Speed Adjustment for Key-Frame Interpolation

Richard H. Bartels
University of Waterloo
Department of Computer Science
Waterloo, Ontario N2L 3G1
Canada

Ines Hardtke
National Film Board of Canada
Studio A, French Animation
Box 6100, Station A
Montreal, PQ H3C 3H5
Canada

Abstract

We present a method of *speed adjustment* as a means of controlling motion in of key-frame animation. This method is designed for key-point trajectories that are expressed as differentiable parametric curves. The animator is given a means of influencing the speed of a trajectory's traversal by the direct manipulation of a *speed-profile curve*. Any adjustment of this curve is translated into an adjustment of the differential rate of the parametric sampling that produces inbetween frames from the trajectory. The sampling rate is correlated with object-space distances along the trajectory. This results in intuitively meaningful control over the motion of the animation.

KEYWORDS: Key-Frame Animation, Splines, Timing Control, Kinematic Adjustment.

1. Introduction

This paper describes a method for controlling the temporal aspects of motion, as distinct from its spatial aspects. The method is adaptable for any key-frame animation system in which the trajectories between key points are expressed as differentiable parametric curves.

In Section 2 we give a brief context in which this work is being carried out. Section 3 reviews parametric trajectories. Section 4 covers the speed-profile curve and its means of control, and Section 5 describes how the speed-profile curve is used to influence motion. Section 6 describes a prototype line-test system prepared at the National Film Board of Canada.

2. Context

Computer-aided animation can be divided roughly into two classes: one is *key-frame animation*, in which motion is determined from a sequence of arbitrarily set positions, and the other is *dynamic simulation*, in which motion is determined from physical laws. We are interested here in key-frame animation, particularly in systems that provide automatic inbetweening and include utilities for subjective timing control; that is, *kinematic adjustment*.

Linear interpolation is the simplest approach to generating inbetween frames automatically, and most early key-frame systems used this method, for example the prototype system developed in the late 1960's by Nestor Burtnyk and Marceli Wein [Burtnyk71]. Linear interpolation, however, is not sufficiently smooth for high-quality animation. It is associated with a number of objectionable artifacts [Kochanek84]. Subsequently, Burtnyk and Wein [Burtnyk76] developed a key-position approach based upon skeletons in which motion was defined using stick figures, which were then "filled out" with polygons for the final rendering. The use of cubic interpolation resulted in smooth motion for the stick figures, but the polygons "clothing" the skeletons still exhibited linear artifacts. In 1977 Martin Tuori [Tuori77] described a way of automatically smoothing out linearly defined motions by a method of *overlapping actions*. This technique removed some of the artifacts of linear interpolation, but it could result in trajectories that missed key positions entirely. In the early 1980's Doris Kochanek, Richard Bartels, and Kellogg Booth produced an automatic inbetweening method that was based upon the cubic interpolation of key positions by *Catmull-Rom splines* [Kochanek82]. Cubic trajectories have become a standard method for defining the spatial aspects of motion in modern key-frame systems.

A step toward providing a measure of kinematic control came with the 2-D system GENSYS developed by Ron Baecker [Baecker69]. Here the animator provided a combined description of the path and timing of a motion by drawing motion paths called *P-curves* on a digitizing tablet. The shape of a P-curve defined the inbetweening trajectory, and the speed at which the curve was drawn dictated the temporal aspects of the motion. Later, William Reeves [Reeves81] investigated *moving point constraints* as a way of generalizing P-curves. This added a further degree of control to the P-curve approach. The requirement that the animator specify the trajectory, however, has prevented the incorporation of the P-curve approach in automatic key-framing systems.

This research was supported by the Canadian Government through the Natural Sciences and Engineering Research Council and the National Film Board.

In 1984 the inbetweening technique of Kochanek, Bartels, and Booth was modified to include three control parameters: *tension*, *continuity*, and *bias* [Kochanek84]. To the automatic definition of a trajectory from key frames was added a limited capacity to select aspects of motion kinematics. But selection does not constitute control. No adjustment of the kinematic aspects separate from the trajectory was provided.

