# Interpolating Splines with Local Tension, Continuity, and Bias Control

*Doris H. U. Kochanek*

Computer Motion Graphics Centre
National Film Board of Canada
Box 6100, Station A, P-36
Montreal, Quebec, Canada, H3C 3H5
(514) 333-3434

*Richard H. Bartels*

Computer Graphics Laboratory
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
(519) 886-1351

**Abstract**

This paper presents a new method for using cubic interpolating splines in a key frame animation system. Three control parameters allow the animator to change the tension, continuity, and bias of the splines. Each of these three parameters can be used for either local or global control. Our technique produces a very general class of interpolating cubic splines which includes the cardinal splines as a proper subset.

CR Categories and Subject Descriptors: G.1.1 [**Numerical Analysis**]: Interpolation – *spline and piecewise polynomial interpolation*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling – *curve, surface, solid and object representations*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism – *animation.*

General Terms: algorithms

Additional Key Words and Phrases: key frames, inbetweening, bias, continuity, tension

## 1. Introduction

One of the oldest techniques used in computer animation is the automatic generation of *inbetweens* (intermediate frames) based on a set of *key frames* supplied by the animator [5]. This same principle is frequently used in computer assisted 3-D animation where camera and object positions are defined only at key points in the animation, leaving the calculation of intermediate positions to the computer. The straightforward *linear interpolation* algorithm used in many systems produces some undesirable side effects which give the animation a mechanical look, often referred to as the "computer signature." The most objectionable characteristic of this type of animation is the lack of smoothness in the motion. The key frames may be clearly visible in the animation because of sudden changes in the direction of motion (Figure 1).

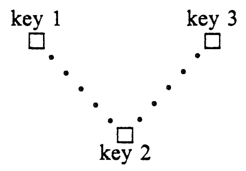

**Figure 1.** Discontinuity in direction with linear interpolation

Discontinuities in the speed of motion may also be visible with linear interpolation, for example when the animator requests a different number of frames between successive keys (Figure 2).

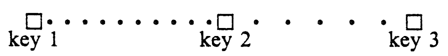

**Figure 2.** Discontinuity in speed with linear interpolation

A third common problem is distortion, which may occur whenever the movement has a rotational component (Figure 3).

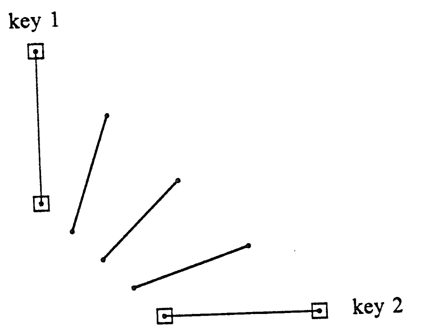

**Figure 3.** Distortion in length when rotation is simulated linearly.

In view of these serious drawbacks of linear interpolation, a number of different methods which produce smoother motion have been published. These techniques include *P-curves* [1], *skeletons* [6], *action overlap* [11], and *moving point constraints* [9]. All of these techniques require the animator to specify additional information other than just the key frames. A completely automatic system which uses only the key positions supplied by the animator was implemented by one of the authors [8]. The approach used there was based on fitting a set of interpolating splines through the key positions, resulting in much smoother animation than can be produced with linear interpolation.

## 2. Interpolating Splines

We assume that each of the objects in the $i^{th}$ key frame in a sequence can be described by a collection of *points*. (As an example, the two designated endpoints of the line segment in key 1 shown in Figure 3 completely define the segment.) We assume that to each point in one key frame there will be a corresponding point in all other key frames of a motion sequence. (For example, the same two endpoints reappear in key 2 of Figure 3 to specify a later position of the line segment.) These assumptions are not as restrictive as they may appear to be. Even quite complicated curved objects can be expressed in terms of small numbers of *control points* using the techniques described in, for example, [7]. For the purposes of the discussion, we will fix our attention on one such point, designated $P_i$ and referred to as the *key position*, in the $i^{th}$ key frame:

$$P_i = (x_i, y_i, z_i)$$

(We will carry on the discussion in terms of 3-D animation.) Given a sequence of corresponding key positions,

$$\cdots, P_{i-1}, P_i, P_{i+1}, \cdots$$

we want to interpolate them using a simple smooth curve. For sufficient generality to handle the multi-valued case, all curves will be treated parametrically as

$$P(s) = (x(s), y(s), z(s))$$

where $s$ varies from 0 to 1 between each two key frames. Thus, we want to find smooth functions $x(s)$, $y(s)$, and $z(s)$ so that, for example,

$$P(0) = (x(0), y(0), z(0)) = (x_i, y_i, z_i) = P_i$$

and

$$P(1) = (x(1), y(1), z(1)) = (x_{i+1}, y_{i+1}, z_{i+1}) = P_{i+1}$$

on the interval between the $i^{th}$ and $i+1^{st}$ key frames. The positions of the inbetween frames which will correspond to the key position in question, then, will be $P(s)$ for $s = \Delta, 2\cdot\Delta, \cdots 1-\Delta$ for

$$\Delta = \frac{1}{N_i} + 1$$

where $N_i$ is the number of inbetweens to be generated between the key frame containing $P_i$ and the key frame containing $P_{i+1}$. Polynomials are a natural choice for the smooth functions because of their simplicity, but using a single interpolating polynomial of high degree for the entire sequence could result in motion which oscillates about the path we expect the animation to follow. A more "natural" fit of the key positions can be obtained by interpolating them with a cubic spline, a curve consisting of a succession of different cubic polynomial segments which are joined together with certain continuity constraints.

Each cubic polynomial extends between two key positions and is uniquely defined by four coefficients which we can determine from four independent constraints. Two constraints are given directly by the interpolation conditions: the spline segment must pass through the key positions at the start and the end of the interval. This leaves two free constraints which we can choose. The choice which defines the most commonly used form of cubic spline imposes first and second derivative continuity at the keys, but this approach is computationally expensive and quite inflexible. Instead, we choose to specify tangent vectors at the two adjacent keys to define each spline segment. In our notation the tangent vector to the curve we wish to construct through key position $P_i$ is given by

$$D_i = \left(\frac{dx}{ds}, \frac{dy}{ds}, \frac{dz}{ds}\right)$$

No conditions will be imposed on second derivatives.

