 to 3 in equation 10, four independent equations are generated that must separately be satisfied. Figure 10 shows the results of our tests.

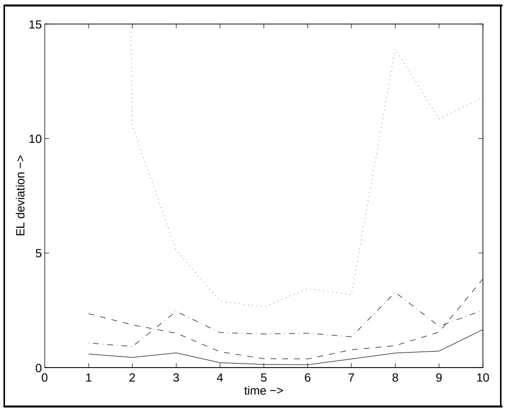

Figure 10: Showing the average Euler-Lagrange deviation as a function of time (we plot the integrated deviation over equal tenths of the path). The dotted line is the deviation of the initial spline path. The dashed line represents the result using one variable point between each fixed point while the solid line represents the result upon using 3 variable points between each fixed point. The dashdot line is a result obtained by adaptively adding variable points at the ends of the path (where the Euler-Lagrange deviation is high). This is a more effective and efficient technique for solution (a total of only 2 variable points was used). It is seen that the result is nearly identical to the result for global subdivision with one variable point between each fixed point (dashed line) at the ends and only slightly worse in the intermediate region. This example clearly shows that the optimizer reduces the covariant acceleration in mainly those regions where there is a large need for improvement. Further, the Euler-Lagrange equations provide a good metric for adaptive refinement (or adaptive subdivision) of the solution which can be much more rapid and effective than addition of variable points globally.

4 COMPARISON TO PREVIOUS METHODS

USE OF SPLINES The use of splines for optimization was introduced by [COHEN 92] where optimization was done over spline coefficients. This corrects the problems with discrete methods like [WITKIN 88] and [BARR ET AL 92]. In those approaches, interpolation to create a continuous representation must be done *after* minimization of the objective and the fact that interpolation itself alters the objective is not taken into account. Thus, a much larger number of variable points is needed to get the same accuracy as a continuous approach. This is one of the reasons for the speed up of our method when applied to creation of quaternion spline paths. More sophisticated basis functions like wavelets [Liu et al 94] have also been proposed.

VARIABLE POINTS While our use of splines is not completely new, we do not optimize over spline coefficients as in [Cohen 92], but over the value of variables describing the state of the animation at intermediate or variable points. Our method has the following advantages:

- For the end user, the value of coefficients is not always an intuitive way of thinking about the optimization. Also, large coefficient changes do not necessarily correspond in a very direct way with changes in the actual path. By contrast, we use variables more intuitive under several circumstances — actual animation states. Further, changes in their values correspond to clear changes in the animation path.
- Hard equality constraints in the minimizer are not needed in our approach to enforce the desired values at key frames. In [cohen 92], the key frame constraints are treated as similar to other constraints with the same complicated numerical methods used to enforce them.

USE OF UNCONSTRAINED MINIMIZATION We have shown that constrained optimization methods are not necessary. This has been done by the following devices:

1. We have used variable points instead of B-spline coefficients. Thus, no constraints are needed to ensure that the objects pass correctly and at all times through the key frames.
2. By including soft-constraints in our objective, we have shown that constrained optimization is not always needed.

We believe our methods to show certain improvements over previous methods. For instance, [Welch & Witkin 92] used Lagrange multipliers or penalty functions to enforce linear constraints. Our keyframe constraints are indeed linear in the coefficients of the spline basis. However, the method in [Welch & Witkin 92] either *increases* the number of variables as more variables must be used for Lagrange multipliers or allows for (possibly large) constraint violations through the use of a penalty function. By contrast, our approach uses *fewer* variables *and* ensures that the constraints are always satisfied exactly (within double precision error).