One of the important things animators learn is that it is subtle variations in temporal aspects that give personality to motion [Lasseter87]. Thus, in the computer interpolation process it is desirable that the temporal element be isolated from the spatial, to be available to the animator under separate control. In 1985 Scott Steketee and Norman Badler [Steketee85] addressed the problem of separating the kinematic from the spatial aspects of key-frame animation through the use of two functions employed in composition, one to incorporate frame numbers and the other to specify frame times. The function for frame times was available for the animator to modify.

In 1987, Bartels and Hardtke [Bartels87] investigated two methods of controlling the kinematics of a key-frame animation. One of these offered control over frame times, much like the method of Steketee and Badler, and the other provided a means of directly influencing the relative speed over which portions of the trajectory are traversed. We report on the method of speed control here. Unlike the method of Badler and Steketee, or methods used in the commercial systems known to the authors, the proposals presented here provide the animator a direct manipulative grasp on object-space speed rather than the indirect control of speed that is achieved through modification of frame times or of *ad hoc* adjustments to the sampling rate on a parametric trajectory curve. Animators will have need of various tools to achieve their goals. For certain purposes, control over frame times may be less desirable than direct control over motion velocity. With this in mind, speed control was provided to animators at the National Film Board of Canada in a prototype line-test system which was used in conjunction with the WaveFront animation system to produce a trial animated sequence. Response from the animators has been favorable.

3. Trajectory

Ideally, key frames should be able to define a continuous spatial path for the motion to follow. The frames recording that motion should be interpolated from the spatial path according to some temporal sampling pattern provided in an intuitive fashion by the animator. Changing the sampling pattern should not change the spatial path defined by the key positions.

Let $\mathbf{P}_0$, $\mathbf{P}_1$, . . . , $\mathbf{P}_n$ be the position of a single key point throughout a key-frame sequence. We assume that an inbetweening system has produced a parametric trajectory

$$\mathbf{P}(s) = (x(s), y(s))$$

joining the locations $\mathbf{P}_i$,

$$\mathbf{P}_i = (x_i, y_i) = (x(s_i), y(s_i)) = \mathbf{P}(s_i) \ .$$

We shall present the discussion here in terms of trajectories that are 2-D curves in x-y space. The discussion is equally applicable to 3-D curves, and it can be modified to handle trajectories in other coordinates of motion; e.g., angles of rotation. All we will need to require is that the trajectory be parametric, that it be differentiable, and that appearance of motion be directly related to some measure of spatial progress along the trajectory, such as arc length.

The parameter s satisfies $s_0 \le s \le s_n$ for the trajectory, with $s = s_i$ for the i^{th} key frame. The parameter s need not have any physical significance, and its relation to the amount of motion over any portion of the trajectory is often arbitrary and varys from location to location. A simple animation system would compute the inbetween frames from $\mathbf{P}_i$ to $\mathbf{P}_{i+1}$ by sampling s at uniformly spaced values over the interval from s_i to s_{i+1}. The resulting appearance of motion is often unsatisfying due to the arbitrary association of values, s, in parameter space with points, P, in object space along the trajectory. The motion can be adjusted by modifying the sampling of s in some fashion. Commercial systems known to the authors arrange to do this by providing access to s through some form of "sampling function," $s = s(t)$, that the animator can control. System-defined steps in t provide parameter values in s that are nonuniformly spaced, but this leaves the animator with a trial-and-error process of manipulating the sampling function until the desired appearance of motion is attained. The method of Steketee and Badler [Steketee85], on the other hand, splits the description of motion into a "position component," giving location in terms of frame number, and a "kinetic component," giving frame number in terms of time. The two component functions are composed to give the final motion. The animator has precise timing control and, indirectly through the frame times, control over such aspects of motion as speed.

The adjustment we shall be discussing is a method to alter the default sampling to use nonuniformly spaced of values of s. But the method of achieving this will be done with a mechanism that relates directly to speed of progress along the trajectory. It provides a method that works directly in the object space of the animation scene. This is intended to complement the method of Steketee and Badler.