By default, we determine appropriate tangent vectors from the geometry of the surrounding keys. This approach can be generalized to produce a very flexible class of cubic splines by the introduction of control parameters which modify the length and direction of the tangent vectors.

Any cubic polynomial can be expressed as a scaled sum of four basis functions. Frequently these functions are taken to be the monomials $s^3$, $s^2$, $s$, 1, however, for our purposes the *Hermite interpolation basis functions* shown in Figure 4 are more useful.

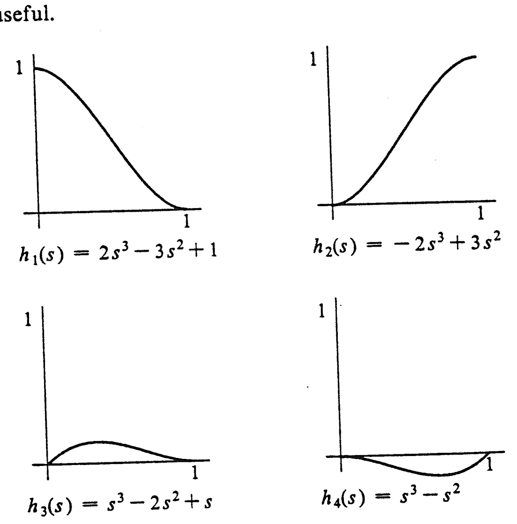

**Figure 4.** Basis functions for Hermite interpolation.

These functions have the following convenient properties:

| | $h_1$ | $h_2$ | $h_3$ | $h_4$ |
|---|---|---|---|---|
| function value at $s=0$ | 1 | 0 | 0 | 0 |
| function value at $s=1$ | 0 | 1 | 0 | 0 |
| derivative at $s=0$ | 0 | 0 | 1 | 0 |
| derivative at $s=1$ | 0 | 0 | 0 | 1 |

Note that $h_1$ alone determines the function value of the composite cubic at the start of the interval. Therefore, $h_1$ can be scaled with a coefficient of $P_i$ to obtain the desired point $P_i$ at $s=0$. Similarly, $h_2$ can be scaled with $P_{i+1}$. The derivative of the composite cubic is determined by $h_3$ at the start and by $h_4$ at the end of the interval, therefore $D_i$ and $D_{i+1}$, the desired derivatives at the interval ends, can be used as scaling factors for $h_3$ and $h_4$. These observations lead to a triplet of cubic polynomials given by

$$\begin{aligned} P(s) &= (x(s),y(s),z(s)) \\ &= P_i \cdot h_1(s) + P_{i+1} \cdot h_2(s) + D_i \cdot h_3(s) + D_{i+1} \cdot h_4(s) \end{aligned}$$

**Equation 1.** Parametric cubic curve using the $h$ basis.

In matrix form this expression reduces to

$$\begin{aligned} P(s) &= \mathbf{s} \cdot \mathbf{h} \cdot \mathbf{C} \\ &= \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_i \\ P_{i+1} \\ D_i \\ D_{i+1} \end{bmatrix} \end{aligned}$$

**Equation 2.** Matrix format for parametric cubics using the $h$ basis.

Note that the vector **s** only changes from one frame in the animation to the next. Within a given frame it applies to the $x$, $y$, and $z$ components of all key positions which are being interpolated. The matrix **h** contains the coefficients of the Hermite interpolation basis functions and is therefore constant for all frames and all key positions. In practice, **s**·**h** is calculated only once per frame. By contrast, each **C**, which is a 4X3 matrix, corresponds to a single key position and is independent of the **C** associated with any of the other key positions being interpolated. It does not change from one frame to another (except at a key frame), and the independence implies that all key positions can be interpolated "in parallel".

## 3. A General Class of Interpolating Cubic Splines

Using this formulation as a framework, the remaining open question is how to find "appropriate" values for the components of $D_i$ and $D_{i+1}$, the tangent vectors at the key positions, needed to fully specify $P(s)$. [10] describes the approach used for a class of cubic splines which are commonly called *cardinal splines*. Even though cardinal splines are not usually formulated in terms of the Hermite interpolation basis functions, the tangent vectors at the key positions are used to constrain the cubic segments. The tangent vector at $P_i$ is calculated as $D_i = a \cdot (P_{i+1} - P_{i-1})$, where $a$ is a constant which affects the tightness of the curve. A particular example of this class of splines is the Catmull-Rom spline for which $a = \frac{1}{2}$. Thus the tangent vector for the Catmull-Rom spline is

$$D_i = \frac{1}{2} \cdot (P_{i+1} - P_{i-1}) = \frac{1}{2} \cdot \left( (P_{i+1} - P_i) + (P_i - P_{i-1}) \right)$$

**Equation 3.** The Catmull-Rom spline.

which is simply the average of the *source chord* $P_i - P_{i-1}$ and the *destination chord* $P_{i+1} - P_i$. The technique presented in this paper uses this average of adjacent chords as the default tangent vector. Thus our default spline, even though formulated differently, is exactly the Catmull-Rom spline (Figure 5).

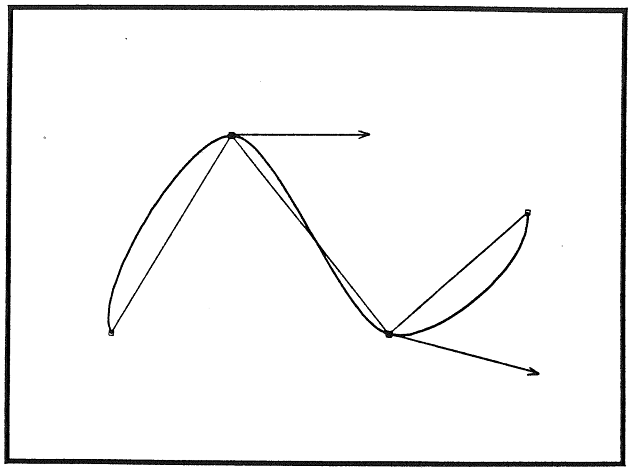

**Figure 5.** The Catmull-Rom spline.

At the beginning of a motion sequence some arbitrary choice for the source chord (e.g. (0,0,0)) must be made. Similarly, the destination chord must be specified arbitrarily at the end of the sequence. Alternatively a specification of the tangent beginning and ending vectors can be made without regard to any chords. Setting $D$ = (0,0,0) was tried with some success in [8].