There is a further important disadvantage of constrained methods like [Welch & Witkin 92] or [Cohen 92]. Constraints may be violated by large amounts in the search for a minimum even if the final constraint violation is small or zero on convergence. Since we want the user to interact with the solution process, an approach where the constraints are always met even without convergence is attractive. Our method forces the keyframe constraints to be met throughout the optimization process.

USE OF SOFT CONSTRAINTS We have used soft rather than hard constraints (which must be met on convergence) for the following reasons:

1. In [Welch & Witkin 92], there is a discussion of how a finite basis cannot satisfy the constraints completely. The authors of that paper used a least squares fit. However, their functionals were of a quadratic form, while we seek to minimize highly nonlinear nonquadratic objectives. Thus, we cannot use a least-squares approximation. Since the constraints cannot be satisfied completely, use of hard constraints is inappropriate. To clarify this point, consider the case of maintaining quaternion magnitude. Because the number of basis functions is finite, the basis is incomplete: quaternion magnitude *cannot* be maintained over the entire animation path and we cannot have a hard constraint forcing the unitariness of quaternions to be exactly satisfied everywhere. We could have the unitary condition exactly satisfied at a finite number of points, but this would not guarantee unitary quaternions over the entire path as we desire.
2. Use of soft constraints speeds up the solution process considerably as compared to the length of time it would take if hard constraints were used. Calculation of constraint violation and constraint gradients are not required. This can increase the speed tremendously.
3. Simpler minimization packages may be used.

A few points deserve note. Even with a complicated minimization algorithm such as sequential quadratic programming that can handle constraints, the absence of constraint evaluations (and constraint gradients) makes the speed of using soft constraints very attractive. Note that it is also possible to introduce an inequality constraint thereby requiring the magnitude of constraint violation to be less than some small value. However, most minimization packages cannot deal with this. Even a more complete package such as the E04UCF routine of the NAG libraries is extremely slow when dealing with inequality constraints as compared to even hard constraints.

SUBDIVISION SCHEMES We believe our subdivision approach based on the Euler-Lagrange equations to be new and superior to previous subdivision schemes discussed earlier.

5 Foot placement for a Running Articulated System

In this section, we apply the method to the animation of a person-like running figure. We treat the person as an articulated body, and choose to represent the rotation of each body by a quaternion that gives its rotation in world co-ordinates. These rotations along with the translation of the root body are sufficient to completely specify the state of the body. (See Appendix A for representing articulated bodies with quaternions and computing forces and torques.) This example is intended to demonstrate the use of our technique for animation of complicated objects. The running figure has thirty nine degrees of freedom and is thus a high-dimensional system. Unlike many previous optimization methods, our approach is fast enough to handle this complicated system. Our tests took no longer than a few minutes at most.

Improvement of Motion Several simple improvements of the original spline motion can be made. We describe one such example where our technique helps fine-tune the animation to prevent penetration of the runner's leg with the floor and a step. Note that here again, for the sake of speed, we use soft constraints to enforce non interpenetration. The advantages of soft constraints have been dealt with earlier. In this example they provide a means of making fast automatic corrections to a specific problem in the original animation.

Penetration of the floor by the foot We calculate the position of the heel and toe on both feet. Refer to Appendix A for details. Then, we define for each of the four points considered, the left heel and toe as well as the right heel and toe, the constraint violation to be equal to the distance from the point to the floor if the point is below the floor and 0 otherwise. Computation of distance from a given point to the floor is an elementary co-ordinate geometry problem. If the floor lies along one of the co-ordinate planes, this computation is especially simple. For instance, for the floor at $y = 0$ and the direction of $y > 0$ being above the floor, we have:

$$Constraint = \sum_{y=1}^{4} -footpos_y \ if \ footpos_y < 0 \quad (12)$$

where the subscript refers to the four points being considered. We implement other auxiliary constraints as soft constraints such as:

- Quaternions should remain unitary.
- Elbows and Knees should not bend backward.

In our tests it is clear that ordinary spline interpolation is unable to handle the floor penetration and overbending constraints; our technique copes well with these.

Figure 11. We show keyframes and a sample frame from a human figure running and jumping onto a step. This is the original spline and a foot passes through the step.