The effect of nonuniform parametric sampling can be understood in either of two ways. From one point of view, the playback of an animation sequence is locked into a fixed number of frames per second (e.g., 24 for film or 30 for video), and a nonuniform sampling rate acts as a variable-speed shutter on a camera. Assuming that the sampling density in the parameter s is coupled with the arc length along the trajectory; that is, with the actual distance covered by the motion, dense sampling expands the detail of the trajectory, providing a slow-motion version of the action, and sparse sampling compresses the detail, providing a time-lapse version of the action. The playback has an exactly predictable timing based upon the number of frames in the sequence (this is of importance in planning an animation), but the kinematic effect of the action will be determined by the sampling density. The motion will seem to pick up speed where the differential between shutter clicks is high (a slow shutter) and will seem to lose speed where the differential between shutter clicks is low (a fast shutter). From the other point of view, the shutter is regarded as recording the motion at a fixed rate, and dense sampling occurs at points of the trajectory where motion is slow while sparse sampling occurs where motion is fast. A

fixed number of shutter clicks is allocated to the sequence of motion, and the sampling rate reflects an orchestration of the speeds along the path of motion required to complete the sequence in the number of frames given. This orchestration of speedups and slowdowns constitutes the *kinematic* adjustment.

4. Speed-Profile Curve

The technical background for this approach comes from [Mastin86], where similar methods are used to sample parametric curves and surfaces nonuniformly, using reparameterization, for the purposes of creating finite-element grids. In the case of finite elements, the reparameterization of a curve is made to be sensitive to some physical property such as local curvature, causing it to act as a "profiling function" governing the sampling of the curve. In our case we wish to construct the reparameterization according to relative variations in speed that an animator can set interactively. To do this we must represent the parameter s as a function of another parameter, α, $s = s(\alpha)$, and create a function $\phi(\alpha)$ that, in a way to be explained in the next section, forces equally spaced values of α to produce values of s spaced along some desired pattern of points on the trajectory $\mathbf{P}(s)$. We will call $\phi(\alpha)$ the *speed-profile curve*. Peaks in the curve $\phi(\alpha)$ should yield widely spaced points on the trajectory, producing accelerated motion, and valleys in the curve $\phi(\alpha)$ should yield closely spaced points on the trajectory, producing slowed motion, where closeness of spacing is measured in Euclidian distance along the arc of the trajectory. This attention to arc-length, object-space distance is required to provide a direct linkage between the input of the animator and the effect on the speed of motion.

In the prototype system, $\phi(\alpha)$ is presented as a B-spline curve defined by control values $v_0, \ldots, v_f$ that the animator can insert, delete, raise, and lower. The range of values that each v_j can take on dictates the shape of the speed-profile curve and the influence that the curve will have over some portion of the sampling process. The nature of a B-spline curve is such that the maximum and minimum values of the v's will contain the maximum and minimum values of $\phi(\alpha)$. (This *convex hull property* B-splines is established and described in [Bartels87b], together with other properties.) Some tuning will be necessary to suit the desires of the individual animator, but it is unlikely that a range greater than $0 < \phi(\alpha) < 100$ will be needed. Figures 4, 5, and 6, for example, were produced for $\phi(\alpha)$ between 1 and 25.

An example of a B-spline speed-profile curve is shown in Figure 1, and Figure 2 shows the curve after the control value v_4 has been lowered.

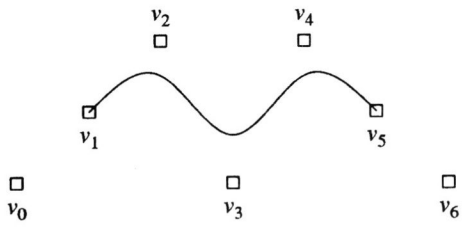

Figure 1. A B-spline curve.

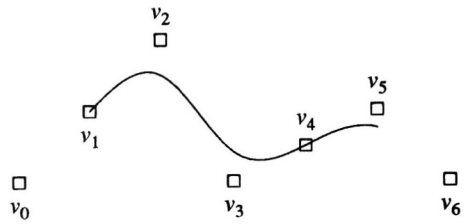

Figure 2. An adjusted B-spline curve.