A "standard" smooth motion through a given set of keys does not always produce the effect desired by the animator. In certain cases he may want the motion to follow a wider, more exaggerated curve, while in other cases he may want the motion path to be much tighter, maybe almost linear. Even continuity in the direction and speed of motion is not necessarily desirable at all times. Animating a bouncing ball, for example, actually requires the introduction of a discontinuity in the motion at the point of impact.

The research described in this paper replaces the standard interpolating spline used in [8] by a highly flexible class of cubic splines which interpolate the same key positions but vary in several control parameters. These three parameters, *tension*, *continuity* and *bias*, allow the animator to fine-tune the animated sequence by changing certain characteristics of the "standard" interpolating spline either locally (applying only in the vicinity of a specific key frame), or globally (applying to the entire motion sequence). The introduction of these three control parameters produces a highly flexible class of interpolating cubic splines which include the cardinal splines as a proper subset.

The concepts of tension and bias have been introduced before in connection with *approximating splines* in [2], [3], [4]. Our use of the term "bias" is similar to the concept being used by the authors of these references. The concept of "tension" which we are using is different, however. Their use of the word "tension" refers to an effect produced by adjusting the match between the second derivatives of adjoining polynomial segments, and we are exercising control only over first derivatives. We are able to produce visually similar effects, however, and so have chosen to borrow the use of their term. An excellent introduction to the theory of interpolating and approximating splines for computer animation can be found in [10].

The three control parameters tension, continuity, and bias are introduced by the convention of separating each tangent at the $i^{th}$ key position into an *incoming* and an *outgoing* part, respectively the *source derivative* $DS_i$ and the *destination derivative* $DD_i$ as indicated in Figure 6.

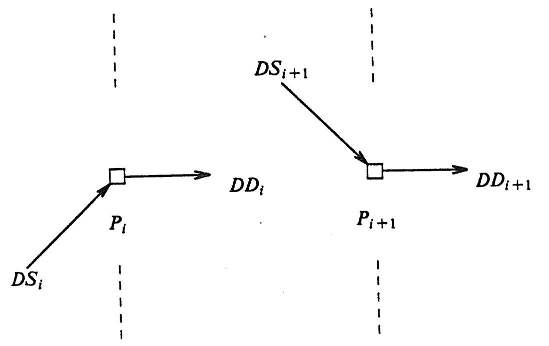

**Figure 6.** Incoming and outgoing tangents of two key positions.

These replace the single derivative $D$ of the Catmull–Rom spline (Equation 3). Furthermore, the average $a=\frac{1}{2}$ of Equation 3 is relaxed in favor of a more selective average of the source and destination chord.

In the following three sections we will treat each of these three parameters independently of the other two. Then the three will be tied together in a fourth section.

### 3.1. Tension

The tension parameter $t$ controls how sharply the curve bends at a key position (Figure 7).

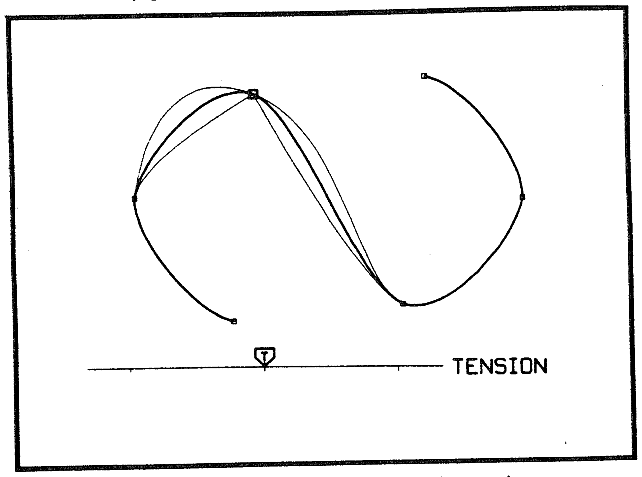

**Figure 7.** Bending of the curve under various tensions.

Tension is implemented as a scale factor which changes the length of both the incoming and outgoing parts of the tangent vector equally at a key position:

$$DS_i = DD_i = (1-t)\cdot\frac{1}{2}\cdot\Big((P_{i+1}-P_i)+(P_i-P_{i-1})\Big)$$

**Equation 4.** Tension equation.

For the default curve $t=0$, and the tangent vector is the average of the two adjacent chords (Figure 8).

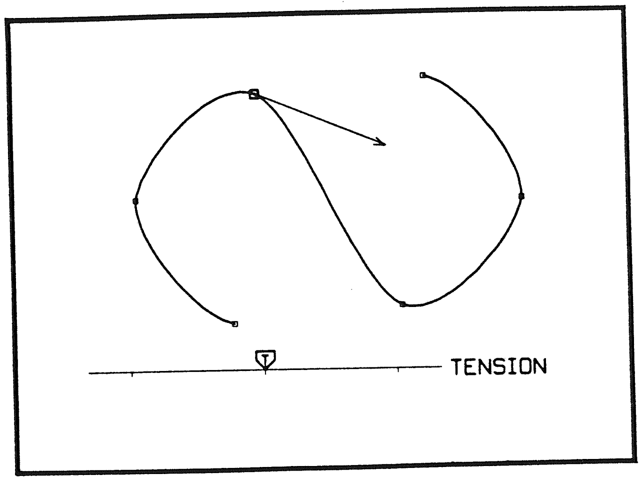

**Figure 8.** Default tension.

Increasing the tension to $t=1$ reduces the length of the tangent vector to zero and thus tightens the curve (Figure 9).