Figure 12. Optimizing the animation from the same keyframes, the foot does not pass through the step. The soft constraints have lifted the foot out of the way.

6 Conclusions and Future Work

In this paper we have presented a technique to smoothly interpolate key frames in animation, subject to covariant acceleration constraints and to force and torque constraints. The algorithm can provide improvements to ordinary spline-based interpolation while leaving control in the animator's hands. We have presented a general method for producing more physically realistic motion from the spline paths.

When the objective functions are well known such as in covariant interpolation of quaternions or maintenance of non-penetration constraints, the method works well, with few significant problems.

For complex behaviors, such as human locomotion, ideal objective functions are not easily derived. In that case, there are difficulties in the algorithm, the resolution of which is future work. Unruly spline behavior can cause skidding, floating, and undesirable backwards motion; the optimizer has difficulty eliminating the objectionable behavior completely. It is also possible to try to make articulated body motion more physical by requiring the net torque on each body to be the sum of torques exerted on it at each of its joints. Further, we require equal and opposite torques exerted on the objects held together at a joint. We have observed however, that simple ideas like this are insufficient to significantly improve the realism of the motion. Considerable further research is required to find appropriate objective functions for articulated figure animation.

7 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.

Bibliography

[BARR ET AL 92] A. H. Barr, B. Currin, S. Gabriel and J. F. Hughes, "Smooth interpolation of orientations with angular velocity constraints using quaternions", Computer Graphics SIGGRAPH '92 Proceedings

[BARZEL & BARR 88] "A Modeling System based on Dynamic Constraints," Siggraph 1988.

[BARTELS ET AL 87] R. Bartels, J. Beatty, and B. Barsky. An Introduction to Splines for use in Computer Graphics and Geometric Modeling. Morgan Kaufmann, Los Angeles, 1987.

[COHEN 92] Michael F. Cohen, "Interactive spacetime control for animation," Computer Graphics (SIGGRAPH '92 Proceedings)

[FARIN 92] . G. Farin, From Conics to Nurbs, IEEE CG&A, 1992.

[GABRIEL & KAJIYA 85] S. Gabriel and J. Kajiya, "Spline interpolation in curved space." In state of the art in Image Synthesis, Siggraph 1985.

[GOLDSTEIN 1950] H. Goldstein, Classical Mechanics, Addison Wesley, 1950.

[HAMILTON 1853] W.R. Hamilton. Lectures on Quaternions. Hodges and Smith, Dublin, 1853.

[LIU ET AL 94] Zicheng Liu, Steven J. Gortler, Michael F. Cohen, Optimized Computer-Generated Motions for Animation. Hierarchical Spacetime Control Siggraph Proceedings '94

[NAG 92] Numerical Algorithms Group Ltd. NAG fortran library document, 1988.]

[SHOEMAKE 85] K.Shoemake. "Animating rotation with quaternion curves" Computer Graphics 1985.

[TERZOPOULOS & QIN 94] D. Terzopoulos, H. Qin, "Dynamics Nurbs with Geometric Constraints for Interactive Sculpting" Transactions on Graphics v13 1994 Apr.

[WITKIN & KASS 88] Andrew Witkin and Michael Kass, "Spacetime Constraints", Computer Graphics (SIGGRAPH '88 Proceedings)

[WELCH & WITKIN 92] William Welch and Andrew Witkin, "Variational Surface Modeling", Computer Graphics (SIGGRAPH '92 Proceedings)

[ZWILLINGER 89] D. Zwillinger. Handbook of Differential Equations. Academic Press, San Diego, 1989.

[LIU & COHEN 95] Zicheng Liu and Michael F. Cohen. "Keyframe Motion Optimization By Relaxing Speed and Timing", 6th Eurographics Workshop on Animation and Simulation.

Appendix A: Modeling of articulated bodies.

In the articulated body framework considered by us, we compute the positions and velocities of points on the bodies thus:

Here, X^i_{cm} refers to the position of the center of mass of body i. X^i_a refers to the position of an arbitrary point a on body i where the vector a is measured in body co-ordinates. The state of the body is represented by the translation T of the center of mass of the root body and a set of quaternions Q_i representing the orientations of the bodies. Further define for any object, b to be the vector from the center of mass of its parent to the point of attachment, and c_u to be the vector from the center of mass of the body to descendant u. Let d be the vector from the point of attachment of an object to its parent to the center of mass of the object.