5. Sampling

The trajectory forms a locus of points in 2-D, $(x(s), y(s))$, and we are assuming that we can express s as a function of some more deeply underlying parameter α. The arc-length derivative with respect to α along the trajectory is given by

$$\left(\frac{ds}{d\alpha}\right)\sqrt{\left(\frac{dx}{ds}\right)^2+\left(\frac{dy}{ds}\right)^2} \ .$$

An interpretation of this formula is that, if we wish to maintain a constant change in arc length (a constant speed) by taking constant steps in α, then we must take steps in s at a rate (per unit α) proportional to

$$\frac{1}{\sqrt{\left(\frac{dx}{ds}\right)^2+\left(\frac{dy}{ds}\right)^2}} \ ;$$

that is, we must arrange for s to satisfy the differential equation

$$\frac{ds}{d\alpha} = \frac{K}{\sqrt{\left(\frac{dx}{ds}\right)^2+\left(\frac{dy}{ds}\right)^2}} \ ,$$

where K is the constant speed to be imposed. In fact, we wish to vary the speed throughout the trajectory, increasing or decreasing K by a factor at each α as we go along. Letting the function $\phi(\alpha)$ give the scale of increase or decrease, this means that we are interested in making s satisfy the more general differential equation

$$\frac{ds}{d\alpha} = F(\alpha,s) \ ,$$

where

$$F(\alpha,s) = \frac{K\,\phi(\alpha)}{\sqrt{\left(\frac{dx}{ds}\right)^2+\left(\frac{dy}{ds}\right)^2}} \ . \qquad (5.1)$$

The basic speed, K, is free to be set so that the range of α corresponds to the range of s. We are only interested in the relative increase or decrease of speed over the extent of the action. The total time of the action is traditionally set by the animator in terms of the number of frames to be produced from $\mathbf{P}_0$ to $\mathbf{P}_n$, and we must not violate that timing. Our aim is merely to adjust the appearance of local speedups and slowdowns. The variation in α depends upon the number of control values. For $f+1$ control values, $v_0, \ldots, v_f$, the range of α will be $0 \le \alpha \le f$. To fix the traversal time of the trajectory so that all of the frames fit within the time, we require that

$$s(0) = s_0 \qquad (5.2)$$

and that

$$s(f) = s_n \ . \qquad (5.3)$$

Equation (5.2) represents the initial value of the first-order, ordinary differential equation (5.1), and it can be satisfied by setting s to s_0 when α is set to 0 at the start of the integration process. Equation (5.3) implicitly defines the choice of K, to be explained below.

In the prototype system a simple Runge-Kutta integrator is used to obtain the values of s corresponding to the frames. (The Runge-Kutta integration process is described in [Forsythe77] and [Stoer80], for example.) Assuming that $N+1$ frames are to be produced from P_0 (corresponding to frame 0 with $\alpha=0$) to P_n, (corresponding to frame N with $\alpha=f$), then the integrator is responsible for producing values of s corresponding to

$$\alpha=0\,,\ \alpha=\Delta\,,\ \alpha=2\Delta\ ,\ \ldots\ ,\ \alpha=N\Delta=f$$

where $\Delta=f/N$. Such integrators usually demand, for accuracy's sake, that these values of α be obtained from α-steps of size h equal to some fraction of Δ; i.e., $h=\Delta/k$. ($k=10$ is often a reasonable choice.) The Runge-Kutta integrator and the sampling process is shown in Figure 3. The value of *slack* is positive. It represents the amount we may let the integration process overshoot the end of the trajectory.

The intermediate key positions, $\mathbf{P}_1\ ,\ \ldots\ ,\ \mathbf{P}_{n-1}$ will appear in frames only by chance. While the path of motion is guaranteed to go through the key positions, this sampling

```
{set a value of K}
α := 0
s := s0
h := f/(k×N)
i := 0
while ( α < f and s < sn + slack ) do
      F1 := F( α , s )×h
      F2 := F( α+h/2 , s+F1/2 )×h
      F3 := F( α+h/2 , s+F2/2 )×h
      F4 := F( α+h , s+F3 )×h
      s := s + ( F1+F2+F3+F4 ) / 6
      α := α + h
      i := i+1
      if ( i = k ) then
            i := 0
            {sample P at s}
      endif
endwhile
```

Figure 3. The Runge-Kutta integration process.

process does not necessarily result in the "shutter being open" at the "times" corresponding to $s=s_i$, the values of s representing the key positions. Animators have not found this a negative aspect of the method. But, in the event that a key position must be captured in a frame, the animation can be broken into portions in which that key serves as the final key for one part and as the initial key for the next.