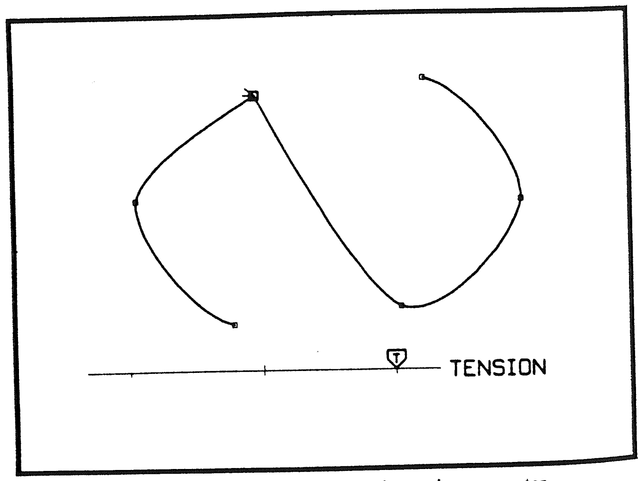

**Figure 9.** The effect of increasing the tension parameter.

Reducing the tension to $t=-1$ increases the tangent vector to twice its default length and produces more slack in the curve (Figure 10).

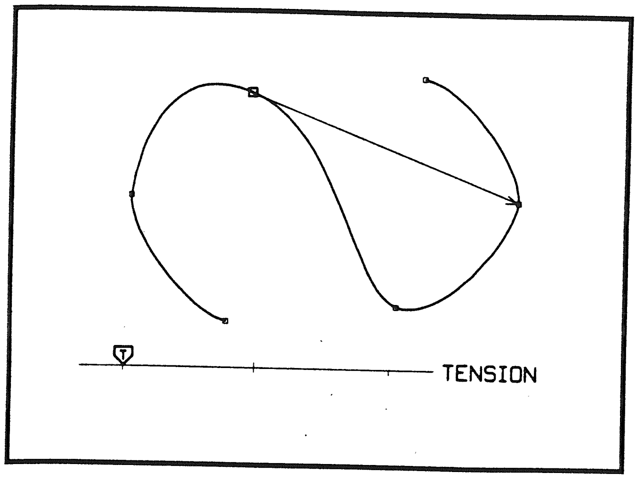

**Figure 10.** The effect of reducing the tension parameter.

If the same value of $t$ is applied to all key positions in the sequence, varying $t$ generates the entire class of cardinal splines with $a=\frac{1-t}{2}$ (Figure 11).

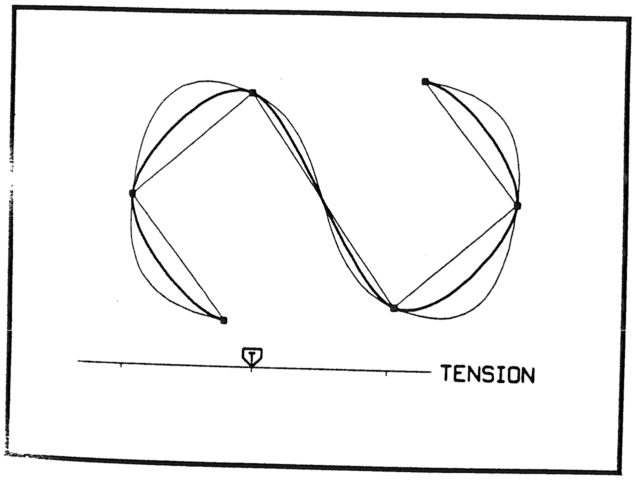

**Figure 11.** Several tension values applied uniformly to all keys.

The default case ($t=0$) is equivalent to the Catmull-Rom spline, where $a=\frac{1}{2}$.

### 3.2. Continuity

The principal reason for using splines in key frame animation is to avoid discontinuities in the direction and speed of motion which are produced by linear interpolation. However, in animation discontinuities are sometimes necessary to create realistic effects such as punching, bouncing, etc. A common technique to introduce such a discontinuity into an otherwise continuous spline is to repeat a key position or to simply terminate the spline at a key and start an entirely independent spline to interpolate the next sequence of key frames.

Neither of these approaches is very satisfactory because the discontinuity cannot be controlled. While it is true that, mathematically speaking, a spline's derivative is either continuous or discontinuous, the artist's view is quite different. He would like to have more control over continuity than a simple on/off switch. In fact, from the animator's point of view two curve segments which have very different tangent vectors at their joint appear "more discontinuous" than two curve segments which have fairly similar tangent vectors. This concept is implemented in our system as a parameter which controls the continuity/discontinuity at a key position (Figure 12).

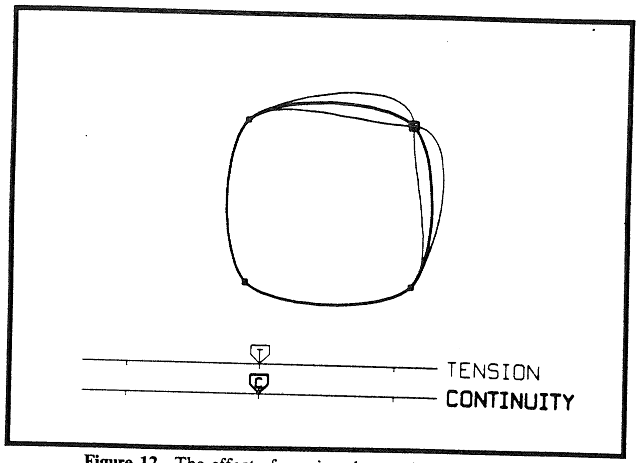

**Figure 12.** The effect of varying the continuity parameter.

Assuming default tension and using $c$ to denote the continuity parameter, we allow the source and destination components of the tangent vector to differ from each other according to:

$$DS_i = \left( \frac{1-c}{2} \cdot (P_i - P_{i-1}) + \frac{1+c}{2} \cdot (P_{i+1} - P_i) \right)$$

**Equation 5.** The "incoming" continuity equation.

$$DD_i = \left( \frac{1+c}{2} \cdot (P_i - P_{i-1}) + \frac{1-c}{2} \cdot (P_{i+1} - P_i) \right)$$

**Equation 6.** The "outgoing" continuity equation.

Note that with $c=0$ (which we use as a default) we obtain $DS_i = DD_i$, which produces a spline with tangent vector continuity at the keys (Figure 13).