We then have the following scheme of equations:

$$\begin{aligned} X_{cm}^{root} &= T \\ X_{cm}^{i} &= X_{cm}^{parent_i} + Q_{parent_i} b {Q_{parent_i}}^{-1} + Q_i d Q_i^{-1} \\ X_a^i &= X_{cm}^i + Q_i a Q_i^{-1} \end{aligned} \tag{13}$$

Since the articulated body has no cycles, the above equations are non-recursive and suffice to supply the position of any point on any body. They can also be analytically differentiated to obtain velocities and accelerations if required. The equations for the position of the center of mass can be differentiated to yield the acceleration of the center of mass. By multiplying by object mass, we can compute the net force on an object.

A note on computation of derivatives. Quaternion products are differentiated similarly to ordinary function products. However, a little care must be taken when differentiating q^{-1}. Note that the formulae below are summed over repeated indices.

Let $q^{\dagger}$ be the quaternion formed from negating the vector components of q. Then, we have:

$$q^{-1} = \frac{q^{\dagger}}{q_i q_i} \tag{14}$$

The correct formulae for first and second derivatives of the inverse are: [Note that ${q^{(n)}}^{\dagger} = {q^{\dagger}}^{(n)}$]

$$q^{(-1)'} = \frac{{q'}^{\dagger}}{q_i q_i} - 2q^{\dagger}\frac{q_j {q_j}'}{(q_i q_i)^2} \tag{15}$$

$$q^{(-1)''} = \frac{{q''}^{\dagger}}{q_i q_i} - 4{q'}^{\dagger}\frac{q_j q_j'}{(q_i q_i)^2} + Cq^{\dagger} \tag{16}$$

where C is given by:

$$C = -2\left(\frac{q_j {q_j}'' + {q_k}' {q_k}'}{(q_i q_i)^2}\right) + 8\left(\frac{(q_j {q_j}')^2}{(q_i q_i)^3}\right) \tag{17}$$

Computation of rotational Inertia, force and Torque Let I be the inertia tensor of a body, ω be the angular velocity, Q be the quaternion rotation, L be the angular momentum, and R be the rotation matrix. Both R and Q are in world co-ordinates. We have:

$$\begin{aligned} I &= R I_{body} R^{\top} \\ I' &= R' I_{body} R^{\top} + R I_{body} {R^{\top}}' \\ &= \omega^* R I_{body} R^{\top} + R I_{body} R^{\top} {\omega^*}^{\top} \\ &= \omega^* I + I {\omega^*}^{\top} \end{aligned} \tag{18}$$

Here, ω^* is the dual matrix of the vector ω. We further have

$$\omega = 2Q'Q^{-1} \tag{19}$$

$$\omega' = 2(Q''Q^{-1} + Q'(Q^{-1})') \tag{20}$$

Finally, we can write:

$$\begin{aligned} Torque &= L' \\ &= (I\omega)' \\ &= I'\omega + I\omega' \end{aligned} \tag{21}$$

See [GOLDSTEIN 50] for further details.

Appendix B: Euler-Lagrange equations

A general extremization problem (generally minimization) can be written as (where $y(t)$ describes the path)

$$min \int_{t_0}^{t_1} F(y, y', \ldots, y^{(n)})\, dt \tag{22}$$

The natural boundary conditions are:

$$\begin{aligned} y(t_0) = X_0, y'(t_0) = X_1, \ldots, y^{(n-1)}(t_0) = X_{n-1} \\ y(t_1) = x_0, y'(t_1) = x_1, \ldots, y^{(n-1)}(t_1) = x_{n-1} \end{aligned} \tag{23}$$

The corresponding Euler-Lagrange equation is given by:

$$\frac{\partial F}{\partial y} - \frac{d}{dt}\frac{\partial F}{\partial y'} \ldots + (-1)^n \frac{d^n}{dt^n}\frac{\partial F}{\partial y^{(n)}} = 0 \tag{24}$$