K must be chosen so that equation (5.3) is satisfied; that is, so that the integration will end at $\alpha=f$ with $s(f)=s_n$. This is a root-finding problem that we can solve using the ZEROIN version of the secant method given in [Forsythe77]. Briefly, s is regarded as a "function" of the "variable" K (in addition to being a function of α), and ZEROIN is used with the s obtained at the end of the integration as the "function value" corresponding to K. To find the value of K which forces s to equal s_n at the end of the integration; that is, the "root" of the equation $s=s_n$, two bounding values of K, K_{low} and K_{high}, are first found by trial integrations so that K_{low} yields $s(f)<s_n$ and K_{high} yields $s_n+slack>s(f)>s_n$. This "brackets" the root (i.e. the value of K in the interval $[K_{\text{low}}, K_{\text{high}}]$ for which $s(f)=s_n$). The ZEROIN process continually refines $[K_{\text{low}}, K_{\text{high}}]$, as well as a trial value of the root K_{trial} within this interval, reducing the width of the interval until a preset tolerance is attained. Each iteration of ZEROIN costs a Runge-Kutta integration from $\alpha=0$ to $\alpha=f$ with a trial value of K. The Runge-Kutta process and ZEROIN are both fast enough, however, that a complete revision of the sampling of a trajectory can be accomplished in a few seconds on the prototype system described in the next section.

Three examples of the effect the speed profile has on the sampling of a trajectory are presented in Figures 4, 5, and 6.

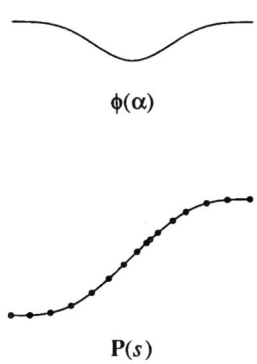

Figure 4. A low speed in the central portion of the trajectory.

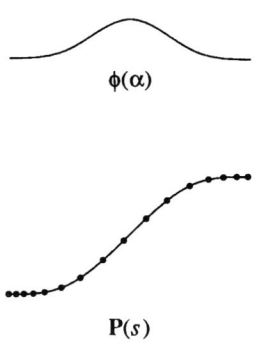

Figure 5. A low speed at the ends of the trajectory.

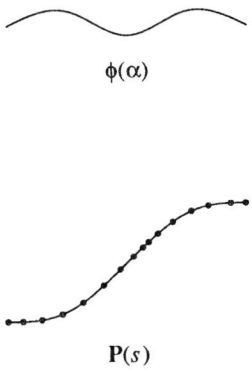

Figure 6. Mild variations of speed throughout the trajectory.

6. Prototype System

These ideas have been implemented in C on a Silicon Graphics 2400 Turbo IRIS workstation. The overall screen layout for the method of speed-profile adjustment is indicated in Figure 7.

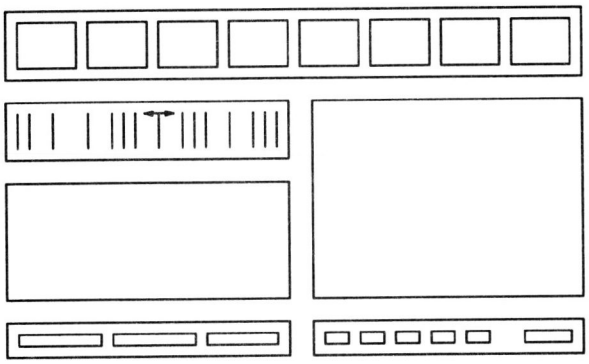

Figure 7. Layout of the prototype system's screen.