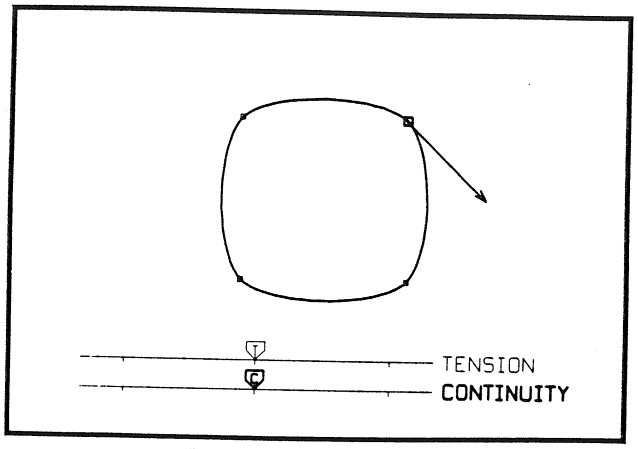

**Figure 13.** Default continuity.

As $|c|$ increases, the two tangent vectors become increasingly distinct. When $c = -1$, the source tangent vector $DS_i$ reduces to the source chord, and the destination tangent vector $DD_i$ is the destination chord, producing a pronounced corner in the curve (Figure 14).

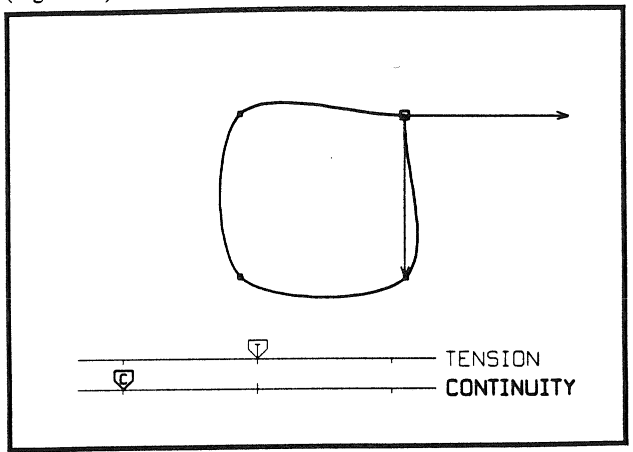

**Figure 14.** The effect of decreasing the continuity parameter.

Going in the opposite direction, at $c=1$, $DS_i = P_{i+1} - P_i$ and $DD_i = P_i - P_{i-1}$, produces a corner pointing in the opposite direction (Figure 15).

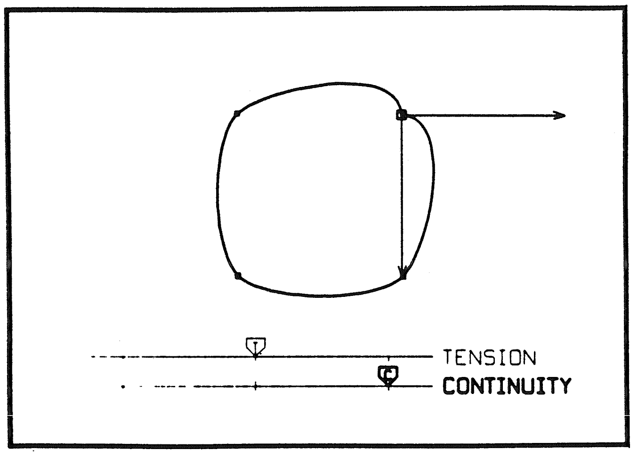

**Figure 15.** The effect of increasing the continuity parameter.

In Figure 16 some results of applying the same value of $c$ to all key positions are depicted.

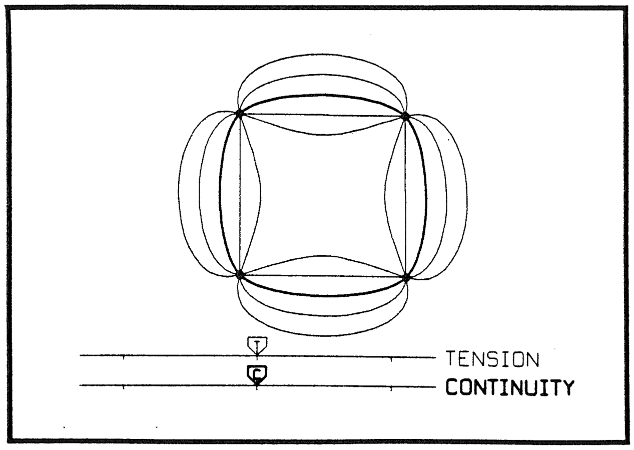

**Figure 16.** Several continuity values applied uniformly to all keys.

Looking only at the *path* of motion, the effects of increasing the tension or reducing the continuity seem to be rather similar (Figures 17 and 18).