For instance, minimization of $\int_{t_0}^{t_1} (x'')^2\, dt$ subject to known velocities and positions for $x(t)$ at t_0 and t_1 gives $x'''' = 0$. For acceleration in flat space, we want to minimize $\int_{t_0}^{t_1} [(x'')^2 + (y'')^2 + (z'')^2]\, dt$. This gives three independent equations for $x(t)$, $y(t)$, $z(t)$. Thus, we have: $x'''' = y'''' = z'''' = 0$. This is why cubic splines are very suitable for interpolation in flat space since these equations are always satisfied. However, the equations for quaternion space are more complicated — no analytic solution has been found — and cubic splines are no longer sufficient.

To derive the Euler Lagrange equation for splines in quaternion space, we use (sum over repeated indices):

$$min \int_{t_0}^{t_1} \mid q_i'' - q_i \frac{q_j'' q_j}{q_l q_l} \mid^2 + \beta \mid q_n q_n - 1 \mid^2 \, dt$$
$$= min \int_{t_0}^{t_1} q_i'' q_i'' - \frac{(q_j'' q_j)^2}{q_l q_l} + \beta (q_n q_n - 1)^2 \, dt \quad (25)$$

To derive the corresponding four independent Euler-Lagrange equations, we simply use equation 24 for each of the quaternion components. The Euler-Lagrange equations are given as equation 10. A brief sketch of the derivation is presented below:

We have:

$$F_1 = q_i'' q_i'' - \frac{(q_j'' q_j)^2}{q_l q_l}$$
$$F_2 = \beta (q_n q_n - 1)^2$$
$$F = F_1 + F_2$$

As per the canonical equation 24, we calculate:

$$\frac{\partial F_1}{\partial q_i} = 2(q_i [\frac{q_j'' q_j}{q_l q_l}]^2 - q_i'' \frac{q_j'' q_j}{q_l q_l})$$
$$\frac{\partial F_1}{\partial q_i'} = 0$$
$$\frac{\partial F_1}{\partial q_i''} = 2(q_i'' - q_i [\frac{q_j'' q_j}{q_l q_l}])$$
$$\frac{\partial F_2}{\partial q_i} = 4 \beta q_i (q_n q_n - 1)$$

Now, plugging into equation 24, and simplifying, we derive equation 10. The terms $X(i)$, $Y(i)$, $Z(i)$ of equation 10 correspond in order to the nonzero terms of the above derivation.

Appendix C: Physical Interpretation of Euler-Lagrange equations

This section discusses the physical interpretation of the Euler-Lagrange equations. A summary of the key results is given in the section on subdivision in the main text.

We are seeking a relation between the actual minimal solution of the Euler-Lagrange equations $EL(Y) = 0$, and the approximate optimized solution obtained by minimization with an (incomplete) set of spline basis functions.

Let Y_{soln} refer to the actual optimal solution of the Euler-Lagrange equations $EL(Y) = 0$. Let another arbitrary path be represented as:

$$Y(\epsilon, \phi(t), t) = Y_{soln}(t) + \epsilon \phi(t) \quad (26)$$

where $\phi(t)$ has norm ($\int_{t_0}^{t_1} \mid phi(t) \mid$) equal to unity and (positive) ϵ measures the magnitude of the deviation from Y_{soln}. $Y_{soln}(t)$ can be written in this notation as $Y(0, \alpha, t)$ where α is arbitrary.