The display is divided into three regions: a key-frame scripting area, a timing control area and a preview port. The top portion of the screen contains the key-frame scripting area. In this region keys can be moved around, copied, deleted and added. The area underneath the scripting portion, on the right, provides a line-test preview of the sequence. To control previewing, at the bottom right of the screen is a control panel whose form and function resemble that of a videotape player. It provides a frame counter and buttons for single stepping or 24-frame-per-second cycling in either forward or backward directions. The left side of the screen is the timing control area, which is divided into rectangular sub-areas for the display of timing information. The bottom sub-area is for kinematic "fine tuning" and displays the speed-profile curve. Operations to move, add, or delete control points are invoked from a menu at the bottom left. Above the speed-profile area is another rectangular sub-area containing a stripe for each frame, with the frame stripes spaced in proportion to their distance along the trajectory. The animator may interpret this as representing time of display. This manner of showing the timing of a script is analogous to the time bars animators traditionally draw and use [Whitaker81]. For coarse changes to the timing of the script, the animator is permitted to change the actual time of any selected key frame. To do this, arrow-shaped buttons are provided in the time-bar area that shift any selected key one frame forward or backward in the sequence of frames.

On the subject of separating the the temporal element of motion from the spatial, the reactions of animators and of people otherwise familiar with computer animation have been overwhelmingly positive. On the whole, animators appear to feel quite comfortable with the notion of a speed-profile curve and with the ability to directly manipulate the curve itself. In particular, the freedom to define a curve that can but doesn't have to be anchored to key-frames seems to be quite attractive.

7. Acknowledgements

We have benefited from discussions, support, encouragement, and opinions provided by many people. We would particularly like to thank Marc Aubry, Kelly Booth, Robert Forget, Michel Hébert, Terry Higgins, Doris Kochanek, Daniel Langlois, Dave Martindale, Richard Mercille, and Bruno Tezenas du Montcel.

8. References

Baecker69.
Ron Baecker, "Interactive Computer-Mediated Animation," MAC-TR-61, PhD Thesis, Massachusetts Institute of Technology (1969).

Bartels87a.
Richard Bartels and Ines Hardtke, "Kinetics for Key-Frame Interpolation," CS-87-07, Computer Graphics Laboratory, Computer Science Department, University of Waterloo (1987).

Bartels87b
Richard Bartels, John Beatty, and Brian Barsky, **An Introduction to Splines for Use in Computer Graphics and Geometric Modeling**, Morgan Kaufmann Publishers, Palo Alto, California (1987).

Burtnyk71.
Nestor Burtnyk and Marceli Wein, "Computer Generated Key Frame Animation," *Journal of the SMPTE*, **80**, pp. 149-153 (1971).

Burtnyk76.
Nestor Burtnyk and Marceli Wein, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation," *Communications of the ACM*, **10**, pp. 564-569 (1976).

Forsythe77.
George Forsythe, Michael Malcolm, and Cleve Moler, **Computer Methods for Mathematical Computations**, Prentice-Hall (1977).

Kochanek82.
Doris Kochanek, Richard Bartels, and Kellogg Booth, "A Computer System for Smooth Keyframe Animation," CS-82-42, Computer Graphics Laboratory, Computer Science Department, University of Waterloo (1982).

Kochanek84.
Doris Kochanek and Richard Bartels, "Interpolating Splines with Local Tension, Continuity and Bias Control," *Computer Graphics*, **18** (3), pp. 33-41 (1984) [Proceedings of the SIGGRAPH '84 Conference].

Lasseter87.
John Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation," *Computer Graphics*, **21** (4), pp. 35-44 (1987) [Proceedings of the SIGGRAPH '87 Conference].

Mastin86.
C. Wayne Mastin, "Parameterization in Grid Generation," *Computer-Aided Design*, **18** (1), pp. 22-24 (1986).

Reeves81.
William T. Reeves, "Inbetweening for Computer Animation Utilizing Moving Point Constraints," *Computer Graphics*, **15** (3), pp. 263-269 (1981) [Proceedings of the SIGGRAPH '81 Conference].

Steketee85.
Scott Steketee and Norman Badler, "Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control," *Computer Graphics*, **19** (3), pp. 255-262 (1985) [Proceedings of the SIGGRAPH '85 Conference].

Stoer80.
Josef Stoer and Roland Bulirsch, **Introduction to Numerical Analysis**, Springer Verlag (1980).

Tuori77.
Martin I. Tuori, "Tools and Techniques for Computer-aided Animation," Master's Thesis, Computer Science Department, University of Toronto (1977).

Whitaker81.
Harold Whitaker and John Halas, **Timing for Animation**, Focal Press Limited, London, England (1981).