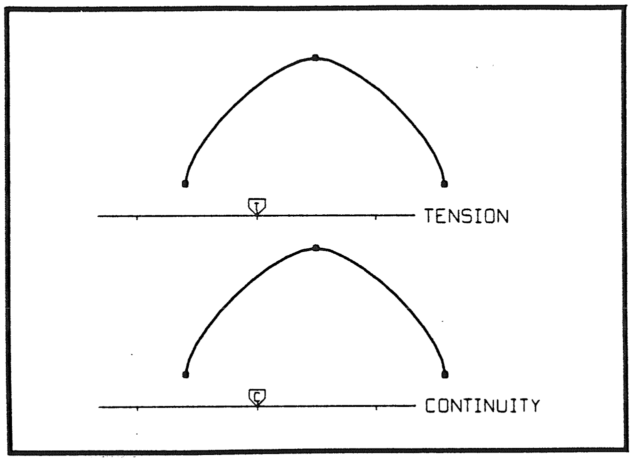

**Figure 17.** Default tension and continuity.

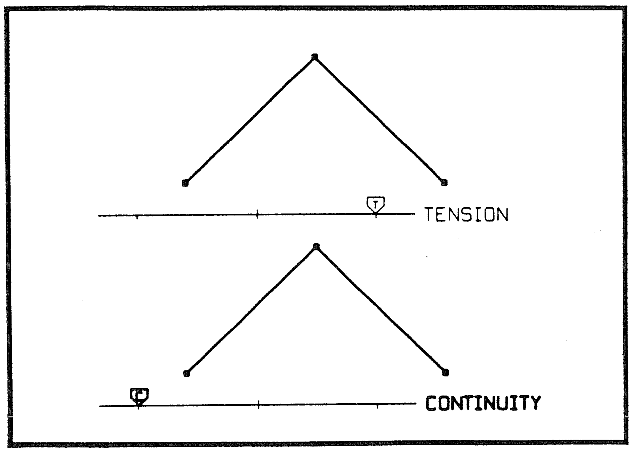

**Figure 18.** Increasing tension, reducing continuity.

However, the *motion dynamics* are quite different (Figure 19).

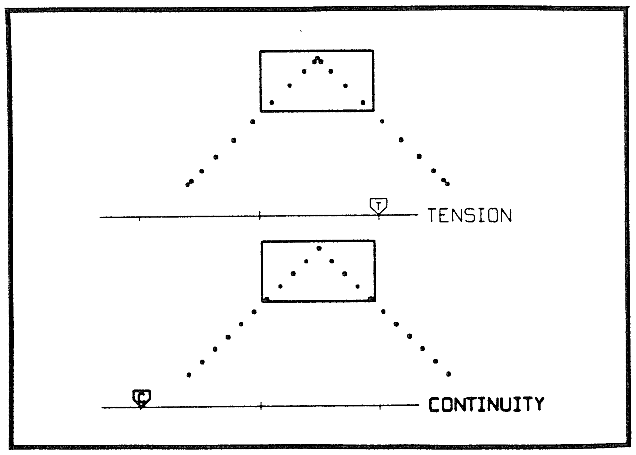

**Figure 19.** Differences in motion: tension vs. continuity.

Increasing the tension does *not* introduce discontinuities in the velocity. Instead the length of the tangent vector, and therefore the speed, is eventually reduced to zero. By contrast, reducing the continuity produces an abrupt change in the direction of motion at the key position while the speed remains constant. This effect is often necessary to create realistic animation. For example, to make the movement of a ball careening off a tree look convincing, the animator must introduce a sharp corner in the path as the ball hits the tree. Increasing the tension would produce a corner in the path, but the speed at this corner would be zero, resulting in a *deceleration* before the ball actually hits. Reducing the continuity would produce the desired abrupt change, with the ball altering its direction of motion at the point of impact without slowing down ahead of time.

Generating the appropriate motion dynamics is extremely important in animation. If we look only at the motion path, the use of maximum tension appears to generate a discontinuity. However, this effect is simply a *geometric* discontinuity. What is needed in the case of the ball's encounter with the tree is a *parametric* discontinuity, a sudden change in the magnitude and/or direction of the velocity. This effect cannot be achieved by changing the tension because the tangent vector will always remain continuous under such a change, thus guaranteeing parametric continuity.

## 3.3. Bias

The bias parameter $b$ controls the direction of the path as it passes through a key position (Figure 20).

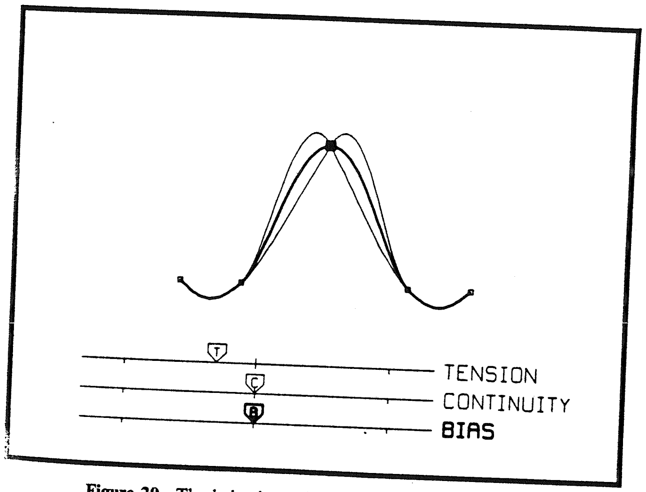

**Figure 20.** The behaviour of a curve under changing bias.

Both incoming and outgoing parts of the tangent are formed as an average of the incoming and outgoing chords, but the bias assigns different weights to the two chords when forming the average. Assuming default tension and continuity ($t=c=0$), the tangent vector is given by:

$$DS_i = DD_i = \frac{1+b}{2} \cdot (P_i - P_{i-1}) + \frac{1-b}{2} \cdot (P_{i+1} - P_i)$$

**Equation 7.** The bias equation.

Note that with $b=0$ the two chords are weighted equally, and we obtain the default spline shown in Figure 21. When $b=-1$, the tangent vector is completely determined by the destination chord (Figure 22),