Let a path (subscripted with "opt" since in our applications, this will be an optimized path) Y_{opt} be close to Y_{soln}. We assume that Y_{opt} is close enough to Y_{soln} so that the partial derivative of the integrated objective function $E(Y(\epsilon, \phi(t), t))$ with respect to ϵ is 0 for the optimal path Y_{soln} and increases as we move away from Y_{soln} toward Y_{opt}. Note that E is now a function of ϵ and $\phi(t)$. We assume that the variation of E with ϵ is quadratic near Y_{soln}. This is because Y_{soln} is assumed to be a local minimum and Y_{opt} is assumed to be sufficiently close to Y_{soln} so that for $0 < \epsilon \leq \epsilon_{opt}$, $d^2E/d\epsilon^2 > 0$. Here, ϵ_{opt} refers to the value of ϵ corresponding to the specific path Y_{opt}. Similarly, $\phi_{opt}(t)$ refers to the value of ϕ corresponding to the specific path $Y_{opt}(t)$

Thus, we have (where $EL(Y)$ stands for the left hand side of the Euler-Lagrange equation — equation 24) :

$$(\frac{\partial E}{\partial \epsilon})_{\epsilon = \epsilon_{opt}, \phi = \phi_{opt}} = \int_{t_0}^{t_1} EL(Y_{opt}) \phi_{opt}(t) \, dt \quad (27)$$

$$\triangle E = \int_0^{\epsilon_{opt}} (\frac{\partial E}{\partial \epsilon})_{\phi = \phi_{opt}} \, d\epsilon \quad (28)$$

where $\triangle E$ represents the gain in the objective from the minimal objective at Y_{soln}—$E(Y_{opt}) - E(Y_{soln})$.

Since we know $\partial E / d\epsilon$ is maximal at ϵ_{opt} [for $0 \leq \epsilon \leq \epsilon_{opt}$], we can write:

$$\triangle E \leq \epsilon_{opt} \int_{t_0}^{t_1} \mid EL(Y_{opt}) \mid \, dt \quad (29)$$

since $\mid EL(Y) \phi(t) \mid \leq \mid EL(Y) \mid \mid \phi(t) \mid$. If EL_{max} is the maximal magnitude of $EL(Y_{opt})$ between t_0 and t_1, we can write:

$$\triangle E \leq (t_1 - t_0) \epsilon_{opt} EL_{max} \quad (30)$$

$$\frac{\triangle E / (t_1 - t_0)}{\epsilon_{opt}} \leq EL_{max} \quad (31)$$

This states that the net time averaged violation (beyond what is required) in the penalty or objective function per unit displacement (from the optimal path) is less than or equal to the maximal magnitude of the left-hand side of the Euler-Lagrange equation. We can also apply this equation locally to claim that the time averaged violation per unit displacement at a point is less than or equal to the magnitude of the left hand side of the Euler-Lagrange equation evaluated at that point.

Equation 30 can be written in pointwise form as:

$$\frac{\triangle E}{\triangle t} \leq \epsilon_{opt}(t) EL(t) \tag{32}$$

Here the left hand side represents the unnecessary "unphysicality" (as measured by the integrated objective E) per unit time in the neighborhood of time t. $\epsilon_{opt}(t)$ represents the magnitude of the deviation of Y_{opt} from Y_{soln} in the neighborhood of time t (in this formulation ϵ_{opt} is a function of time because we consider an infinitesimal neighborhood near t fixing the position and velocities at the end points to be those of the spline solution). If ϵ_{opt} is relatively independent of t compared to EL, we can estimate the unnecessary unphysicality merely by considering the magnitude of the left hand side of the Euler-Lagrange equations (EL). This is the approach we use.

Equation 31 can be used more accurately to estimate the time averaged violation in the penalty function if a value for ϵ_Y can be calculated. This can be done either by solving the Euler-Lagrange equation $EL(Y) = 0$ and comparing this solution to the optimized spline solution calculated after minimization. Here, the boundary conditions (described in Appendix B) at t_0 and t_1 are fixed by the corresponding values for the optimized spline solution. An alternative is to calculate a solution to the Euler-Lagrange equations locally (for instance with a simple low-order power-series solution). We then have:

$$\epsilon_{opt} = \int_{t_0}^{t_1} \mid Y_{opt}(t) - Y_{EL}(t) \mid dt \tag{33}$$

where Y_{opt} is the optimized solution, and Y_{EL} is the solution calculated by the methods discussed above. We can also use this equation pointwise :

$$\epsilon_{opt}(t) = \mid Y_{opt}(t) - Y_{EL}(t) \mid \tag{34}$$

where ϵ_{opt} is now a function of time.