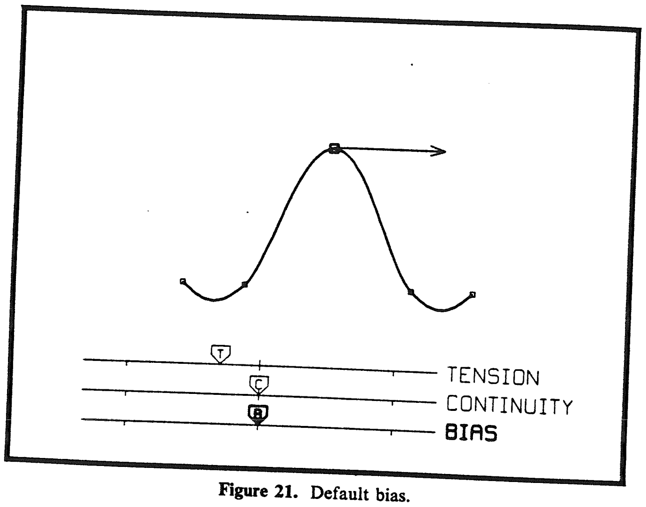

**Figure 21.** Default bias.

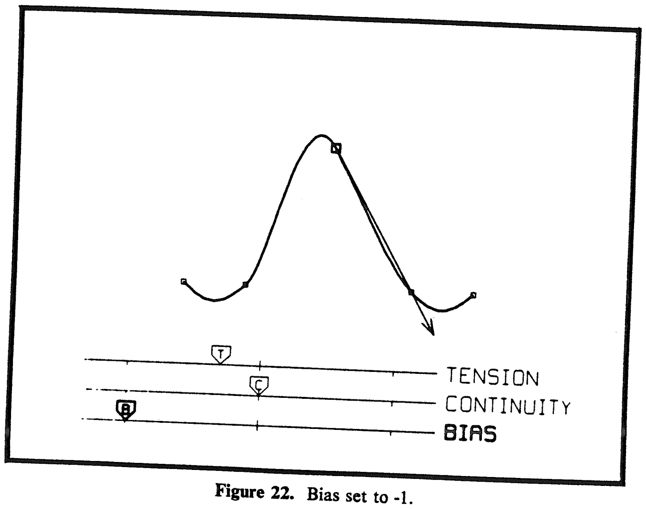

**Figure 22.** Bias set to -1.

whereas with $b=1$ it is completely determined by the source chord (Figure 23).

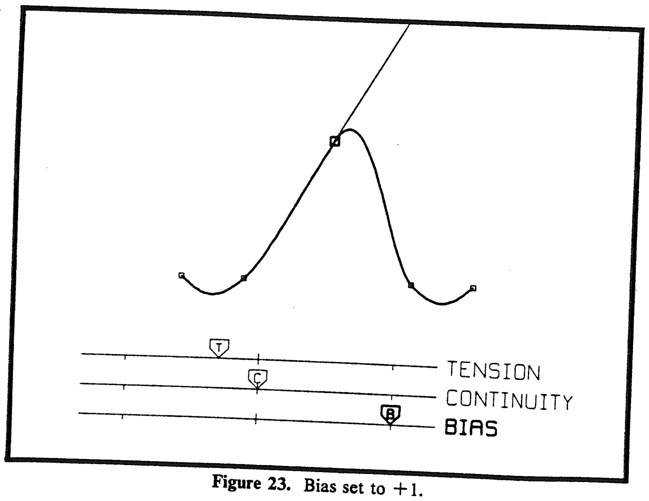

**Figure 23.** Bias set to +1.

The bias parameter easily simulates the traditional animation effect of following through after an action by "overshooting" the key position ($b=1$), or exaggerating a movement by "undershooting" a key position ($b=-1$).

### 3.4. Composite Control

Combining the tension, continuity, and bias control parameters we obtain the following general equations for the source and destination tangent vectors at the key position $P_i$:

$$DS_i = \frac{(1-t)\cdot(1-c)\cdot(1+b)}{2}\cdot(P_i-P_{i-1}) + \frac{(1-t)\cdot(1+c)\cdot(1-b)}{2}\cdot(P_{i+1}-P_i)$$

**Equation 8.** The "incoming" composite equation.

$$DD_i = \frac{(1-t)\cdot(1+c)\cdot(1+b)}{2}\cdot(P_i-P_{i-1}) + \frac{(1-t)\cdot(1-c)\cdot(1-b)}{2}\cdot(P_{i+1}-P_i)$$

**Equation 9.** The "outgoing" composite equation.

The spline segment between $P_i$ and $P_{i+1}$ can now be defined in terms of $P_i$, $P_{i+1}$, $DD_i$, and $DS_{i+1}$. All inbetween positions within this interval can then be generated by using Equation 2, varying the interpolation parameter $s$ from 0 to 1 over the interval.

## 4. Adjustments for Parameter Step Size

If we assume default continuity ($c=0$) at key $P_i$, the spline segment between $P_i$ and $P_{i+1}$ should join smoothly with the segment used between $P_{i-1}$ and $P_i$. While this is true when looking only at the path of the motion, it may not necessarily be true for the speed of motion. The formulas given in Equation 8 and Equation 9 assume an equal time spacing of key frames, implying an equal number of inbetweens within each key interval. A problem can exist if the animator requests a different number of inbetweens for adjacent intervals. Consider the case where 10 inbetweens are supposed to be generated in the interval from $P_{i-1}$ to $P_i$, but only 5 inbetweens between $P_i$ and $P_{i+1}$ as was shown in Figure 2. In the first interval, the step size for the interpolation parameter $s$ will be $\Delta_1 = \frac{1}{11}$ whereas for the second interval the step size will be $\Delta_2 = \frac{1}{6}$. If the same parametric derivative is used for both splines at $P_i$, these different step sizes will cause a discontinuity in the speed of motion. What is required, if this discontinuity is not intentional, is a means of making a local adjustment to the interval separating successive frames before and after the key frame so that the speed of entry matches the speed of exit. This can be accomplished by adjusting the specification of the tangent vector at the key frame based on the number of inbetweens in the adjacent intervals. In practice this turns out to be very simple, because we have already made provisions for two distinct tangent vectors at each key position in order to accommodate the continuity control parameter. Once the tangent vectors have been found for an equal number of inbetweens in the adjacent intervals, the adjustment required for different numbers of inbetweens ($N_{i-1}$ frames between $P_{i-1}$ and $P_i$ followed by $N_i$ frames between $P_i$ and $P_{i+1}$) can be made by weighting the tangent vectors appropriately:

$$\text{adjusted } DD_i = DD_i \cdot \frac{2\cdot N_{i-1}}{N_{i-1}+N_i}$$
$$\text{adjusted } DS_i = DS_i \cdot \frac{2\cdot N_i}{N_{i-1}+N_i}$$

**Equation 10.** Adjustment for parameter stepsize.

## 5. Current Experience and Future Developments

The tension, continuity, and bias parameters were designed to correspond closely to traditional animation effects. These ideas were tested and refined using a simple interactive spline display package. In one test we asked the animators to interactively modify the three control parameters until a spline passing through a set of key positions looked "natural" to them. We found that most animators left the continuity and bias at their defaults (c=0, b=0) but reduced the tension parameter. The tension values which animators considered to produce a "natural" looking curve for a given set of keys ranged from -0.1 to -0.4. It was interesting to note that while there was some disagreement between animators about the "best" tension value, each animator was surprisingly consistent in his choice.

These individual preferences are most likely related to differences in personal style. For example, some animators tend to animate movements more tightly (larger tension value) than others. This indicates that the tension is especially useful as a global default parameter which the animator can define according to his style preference. The continuity and bias parameters are mostly used in a local context, i.e. to achieve a particular effect at a specific key position. In practice, a small number of parameter combinations will probably be sufficient for most sequences. Several animators noted that the system should **include but not be limited to** a small set of predefined effects (e.g. "overshoot": t=-0.3, c=0, b=1). They still wanted to have the full power and flexibility of the three independent parameters available for more advanced users.

The algorithm described in this paper is currently being implemented at the National Film Board of Canada (free-form drawn 2-D and 2 1/2-D multiplane key frames) and the University of Waterloo (3-D key positions with skeleton control). These implementations will be used to test various user interface strategies for both inexperienced and advanced users.

## 6. Summary

The introduction of tension, continuity, and bias control produces a very general class of interpolating cubic splines. The flexibility which these parameters provide is especially useful in key frame animation, because it allows the animator to adjust the movement of objects without having to adjust or redraw the key frames (Figure 24).

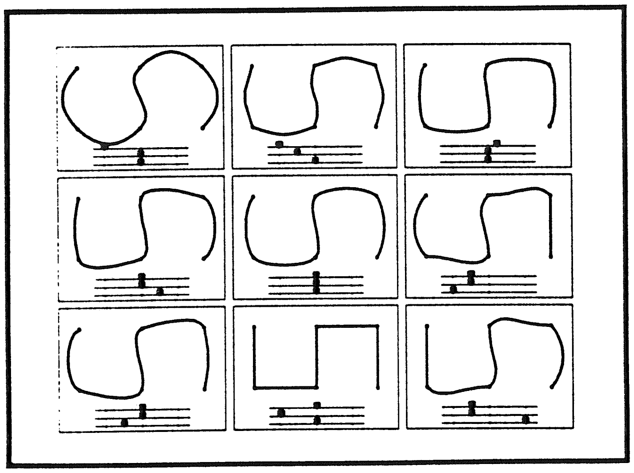

**Figure 24.** The effect of various parameter settings used together.

We have provided these control parameters by a simple technique of separating the tangent vector at each key frame into an incoming and an outgoing part and specifying these parts as a weighted average of the chords defined by the nearest neighbor key frames. Our system has the additional benefit that it can make adjustments to overcome speed discontinuities when the number of inbetweens is changed between key frames.

## 7. Acknowledgements

Feedback from animators is particularly important in a project such as this. We would especially like to thank Robert Forget, Rene Jodoin, Daniel Langlois, and Marc Aubry from the French Animation Studio of the National Film Board of Canada who as "test users" made many valuable suggestions which helped to clarify and refine our ideas.

Most of the illustrations in this paper are frames from the film "Interpolating Splines with Variable Tension, Continuity, and Bias" (National Film Board of Canada, 1984). We gratefully acknowledge the assistance of the Defence and Civil Institute of Environmental Medicine, Toronto, in the filming of this production.

This research was supported in part by the National Film Board of Canada, the National Sciences and Engineering Research Council of Canada, the Province of Ontario's BILD program, the University of Waterloo, and the Computer Graphics Laboratory.

## 8. Bibliography

[1] R. Baecker, "Interactive Computer-Mediated Animation", PhD Thesis, MIT, Project MAC Technical Report MAC-TR-61, 1969.

[2] B. Barsky, "The Beta-spline: A Local Representation Based on Shape Parameters and Fundamental Geometric Measures", Ph. D. dissertation, Department of Computer Science, University of Utah, December, 1981.

[3] B. Barsky and J. Beatty, "Local Control of Bias and Tension in Beta-Splines", *Computer Graphics (SIGGRAPH '83)*, **17** (**3**), pp. 193-218, July, 1983.

[4] R. Bartels, J. Beatty, and B. Barsky, "An Introduction to the Use of Splines in Computer Graphics", Department of Computer Science, University of Waterloo, TR CS-83-09, August 1983.

[5] N. Burtnyk and M. Wein, "Computer Generated Key Frame Animation", *Journal of the SMPTE 80*, pp. 149-153, March, 1971.

[6] N. Burtnyk and M. Wein, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation", Communications of the ACM **19** (**10**) pp. 564-569, October, 1976.

[7] D. Greenberg, S. Wu, and J. Abel, "An Interactive Computer Graphics Approach to Surface Representation", Communications of the ACM **20** (**10**) pp. 703-712, October, 1977.

[8] D. Kochanek, "A Computer System for Smooth Keyframe Animation", MMath Thesis, Department of Computer Science, University of Waterloo, Technical Report CS-82-42, December, 1982.

[9] W. Reeves, "Inbetweening for Computer Animation Utilizing Moving Point Constraints", *Computer Graphics (SIGGRAPH '81)*, **15** (**3**), pp. 263-269, August, 1981.

[10] A. Smith, "Spline Tutorial Notes – Technical Memo No. 77", *SIGGRAPH '83 Tutorial Notes: Introduction to Computer Animation*, pp. 64-75, July, 1983.

[11] M. Tuori, "Tools and Techniques for Computer-aided Animation", MSc Thesis, Department of Computer Science, University of Toronto, 1977.